Title
Towards Interpretability and Robustness of Machine Learning Models

Author
Chen, Jianbo

Publication Date
2019

Peer reviewed|Thesis/dissertation

Towards Interpretability and Robustness of Machine Learning Models

by

Jianbo Chen

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Statistics

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor Michael I. Jordan, Co-chair
Professor Martin J. Wainwright, Co-chair
Assistant Professor Thomas Courtade
Assistant Professor William Fithian

Fall 2019

Towards Interpretability and Robustness of Machine Learning Models

Abstract

Towards Interpretability and Robustness of Machine Learning Models

by

Jianbo Chen

Doctor of Philosophy in Statistics

University of California, Berkeley

Professor Michael I. Jordan, Co-chair

Professor Martin J. Wainwright, Co-chair

Modern machine learning models can be difficult to probe and understand after they have been trained. This is a major problem for the field, with consequences for trustworthiness, diagnostics, debugging, robustness, and a range of other engineering and human interaction issues surrounding the deployment of a model. Another problem of modern machine learning models is their vulnerability to small adversarial perturbations to the input, which incurs a security risk when they are applied to critical areas.

In this thesis, we develop systematic and efficient tools for interpreting machine learning models and evaluating their adversarial robustness. Part I focuses on model interpretation. We derive an efficient feature scoring method by exploiting the graph structure in data. We also develop a learning-based method under an information-based framework. As an attempt to leverage prior knowledge about what constitutes a satisfying interpretation in a given domain, we propose a systematic approach to exploiting syntactic constituency structure by leveraging a parse tree for interpretation of models in the setting of linguistic data. Part II focuses on the evaluation of adversarial robustness. We first propose a probabilistic framework for generating adversarial examples on discrete data, and develop two algorithms to implement it. We also introduce a novel attack method in the setting where the attacker has access to model decisions alone. We investigate the robustness of various machine learning models and existing defense mechanisms under the proposed attack method. In Part III, we build a connection between the two fields by developing a method for detecting adversarial examples via tools in model interpretation.

To my family

Contents

Acknowledgments

This thesis would not exist without the support of many people who have helped me during the past four years.

The most valuable and fortunate thing in my Ph.D. study is that I have Michael I. Jordan and Martin J. Wainwright as my advisors. Both of them have provided me with tremendous support along my wonderful journey. I met Mike and Martin in their classes first, and decided to seek their advising after that. Mike and Martin have given me full freedom to explore my new ideas and investigate new directions, and have guided me to overcome failure and difficulties in an encouraging and motivational manner. With their keen intellect and broad knowledge, they are always ready to provide me with thoughtful suggestions when I am faced with obstacles. I still remember how astonished I was when Mike suggested a connection between two seemingly unrelated problems in two different fields, and when Martin located the solution to a cutting edge problem at a book in the 90s on his shelf. These things happened during many of our discussions. Not only have they advised me academically, but they have also been my life mentors. Mike has provided countless thoughtful suggestions on my career path. Martin has taught me it is never more important to maintain a balance between life, family and work. Both of them have offered me multiple career opportunities to explore my own interest, and are willing to spend time discussing trade-offs in my career choice. Mike also holds parties at his home every semester. Surrounded by my group mates, I often feel I have a second family.

I would also like to express my thanks to faculty and staff members in Statistics and EECS. Will Fithian, a committee member in my qualifying exam and dissertation, has given constructive suggestions on my projects in multiple hypothesis testing. Peter Bartlett's group meeting has equipped me with a theoretical understanding of neural networks and online learning. I would like to thank Thomas Courtade, who was willing to be a committee member in both the qualifying exam and the dissertation, and provided useful feedback. I also learned a lot from lectures and courses given by many professors at Berkeley, including Michael I. Jordan, Peter Bartlett, Chris Paciorek, Martin J. Wainwright, Ben Recht, Jitendra Malik, and Alexei A. Efros. Staff members at the department of statistics have also supported me throughout my graduate study. La Shana Porlaris and Mary Melinn are always ready to resolve my various questions and problems when I have trouble.

I owe a lot to my collaborators as well. My research interest in model interpretation was motivated by Prof. Le Song at Georgia Institute of Technology, who has shared many sharp thoughts with me on several projects. The collaboration with Puyudi Yang, Prof. Jane-Ling Wang at UC Davis and Prof. Cho-Jui Hsieh at UCLA aroused my research interests in adversarial robustness. I am also fortunate to have worked with Aaditya Ramdas, from whom I learned not only knowledge but also principled methodology in doing research. Yelong Shen, Jianfeng Gao, Jingjing Liu and Xiaodong Liu have exposed me to research in industry during my internship at Microsoft Research. I also have a very nice experience in collaborating with my friend Mitchell Stern on feature selection, and with my friends and senior fellows Lihua Lei and Cheng Ju on stochastic optimization.

I am so fortunate to have known many mentors and friends at Berkeley. I came to Berkeley as an exchange student during my undergraduate study, where I spent a fruitful semester. I met Prof. Yan Zhang and Prof. F. Alberto Grünbaum during my exchange study, who have built up my confidence in applying for a Ph.D. program. I am also thankful to my friends at Berkeley who have made my past four years colorful, in particular my academic brothers and sisters in Mike and Martin's groups such as Ashwin Pananjady, Yuchen Zhang, Fanny Yang, Yuting Wei, Raaz Dwivedi, Billy Fang, Joe Borja, Nilesh Tripuraneni, Mitchell Stern, and Koulik Khamaru, my roommate Haoran Tang, my senior fellows Cheng Ju, Lihua Lei, Wenpin Tang, Siqi Wu, and my friends Yuting Ye, Da Xu, Xiao Li and many others.

I would not be at Berkeley without the support of my undergraduate advisors at the University of Hong Kong. Prof. Guangyue Han's reading group equipped me with knowledge in stochastic differential equations, with whom I also spent a wonderful year on information theory and network coding. Under the guidance of Prof. Ngaiming Mok, I had a fantastic tour in the world of complex differential geometry. I finished my first research project in statistics with Prof. Stephen Man Sing Lee, who is so patient and generous, and always available for a discussion. He has cultivated my interest in statistics.

Last but not least, my family has made me who I am. The words cannot express my thanks to my grandfather Shunxu Chen (陈顺序) , who taught me how to write my first Chinese and English words, and my grandmother Fuying Wang (王福英) , who taught me the first digits and the first equations in my life. Both of them have cultivated my interests in art, science and mathematics since my childhood. I also owe numerous credits to my parents Wei Chen (陈巍) and Fen Chen (陈奋) , who have never hesitated in supporting my education. I would like to thank my aunt Miao Chen (陈苗) as well, who takes care of me as if she were another Mom. All of them have devoted themselves to my growth selflessly. Finally, I express my special thanks to my wife Puyudi Yang (杨璞玉迪) , who keeps company with me in Shanghai, Hong Kong, Berkeley, Seattle and Chicago. Without her love, support and care, I would not have the confidence and power to go through the challenges and difficulties I met in the past seven years.

Chapter 1

Introduction

Interpretability and adversarial robustness are important criteria when a machine learning model is applied in critical areas such as finance, medicine, criminal justice and transportation. Many complex models, such as random forests, deep neural networks, and kernel methods, have been developed and employed to optimize prediction accuracy, which can compromise their ease of interpretation and adversarial robustness.

This thesis aims to provide systematic and efficient tools for interpreting machine learning models and evaluating their adversarial robustness. It further builds a connection between the two fields via the application of tools in model interpretation to the detection of adversarial examples.

1.1 Model interpretation

Modern machine learning models can be difficult to probe and understand after they have been trained. This is a major problem for the field, with consequences for trustworthiness, diagnostics, debugging, robustness, and a range of other engineering and human interaction issues surrounding the deployment of a model.

There have been several lines of attack on this problem. One involves changing the model design or training process so as to enhance interpretability. This can involve retreating to simpler models and/or incorporating strong regularizers that effectively simplify a complex model. In both cases, however, there is a possible loss of prediction accuracy. Models can also be changed in more sophisticated ways to enhance interpretability; for example, attention-based methods have yielded deep models for vision and language tasks that improve interpretability at no loss to prediction accuracy [1–7].

Another approach treats interpretability as a separate problem from prediction, which will be a focus of this thesis. Given a predictive model, an interpretation method yields, for each instance to which the model is applied, a vector of importance scores associated with the underlying features. The instancewise property means that this vector, and hence the relative importance of each feature, is allowed to vary across instances. Thus, the importance

scores can act as an explanation for the specific instance, indicating which features are the key for the model to make its prediction on that instance. Most of the stand-alone interpretation methods approximate the model to be interpreted via a locally additive model in order to explain the difference between the model output and some "reference" output in terms of the difference between the input and some "reference" input. The coefficients of the locally additive model are used as importance scores. Such methods are named as *instancewise feature importance scoring* and *feature attribution* interchangeably throughout the thesis. There are two main differences between various methods: the information being used for approximation, and the criterion of a good approximation.

Based on whether the information involves model details, interpretation methods can be classified as being model-agnostic or model-aware. Model-aware methods require additional assumptions, or are specific to a certain class of models [8–13]. The majority of studies on model-aware methods focus on neural network models, which are compositions of functions with certain types, such as linear functions and nonlinear activations. For this type of models, interpretation methods often seek to compose layerwise approximations, or construct a gradient-based approximation. Model-agnostic methods can be applied in a black-box manner to arbitrary models [14–21]. Such methods often draw multiple samples by masking part of the features by some reference value for the given instance, and constructs the linear model via aligning the output of the linear model and the output of the model to be explained in a certain way. While model-aware methods require access to model details and are relatively more inconvenient to implement in practice, model-agnostic methods often suffer from query inefficiency. We will discuss how to improve the efficiency of model-agnostic methods in Chapter 2 and Chapter 3.

The criterion for being a good local approximation of the original model also varies from method to method. Such criteria outline the definition of importance, describe the properties of importance scores, and even suggest potential applications of a method [14, 15]. A common standard in testing the effectiveness of a method is human evaluation, such as the alignment between human explanations and output scores from an interpretation method. However, human evaluation may suffer from cost and variance among different human subjects, and also rely on the design of experiments. A more sophisticated approach is to formalize the set of criteria to be imposed, which are called axioms, and then mathematically derive the method of distributing importance scores based on the provided rules. Recently, a handful of axiomatic frameworks have been employed for the mathematical definition of a faithful interpretation [12, 15–17, 21]. Common examples include the Shapley value [22] and the Banzhaf value [23]. Such axiomatic frameworks are often rooted in economics and game theory, originally proposed as an axiomatic characterization of a fair distribution of a total surplus from all the players in a cooperative game. When they are applied to predictive models, each feature is modeled as a player in the underlying game. There has not been a consensus among researchers which axiomatic framework is the most appropriate for model interpretation yet.

Interpretation methods have a variety of applications. They can help people build trust in machine learning models, and help engineers and researchers delve into the black box of

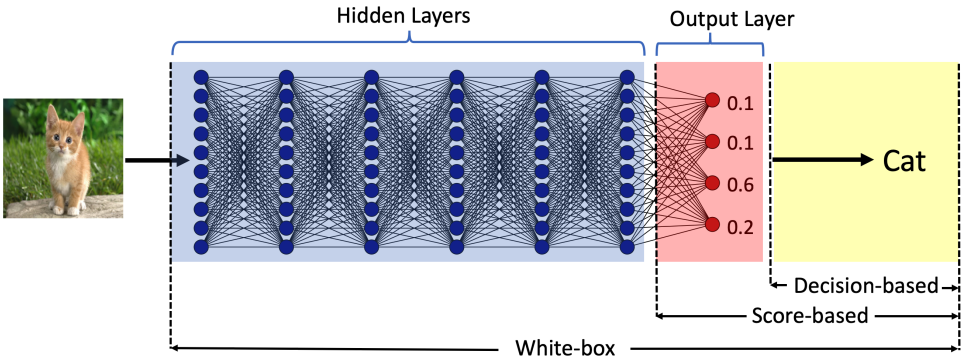


Figure 1.1: An illustration of accessible components of the target model for each of the three threat models. A white-box threat model assumes access to the whole model; a score-based threat model assumes access to the output layer; a decision-based threat model assumes access to the predicted label alone.

machine learning models. See a discussion of such applications in Lipton [24]. We discuss an application of interpretation tools to the detection of adversarial examples in Chapter 7.

1.2 Adversarial robustness

Machine learning models have been shown to be vulnerable to *adversarial examples*—that is, maliciously perturbed examples that are almost identical to original samples in human perception, but cause models to make incorrect decisions [25]. The vulnerability of machine learning models to adversarial examples implies a security risk in applications with real-world consequences, such as self-driving cars, robotics, financial services, and criminal justice; in addition, it highlights fundamental differences between human learning and existing machine-based systems. The study of this phenomenon is carried out from two directions: evaluating the adversarial robustness of a model by designing new algorithms for generating adversarial examples, and improving the adversarial robustness by designing mechanisms to identify and conquer adversarial examples.

1.2.1 Generating adversarial examples

Recent years have witnessed a flurry of research on the design of new algorithms for generating adversarial examples [25–40]. Adversarial examples can be categorized according to at least three different criteria: the similarity metric, the attack goal, and the threat model.

For continuous inputs such as image data, commonly used similarity metrics are ℓ_p-distances between adversarial and original examples with $p \in \{0, 2, \infty\}$. For discrete inputs, the number of distinct features, or the ℓ_0-distance, has been used to quantify the perturbation size. For specific types of data, further constraints may be added to the perturbation, such

as linguistic coherence in text data. We remark here that all similarity metrics are only an approximation of similarity as observed by humans. While they may be far from accurate, a small value under those metrics often suggests the adversarial perturbation does not affect the perception of human observers.

The goal of attack is either untargeted or targeted. The goal of an untargeted attack is to perturb the input so as to cause any type of misclassification, whereas the goal of a targeted attack is to alter the decision of the model to a pre-specific target class. Changing the loss function allows for switching between two types of attacks [27–29].

Perhaps the most important criterion in practice is the *threat model*, of which there are two primary types: white-box and black-box. See Figure 1.1 for an illustration of accessible components of the target neural network classifier for different threat models. In the white-box setting, an attacker has complete access to the model. The majority of work assumes the target model is a neural network with continuous inputs. Under this setting, the generation of adversarial examples is often formulated as an optimization problem, which is solved by gradient-based methods either via treating misclassification loss as a regularization [25, 29] or via tackling the dual as a constrained optimization problem [26, 27, 30]. In the black-box setting, an attacker can only access outputs of the target model. Based on whether one has access to the full probability or the label of a given input, black-box attacks are further divided into score-based and decision-based. Score-based methods use zeroth-order gradient estimation to craft adversarial examples for continuous inputs [32–34]. The most practical threat model is that in which an attacker has access to decisions alone. A widely studied type of the decision-based attack is transfer-based attack [35–37]. In recent work, decision-based attacks relying neither on training data nor on the assumption of transferability have been proposed for continuous inputs [33, 38, 40]. One limitation, however, is that they require a relatively large number of model queries, rendering it impractical for real-world applications. We propose HopSkipJumpAttack in Chapter 6, which leads to a significant improvement in query efficiency.

1.2.2 Improving the adversarial robustness of machine learning models

To improve the robustness of neural networks, the community adopts two directions. One direction is *adversarial defense*, which tries to alter the training or design of the original model, so as to defend against adversarial examples. The other direction is *adversarial detection*, which focuses on detecting adversarial examples in the test stage, by identifying the fundamental difference between original and adversarial examples.

Various approaches have been proposed to defend against adversarial attacks, including adversarial training [26, 27, 30, 41, 42], distributional smoothing [43], defensive distillation [44], generative models [45], feature squeezing [46], randomized models [47–49], and verifiable defense [50, 51]. These defenses often lead to loss of accuracy [52], or involve modifications in the training process of a model, which often require higher computational or

sample complexity [53].

Complimentary to the previous defending techniques, an alternative line of work focuses on screening out adversarial examples in the test stage without touching the training of the original model. To identify the underlying feature discriminating adversarial examples from original examples, existing work has proposed to impose data transformations such as principle component analysis [54–57], to train an alternative neural network or Gaussian discriminant classifier to classify adversarial and original images [58–60], and to use kernel density estimate (KD), Bayesian uncertainty (BU) [61], and Local Intrinsic Dimension (LID) [62]. In Chapter 7, we discuss an attempt to apply tools developed for model interpretation to the detection of adversarial examples, achieving superior performance to existing detection methods across various attacks.

It is important to evaluate techniques for adversarial defense and detection under different threat models. When a black-box threat model is assumed, decision-based attacks may be employed. Decision-based attacks may be used as a simple and efficient first step to test the effectiveness of defense and detection techniques when they are non-differentiable, as we will discuss in Chapter 6. A second threat model is where an attacker has access to model details, but does not have the knowledge of the defense mechanism. Finally, it is also important to evaluate the technique under the setting where an attacker has complete access to both the model and the defense mechanism.

1.3 Contributions of this thesis

This section highlights the contributions of this thesis. At a high level, we hope to address several limitations of current research in model interpretation and adversarial robustness:

- Current model-agnostic interpretation methods fall short of query efficiency in computing importance scores. In Chapter 2, we derive an efficient feature scoring method by exploiting the graph structure in data. In Chapter 3, we propose a learning-based method to construct a global explainer under an information-based framework.

- Current model-agnostic methods provide little opportunity to leverage prior knowledge about what constitutes a satisfying interpretation in a given domain. In Chapter 4, we propose to exploit syntactic constituency structure by leveraging a parse tree when interpreting trained classification models in the setting of linguistic data sets.

- The majority of current algorithms for generating adversarial examples are applicable to continuous data. In Chapter 5, we propose a probabilistic framework for generating adversarial examples on discrete data and create two algorithms to implement it.

- Existing decision-based adversarial attacks require a large number of model queries. In Chapter 6, we introduce HopSkipJumpAttack, which significantly improves the query efficiency. We successfully apply our algorithm to several defense mechanisms, and other machine learning models besides neural networks.

- In Chapter 7, we discuss an application of model interpretation to detecting adversarial examples, achieving state-of-the-art performance against various types of attacks.

The first two chapters are devoted to developing query-efficient algorithms for model interpretation. In Chapter 2, we focus on the settings in which a graph structure is appropriate for describing the relations between features in the data (e.g., chains for sequences and grids for images), and distant features according to the graph have weak interaction. We study feature scoring in the framework of Shapley value. By exploiting the underlying graph structure, the number of model queries is reduced to linear—as opposed to exponential—in the number of features.

In Chapter 3, we further reduce the query complexity by learning a feature selector to exact a subset of features that are most informative for each given example. The selector is trained to maximize the mutual information between selected features and the response variable, where the conditional distribution of the response variable given the input is the model to be explained. In the test stage, the selector directly outputs the subset of important features, without evaluating the model to be interpreted.

In Chapter 4, we study the problem of interpreting trained classification models in the setting of linguistic data sets. We assign least-squares-based importance scores to each word of an instance by exploiting syntactic constituency structure. The proposed scores are related to the framework of the Banzhaf value. We further develop a principled method for detecting and quantifying interactions between words in a sentence, motivated by Cook's distance [63], a classical concept in linear regression.

The next two chapters address the generation of adversarial examples. We propose two algorithms in Chapter 5 based on a probabilistic framework for adversarial attack on discrete data. The first perturbation-based algorithm improves the state-of-the-art across several widely-used language models, and the second learning-based algorithm provides a scalable method for real-time generation of adversarial examples.

In Chapter 6, we develop HopSkipJumpAttack, a family of decision-based algorithms based on a novel estimate of the gradient direction using binary information at the decision boundary. We establish a convergence guarantee for the proposed attack, and demonstrate that it requires significantly fewer model queries than existing attacks. We further show that it also achieves competitive performance in attacking several widely-used defense mechanisms, and it can be used a tool for evaluating robustness of non-differentiable machine learning models.

In Chapter 7, we investigate the application of tools in model interpretation to the detection of adversarial examples. In particular, we introduce a new framework of adversarial detection through thresholding a scale estimate of feature attribution scores. Through vast experiments, our method achieves superior performances in distinguishing adversarial examples from popular attack methods on a variety of real data sets among state-of-the-art detection methods.

Previously published work

This thesis is composed of previously published papers, which are joint work with several other collaborators. Chapter 2 and Chapter 3 are modified from two papers with Le Song, Martin J. Wainwright, and Michael I. Jordan respectively [19, 20]. Chapter 5 is based on the work with Puyudi Yang, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan [64]. Chapter 6 is based on the work with Michael I. Jordan and Martin J. Wainwright [65]. Chapter 7 is based on the work with Puyudi Yang, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I. Jordan [66].

Part I

Interpretability

Chapter 2

Efficient Model Interpretation for Structured Data

Instancewise feature scoring is a method for model interpretation, which yields, for each test instance, a vector of importance scores associated with the feature vector. Methods based on the Shapley score have been proposed as a fair way of computing feature attributions of this kind, but incur an exponential complexity in the number of features for black-box models. This combinatorial explosion arises from the definition of the Shapley value and prevents these methods from being scalable to large data sets and complex models. We focus on settings in which the data have a graph structure, and the contribution of features to the target variable is well-approximated by a graph-structured factorization. In such settings, we develop two algorithms with linear complexity for instancewise feature importance scoring on black-box models. We establish the relationship of our methods to the Shapley value and a closely related concept known as the Myerson value from cooperative game theory. We demonstrate on both language and image data that our algorithms compare favorably with other methods for model interpretation.

2.1 Introduction

In this chapter, we study instancewise feature importance scoring as a specific approach to the problem of interpreting the predictions of black-box models. Given a predictive model, such a method yields, for each instance to which the model is applied, a vector of importance scores associated with the underlying features. The instancewise property means that this vector, and hence the relative importance of each feature, is allowed to vary across instances. Thus, the importance scores can act as an explanation for the specific instance, indicating which features are the key for the model to make its prediction on that instance.

There is now a large body of research focused on the problem of scoring input features based on the prediction of a given instance [see, e.g., 9, 10, 12, 14–17, 67]. Of most relevance to this paper is a line of recent work [15–17] that has developed methods for model inter-

pretation based on Shapley value [22] from cooperative game theory. The Shapley value was originally proposed as an axiomatic characterization of a fair distribution of a total surplus from all the players, and can be applied to predictive models, in which case each feature is modeled as a player in the underlying game. While the Shapley value approach is conceptually appealing, it is also computationally challenging: in general, each evaluation of a Shapley value requires an exponential number of model evaluations. Different approaches to circumventing this complexity barrier have been proposed, including those based on Monte Carlo approximation [16, 17] and methods based on sampled least-squares with weights [15].

In this chapter, we take a complementary point of view, arguing that the problem of explanation is best approached within a model-based paradigm. In this view, explanations are cast in terms of a model, which may or may not be the same model as used to fit the data. Criteria such as Shapley value, which are intractable to compute when no assumptions are made, can be more effectively computed or approximated within the framework of a model. We focus specifically on settings in which a graph structure is appropriate for describing the relations between features in the data (e.g., chains for sequences and grids for images), and distant features according to the graph have weak interaction during the computation of Shapley values. We propose two methods for instancewise feature importance scoring in this framework, which we term *L-Shapley* and *C-Shapley*; here the abbreviations "L" and "C" refer to "local" and "connected," respectively. By exploiting the underlying graph structure, the number of model evaluations is reduced to linear—as opposed to exponential—in the number of features. We demonstrate the relationship of these measures with a constrained form of Shapley value, and we additionally relate C-Shapley with another solution concept from cooperative game theory, known as the Myerson value [68]. The Myerson value is commonly used in graph-restricted games, under a local additivity assumption of the model on disconnected subsets of features. Finally, we apply our feature scoring methods to several state-of-the-art models for both language and image data, and find that our scoring algorithms compare favorably to several existing sampling-based algorithms for instancewise feature importance scoring.

2.2 Background and preliminaries

We begin by introducing some background and notation for instancewise feature importance scoring and the Shapley value.

2.2.1 Importance of a feature subset

We are interested in studying models that are trained to perform prediction, taking as input a feature vector $x \in \mathcal{X} \subset \mathbb{R}^d$ and predicting a response or output variable $y \in \mathcal{Y}$. We assume access to the output of a model via a conditional distribution, denoted by $\mathbb{P}_m(\cdot|x)$, that provides the distribution of the response $Y \in \mathcal{Y}$ conditioned on a given vector $X = x$ of inputs. For any given subset $S \subset \{1, 2, \ldots, d\}$, we use $x_S = \{x_j, j \in S\}$ to denote the

associated sub-vector of features, and we let $\mathbb{P}_m(Y \mid x_S)$ denote the induced conditional distribution when $\mathbb{P}_m$ is restricted to using only the sub-vector x_S. In the corner case in which $S = \emptyset$, we define $\mathbb{P}_m(Y \mid x_\emptyset) := \mathbb{P}_m(Y)$. In terms of this notation, for a given feature vector $x \in \mathcal{X}$, subset S and fitted model distribution $\mathbb{P}_m(Y \mid x)$, we introduce the *importance score*

$$v_x(S) := \mathbb{E}_m \left[- \log \frac{1}{\mathbb{P}_m(Y \mid x_S)} \;\middle|\; x \right],$$

where $\mathbb{E}_m[\cdot \mid x]$ denotes the expectation over $\mathbb{P}_m(\cdot \mid x)$. The importance score $v_x(S)$ has a coding-theoretic interpretation: it corresponds to the negative of the expected number of bits required to encode the output of the model based on the sub-vector x_S. It will be zero when the model makes a deterministic prediction based on x_S, and larger when the model returns a distribution closer to uniform over the output space.

There is also an information-theoretic interpretation to this definition of importance scores, as discussed in Chen et al. [19]. In particular, suppose that for a given integer $k < d$, there is a function $x \mapsto S^*(x)$ such that, for all almost all x, the k-sized subset $S^*(x)$ maximizes $v_x(S)$ over all subsets of size k; then we are guaranteed that the mutual information $I(X_{S^*(X)}, Y)$ between $X_{S^*(X)}$ and Y is maximized, over any conditional distribution that generates a subset of size k given X. The converse is also true.

In many cases, class-specific importance is favored, where one is interested in seeing how important a feature subset S is to the predicted class, instead of the prediction as a conditional distribution. In order to handle such cases, it is convenient to introduce the degenerate conditional distribution

$$\hat{\mathbb{P}}_m(y \mid x) := \begin{cases} 1 \text{ if } y \in \arg\max_{y'} \mathbb{P}_m(y' \mid x), \\ 0 \text{ otherwise.} \end{cases}$$

We can then define the importance of a subset S with respect to $\hat{\mathbb{P}}_m$ using the modified score

$$v_x(S) := \hat{\mathbb{E}}_m \left[- \log \frac{1}{\mathbb{P}_m(Y \mid x_S)} \;\middle|\; x \right],$$

which is the expected log probability of the predicted class given the features in S.

Estimating the conditional distribution: In practice, we need to estimate—for any given feature vector $\bar{x} \in \mathcal{X}$—the conditional probability functions $\mathbb{P}_m(y \mid \bar{x}_S)$ based on observed data. Past work has used one of two approaches: either estimation based on empirical averages [16], or plug-in estimation using a reference point [15, 17].

Empirical average estimation: In this approach, we first draw a set of feature vector $\{x^j\}_{j=1}^M$ by sampling with replacement from the full data set. For each sample x^j, we define a new vector $\tilde{x}^j \in \mathbb{R}^d$ with components $(\tilde{x}^j)_i$ equal to x_i^j if $i \in S$ and $\bar{x}_i$ otherwise. Taking the empirical mean of $\mathbb{P}_m(y \mid \tilde{x}^j)$ over $\{\tilde{x}^j\}$ then provides an estimate of $\mathbb{P}_m(y \mid \bar{x}_S)$.

Plug-in estimation: In this approach, the first step is to specify a reference vector $x^0 \in \mathbb{R}^d$ is specified. We then define the vector $\tilde{x} \in \mathbb{R}^d$ with components $(\tilde{x})_i$ equal to x_i if $i \in S$ and x_i^0 otherwise. Finally, we use the conditional probability $\mathbb{P}_m(y \mid \tilde{x})$ as an approximation

to $\mathbb{P}_m(y \mid \bar{x}_S)$. The plug-in estimate is more computationally efficient than the empirical average estimator, and works well when there exist appropriate choices of reference points. We use this method for our experiments, where we use the index of padding for language data, and the average pixel strength of an image for vision data.

2.2.2 Shapley value for measuring interaction between features

Consider the problem of quantifying the importance of a given feature index i for feature vector x. A naive way of doing so would be by computing the importance score $v_x(\{i\})$ of feature i on its own. However, doing so ignores interactions between features, which are likely to be very important in applications. As a simple example, suppose that we were interested in performing sentiment analysis on the following sentence:

It is not heartwarming or entertaining. It just sucks. ($\star$)

This sentence is contained in a movie review from the IMDB movie data set [69], and it is classified as negative sentiment by a machine learning model to be discussed in the sequel. Now suppose we wish to quantify the importance of feature "*not*" in prediction. The word "*not*" plays an important role in the overall sentence as being classified as negative, and thus should be attributed a significant weight. However, viewed in isolation, the word "*not*" has neither negative nor positive sentiment, so that one would expect that $v_x(\{\text{“}not\text{”}\}) \approx 0$.

Thus, it is essential to consider the interaction of a given feature i with other features. For a given subset S containing i, a natural way in which to assess how i interacts with the other features in S is by computing the difference between the importance of all features in S, with and without i. This difference is called the *marginal contribution* of i to S, and given by

$$m_x(S, i) := v_x(S) - v_x(S \setminus \{i\}). \tag{2.1}$$

In order to obtain a simple scalar measure for feature i, we need to aggregate these marginal contributions over all subsets that contain i. The *Shapley value* [22] is one principled way of doing so. For each integer $k = 1, \ldots, d$, we let $\mathcal{S}_k(i)$ denote the set of k-sized subsets that contain i. The Shapley value is obtained by averaging the marginal contributions, first over the set $\mathcal{S}_k(i)$ for a fixed k, and then over all possible choices of set size k:

$$\phi_x(\mathbb{P}_m, i) := \frac{1}{d} \sum_{k=1}^{d} \frac{1}{\binom{d-1}{k-1}} \sum_{S \in \mathcal{S}_k(i)} m_x(S, i). \tag{2.2}$$

Since the model $\mathbb{P}_m$ remains fixed throughout our analysis, we frequently omit the dependence of ϕ_x on $\mathbb{P}_m$, instead adopting the more compact notation $\phi_x(i)$.

The concept of Shapley value was first introduced in cooperative game theory [22], and it has been used in a line of recent work on instancewise feature importance ranking [15–17]. It can be justified on an axiomatic basis [22, 70] as being the unique function from a collection of 2^d numbers (one for each subset S) to a collection of d numbers (one for each feature i) with the following properties: (i) [Additivity] The sum of the Shapley values $\sum_{i=1}^{d} \phi_x(i)$ is equal to the difference $v_x(\{1, \ldots, d\}) - v_x(\emptyset)$. (ii) [Equal contributions] If $v_x(S \cup \{i\}) = v_x(S \cup \{j\})$

for all subsets S, then $\phi_x(i) = \phi_x(j)$. (iii) [Monotonicity] Given two models $\mathbb{P}_m$ and $\mathbb{P}_m$, let m_x and m'_x denote the associated marginal contribution functions, and let ϕ_x and ϕ'_x denote the associated Shapley values. If $m_x(S, i) \geq m'_x(S, i)$ for all subsets S, then we are guaranteed that $\phi_x(i) \geq \phi'_x(i)$. Note that all three of these axioms are reasonable in our feature selection context.

2.2.3 The challenge with computing Shapley values

The exact computation of the Shapley value $\phi_x(i)$ takes into account the interaction of feature i with all 2^{d-1} subsets that contain i, thereby leading to computational difficulties. Various approximation methods have been developed with the goal of reducing complexity. For example, Štrumbelj and Kononenko [16] proposed to estimate the Shapley values via a Monte Carlo approximation built on an alternative permutation-based definition of the Shapley value. Lundberg and Lee [15] proposed to evaluate the model over randomly sampled subsets and use a weighted linear regression to approximate the Shapley values based on the collected model evaluations.

In practice, such sampling-based approximations may suffer from high variance when the number of samples to be collected per instance is limited. (See Section 2.5.6 for an empirical evaluation.) For large-scale predictive models, the number of features is often relatively large, meaning that the number of samples required to obtain stable estimates can be prohibitively large. The main contribution of this paper is to address this challenge in a model-based paradigm, where the contribution of features to the response variable respects the structure of an underlying graph. In this setting, we propose efficient algorithms and provide bounds on the quality of the resulting approximation. As we discuss in more detail later, our approach should be viewed as complementary to sampling-based or regresssion-based approximations of the Shapley value. In particular, these methods can be combined with the approach of this paper so as to speed up the computation of the L-Shapley and C-Shapley values that we propose.

2.3 Methods

In many applications, the features can be associated with the nodes of a graph, and we can define distances between pairs of features based on the graph structure. Intuitively, features distant in the graph have weak interactions with each other, and hence excluding those features in the computation of Shapley value has little effect. For instance, each feature vector x in sequence data (such as language, music etc.), can be associated with a line graph, where positions too far apart in a sequence may not affect each other in Shapley value computation; similarly, each image data is naturally modeled with a grid graph, such that pixels that are far apart may have little effect on each other in the computation of Shapley value.

In this section, we propose modified forms of the Shapley values, referred to as L-Shapley and C-Shapley values, that can be computed more efficiently than the Shapley value by excluding those weak interactions in the structured data. We also show that under certain probabilistic assumptions on the marginal distribution over the features, these quantities yield good approximations to the original Shapley values.

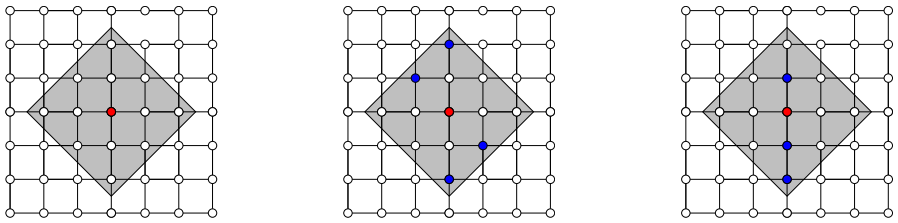
More precisely, given feature vectors $x \in \mathbb{R}^d$, we let $G = (V, E)$ denote a connected graph with nodes V and edges $E \subset V \times V$, where each feature i is associated with a a node $i \in V$, and edges represent interactions between features. The graph induces a distance function on $V \times V$, given by

$$d_G(\ell, m) = \text{number of edges in shortest path joining } \ell \text{ to } m. \tag{2.3}$$

In the line graph, this graph distance corresponds to the number of edges in the unique path joining them, whereas it corresponds to the Manhattan distance in the grid graph. For a given node $i \in V$, its *k-neighborhood* is the set

$$\mathcal{N}_k(i) := \{j \in V \mid d_G(i, j) \leq k\} \tag{2.4}$$

of all nodes at graph distance at most k. See Figure 2.1 for an illustration for the 2D grid graph.



(a) (b) (c)

Figure 2.1: In all cases, the red node denotes the target feature i. (a) Illustration of the $k = 2$ graph neighborhood $\mathcal{N}_2(i)$ on the grid graph. All nodes within the shaded gray triangle lie within the neighborhood $\mathcal{N}_2(i)$. (b) A disconnected subset of $\mathcal{N}_2(i)$ that is summed over in L-Shapley but not C-Shapley. (c) A connected subset of $\mathcal{N}_2(i)$ that is summed over in both L-Shapley and C-Shapley.

We propose two algorithms for approximating Shapley value in which features that are either far apart on the graph or features that are not directly connected have an accordingly weaker interaction.

2.3.1 Local Shapley

In order to motivate our first graph-structured Shapley score, let us take a deeper look at Example ($\star$). In order to compute the importance score of "*not,*" the most important words to be included are "*heartwarming*" and "*entertaining.*" Intuitively, the words distant from them have a weaker influence on the importance of a given word in a document, and

therefore have relatively less effect on the Shapley score. Accordingly, as one approximation, we propose the L-Shapley score, which only perturbs the neighboring features of a given feature when evaluating its importance:

Definition 1. *Given a model $\mathbb{P}_m$, a sample x and a feature i, the* L-Shapley estimate of order k *on a graph G is given by*

$$\hat{\phi}_x^k(i) := \frac{1}{|\mathcal{N}_k(i)|} \sum_{\substack{T \ni i \\ T \subseteq \mathcal{N}_k(i)}} \frac{1}{\binom{|\mathcal{N}_k(i)|-1}{|T|-1}} m_x(T, i). \tag{2.5}$$

The coefficients in front of the marginal contributions of feature i are chosen to match the coefficients in the definition of the Shapley value restricted to the neighborhood $\mathcal{N}_k(i)$. We show in Section 2.4 that this choice controls the error under certain probabilistic assumptions. In practice, the choice of the integer k is dictated by computational considerations. By the definition of k-neighborhoods, evaluating all d L-Shapley scores on a line graph requires $2^{2k}d$ model evaluations. (In particular, computing each feature takes 2^{2k+1} model evaluations, half of which overlap with those of its preceding feature.) A similar calculation shows that computing all d L-Shapley scores on a grid graph requires $2^{4k^2}d$ function evaluations.

2.3.2 Connected Shapley

We also propose a second algorithm, C-Shapley, that further reduces the complexity of approximating the Shapley value. Coming back to Example ($\star$) where we evaluate the importance of "*not,*" both the L-Shapley estimate of order larger than two and the exact Shapley value estimate would evaluate the model on the word subset "*It not heartwarming,*" which rarely appears in real data and may not make sense to a human or a model trained on real-world data. The marginal contribution of "*not*" relative to "*It not heartwarming*" may be well approximated by the marginal contribution of "not" to "*not heartwarming.*" This motivates us to proprose *C-Shapley*:

Definition 2. *Given a model $\mathbb{P}_m$, a sample x and a feature i, the* C-Shapley estimate of order k *on a graph G is given by*

$$\tilde{\phi}_x^k(i) := \sum_{U \in \mathcal{C}_k(i)} \frac{2}{(|U|+2)(|U|+1)|U|} m_x(U, i), \tag{2.6}$$

where $\mathcal{C}_k(i)$ denotes the set of all subsets of $\mathcal{N}_k(i)$ that contain node i, and are connected in G.

The coefficients in front of the marginal contributions are a result of using Myerson value to characterize a new coalitional game over the graph G, in which the influence of disconnected subsets of features are additive. The error between C-Shapley and the Shapley value can also be controlled under certain statistical assumptions. See Section 2.4 for details.

For text data, C-Shapley is equivalent to only evaluating n-grams in a neighborhood of the word to be explained. By the definition of k-neighborhoods, evaluating the C-Shapley

scores for all d features takes $\mathcal{O}(k^2 d)$ model evaluations on a line graph, as each feature takes $\mathcal{O}(k^2)$ model evaluations.

2.4 Properties

In this section, we study some basic properties of the L-Shapley and C-Shapley values. In particular, under certain probabilistic assumptions on the features, we show that they provide good approximations to the original Shapley values. We also show their relationship to another concept from cooperative game theory, namely that of Myerson values, when the model satisfies certain local additivity assumptions.

2.4.1 Approximation of Shapley value

In order to characterize the relationship between L-Shapley and the Shapley value in terms of some conditional independence assumption between features, we introduce *absolute mutual information* as a measure of dependence. Given two random variables X and Y, the absolute mutual information $I_a(X;Y)$ between X and Y is defined as

$$I_a(X;Y) = \mathbb{E}\left[\left|\log \frac{P(X,Y)}{P(X)P(Y)}\right|\right], \tag{2.7}$$

where the expectation is taken jointly over X, Y. Based on the definition of independence, we have $I_a(X;Y) = 0$ if and only if $X \perp\!\!\!\perp Y$. Recall the mutual information [71] is defined as $I(X;Y) = \mathbb{E}[\log \frac{P(X,Y)}{P(X)P(Y)}]$. The new measure is more stringent than the mutual information in the sense that $I(X;Y) \leq I_a(X;Y)$. The absolute conditional mutual information can be defined in an analogous way. Given three random variables X, Y and Z, we define the absolute conditional mutual information to be $I_a(X;Y \mid Z) = \mathbb{E}[|\log \frac{P(X,Y|Z)}{P(X|Z)P(Y|Z)}|]$, where the expectation is taken jointly over X, Y, Z. Recall that $I_a(X;Y \mid Z)$ is zero if and only if $X \perp\!\!\!\perp Y|Z$.

Theorem 1 and Theorem 2 show that L-Shapley and C-Shapley values, respectively, are related to the Shapley value whenever the model obeys a Markovian structure that is encoded by the graph. We leave their proofs to Section 2.6.

Theorem 1. *Suppose there exists a feature subset $S \subset \mathcal{N}_k(i)$ with $i \in S$, such that*

$$\sup_{U \subset S\setminus\{i\}, V \subset [d]\setminus S} I_a(X_i; X_V | X_U, Y) \leq \varepsilon; \quad \sup_{U \subset S\setminus\{i\}, V \subset [d]\setminus S} I_a(X_i; X_V | X_U) \leq \varepsilon, \tag{2.8}$$

where we identify $I_a(X_i; X_V|X_\emptyset)$ with $I_a(X_i; X_V)$ for notational convenience. Then the expected error between the L-Shapley estimate $\hat{\phi}_X^k(i)$ and the true Shapley-value-based importance score $\phi_i(\mathbb{P}_m, x)$ is bounded by 4ε:

$$\mathbb{E}_X|\hat{\phi}_X^k(i) - \phi_X(i)| \leq 4\varepsilon. \tag{2.9}$$

In particular, we have $\hat{\phi}_X^k(i) = \phi_X(i)$ almost surely if we have $X_i \perp\!\!\!\perp X_{[d]\setminus S}|X_T$ and $X_i \perp\!\!\!\perp X_{[d]\setminus S}|X_T, Y$ for any $T \subset S \setminus \{i\}$.

Theorem 2. *Suppose there exists a neighborhood $S \subset \mathcal{N}_k(i)$ of i, with $i \in S$, such that Condition 2.8 is satisfied. Moreover, for any connected subset $U \subset S$ with $i \in U$, we have*

$$\sup_{V \subset R(U)} I_a(X_i; X_V | X_{U\setminus\{i\}}, Y) \leq \varepsilon; \ \sup_{V \subset R(U)} I_a(X_i; X_V | X_{U\setminus\{i\}}) \leq \varepsilon, \tag{2.10}$$

where $R(U) := \{i \in [d] - U : \text{ for any } j \in U, (i,j) \notin E\}$. Then the expected error between the C-Shapley estimate $\tilde{\phi}_X^k(i)$ and the true Shapley-value-based importance score $\phi_i(\mathbb{P}_m, x)$ is bounded by 6ε:

$$\mathbb{E}_X |\tilde{\phi}_X^k(i) - \phi_X(i)| \leq 6\varepsilon. \tag{2.11}$$

In particular, we have $\hat{\phi}_X^d(i) = \phi_X(i)$ almost surely if we have $X_i \perp\!\!\!\perp X_{R(U)} | X_{U\setminus\{i\}}$ and $X_i \perp\!\!\!\perp X_{R(U)} | X_{U\setminus\{i\}}, Y$ for any $U \subset [d]$.

2.4.2 Relating the C-Shapley value to the Myerson value

Let us now discuss how the C-Shapley value can be related to the Myerson value, which was introduced by Myerson [68] as an approach for characterizing a coalitional game over a graph G. Given a subset of nodes S in the graph G, let $\mathcal{C}_G(S)$ denote the set of connected components of S. Thus, if S is a connected subset of G, then $\mathcal{C}_G(S)$ consists only of S; otherwise, it contains a collection of subsets whose disjoint union is equal to S.

Consider a score function $T \mapsto v(T)$ that satisfies the following decomposability condition: for any subset of nodes S, the score $v(S)$ is equal to the sum of the scores over the connected components of S:

$$v(S) = \sum_{T \in \mathcal{C}_G(S)} v(T). \tag{2.12}$$

For any such score function, we can define the associated Shapley value, and it is known as the *Myerson value* on G with respect to v. Myerson [68] showed that the Myerson value is the unique quantity that satisfies both the decomposability property, as well as the properties additivity, equal contributions and monotonicity given in Section 2.2.2.

In our setting, if we use a plug-in estimate for conditional probability, the decomposability condition (2.12) is equivalent to assuming that the influence of disconnected subsets of features are additive at sample x, and C-Shapley of order $k = d$ is exactly the Myerson value over G. In fact, if we partition each subset S into connected components, as in the definition of Myerson value, and sum up the coefficients (using Lemma 1 in Section 2.6), then the Myerson value is equivalent to (2.6).

2.4.3 Connections with related work

Let us how methods useful for approximating the Shapley value can be used to speed up the evaluation of approximate L-Shapley and C-Shapley values.

Sampling-based methods An alternative definition of the Shapley value defines the contribution of a feature i as the average of the marginal contribution of i to its preceding features over the set of all permutations of d features. Based on this definition, Štrumbelj

Data Set	Classes	Train Samples	Test Samples	Average #w	Model	Parameters	Accuracy
IMDB Review [69]	2	25,000	25,000	325.6	WordCNN	351,002	90.1%
AG's News [72]	4	120,000	7,600	43.3	CharCNN	11,337,988	90.09%
Yahoo! Answers [72]	10	1,400,000	60,000	108.4	LSTM	7,146,166	70.84%

Table 2.1: A summary of data sets and models in three experiments. "Average #w" is the average number of words per sentence. "Accuracy" is the model accuracy on test samples.

and Kononenko [16] propose a Monte Carlo approximation, based on randomly sampling permutations. While L-Shapley is deterministic in nature, it is possible to combine it with this and other sampling-based methods. For example, if one hopes to consider the interaction of features in a large neighborhood $\mathcal{N}_k(i)$ with a feature i, where exponential complexity in k becomes a barrier, sampling based on random permutation of local features may be used to alleviate the computational burden.

Regression-based methods Lundberg and Lee [15] proposed to sample feature subsets based on a weighted kernel, and carry out a weighted linear regression to estimate the Shapley value. Strong empirical results were provided using the regression-based approximation, referred to as KernelSHAP; see, in particular, Section 5.1 and Figure 3 of their paper. We can combine such a regression-based approximation with our modified Shapley values to further reduce the evaluation complexity of the C-Shapley values. In particular, for a chain graph, we evaluate the score function over all connected subsequences of length $\leq k$; similarly, on a grid graph, we evaluate it over all connected squares of size $\leq k \times k$.

2.5 Experiments

We evaluate the performance of L-Shapley and C-Shapley on real-world data sets involving text and image classification. We compare L-Shapley and C-Shapley with several competitive algorithms for instancewise feature importance scoring on black-box models, including the regression-based approximation known as KernelSHAP [15], SampleShapley [16], and the LIME method [14]. We emphasize that our focus is model-agnostic interpretation, and we omit the comparison with methods requiring additional assumptions or specific to a certain class models (e.g., [9, 10, 12, 73–75]). For all methods, we choose the objective to be the log probability of the predicted class, and use the plug-in estimate of conditional probability across all methods (see Section 2.2.1). We further evaluate the correlation of different methods with the Shapley value directly in Section 2.5.3, and analyze the sensitivity of L-Shapley and C-Shapley with the size of neighborhood in Section 2.5.4. Human evaluation is carried out in Section 2.5.5. Section 2.5.6 evaluates the variance of of SampleShapley and KernelSHAP in the setting where the sample size is linear in the number of features.

2.5.1 Text Classification

Text classification is a classical problem in natural language processing, in which text documents are assigned to predefined categories. We study the performance of L-Shapley and

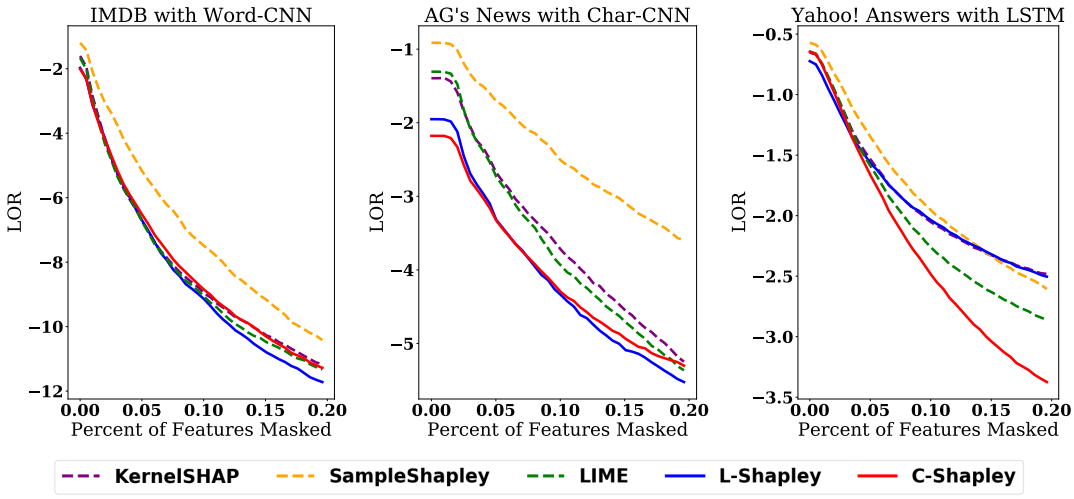


Figure 2.2: The above plots show the change in log odds ratio of the predicted class as a function of the percent of masked features, on the three text data sets. Lower log odds ratios are better.

C-Shapley on three popular neural models for text classification: word-based CNNs [76], character-based CNNs [72], and long-short term memory (LSTM) recurrent neural networks [77], with the following three data sets on different scales. See Table 2.1 for a summary.

IMDB Review with Word-CNN The Internet Movie Review Dataset (IMDB) is a dataset of movie reviews for sentiment classification [69], which contains $50,000$ binary labeled movie reviews, with a split of $25,000$ for training and $25,000$ for testing. The word-based CNN model is composed of a 50-dimensional word embedding, a 1-D convolutional layer of 250 filters and kernel size three, a max-pooling and a 250-dimensional dense layer as hidden layers. Both the convolutional and the dense layers are followed by ReLU as nonlinearity, and Dropout [78] as regularization. The model is trained with rmsprop [79]. The model achieves an accuracy of 90.1% on the test data set.

AG's news with Char-CNN The AG news corpus is composed of titles and descriptions of $196,000$ news articles from $2,000$ news sources [72]. It is segmented into four classes, each containing $30,000$ training samples and $1,900$ testing samples. The character-based CNN has the same structure as the one proposed in Zhang, Zhao, and LeCun [72], composed of six convolutional layers, three max-pooling layers, and two dense layers. The model is trained with SGD with momentum 0.9 and decreasing step size initialized at 0.01. (Details can be found in Zhang, Zhao, and LeCun [72].) The model reaches accuracy of 90.09% on the test data set.

Yahoo! Answers with LSTM The corpus of Yahoo! Answers Topic Classification Dataset is divided into ten categories, each class containing $140,000$ training samples and $5,000$ test-

Method	Explanation
Shapley	It is not heartwarming or entertaining . It just sucks .
C-Shapley	It is not heartwarming or entertaining . It just sucks .
L-Shapley	It is not heartwarming or entertaining . It just sucks .
KernelSHAP	It is not heartwarming or entertaining . It just sucks .
SampleShapley	It is not heartwarming or entertaining . It just sucks .

Table 2.2: Each word is highlighted with the RGB color as a linear function of its importance score. The background colors of words with positive and negative scores are linearly interpolated between blue and white, red and white respectively.

ing samples. Each input text includes the question title, content and best answer. The network consists of a 300-dimensional randomly-initialized word embedding, a bidirectional LSTM, each LSTM unit of dimension 256, and a dropout layer as hidden layers. The model is trained with rmsprop [79]. The model reaches accuracy of 70.84% on the test data set, close to the state-of-the-art accuracy of 71.2% obtained by character-based CNN [72].

We choose zero paddings as the reference point for all methods, and make $4 \times d$ model evaluations, where d is the number of words for each input. Given the average length of each input (see Table 2.1), this choice controls the number of model evaluations under $1,000$, taking less than one second in TensorFlow on a Tesla K80 GPU for all the three models. For L-Shapley, we are able to consider the interaction of each word i with the two neighboring words in $\mathcal{N}_1(i)$ given the budget. For C-Shapley, the budget allows the regression-based version to evaluate all n-grams with $n \leq 4$.

The change in log-odds scores before and after masking the top features ranked by importance scores is used as a metric for evaluating performance, where masked words are replaced by zero paddings. This metric has been used in previous literature in model interpretation [10, 15]. We study how the average log-odds score of the predicted class decreases as the percentage of masked features over the total number of features increases on $1,000$ samples from the test set. Results are plotted in Figure 2.2.

On IMDB with Word-CNN, the simplest model among the three, L-Shapley, achieves the best performance while LIME, KernelSHAP and C-Shapley achieve slightly worse performance. On AG's news with Char-CNN, L-Shapley and C-Shapley both outperform other algorithms. On Yahoo! Answers with LSTM, C-Shapley outperforms the rest of the algorithms by a large margin, followed by LIME. L-Shapley with order 1, SampleShapley, and KernelSHAP do not perform well for LSTM model, probably because some of the signals captured by LSTM are relatively long n-grams.

We also visualize the importance scores produced by different Shapley-based methods on Example ($\star$), which is part of a negative movie review taken from IMDB. The result is shown in Table 2.2.

2.5.2 Image Classification

We carry out experiments in image classification on the MNIST and CIFAR10 data sets:

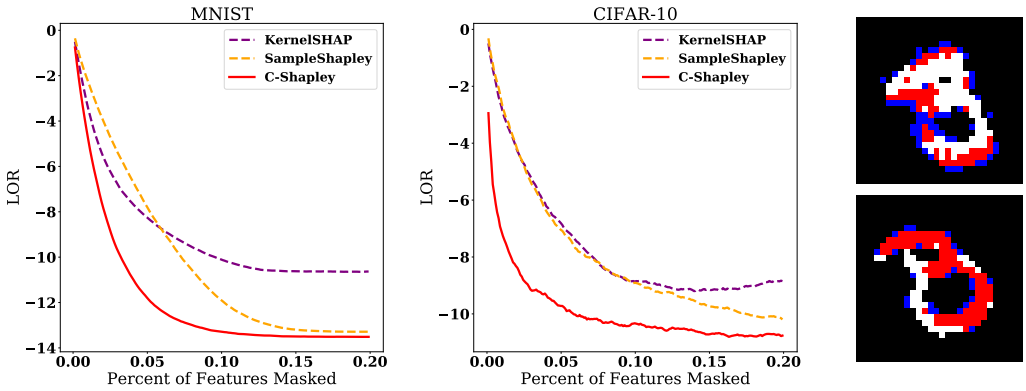


Figure 2.3: Left and Middle: change in log-odds ratio vs. the percent of pixels masked on MNIST and CIFAR10. Right: top pixels ranked by C-Shapley for a "3" and an "8" misclassified into "8" and "3" respectively. The masked pixels are colored with red if activated (white) and blue otherwise.

MNIST The MNIST data set contains 28×28 images of handwritten digits with ten categories $0 - 9$ [80]. A subset of MNIST data set composed of digits 3 and 8 is used for better visualization, with $12,000$ images for training and $1,000$ images for testing. A simple CNN model is trained on the data set, which achieves 99.7% accuracy on the test data set. It is composed of two convolutional layers of kernel size 5×5 and a dense linear layer at last. The two convolutional layers contain 8 and 16 filters respectively, and both are followed by a max-pooling layer of pool size two.

CIFAR10 The CIFAR10 data set [81] contains 32×32 images in ten classes. A subset of CIFAR10 data set composed of deers and horses is used for better visualization, with $10,000$ images for training and $2,000$ images for testing. A convolutional neural network modified from AlexNet [82] is trained on the subset. It is composed of six convolutional layers of kernel size 3×3 and two dense linear layers of dimension 512 and 256 at last. The six convolutional layers contain 48,48,96,96,192,192 filters respectively, and every two convolutional layers are followed by a max-pooling layer of pool size two and a dropout layer. The CNN model is trained with the Adam optimizer [83] and achieves 96.1% accuracy on the test data set.

We take each pixel as a single feature for both MNIST and CIFAR10. We choose the average pixel strength and the black pixel strength respectively as the reference point for all methods, and make $4 \times d$ model evaluations, where d is the number of pixels for each input image, which keeps the number of model evaluations under $4,000$.

LIME and L-Shapley are not used for comparison because LIME takes "superpixels" instead of raw pixels segmented by segmentation algorithms as single features, and L-Shapley requires nearly sixteen thousand model evaluations when applied to raw pixels.[1] For C-

[1]L-Shapley becomes practical if we take small patches of images instead of pixels as single features.

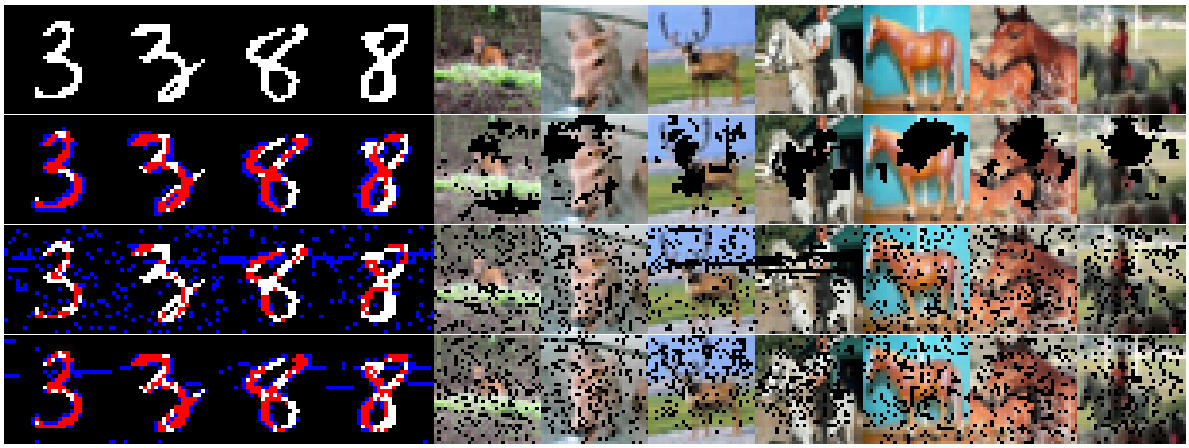


Figure 2.4: Some examples of explanations obtained for the MNIST and CIFAR10 data sets. The first row corresponds to the original images, with the rows below showing images masked based on scores produced by C-Shapley, KernelSHAP and SampleShapley respectively. For MNIST, the masked pixels are colored with red if activated (white) and blue otherwise.

Shapley, the budget allows the regression-based version to evaluate all $n \times n$ image patches with $n \leq 4$.

Figure 2.3 shows the decrease in log-odds scores before and after masking the top pixels ranked by importance scores as the percentage of masked pixels over the total number of pixels increases on $1,000$ test samples on MNIST and CIFAR10 data sets. C-Shapley consistently outperforms other methods on both data sets. Figure 2.3 also shows two misclassified digits by the CNN model. Interestingly, the top pixels chosen by C-Shapley visualize the "reasoning" of the model: the important pixels to the model are exactly those which could form a digit from the opposite class.

Figure 2.4 provides additional visualization of the results. By masking the top pixels ranked by various methods, we find that the pixels picked by C-Shapley concentrate around and inside the digits in MNIST. For SampleShapley and KernelSHAP, unactivated pixels in MNIST are attributed nonzero scores when evaluated jointly with activated pixels. While one could use post-processing by not choosing unactivated pixels, we choose to visualize the original outputs from all algorithms for fairness of comparison. The C-Shapley also yields the most interpretable results in CIFAR10. In particular, C-Shapley tends to mask the parts of head and body that distinguish deers and horses, and the human riding the horse.

2.5.3 Rank correlation with the Shapley value

We address the problem of how the rank of features produced by various approximation algorithms correlates with the rank produced by the true Shapley value. We sample a subset of test data from Yahoo! Answers with 9-12 words, so that the underlying Shapley scores can be accurately computed. We employ two common metrics, Kendall's τ and Spearman's ρ [84], to measure the similarity (correlation) between two ranks.

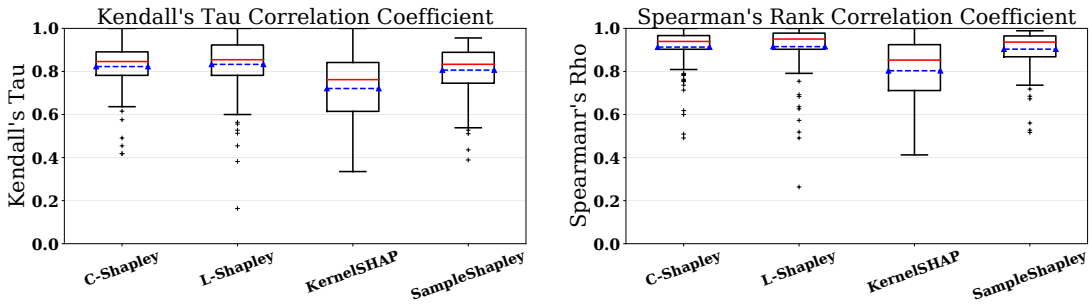


Figure 2.5: The box plots of Kendall's τ and Spearman's ρ between various algorithms (with the same computational complexity) and the Shapley value. The red line and the dotted blue line on each box are the median and the mean respectively. (Higher is better.)

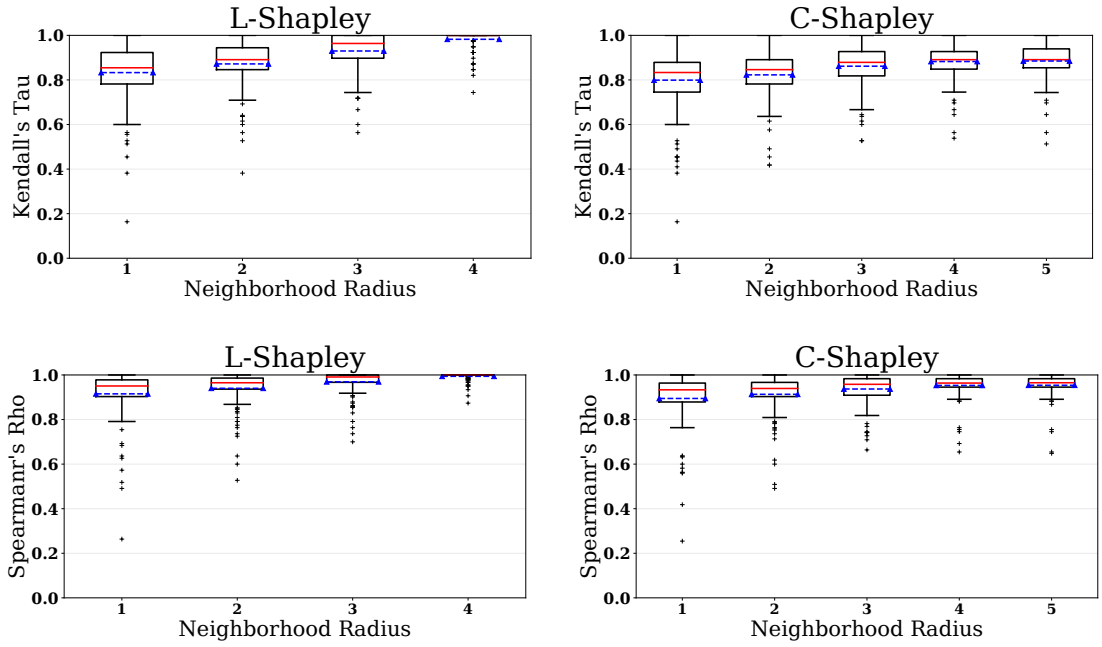


Figure 2.6: Kendall's τ and Spearman's ρ between L-Shapley and the Shapley value, C-Shapley and the Shapley value vs. the neighborhood radius. The red line and the dotted blue line on each box are the median and the mean respectively. (Higher is better.)

The result is shown in Figure 2.5. The rank correlation between L-Shapley and the Shapley value is the highest, followed by C-Shapley, consistent across both of the two metrics. Given the limited length of each instance, the search space for sampling based algorithms is relatively small. Thus there is only a slight performance gain of our algorithms over KernelSHAP and SampleShapley.

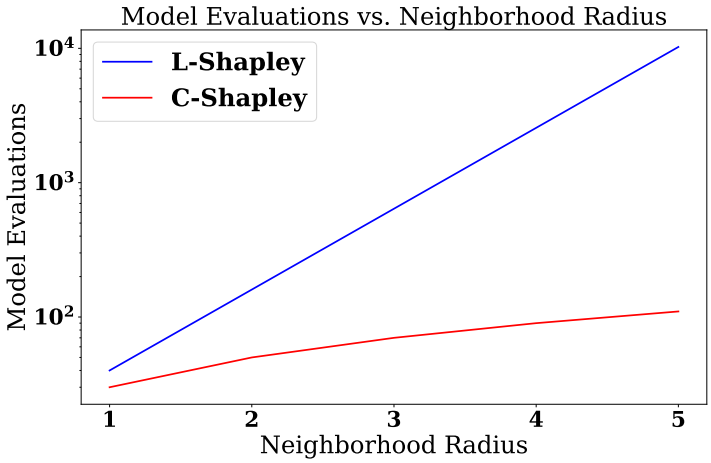


Figure 2.7: Number of model evaluations vs. neighborhood radius (in log scale) on an instance with ten features.

2.5.4 Sensitivity of L-Shapley and C-Shapley

We study the sensitivity of L-Shapley and C-Shapley to the radius of neighborhood on the subsampled data from Yahoo! Answers in Section 2.5.3. We employ Kendall's τ and Spearman's ρ [84] to measure the rank correlation between scores from the proposed methods and the Shapley value[2].

Figure 2.6 shows how Kendall's τ and Spearman's ρ between the proposed algorithms and the Shapley value vary with the radius of neighborhood. We observe the bias of the proposed algorithms decreases gradually with increasing neighborhood radius. Figure 2.7 plots the number of model evaluations as a function of neighborhood radius for both algorithms, on an example instance with ten features[3]. The complexity of L-Shapley grows exponentially with the neighborhood radius while the complexity of C-Shapley grows linearly.

2.5.5 Evaluation with human subjects

We use human annotators on Amazon Mechanical Turk (AMT) to compare L-Shapley, C-Shapley and KernelSHAP on IMDB movie reviews. We aim to address two problems: (i) Are humans able to make a decision with top words alone? (ii) Are humans unable to make a decision with top words masked?

We randomly sample 200 movie reviews that are correctly classified by the model. Each review is assigned to five annotators. We ask humans on AMT to classify the sentiment of

[2]The nonparametric rank correlation is used instead of Pearson correlation coefficient because of the violation of identical assumption across different instances

[3]Model evaluations can be easily paralleled on a modern GPU. Hence we plot the number of model evaluations instead of real running time, which depends on the availability of computational resource.

Algorithm	Modification	Consistency	Standard Deviation	Abs. Score	Words Masked
Raw	None	0.880	0.960	0.811	N/A
L-Shapley	Selected	0.970	0.891	1.118	N/A
	Masked	**0.615**	**1.077**	**0.474**	**14.36%**
C-Shapley	Selected	**0.990**	**0.500**	**1.441**	N/A
	Masked	0.830	0.778	0.743	14.75%
KernelSHAP	Selected	0.960	0.627	1.036	N/A
	Masked	0.660	0.818	0.492	31.60%

Table 2.3: Results of human evaluation. "Selected" and "Masked" indicate selected words and masked reviews respectively. Results are averaged over 200 samples. (The best numbers are highlighted.)

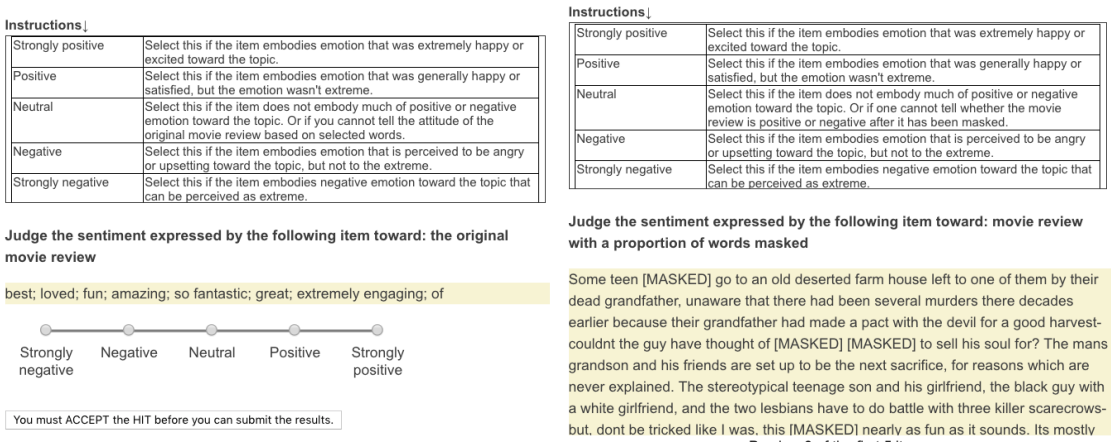


Figure 2.8: Interfaces of Amazon Mechanical Turk where annotators are asked to infer the sentiment of the original reviews based on selected words and masked reviews.

texts into five categories: strongly positive (+2), positive (+1), neutral (0), negative (-1), strongly negative (-2). See Figure 2.8 for an example interface.

Texts have three types: (i) raw reviews; (ii) top ten words of each review ranked by L-Shapley, C-Shapley and KernelSHAP, where adjacent words, like "not satisfying or entertaining", keep their adjacency if selected simultaneously; and (iii) reviews with top words being masked. In the third type of texts, words are replaced with "[MASKED]" one after another, in the order produced by the respective algorithms, until the probability score of the correct class produced by the model is lower than 10%. We adopt the above design to make sure the majority of key words sensitive to the model have been masked. On average, around 14% of words in each review are masked for L-Shapley and C-Shapley, while 31.6% for KernelSHAP.

We measure the consistency (0 or 1) between the true labels and labels from human annotators, where a human label is positive if the average score over five annotators are larger than zero. Reviews with an average score of zero are neither put in the positive nor in the negative class. We also employ the standard deviation of scores on each review as a measure of disagreement between humans. Finally, the absolute value of the average scores

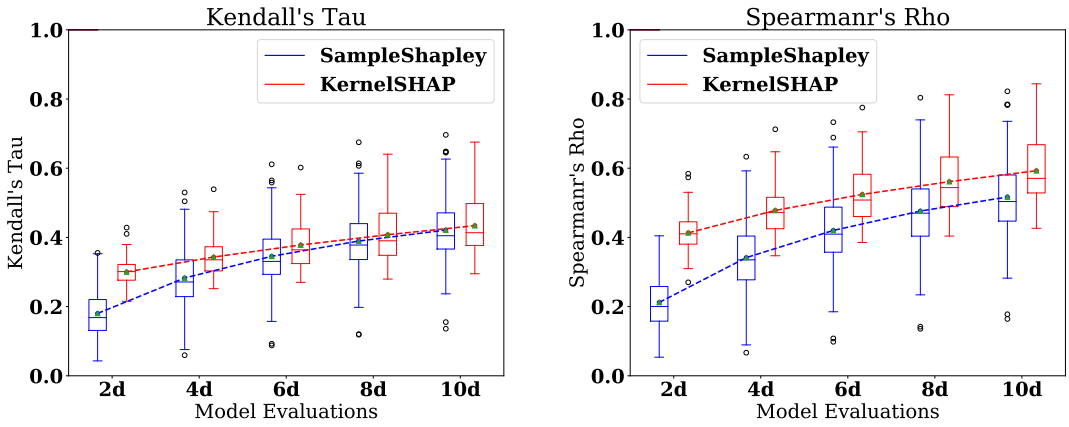


Figure 2.9: The box plots of average pairwise Kendall's τ and Spearman's ρ vs. the number of model evaluations over 30 replicates. (Higher is better.) Dash lines (with green dots) plots the mean, and the solid lines in box plots are medians. The number of features d varies across different instances.

from five annotators is used as a measure of confidence of decision.

The results of the two experiments are shown in Table 2.3. We observe humans become more consistent with the truth and more confident, and also have less disagreement with each other when they are presented with top words. Among the three algorithms, C-Shapley yields the highest performance in terms of consistency, agreement, and confidence. On the other hand, when top words are masked, humans are easier to make mistakes and are less certain about their judgement. L-Shapley harms the human judgement the most among the three algorithms, although KernelSHAP masks two times more words. The above experiments show that (i) Key words to the model convey an attitude toward a movie;, and (ii) Our algorithms find the key words more accurately.

2.5.6 Variability of sampling based algorithms

We empirically evaluate the variance of SampleShapley and KernelSHAP in the setting where the sample size is linear in the number of features. The experiment is carried out on the test data set of IMDB movie reviews. For each method, we run 30 replications on every instance, which generates 30 scores. Given the varied scalability of underlying Shapley values, we again seek a nonparametric approach to measure the variability of sampling based algorithms. On each instance, we compute the pairwise Kendall's τ[4] and Spearman's ρ among the 30 runs of a single method. Then we use the average of $\binom{30}{2}$ τs and ρs respectively as measures of statistical dispersion[5].

[4] τ-b version is used which can account for ties [84].

[5] The former can be linked to the variance when one models permutation with the Mallows Model. A discussion can be found in [85]

Figure 2.9 shows the variability of SampleShapley and KernelSHAP as a function of the number of model evaluations. The ticks $2d, 4d, \ldots$ on the x-axis are the number of model evaluations, where d is the number of features which varies across different instances. As a concrete example, on the rightmost box plot, KernelSHAP carries out $10d = 2,000$ model evaluations on an instance with $d = 200$ features.

2.6 Proof of Main Theorems

In this section, we collect the proofs of Theorem 1 and Theorem 2.

2.6.1 Proof of Theorem 1

We state an elementary combinatorial equality required for the proof of the main theorem:

Lemma 1 (A combinatorial equality)**.** *For any positive integer n, and any pair of non-negative integers with $s \geq t$, we have*

$$\sum_{j=0}^{n} \frac{1}{\binom{n+s}{j+t}} \binom{n}{j} = \frac{s+1+n}{(s+1)\binom{s}{t}} \tag{2.13}$$

Proof. By the binomial theorem for negative integer exponents, we have

$$\frac{1}{(1-x)^{t+1}} = \sum_{j=0}^{\infty} \binom{j+t}{j} x^j.$$

The identity can be found by examination of the coefficient of x^n in the expansion of

$$\frac{1}{(1-x)^{t+1}} \cdot \frac{1}{(1-x)^{s-t+1}} = \frac{1}{(1-x)^{s+1+1}}. \tag{2.14}$$

In fact, equating the coefficients of x^n in the left and the right hand sides, we get

$$\sum_{j=0}^{n} \binom{j+t}{j} \binom{(n-j)+(s-t)}{n-j} = \binom{n+s+1}{n} = \frac{n+s+1}{s+1} \binom{n+s}{n}. \tag{2.15}$$

Moving $\binom{n+s}{n}$ to the right hand side and expanding the binomial coefficients, we have

$$\sum_{j=0}^{n} \frac{(j+t)!}{j!t!} \cdot \frac{(n-j+s-t)!}{(n-j)!(s-t)!} \cdot \frac{n!s!}{(n+s)!} = \frac{n+s+1}{s+1}, \tag{2.16}$$

which implies

$$\begin{aligned} \sum_{j=0}^{n} \binom{n}{j} \binom{s}{t} \Big/ \binom{n+s}{j+t} &= \sum_{j=0}^{n} \frac{n!}{(n-j)!j!} \cdot \frac{s!}{t!(s-t)!} \cdot \frac{((n+s)-(j+t))!(j+t)!}{(n+s)!} \\ &= \sum_{j=0}^{n} \frac{(j+t)!}{j!t!} \cdot \frac{(n-j+s-t)!}{(n-j)!(s-t)!} \cdot \frac{n!s!}{(n+s)!} = \frac{n+s+1}{s+1}. \end{aligned}$$

□

Taking this lemma, we now prove the theorem. We split our analysis into two cases, namely $S = \mathcal{N}_k(i)$ versus $S \subset \mathcal{N}_k(i)$. For notational convenience, we extend the definition of L-Shapley estimate for feature i to an arbitrary feature subset S containing i. In particular, we define

$$\hat{\phi}_x^S(i) := \frac{1}{|S|} \sum_{\substack{T \ni i \\ T \subseteq S}} \frac{1}{\binom{|S|-1}{|T|-1}} m_x(T, i). \tag{2.17}$$

Case 1: First, suppose that $S = \mathcal{N}_k(i)$. For any subset $A \subset [d]$, we introduce the shorthand notation $U_S(A) := A \cap S$ and $V_S(A) := A \cap S^c$, and note that $A = U_S(A) \cup V_S(A)$. Recalling the definition of the Shapley value, let us partition all the subsets A based on $U_S(A)$, in particular writing

$$\phi_X(i) = \frac{1}{d} \sum_{\substack{A \subseteq [d] \\ A \ni i}} \frac{1}{\binom{d-1}{|A|-1}} m_X(A, i) \;=\; \frac{1}{d} \sum_{\substack{U \subseteq S \\ U \ni i}} \sum_{\substack{A \subseteq [d] \\ U_S(A) = U}} \frac{1}{\binom{d-1}{|A|-1}} m_X(A, i).$$

Based on this partitioning, the expected error between $\hat{\phi}_X^S(i)$ and $\phi_X(i)$ can be written as

$$\mathbb{E}\left|\hat{\phi}_X^S(i) - \phi_X(i)\right| = \mathbb{E}\left|\frac{1}{|S|} \sum_{\substack{U \subseteq S \\ U \ni i}} \frac{1}{\binom{|S|-1}{|U|-1}} m_X(U, i) - \frac{1}{d} \sum_{\substack{U \subseteq S \\ U \ni i}} \sum_{\substack{A \subseteq [d] \\ U_S(A) = U}} \frac{1}{\binom{d-1}{|A|-1}} m_X(A, i)\right|. \tag{2.18}$$

Partitioning the set $\{A : U_S(A) = U\}$ by the size of $V_S(A) = A \cap S^c$, we observe that

$$\begin{aligned}
\sum_{\substack{A \subseteq [d] \\ U_S(A) = U}} \frac{1}{\binom{d-1}{|A|-1}} &= \sum_{i=0}^{d-|S|} \frac{1}{\binom{d-1}{i+|U|-1}} \binom{d-|S|}{i} \\
&= \frac{(|S|-1) + 1 + (d-|S|)}{((|S|-1)+1)\binom{|S|-1}{|U|-1}} \\
&= \frac{d}{|S|} \frac{1}{\binom{|S|-1}{|U|-1}},
\end{aligned}$$

where we have applied Lemma 1 with $n = d - |S|$, $s = |S| - 1$, and $t = |U| - 1$. Substituting this equivalence into equation (2.18), we find that the expected error can be upper bounded by

$$\mathbb{E}|\hat{\phi}_X^S(i) - \phi_X(i)| \leq \frac{1}{d} \sum_{\substack{U \subseteq S \\ U \ni i}} \sum_{\substack{A \subseteq [d] \\ U_S(A) = U}} \frac{1}{\binom{d-1}{|A|-1}} \mathbb{E}\left|m_X(U, i) - m_X(A, i)\right|, \tag{2.19}$$

where we recall that $A = U_S(A) \cup V_S(A)$.

Now omitting the dependence of $U_S(A), V_S(A)$ on A for notational simplicity, we now

write the difference as

$$\begin{aligned} m_X(A,i) - m_X(U,i) &= \mathbb{E}_m\left[\log\frac{\mathbb{P}_m(Y|X_{U\cup V})}{\mathbb{P}_m(Y|X_{U\cup V\setminus\{i\}})} - \log\frac{\mathbb{P}_m(Y|X_U)}{\mathbb{P}_m(Y|X_{U\setminus\{i\}})} \mid X\right] \\ &= \mathbb{E}_m\left[\log\frac{\mathbb{P}(Y,X_{U\setminus\{i\}})\mathbb{P}(X_U)P(X_{U\cup V\setminus\{i\}})P(X_{U\cup V},Y)}{\mathbb{P}(Y,X_U)\mathbb{P}(X_{U\setminus\{i\}})P(X_{U\cup V})P(X_{U\cup V\setminus\{i\}},Y)} \mid X\right] \\ &= \mathbb{E}_m\Bigg[\log\frac{\mathbb{P}(X_i,X_V \mid X_{U\setminus\{i\}},Y)}{\mathbb{P}(X_i \mid X_{U\setminus\{i\}},Y)\mathbb{P}(X_V \mid X_{U\setminus\{i\}},Y)} \\ &\qquad - \log\frac{\mathbb{P}(X_i,X_V|X_{U\setminus\{i\}})}{\mathbb{P}(X_i|X_{U\setminus\{i\}})\mathbb{P}(X_V \mid X_{U\setminus\{i\}})} \mid X\Bigg]. \end{aligned}$$

Substituting this equivalence into our earlier bound (2.19) and taking an expectation over X on both sides, we find that the expected error is upper bounded as

$$\begin{aligned} \mathbb{E}|\hat{\phi}_X^S(i) - \phi_X(i)| \leq \frac{1}{d}\sum_{\substack{U\subseteq S\\ U\ni i}}\sum_{\substack{A\subseteq[d]\\ U_S(A)=U}} \frac{1}{\binom{d-1}{|A|-1}}\Bigg\{&\mathbb{E}\left|\log\frac{\mathbb{P}(X_i,X_{V_S(A)}|X_{U\setminus\{i\}},Y)}{\mathbb{P}(X_i|X_{U\setminus\{i\}},Y)\mathbb{P}(X_{V_S(A)}|X_{U\setminus\{i\}},Y)}\right| \\ &+ \mathbb{E}\left|\log\frac{\mathbb{P}(X_i,X_{V_S(A)}|X_{U\setminus\{i\}})}{\mathbb{P}(X_i \mid X_{U\setminus\{i\}})\mathbb{P}(X_{V_S(A)}|X_{U\setminus\{i\}})}\right|\Bigg\}. \end{aligned}$$

Recalling the definition of the absolute mutual information, we see that

$$\begin{aligned} \mathbb{E}|\hat{\phi}_X^S(i) - \phi_X(i)| &\leq \frac{1}{d}\sum_{\substack{U\subseteq S\\ U\ni i}}\sum_{\substack{A\subseteq[d]\\ U_S(A)=U}} \frac{1}{\binom{d-1}{|A|-1}}\Big\{I_a(X_i;X_{V_S(A)} \mid X_{U\setminus\{i\}},Y) \\ &\qquad + I_a(X_i;X_{V_S(A)} \mid X_{U\setminus\{i\}})\Big\} \\ &\leq 2\varepsilon, \end{aligned}$$

which completes the proof of the claimed bound.

Finally, in the special case that $X_i \perp\!\!\!\perp X_{[d]\setminus S}|X_T$ and $X_i \perp\!\!\!\perp X_{[d]\setminus S}|X_T, Y$ for any $T \subset S$, then this inequality holds with $\varepsilon = 0$, which implies $\mathbb{E}|\hat{\phi}_X^S(i) - \phi_X(i)| = 0$. Therefore, we have $\hat{\phi}_X^S(i) = \phi_X(i)$ almost surely, as claimed.

Case 2: We now consider the general case in which $S \subset \mathcal{N}_k(i)$. Using the previous arguments, we can show

$$\mathbb{E}|\hat{\phi}_X^S(i) - \phi_X^k(i)| \leq 2\varepsilon, \quad \text{and} \quad \mathbb{E}|\hat{\phi}_X^S(i) - \phi_X(i)| \leq 2\varepsilon.$$

Appylying the triangle inequality yields $\mathbb{E}|\hat{\phi}_X^k(i) - \phi_X(i)| \leq 4\varepsilon$, which establishes the claim.

2.6.2 Proof of Theorem 2

As in the previous proof, we divide our analysis into two cases.

Proof. **Case 1:** First, suppose that $S = \mathcal{N}_k(i) = [d]$. For any subset $A \subset S$ with $i \in A$, we can partition A into two components $U_S(A)$ and $V_S(A)$, such that $i \in U_S(A)$ and $U_S(A)$ is a connected subsequence. $V_S(A)$ is disconnected from $U_S(A)$. We also define

$$\mathcal{C} = \{U \mid i \in U, U \subset [d], U \text{ is a connected subsequence.}\} \tag{2.20}$$

We partition all the subsets $A \subset S$ based on $U_S(A)$ in the definition of the Shapley value:

$$\begin{aligned}\phi_X(i) &= \frac{1}{d}\sum_{\substack{A\subseteq S\\ A\ni i}} \frac{1}{\binom{d-1}{|A|-1}} m_X(A,i) \\ &= \frac{1}{d}\sum_{U\in\mathcal{C}}\sum_{A:U_S(A)=U} \frac{1}{\binom{d-1}{|A|-1}} m_X(A,i).\end{aligned}$$

The expected error between $\tilde{\phi}_X^{[d]}(i)$ and $\phi_X(i)$ is

$$\mathbb{E}|\tilde{\phi}_X^{[d]}(i) - \phi_X(i)| = \mathbb{E}\left|\frac{1}{d}\sum_{U\in\mathcal{C}} \frac{2d}{(|U|+2)(|U|+1)|U|} m_X(U,i) - \frac{1}{d}\sum_{U\in\mathcal{C}}\sum_{A:U_S(A)=U} \frac{1}{\binom{d-1}{|A|-1}} m_X(A,i)\right|. \tag{2.21}$$

Partitioning $\{A : U_S(A) = U\}$ by the size of $V_S(A)$, we observe that

$$\begin{aligned}\sum_{A:U_S(A)=U} \frac{1}{\binom{d-1}{|A|-1}} &= \sum_{i=0}^{d-|U|-2} \frac{1}{\binom{d-1}{i+|U|-1}} \binom{d-|U|-2}{i} \\ &= \frac{(|U|+1)+1+(d-|U|-2)}{((|U|+1)+1)\binom{|U|+1}{|U|-1}} \\ &= \frac{2d}{(|U|+2)(|U|+1)|U|},\end{aligned} \tag{2.22}$$

where we apply Lemma 1 with $n = d - |U| - 2$, $s = |U| + 1$ and $t = |U| - 1$. From equation (2.21), the expected error can be upper bounded by

$$\mathbb{E}\left|\tilde{\phi}_X^{[d]}(i) - \phi_X(i)\right| \le \frac{1}{d}\sum_{U\in\mathcal{C}}\sum_{A:U_S(A)=U} \frac{1}{\binom{d-1}{|A|-1}} \mathbb{E}\left|m_X(U,i) - m_X(A,i)\right|,$$

where $A = U_S(A) \cup V_S(A)$. We omit the dependence of $U_S(A)$ and $V_S(A)$ on the pair (A, S) for notational simplicity, and observe that the difference between $m_x(A,i)$ and $m_x(U,i)$ is

$$\begin{aligned}m_X(A,i) - m_X(U,i) &= \mathbb{E}_m\left[\log\frac{\mathbb{P}_m(Y|X_{U\cup V})}{\mathbb{P}_m(Y|X_{U\cup V\setminus\{i\}})} - \log\frac{\mathbb{P}_m(Y|X_U)}{\mathbb{P}_m(Y|X_{U\setminus\{i\}})} \mid X\right] \\ &= \mathbb{E}_m\left[\log\frac{\mathbb{P}(Y,X_{U\setminus\{i\}})\mathbb{P}(X_U)P(X_{U\cup V\setminus\{i\}})P(X_{U\cup V},Y)}{\mathbb{P}(Y,X_U)\mathbb{P}(X_{U\setminus\{i\}})P(X_{U\cup V})P(X_{U\cup V\setminus\{i\}},Y)} \mid X\right] \\ &= \mathbb{E}_m\Bigg[\log\frac{\mathbb{P}(X_i,X_V|X_{U\setminus\{i\}},Y)}{\mathbb{P}(X_i|X_{U\setminus\{i\}},Y)\mathbb{P}(X_V|X_{U\setminus\{i\}},Y)} \\ &\qquad - \log\frac{\mathbb{P}(X_i,X_V|X_{U\setminus\{i\}})}{\mathbb{P}(X_i|X_{U\setminus\{i\}})\mathbb{P}(X_V|X_{U\setminus\{i\}})} \mid X\Bigg].\end{aligned}$$

Taking an expectation over X at both sides, we can upper bound the expected error by

$$\begin{aligned}\mathbb{E}|\tilde{\phi}_X^{[d]}(i) - \phi_X(i)| \leq \frac{1}{d}\sum_{U\in\mathcal{C}}\sum_{A:U_S(A)=U}\frac{1}{\binom{d-1}{|A|-1}}(\mathbb{E}\left|\log\frac{\mathbb{P}(X_i, X_{V_S(A)}|X_{U\setminus\{i\}}, Y)}{\mathbb{P}(X_i|X_{U\setminus\{i\}}, Y)\mathbb{P}(X_{V_S(A)}|X_{U\setminus\{i\}}, Y)}\right| \\ + \mathbb{E}\left|\log\frac{\mathbb{P}(X_i, X_{V_S(A)}|X_{U\setminus\{i\}})}{\mathbb{P}(X_i|X_{U\setminus\{i\}})\mathbb{P}(X_{V_S(A)}|X_{U\setminus\{i\}})}\right|) \\ = \frac{1}{d}\sum_{U\in\mathcal{C}}\sum_{A:U_S(A)=U}\frac{1}{\binom{d-1}{|A|-1}}(I_a(X_i; X_{V_S(A)}|X_{U\setminus\{i\}}, Y) + I_a(X_i; X_{V_S(A)}|X_{U\setminus\{i\}})) \\ \leq 2\varepsilon.\end{aligned}$$

Let $R(U) := [d] - U \cup \{\max(u-1, 1), \min(u+l+1, d)\}$. If we have $X_i \perp\!\!\!\perp X_{R(U)}|X_{U\setminus\{i\}}$ and $X_i \perp\!\!\!\perp X_{R(U)}|X_{U\setminus\{i\}}, Y$ for any $U \subset [d]$, then $\varepsilon = 0$, which implies $\mathbb{E}|\tilde{\phi}_X^{[d]}(i) - \phi_X(i)| = 0$. Therefore, we have $\tilde{\phi}_X^{[d]}(i) = \phi_X(i)$ almost surely.

Case 2: We now turn to the general case $S \subset \mathcal{N}_k(i) \subset [d]$. Similar as above, we can show

$$\mathbb{E}|\tilde{\phi}_X^k(i) - \hat{\phi}_X^k(i)| \leq 2\varepsilon.$$

Based on Theorem 1, we have

$$\mathbb{E}|\hat{\phi}_X^k(i) - \phi_X(i)| \leq 4\varepsilon.$$

Applying the triangle yields $\mathbb{E}|\tilde{\phi}_X^k(i) - \phi_X(i)| \leq 6\varepsilon$, which establishes the claim. □

2.7 Discussion

We have proposed two new algorithms—L-Shapley and C-Shapley—for instancewise feature importance scoring, making use of a graphical representation of the data. We have demonstrated the superior performance of these algorithms compared to other methods on black-box models for instancewise feature importance scoring in both text and image classification with both quantitative metrics and human evaluation.

Chapter 3

Learning to Explain: An Information-Theoretic Perspective on Model Interpretation

We focus on *instancewise feature importance scoring* as a methodology for model interpretation. The method proposed in this chapter further improves the efficiency over L-Shapley and C-Shapley as no test-time model query is required.

Our approach is based on learning a feature selector to exact a subset of features that are most informative for each given example. The selector is trained to maximize the mutual information between selected features and the response variable, where the conditional distribution of the response variable given the input is the model to be explained. We develop an efficient variational approximation to the mutual information, and show that the resulting method compares favorably to other model explanation methods on a variety of synthetic and real data sets using both quantitative metrics and human evaluation.

3.1 Introduction

In this chapter, we focus on *instancewise feature importance scoring* as a specific approach for model interpretation. Given a machine learning model, instancewise feature importance scoring asks for the importance score of each feature on the prediction of a given instance, and the relative importance of each feature is allowed to vary across instances. Thus, the importance scores can act as an explanation for the specific instance, indicating which features are the key for the model to make its prediction on that instance. A related concept in machine learning is feature selection, which selects a subset of features that are useful to build a good predictor for a specified response variable [86]. While feature selection produces a global importance of features with respect to the entire labeled data set, instancewise feature importance scoring measures feature importance locally for each instance labeled by the model.

	Training	Efficiency	Additive	Model-agnostic
Parzen [67]	Yes	High	Yes	Yes
Salient map [8]	No	High	Yes	No
LRP [9]	No	High	Yes	No
LIME [14]	No	Low	Yes	Yes
Kernel SHAP [15]	No	Low	Yes	Yes
DeepLIFT [10]	No	High	Yes	No
IG [12]	No	Medium	Yes	No
L2X	Yes	High	No	Yes

Table 3.1: Summary of the properties of different methods. "Training" indicates whether a method requires training on an unlabeled data set. "Efficiency" qualitatively evaluates the computational time during single interpretation. "Additive" indicates whether a method is locally additive. "Model-agnostic" indicates whether a method is generic to black-box models.

Existing work on interpreting models approach the problem from two directions. The first line of work computes the gradient of the output of the correct class with respect to the input vector for the given model, and uses it as a saliency map for masking the input [8, 87]. The gradient is computed using a Parzen window approximation of the original classifier if the original one is not available [67]. Another line of research approximates the model to be interpreted via a locally additive model in order to explain the difference between the model output and some "reference" output in terms of the difference between the input and some "reference" input [9, 10, 12, 14, 15, 88]. Ribeiro, Singh, and Guestrin [14] proposed the LIME, methods which randomly draws instances from a density centered at the sample to be explained, and fits a sparse linear model to predict the model outputs for these instances. Shrikumar, Greenside, and Kundaje [10] presented DeepLIFT, a method designed specifically for neural networks, which decomposes the output of a neural network on a specific input by backpropagating the contribution back to every feature of the input. Lundberg and Lee [15] used Shapley values to quantify the importance of features of a given input, and proposed a sampling based method "kernel SHAP" for approximating Shapley values. [12] proposed Integrated Gradients (IG), which constructs the additive model by cumulating the gradients along the line between the input and the reference point. Essentially, the two directions both approximate the model locally via an additive model, with different definitions of locality. While the first one considers infinitesimal regions on the decision surface and takes the first-order term in the Taylor expansion as the additive model, the second one considers the finite difference between an input vector and a reference vector.

In this chapter, our approach to instancewise feature importance scoring is via mutual information, a conceptually different perspective from existing approaches. We define an "explainer," or instancewise feature selector, as a model which returns a distribution over the subset of features given the input vector.

For a given instance, an ideal explainer should assign the highest probability to the subset of features that are most informative for the associated model response. This motivates us to maximize the mutual information between the selected subset of features and the

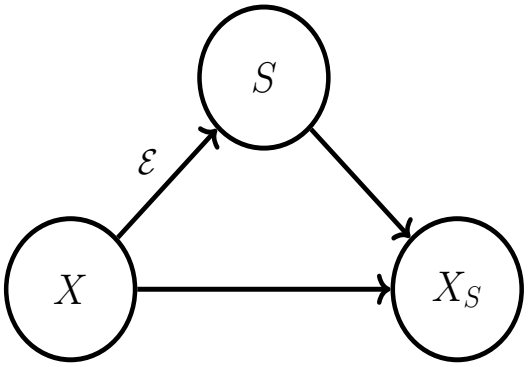


Figure 3.1: The graphical model of obtaining X_S from X.

response variable with respect to the instancewise feature selector. Direct estimation of mutual information and discrete feature subset sampling are intractable; accordingly, we derive a tractable method by first applying a variational lower bound for mutual information, and then developing a continuous reparametrization of the sampling distribution.

At a high level, the primary differences between our approach and past work are the following. First, our framework *globally* learns a *local* explainer, and therefore takes the distribution of inputs into consideration. Second, our framework removes the constraint of local feature additivity on an explainer. These distinctions enable our framework to yield a more efficient, flexible, and natural approach for instancewise feature importance scoring. In summary, our contributions in this work are as follows (see also Table 3.1 for systematic comparisons):

- We propose an information-based framework for instancewise feature importance scoring.
- We introduce a learning-based method for instancewise feature selection, which is both efficient and model-agnostic.

Furthermore, we show that the effectiveness of our method on a variety of synthetic and real data sets using both quantitative metric and human evaluation on Amazon Mechanical Turk.

3.2 Our framework

We now lay out the primary ingredients of our general approach. While our framework is generic and can be applied to both classification and regression models, the current discussion is restricted to classification models. We assume one has access to the output of a model as a conditional distribution, $\mathbb{P}_m(\cdot \mid x)$, of the response variable Y given the realization of the input random variable $X = x \in \mathbb{R}^d$.

3.2.1 Mutual information

Our method is derived from considering the mutual information between a particular pair of random vectors, so we begin by providing some basic background. Given two random vectors X and Y, the *mutual information* $I(X;Y)$ is a measure of dependence between them; intuitively, it corresponds to how much knowledge of one random vector reduces the uncertainty about the other. More precisely, the mutual information is given by the Kullback-Leibler divergence of the product of marginal distributions of X and Y from the joint distribution of X and Y [71]; it takes the form

$$I(X;Y) = \mathbb{E}_{X,Y}\left[\log \frac{p_{XY}(X,Y)}{p_X(X)p_Y(Y)}\right],$$

where p_{XY} and p_X, p_Y are the joint and marginal probability densities if X, Y are continuous, or the joint and marginal probability mass functions if they are discrete. The expectation is taken with respect to the joint distribution of X and Y. One can show the mutual information is nonnegative and symmetric in two random variables. The mutual information has been a popular criteria in feature selection, where one selects the subset of features that approximately maximizes the mutual information between the response variable and the selected features [89, 90]. Here we propose to use mutual information as a criteria for instancewise feature selection.

3.2.2 How to construct explanations

We now describe how to construct explanations using mutual information. In our specific setting, the pair (X, Y) are characterized by the marginal distribution $X \sim \mathbb{P}_X(\cdot)$, and a family of conditional distributions of the form $(Y \mid x) \sim \mathbb{P}_m(\cdot \mid x)$. For a given positive integer k, let $\wp_k = \{S \subset 2^d \mid |S| = k\}$ be the set of all subsets of size k. An *explainer* $\mathcal{E}$ of size k is a mapping from the feature space $\mathbb{R}^d$ to the power set $\wp_k$; we allow the mapping to be randomized, meaning that we can also think of $\mathcal{E}$ as mapping x to a conditional distribution $\mathbb{P}(S \mid x)$ over $S \in \wp_k$. Given the chosen subset $S = \mathcal{E}(x)$, we use x_S to denote the sub-vector formed by the chosen features. We view the choice of the number of explaining features k as best left in the hands of the user, but it can also be tuned as a hyper-parameter.

We have thus defined a new random vector $X_S \in \mathbb{R}^k$; see Figure 3.1 for a probabilistic graphical model representing its construction. We formulate instancewise feature selection as seeking explainer that optimizes the criterion

$$\max_{\mathcal{E}} I(X_S;Y) \quad \text{subject to} \qquad S \sim \mathcal{E}(X). \tag{3.1}$$

In words, we aim to maximize the mutual information between the response variable from the model and the selected features, as a function of the choice of selection rule.

It turns out that a global optimum of Problem (3.1) has a natural information-theoretic interpretation: it corresponds to the minimization of the expected length of encoded message for the model $\mathbb{P}_m(Y \mid x)$ using $\mathbb{P}_m(Y|x_S)$, where the latter corresponds to the conditional distribution of Y upon observing the selected sub-vector. More concretely, we have the

following:

Theorem 3. *Letting* $\mathbb{E}_m[\cdot \mid x]$ *denote the expectation over* $\mathbb{P}_m(\cdot \mid x)$*, define*

$$\mathcal{E}^*(x) := \arg\min_S \ \mathbb{E}_m \left[\log \frac{1}{\mathbb{P}_m(Y \mid x_S)} \,\middle|\, x \right]. \tag{3.2}$$

Then $\mathcal{E}^*$ *is a global optimum of Problem* (3.1)*. Conversely, any global optimum of Problem* (3.1) *degenerates to* $\mathcal{E}^*$ *almost surely over the marginal distribution* $\mathbb{P}_X$*.*

The proof of Theorem 3 is left to Section 3.5. In practice, the above global optimum is obtained only if the explanation family $\mathcal{E}$ is sufficiently large. In the case when $\mathbb{P}_m(Y|x_S)$ is unknown or computationally expensive to estimate accurately, we can choose to restrict $\mathcal{E}$ to suitably controlled families so as to prevent overfitting.

3.3 Our proposed method

A direct solution to Problem (3.1) is not possible, so that we need to approach it by a variational approximation. In particular, we derive a lower bound on the mutual information, and we approximate the model conditional distribution $\mathbb{P}_m$ by a suitably rich family of functions.

3.3.1 Obtaining a tractable variational formulation

We now describe the steps taken to obtain a tractable variational formulation.

A variational lower bound: Mutual information between X_S and Y can be expressed in terms of the conditional distribution of Y given X_S:

$$\begin{aligned} I(X_S; Y) &= \mathbb{E}\Big[\log \frac{\mathbb{P}_m(X_S, Y)}{\mathbb{P}(X_S)\mathbb{P}_m(Y)} \Big] = \mathbb{E}\Big[\log \frac{\mathbb{P}_m(Y|X_S)}{\mathbb{P}_m(Y)} \Big] \\ &= \mathbb{E}\Big[\log \mathbb{P}_m(Y|X_S) \Big] + \text{Const.} \\ &= \mathbb{E}_X \mathbb{E}_{S|X} \mathbb{E}_{Y|X_S} \Big[\log \mathbb{P}_m(Y|X_S) \Big] + \text{Const.} \end{aligned}$$

For a generic model, it is impossible to compute expectations under the conditional distribution $\mathbb{P}_m(\cdot \mid x_S)$. Hence we introduce a variational family for approximation:

$$\mathcal{Q} := \Big\{ \mathbb{Q} \mid \mathbb{Q} = \{ x_S \to \mathbb{Q}_S(Y|x_S), S \in \wp_k \} \Big\}. \tag{3.3}$$

Note each member $\mathbb{Q}$ of the family $\mathcal{Q}$ is a collection of conditional distributions $\mathbb{Q}_S(Y|x_S)$, one for each choice of k-sized feature subset S. For any $\mathbb{Q}$, an application of Jensen's inequality

yields the lower bound

$$\begin{aligned}\mathbb{E}_{Y|X_S}[\log \mathbb{P}_m(Y|X_S)] &\geq \int \mathbb{P}_m(Y|X_S) \log \mathbb{Q}_S(Y|X_S) \\ &= \mathbb{E}_{Y|X_S}[\log \mathbb{Q}_S(Y|X_S)],\end{aligned}$$

where equality holds if and only if $\mathbb{P}_m(Y|X_S)$ and $\mathbb{Q}_S(Y|X_S)$ are equal in distribution. We have thus obtained a variational lower bound of the mutual information $I(X_S; Y)$. Problem (3.1) can thus be relaxed as maximizing the variational lower bound, over both the explanation $\mathcal{E}$ and the conditional distribution $\mathbb{Q}$:

$$\max_{\mathcal{E},\mathbb{Q}} \mathbb{E}\Big[\log \mathbb{Q}_S(Y \mid X_S)\Big] \qquad \text{such that } S \sim \mathcal{E}(X). \tag{3.4}$$

For generic choices $\mathbb{Q}$ and $\mathcal{E}$, it is still difficult to solve the variational approximation (3.4). In order to obtain a tractable method, we need to restrict both $\mathbb{Q}$ and $\mathcal{E}$ to suitable families over which it is efficient to perform optimization.

A single neural network for parametrizing $\mathbb{Q}$: Recall that $\mathbb{Q} = \{\mathbb{Q}_S(\cdot \mid x_S),\ S \in \wp_k\}$ is a collection of conditional distributions with cardinality $|\mathbb{Q}| = \binom{d}{k}$. We assume X is a continuous random vector, and $\mathbb{P}_m(Y \mid x)$ is continuous with respect to x. Then we introduce a single neural network function $g_\alpha : \mathbb{R}^d \times [c] \to [0, 1]$ for parametrizing $\mathbb{Q}$, where $[c] = \{0, 1, \ldots, c-1\}$ denotes the set of possible classes, and α denotes the learnable parameters. We define $\mathbb{Q}_S(Y|x_S) := g_\alpha(\tilde{x}_S, Y)$, where $\tilde{x}_S \in \mathbb{R}^d$ is transformed from x by replacing entries not in S with zeros:

$$(\tilde{x}_S)_i = \begin{cases} x_i, i \in S, \\ 0, i \notin S. \end{cases}$$

When X contains discrete features, we embed each discrete feature with a vector, and the vector representing a specific feature is set to zero simultaneously when the corresponding feature is not in S.

3.3.2 Continuous relaxation of subset sampling

Direct estimation of the objective function in equation (3.4) requires summing over $\binom{d}{k}$ combinations of feature subsets after the variational approximation. Several tricks exist for tackling this issue, like REINFORCE-type Algorithms [91], or weighted sum of features parametrized by deterministic functions of X. (A similar concept to the second trick is the "soft attention" structure in vision [1] and NLP [92] where the weight of each feature is parametrized by a function of the respective feature itself.) We employ an alternative approach generalized from Concrete Relaxation (Gumbel-softmax trick) [93–95], which empirically has a lower variance than REINFORCE and encourages discreteness [96].

The Gumbel-softmax trick uses the concrete distribution as a continuous differentiable approximation to a categorical distribution. In particular, suppose we want to approximate a categorical random variable represented as a one-hot vector in $\mathbb{R}^d$ with category probability

$p_1, p_2, \ldots, p_d$. The random perturbation for each category is independently generated from a Gumbel$(0, 1)$ distribution:

$$G_i = -\log(-\log u_i), u_i \sim \text{Uniform}(0, 1).$$

We add the random perturbation to the log probability of each category and take a temperature-dependent softmax over the d-dimensional vector:

$$C_i = \frac{\exp\{(\log p_i + G_i)/\tau\}}{\sum_{j=1}^{d} \exp\{(\log p_j + G_j)/\tau\}}.$$

The resulting random vector $C = (C_1, \ldots, C_d)$ is called a Concrete random vector, which we denote by

$$C \sim \text{Concrete}(\log p_1, \ldots, \log p_d).$$

We apply the Gumbel-softmax trick to approximate weighted subset sampling. We would like to sample a subset S of k distinct features out of the d dimensions. The sampling scheme for S can be equivalently viewed as sampling a k-hot random vector Z from $D_k^d := \{z \in \{0, 1\}^d \mid \sum z_i = k\}$, with each entry of z being one if it is in the selected subset S and being zero otherwise. An importance score which depends on the input vector is assigned for each feature. Concretely, we define $w_\theta \colon \mathbb{R}^d \to \mathbb{R}^d$ that maps the input to a d-dimensional vector, with the ith entry of $w_\theta(X)$ representing the importance score of the ith feature.

We start with approximating sampling k distinct features out of d features by the sampling scheme below: Sample a single feature out of d features independently for k times. Discard the overlapping features and keep the rest. Such a scheme samples at most k features, and is easier to approximate by a continuous relaxation. We further approximate the above scheme by independently sampling k independent Concrete random vectors, and then we define a d-dimensional random vector V that is the elementwise maximum of $C^1, C^2, \ldots, C^k$:

$$C^j \sim \text{Concrete}(w_\theta(X)) \text{ i.i.d. for } j = 1, 2, \ldots, k,$$
$$V = (V_1, V_2, \ldots, V_d), \quad V_i = \max_j C_i^j.$$

The random vector V is then used to approximate the k-hot random vector Z during training.

We write $V = V(\theta, \zeta)$ as V is a function of θ and a collection of auxiliary random variables ζ sampled independently from the Gumbel distribution. Then we use the elementwise product $V(\theta, \zeta) \odot X$ between V and X as an approximation of $\tilde{X}_S$.

3.3.3 The final objective and its optimization

After having applied the continuous approximation of feature subset sampling, we have reduced Problem (3.4) to the following:

$$\max_{\theta, \alpha} \mathbb{E}_{X,Y,\zeta}\Big[\log g_\alpha(V(\theta, \zeta) \odot X, Y)\Big], \tag{3.5}$$

where g_α denotes the neural network used to approximate the model conditional distribution, and the quantity θ is used to parametrize the explainer. In the case of classification with c

classes, we can write

$$\mathbb{E}_{X,\zeta}\Big[\sum_{y=1}^{c}[\mathbb{P}_m(y \mid X)\log g_\alpha(V(\theta,\zeta)\odot X, y)\Big]. \tag{3.6}$$

Note that the expectation operator $\mathbb{E}_{X,\zeta}$ does not depend on the parameters (α,θ), so that during the training stage, we can apply stochastic gradient methods to jointly optimize the pair (α,θ). In each update, we sample a mini-batch of unlabeled data with their class distributions from the model to be explained, and the auxiliary random variables ζ, and we then compute a Monte Carlo estimate of the gradient of the objective function (3.6).

3.3.4 The explaining stage

During the explaining stage, the learned explainer maps each sample X to a weight vector $w_\theta(X)$ of dimension d, each entry representing the importance of the corresponding feature for the specific sample X. In order to provide a deterministic explanation for a given sample, we rank features according to the weight vector, and the k features with the largest weights are picked as the explaining features.

For each sample, only a single forward pass through the neural network parametrizing the explainer is required to yield explanation. Thus our algorithm is much more efficient in the explaining stage compared to other model-agnostic explainers like LIME or Kernel SHAP which require thousands of evaluations of the original model per sample.

3.4 Experiments

We carry out experiments on both synthetic and real data sets. For all experiments, we use RMSprop [95] with the default hyperparameters for optimization. We also fix the step size to be 0.001 across experiments. The temperature for Gumbel-softmax approximation is fixed to be 0.1. Codes for reproducing the key results are available online at `https://github.com/Jianbo-Lab/L2X`.

3.4.1 Synthetic Data

We begin with experiments on four synthetic data sets:

- 2-dimensional XOR as binary classification. The input vector X is generated from a 10-dimensional standard Gaussian. The response variable Y is generated from $P(Y = 1|X) \propto \exp\{X_1X_2\}$.
- Orange Skin. The input vector X is generated from a 10-dimensional standard Gaussian. The response variable Y is generated from $P(Y = 1|X) \propto \exp\{\sum_{i=1}^{4} X_i^2 - 4\}$.
- Nonlinear additive model. Generate X from a 10-dimensional standard Gaussian. The response variable Y is generated from $P(Y = 1|X) \propto \exp\{-100\sin(2X_1) + 2|X_2| + X_3 + \exp\{-X_4\}\}$.

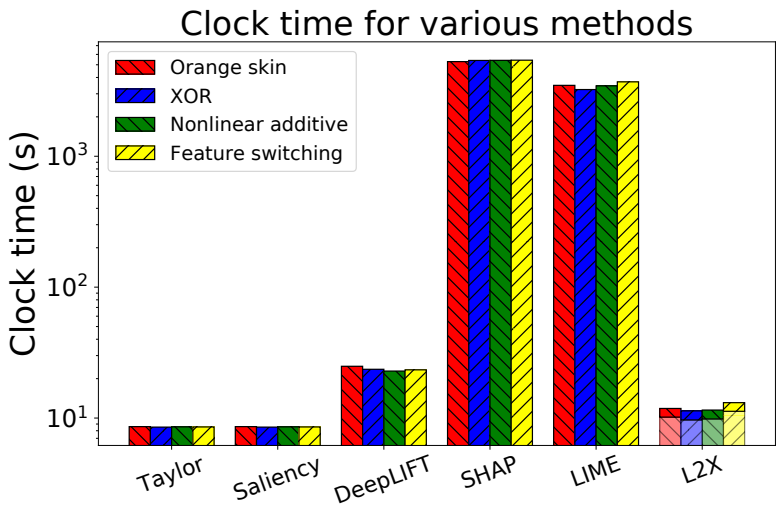


Figure 3.2: The clock time (in log scale) of explaining $10,000$ samples for each method. The training time of L2X is shown in translucent bars.

- Switch feature. Generate X_1 from a mixture of two Gaussians centered at ± 3 respectively with equal probability. If X_1 is generated from the Gaussian centered at 3, the $2-5$th dimensions are used to generate Y like the orange skin model. Otherwise, the $6-9th$ dimensions are used to generate Y from the nonlinear additive model.

The first three data sets are modified from commonly used data sets in the feature selection literature [97]. The fourth data set is designed specifically for instancewise feature importance scoring. Every sample in the first data set has the first two dimensions as true features, where each dimension itself is independent of the response variable Y but the combination of them has a joint effect on Y. In the second data set, the samples with positive labels centered around a sphere in a four-dimensional space. The sufficient statistic is formed by an additive model of the first four features. The response variable in the third data set is generated from a nonlinear additive model using the first four features. The last data set switches important features (roughly) based on the sign of the first feature. The $1-5$ features are true for samples with X_1 generated from the Gaussian centered at -3, and the $1, 6-9$ features are true otherwise.

We compare our method L2X (for "Learning to Explain") with several strong existing algorithms for instancewise feature importance scoring, including Saliency [8], DeepLIFT [10], SHAP [15], LIME [14]. Saliency refers to the method that computes the gradient of the selected class with respect to the input feature and uses the absolute values as importance scores. SHAP refers to Kernel SHAP. The number of samples used for explaining each instance for LIME and SHAP is set as default for all experiments. We also compare with a method that ranks features by the input feature times the gradient of the selected class with respect to the input feature. Shrikumar, Greenside, and Kundaje [10] showed it is equivalent to LRP [9] when activations are piecewise linear, and used it in Shrikumar, Greenside, and Kundaje [10] as a strong baseline. We call it "Taylor" as it is the first-order Taylor

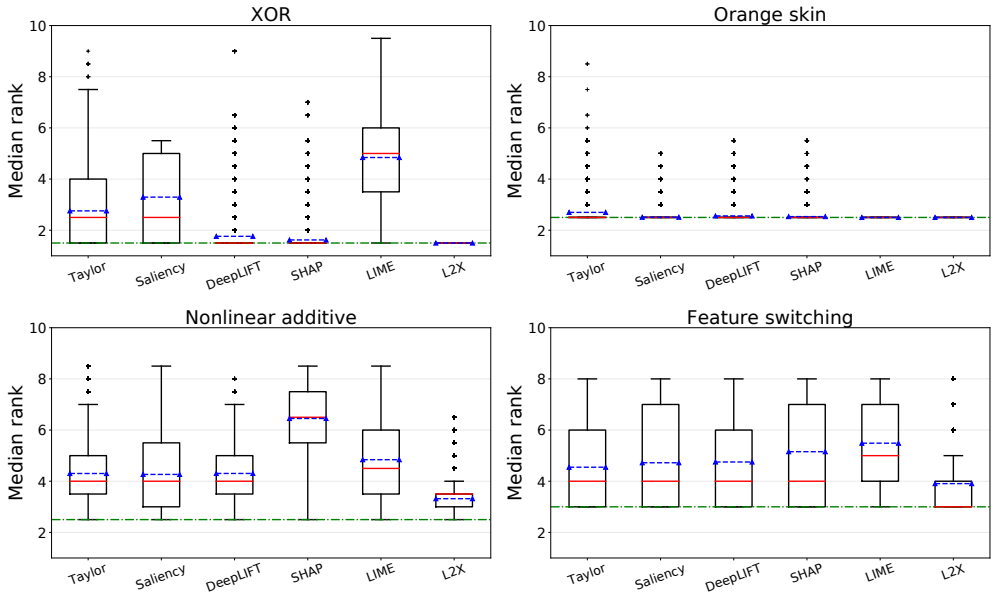


Figure 3.3: The box plots for the median ranks of the influential features by each sample, over 10,000 samples for each data set. The red line and the dotted blue line on each box is the median and the mean respectively. Lower median ranks are better. The dotted green lines indicate the optimal median rank.

approximation of the model.

Our experimental setup is as follows. For each data set, we train a neural network model with three hidden dense layers. We can safely assume the neural network has successfully captured the important features, and ignored noise features, based on its error rate. Then we use Taylor, Saliency, DeepLIFT, SHAP, LIME, and L2X for instancewise feature importance scoring on the trained neural network models. For L2X, the explainer is a neural network composed of two hidden layers. The variational family is composed of three hidden layers. All layers are linear with dimension 200. The number of desired features k is set to the number of true features.

The underlying true features are known for each sample, and hence the median ranks of selected features for each sample in a validation data set are reported as a performance metric, the box plots of which have been plotted in Figure 3.3. We observe that L2X outperforms all other methods on nonlinear additive and feature switching data sets. On the XOR model, DeepLIFT, SHAP and L2X achieve the best performance. On the orange skin model, all algorithms have near optimal performance, with L2X and LIME achieving the most stable performance across samples.

We also report the clock time of each method in Figure 3.2, where all experiments were performed on a single NVidia Tesla k80 GPU, coded in TensorFlow. Across all the four data sets, SHAP and LIME are the least efficient as they require multiple evaluations of the model. DeepLIFT, Taylor and Saliency requires a backward pass of the model. DeepLIFT is the slowest among the three, probably due to the fact that backpropagation of gradients

Truth	Model	Key words
positive	positive	Ray Liotta and Tom Hulce shine in this sterling example of brotherly love and commitment. Hulce plays Dominick, (nicky) a mildly mentally handicapped young man who is putting his 12 minutes younger, twin brother, Liotta, who plays Eugene, through medical school. It is set in Baltimore and deals with the issues of sibling rivalry, the unbreakable bond of twins, child abuse and good always winning out over evil. It is captivating, and filled with laughter and tears. If you have not yet seen this film, please rent it, I promise, you'll be amazed at how such a wonderful film could go unnoticed.
negative	negative	Sorry to go against the flow but I thought this film was unrealistic, boring and way too long. I got tired of watching Gena Rowlands long arduous battle with herself and the crisis she was experiencing. Maybe the film has some cinematic value or represented an important step for the director but for pure entertainment value. I wish I would have skipped it.
negative	positive	This movie is chilling reminder of Bollywood being just a parasite of Hollywood. Bollywood also tends to feed on past blockbusters for furthering its industry. Vidhu Vinod Chopra made this movie with the reasoning that a cocktail mix of deewar and on the waterfront will bring home an oscar. It turned out to be rookie mistake. Even the idea of the title is inspired from the Elia Kazan classic. In the original, Brando is shown as raising doves as symbolism of peace. Bollywood must move out of Hollywoods shadow if it needs to be taken seriously.
positive	negative	When a small town is threatened by a child killer, a lady police officer goes after him by pretending to be his friend. As she becomes more and more emotionally involved with the murderer her psyche begins to take a beating causing her to lose focus on the job of catching the criminal. Not a film of high voltage excitement, but solid police work and a good depiction of the faulty mind of a psychotic loser.

Table 3.2: True labels and labels predicted by the model are in the first two columns. Key words picked by L2X are highlighted in yellow.

for Taylor and Saliency are built-in operations of TensorFlow, while backpropagation in DeepLIFT is implemented with high-level operations in TensorFlow. Our method L2X is the most efficient in the explanation stage as it only requires a forward pass of the subset sampler. It is much more efficient compared to SHAP and LIME even after the training time has been taken into consideration, when a moderate number of samples (10,000) need to be explained. As the scale of the data to be explained increases, the training of L2X accounts for a smaller proportion of the over-all time. Thus the relative efficiency of L2X to other algorithms increases with the size of a data set.

3.4.2 IMDB

The Large Movie Review Dataset (IMDB) is a dataset of movie reviews for sentiment classification [69]. It contains $50,000$ labeled movie reviews, with a split of $25,000$ for training and $25,000$ for testing. The average document length is 231 words, and 10.7 sentences. We use L2X to study two popular classes of models for sentiment analysis on the IMDB data set.

Truth	Predicted	Key sentence
positive	positive	There are few really hilarious films about science fiction but this one will knock your sox off. The lead Martians Jack Nicholson take-off is side-splitting. The plot has a very clever twist that has be seen to be enjoyed. This is a movie with heart and excellent acting by all. Make some popcorn and have a great evening.
negative	negative	You get 5 writers together, have each write a different story with a different genre, and then you try to make one movie out of it. Its action, its adventure, its sci-fi, its western, its a mess. Sorry, but this movie absolutely stinks. 4.5 is giving it an awefully high rating. That said, its movies like this that make me think I could write movies, and I can barely write.
negative	positive	This movie is not the same as the 1954 version with Judy garland and James mason, and that is a shame because the 1954 version is, in my opinion, much better. I am not denying Barbra Streisand's talent at all. She is a good actress and brilliant singer. I am not acquainted with Kris Kristofferson's other work and therefore I can't pass judgment on it. However, this movie leaves much to be desired. It is paced slowly, it has gratuitous nudity and foul language, and can be very difficult to sit through. However, I am not a big fan of rock music, so its only natural that I would like the judy garland version better. See the 1976 film with Barbra and Kris, and judge for yourself.
positive	negative	The first time you see the second renaissance it may look boring. Look at it at least twice and definitely watch part 2. it will change your view of the matrix. Are the human people the ones who started the war? Is ai a bad thing?

Table 3.3: True labels and labels from the model are shown in the first two columns. Key sentences picked by L2X highlighted in yellow.

Explaining a CNN model with key words

Convolutional neural networks (CNN) have shown excellent performance for sentiment analysis [76, 98]. We use a simple CNN model on Keras [99] for the IMDB data set, which is composed of a word embedding of dimension 50, a 1-D convolutional layer of kernel size 3 with 250 filters, a max-pooling layer and a dense layer of dimension 250 as hidden layers. Both the convolutional and the dense layers are followed by ReLU as nonlinearity, and Dropout [78] as regularization. Each review is padded/cut to 400 words. The CNN model achieves 90% accuracy on the test data, close to the state-of-the-art performance (around 94%). We would like to find out which k words make the most influence on the decision of the model in a specific review. The number of key words is fixed to be $k = 10$ for all the experiments.

The explainer of L2X is composed of a global component and a local component (See Figure 2 in Yang et al. [64]). The input is initially fed into a common embedding layer followed by a convolutional layer with 100 filters. Then the local component processes the common output using two convolutional layers with 50 filters, and the global component processes the common output using a max-pooling layer followed by a 100-dimensional dense layer. Then we concatenate the global and local outputs corresponding to each feature, and process them through one convolutional layer with 50 filters, followed by a Dropout layer

[78]. Finally a convolutional network with kernel size 1 is used to yield the output. All previous convolutional layers are of kernel size 3, and ReLU is used as nonlinearity. The variational family is composed of an word embedding layer of the same size, followed by an average pooling and a 250-dimensional dense layer. Each entry of the output vector V from the explainer is multiplied with the embedding of the respective word in the variational family. We use both automatic metrics and human annotators to validate the effectiveness of L2X.

Post-hoc accuracy. We introduce *post-hoc accuracy* for quantitatively validating the effectiveness of our method. Each model explainer outputs a subset of features X_S for each specific sample X. We use $\mathbb{P}_m(y \mid \tilde{X}_S)$ to approximate $\mathbb{P}_m(y \mid X_S)$. That is, we feed in the sample X to the model with unselected words masked by zero paddings. Then we compute the accuracy of using $\mathbb{P}_m(y \mid \tilde{X}_S)$ to predict samples in the test data set labeled by $\mathbb{P}_m(y \mid X)$, which we call *post-hoc accuracy* as it is computed after instancewise feature selection.

Human accuracy. When designing human experiments, we assume that the key words convey an attitude toward a movie, and can thus be used by a human to infer the review sentiment. This assumption has been partially validated given the aligned outcomes provided by post-hoc accuracy and by human judges, because the alignment implies the consistency between the sentiment judgement based on selected words from the original model and that from humans. Based on this assumption, we ask humans on Amazon Mechanical Turk (AMT) to infer the sentiment of a review given the ten key words selected by each explainer. The words adjacent to each other, like "not good at all," keep their adjacency on the AMT interface if they are selected simultaneously. The reviews from different explainers have been mixed randomly, and the final sentiment of each review is averaged over the results of multiple human annotators. We measure whether the labels from human based on selected words align with the labels provided by the model, in terms of the average accuracy over 500 reviews in the test data set. Some reviews are labeled as "neutral" based on selected words, which is because the selected key words do not contain sentiment, or the selected key words contain comparable numbers of positive and negative words. Thus these reviews are neither put in the positive nor in the negative class when we compute accuracy. We call this metric *human accuracy*.

The result is reported in Table 3.4. We observe that the model prediction based on only ten words selected by L2X align with the original prediction for over 90% of the data. The human judgement given ten words also aligns with the model prediction for 84.4% of the data. The human accuracy is even higher than that based on the original review, which is 83.3% [64]. This indicates the selected words by L2X can serve as key words for human to understand the model behavior. Table 3.2 shows the results of our model on four examples.

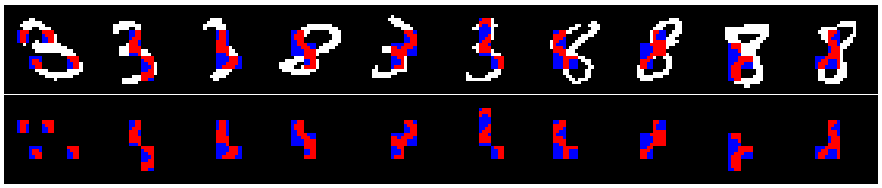


Figure 3.4: The above figure shows ten randomly selected figures of 3 and 8 in the validation set. The first line include the original digits while the second line does not. The selected patches are colored with red if the pixel is activated (white) and blue otherwise.

	IMDB-Word	IMDB-Sent	MNIST
Post-hoc accuracy	0.908	0.849	0.958
Human accuracy	0.844	0.774	NA

Table 3.4: Post-hoc accuracy and human accuracy of L2X on three models: a word-based CNN model on IMDB, a hierarchical LSTM model on IMDB, and a CNN model on MNIST.

Explaining hierarchical LSTM

Another competitive class of models in sentiment analysis uses hierarchical LSTM [77, 100]. We build a simple hierarchical LSTM by putting one layer of LSTM on top of word embeddings, which yields a representation vector for each sentence, and then using another LSTM to encoder all sentence vectors. The output representation vector by the second LSTM is passed to the class distribution via a linear layer. Both the two LSTMs and the word embedding are of dimension 100. The word embedding is pretrained on a large corpus [101]. Each review is padded to contain 15 sentences. The hierarchical LSTM model gets around 90% accuracy on the test data. We take each sentence as a single feature group, and study which sentence is the most important in each review for the model.

The explainer of L2X is composed of a 100-dimensional word embedding followed by a convolutional layer and a max pooling layer to encode each sentence. The encoded sentence vectors are fed through three convolutional layers and a dense layer to get sampling weights for each sentence. The variational family also encodes each sentence with a convolutional layer and a max pooling layer. The encoding vectors are weighted by the output of the subset sampler, and passed through an average pooling layer and a dense layer to the class probability. All convolutional layers are of filter size 150 and kernel size 3. In this setting, L2X can be interpreted as a hard attention model [2] that employs the Gumbel-softmax trick.

Comparison is carried out with the same metrics. For *human accuracy*, one selected sentence for each review is shown to human annotators. The other experimental setups are kept the same as above. We observe that post-hoc accuracy reaches 84.4% with one sentence selected by L2X, and human judgements using one sentence align with the original model prediction for 77.4% of data. Table 3.3 shows the explanations from our model on four examples.

3.4.3 MNIST

The MNIST data set contains 28×28 images of handwritten digits [80]. We form a subset of the MNIST data set by choosing images of digits 3 and 8, with $11,982$ images for training and $1,984$ images for testing. Then we train a simple neural network for binary classification over the subset, which achieves accuracy 99.7% on the test data set. The neural network is composed of two convolutional layers of kernel size 5 and a dense linear layer at last. The two convolutional layers contains 8 and 16 filters respectively, and both are followed by a max pooling layer of pool size 2. We try to explain each sample image with $k = 4$ image patches on the neural network model, where each patch contains 4×4 pixels, obtained by dividing each 28×28 image into 7×7 patches. We use patches instead of raw pixels as features for better visualization.

We parametrize the explainer and the variational family with three-layer and two-layer convolutional networks respectively, with max pooling added after each hidden layer. The 7×7 vector sampled from the explainer is upsampled (with repetition) to size 28×28 and multiplied with the input raw pixels.

We use only the post-hoc accuracy for experiment, with results shown in Table 3.4. The predictions based on 4 patches selected by L2X out of 49 align with those from original images for 95.8% of data. Randomly selected examples with explanations are shown in Figure 3.4. We observe that L2X captures most of the informative patches, in particular those containing patterns that can distinguish 3 and 8.

3.5 Proof of Theorem 1

Proof. **Forward direction:** Any explanation is represented as a conditional distribution of the feature subset over the input vector. Given the definition of S^*, we have for any X, and any explanation $\mathcal{E} : S|X$,

$$\mathbb{E}_{S|X}\mathbb{E}_m[\log P_m(Y|X_S)|X] \leq \mathbb{E}_m[\log P_m(Y|X_{S^*(X)})|X].$$

In the case when $S^*(X)$ is a set instead of a singleton, we identify $S^*(X)$ with any distribution that assigns arbitrary probability to each elements in $S^*(X)$ with zero probability outside $S^*(X)$. With abuse of notation, S^* indicates both the set function that maps every X to a set $S^*(X)$ and any real-valued function that maps X to an element in $S^*(X)$.

Taking expectation over the distribution of X, and adding $\mathbb{E}\log P_m(Y)$ at both sides, we have

$$I(X_S; Y) \leq I(X_{S^*}; Y)$$

for any explanation $\mathcal{E} : S|X$.

Reverse direction: The reverse direction is proved by contradiction. Assume the optimal explanation $P(S|X)$ is such that there exists a set M of nonzero probability, over

which $P(S|X)$ does not degenerates to an element in $S^*(X)$. Concretely, we define M as

$$M = \{x : P(S \notin S^*(x)|X = x) > 0\}.$$

For any $x \in M$, we have

$$\mathbb{E}_{S|X}\mathbb{E}_m[\log P_m(Y|X_S)|X = x] < \mathbb{E}_m[\log P_m(Y|X_{S^*(x)})|X = x], \tag{3.7}$$

where $S^*(x)$ is a deterministic function in the set of distributions that assign arbitrary probability to each elements in $S^*(x)$ with zero probability outside $S^*(x)$. Outside M, we always have

$$\mathbb{E}_{S|X}\mathbb{E}_m[\log P_m(Y|X_S)|X = x] \leq \mathbb{E}_m[\log P_m(Y|X_{S^*(x)})|X = x] \tag{3.8}$$

from the definition of S^*. As M is of nonzero size over $P(X)$, combining Equation 3.7 and Equation 3.8 and taking expectation with respect to $P(X)$, we have

$$I(X_S; Y) < I(X_{S^*}; Y), \tag{3.9}$$

which is a contradiction. □

3.6 Conclusion

We have proposed a framework for instancewise feature importance scoring via mutual information, and a method L2X which seeks a variational approximation of the mutual information, and makes use of a Gumbel-softmax relaxation of discrete subset sampling during training. To our best knowledge, L2X is the first method to realize real-time interpretation of a black-box model. We have shown the efficiency and the capacity of L2X for instancewise feature importance scoring on both synthetic and real data sets.

Chapter 4

LS-Tree: Model Interpretation When the Data Are Linguistic

In this chapter, we study the problem of interpreting trained classification models in the setting of linguistic data sets. The framework developed in this chapter provides a guideline for incorporating into importance scores the prior knowledge about what constitutes a satisfying interpretation in a given domain. It also outlines a framework for quantifying the importance of interactions between words, going beyond the focus of previous chapters where importance scores are assigned to single features.

Leveraging a parse tree, we propose to assign least-squares-based importance scores to each word of an instance by exploiting syntactic constituency structure. We establish an axiomatic characterization of these importance scores by relating them to the Banzhaf value in coalitional game theory. Based on these importance scores, we develop a principled method for detecting and quantifying interactions between words in a sentence. We demonstrate that the proposed method can aid in interpretability and diagnostics for several widely-used language models.

4.1 Introduction

While the generality of the stand-alone approach to interpretation is appealing, current methods provide little opportunity to leverage prior knowledge about what constitutes a satisfying interpretation in a given domain. Such interpretive capabilities are available most notably in the setting of natural-language processing (NLP), where there is an ongoing effort to incorporate linguistic structure (syntactic, semantic and pragmatic) in machine learning models. Such structure can be brought to bear in the model construction, the interpretation of a model, or both. For example, Socher et al. [102] introduced a recursive deep model to understand and leverage compositionality in tasks such as sentiment detection. Lei, Barzilay, and Jaakkola [103] proposed to use a combination of two modular components, generator and encoder, to explicitly generate rationales and make prediction for NLP tasks.

Compositionality, expressed in the rules used to construct a sentence from its constituent expressions, is an important property of natural language. While current interpretation methods fall short of quantifying compositionality directly, there has been a growing interest in investigating the manner in which existing deep models capture the interactions between constituent expressions that are critical for successful prediction [13, 18, 103, 104]. However, existing approaches generally fall short of providing a systematic, quantitative treatment of interactions, and the generality to be applied to arbitrary models.

In the current chapter, we focus on the model-agnostic interpretation of NLP models. Our approach quantifies the importance of words by leveraging the syntactic structure of linguistic data, as represented by constituency-based parse trees. In particular, we develop the *LS-Tree value*, a procedure that provides instance-wise importance scores for a model by minimizing the sum-of-squared residuals at every node of a parse tree for the sentence in consideration. We provide theoretical support for this by relating it to the Banzhaf value in coalitional game theory [23].

Our framework also provides a seedbed for studying compositionality in natural language. Based on the LS-Tree value, we develop a novel method for quantifying interactions between sibling nodes on a parse tree captured by the target model, by exploiting Cook's distance in linear regression [63]. We show that the proposed algorithm can be used to analyze several aspects of widely-used NLP models, including nonlinearity, the ability to capture adversative relations, and overfitting. In particular, we carry out a series of experiments studying four models—a linear model with Bag-Of-Word features, a convolutional neural network [76], an LSTM [77], and the recently proposed BERT model [105].

4.2 Least squares on parse trees

For simplicity, we restrict ourselves to classification. Assume a model maps a sentence to a vector of class probabilities. We use f to denote the function that maps an input sentence $x = (x_1, \ldots, x_d)$ to the log-probability score of a selected class. Let $2^{[d]}$ denote the power set of $[d] := \{1, 2, \ldots, d\}$. The parse tree maps the sentence to a collection of subsets, denoted as $\wp \subset 2^{[d]}$, where each subset $S \in \wp$ contains the indices of words corresponding to one node in the parse tree. See Figure 4.1 for an example. By abuse of notation, we use $f(S)$ to denote the output of the model evaluated on the words with indices S, with the rest of the words replaced by zero paddings or some reference placeholder. We call $v : \wp \to \mathbb{R}$ defined by $v(S) := f(S) - f(\emptyset)$ a *characteristic function*, which captures the importance of each word subset to the prediction.

We seek the optimal linear function on the Boolean hypercube to approximate the characteristic function on $\wp$, and use the coefficients as importance scores assigned to each word. Concretely, we solve the following least squares problem:

$$\min_{\psi \in \mathbb{R}^d} \sum_{S \in \wp} [v(S) - \sum_{i \in S} \psi_i]^2, \tag{4.1}$$

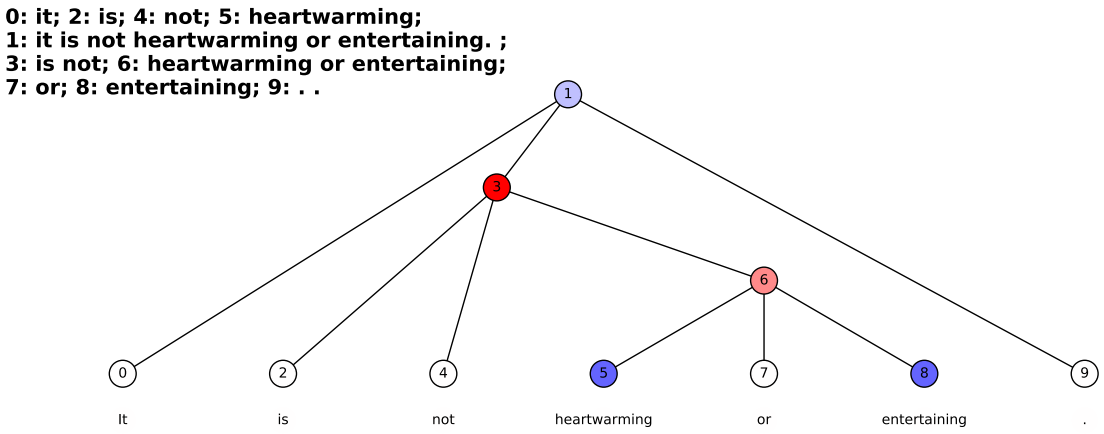


Figure 4.1: An example parse tree. Top left shows how each node corresponds to a word subset. Color indicates the direction and strength of interaction as assigned by Algorithm 1. Red is used for the direction of positive class, and blue otherwise.

where component ψ_i of the optimal ψ is the importance score of the word with index i. We refer to the map from $(\wp\mathcal{O}, v)$ to the solution of Equation (4.1) as the *LS-Tree value*, because it results from least squares (LS) on parse trees, and can be considered as a *value* in coalitional game theory.

4.3 Connection to coalitional game theory

In this section, we give an interpretation of the LS-Tree value from the perspective of coalitional game theory.

Model interpretation has been studied using tools from coalitional game theory [15–17, 20]. We build on this line of research by considering a restriction on coalitions induced by the syntactic structure of the input.

Let $\wp\mathcal{O} \subset 2^{[d]}$ be the collection of word subsets constructed from the parse tree. Taking each word as a player, we can define a coalitional game between d words in a sentence as a pair $(\wp\mathcal{O}, v)$, where $\wp\mathcal{O} \subset 2^{[d]}$ enforces restrictions on coalition among players and $v : \wp\mathcal{O} \to \mathbb{R}$ with $v(\emptyset) = 0$ is the characteristic function defined by the model evaluated on each coalition. A *value* is a mapping that associates a d-dimensional payoff vector $\psi(\wp\mathcal{O}, v)$ to each game $(\wp\mathcal{O}, v)$, each entry corresponding to a word. The value provides rules which give allocations to each player for any game.

The problem of defining a fair value in the setting of full coalition (when $\wp\mathcal{O} = 2^{[d]}$) has been studied extensively in coalitional game theory [22, 23]. One popular value is the Banzhaf value introduced by Banzhaf III [23]. For each $i \in [d]$ it defines the value:

$$\phi_i(2^{[d]}, v) = \frac{1}{2^{d-1}} \sum_{S \subset N \setminus i} [v(S \cup i) - v(S)].$$

The Banzhaf value can be characterized as the unique value that satisfies the following four properties [106]:

i) Symmetry: If $v(S \cup i) = v(S \cup j)$ for all $S \subset [d] \setminus \{i, j\}$, we have $\phi_i(2^{[d]}, v) = \phi_j(2^{[d]}, v)$.

ii) Dummy player property: If $v(S \cup i) = v(S) + v(i)$ for all $S \subset [d] \setminus i$, we have $\phi_i(2^{[d]}, v) = v(i)$.

iii) Marginal contributions: For any two characteristic functions v, w such that $v(S \cup i) - v(S) = w(S \cup i) - w(S)$ for any $S \subset [d]$, we have $\phi_i(2^{[d]}, v) = \phi_i(2^{[d]}, w)$.

iv) 2-Efficiency: If $i, j \in [d]$ merges into a new player p, then $\phi_p(2^{[d]\setminus\{i,j\}\cup p}, v^{ij}) = \phi_i(2^{[d]}, v) + \phi_j(2^{[d]}, v)$, where $v^{ij}(S) := v(S)$ if $p \notin S$ and $v^{ij}(S) := v(S \setminus p \cup i \cup j)$ otherwise, for any $S \subset [d] \setminus \{i, j\} \cup p$.

These properties are natural for allocation of importance to prediction in model interpretation. Symmetry states that two features have the same allocation if their marginal contributions to feature subsets are the same. The dummy property states that a feature is allocated the same amount as the contribution of itself alone if its marginal contribution always equals the model evaluation on its own. The linear model yields such an example. Marginal contributions states that a feature which has the same marginal contribution between two models for any word subset has the same amount of allocation. 2-Efficiency states that allocation of importance is immune to artificial merging of two features.

To employ game-theoretic concepts such as the Banzhaf value in the interpretation of NLP models, we need to recognize that arbitrary combinations of words are not likely to be accepted as valid interpretations by humans. We might wish to start with a set of combinations that are likely to be interpretable by humans, and can be obtained via human-interpretable data, and then define the worth of other combinations of words via extrapolation. It turns out that the LS-Tree value as defined in the previous section can be interpreted as exactly such an extrapolation, where each node of the parse tree represents an interpretable word combination:

Theorem 4. *Suppose a value ψ coincides with the Banzhaf value ϕ for any game of full coalition, and for every game $(\mathcal{O}, v)$ with restricted coalition, it is consistent under the addition of an arbitrary subset $S \notin \mathcal{O}$:*

$$\psi(\mathcal{O}, v) = \psi(\mathcal{O} \cup \{S\}, v'), \tag{4.2}$$

where v' is defined as $v'(T) = v(T)$ for $T \neq S$ and $v'(S) = \sum_{i \in S} \psi_i(\mathcal{O}, v)$. Then ψ coincides with the LS-Tree value.

Proof. It was shown in Hammer and Holzman [107] that the Banzhaf value assigns to each player i the corresponding coefficient in the best linear approximation of v. That is,

$$\phi(2^{[d]}, v) = \arg\min_{\psi \in \mathbb{R}^d} \sum_{S \subset [d]} [v(S) - \sum_{i \in S} \psi_i]^2.$$

Based on the proof of Theorem 3.3 in Katsev [108],[1] it follows directly that ψ^*, as is defined

[1]The original theorem is established for the solution to Problem (4.3) with the efficiency constraint that $\sum_{i \in [d]} x_i = v([d])$. But the same proof follows for the unconstrained version.

by Equation (4.3), is the unique value that coincides with $v \to \psi^*(2^{[d]}, v)$ with full coalition and is consistent under the addition of an arbitrary subset:

$$\psi^*(\mathcal{O}, v) = \arg\min_{\psi \in \mathbb{R}^d} \sum_{S \in \mathcal{O}} w_S [v(S) - \sum_{i \in S} \psi_i]^2. \tag{4.3}$$

Taking $w_S \equiv 1$, the theorem is established. □

4.4 Detecting interactions

We aim to detect and quantify interactions between words in a sentence that have been captured by the target model. While there are exponentially many possible interactions between arbitrary words, we restrict ourselves to the ones permitted by the structure of language. Concretely, we focus on interactions between siblings, or nodes with a common parent, in the parse tree. As an example, node 3 in Figure 4.1 represents interaction between "is," "not" and "heartwarming or entertaining."

We define interaction as *deviation of composition from linearity* in a given sentence. As a result, all non-leaf nodes in the tree are expected to admit zero interaction for a linear model. The above definition suggests that interaction can be quantified by studying how the inclusion of a common parent representing the interaction affects the coefficients of the linear approximation of the model.

Cook's distance is a classic metric in linear regression that captures the influence of a data point [63]. It is defined as a constant multiple of the squared distance between coefficients after a data point is moved, where the distance metric is defined by the data matrix $X \in \mathbb{R}^{n \times d}$:

$$D_i = \text{Const.} \cdot (\hat{\beta}_{(i)} - \hat{\beta})^T X^T X (\hat{\beta}_{(i)} - \hat{\beta}),$$

where $\hat{\beta}_{(i)}$ and $\hat{\beta}$ are the least-squares estimate with the ith data point deleted and the original least-squares estimate respectively. A larger Cook's distance indicates a larger influence of the corresponding data point.

In our setting, the data matrix X is a Boolean matrix where each row corresponds to a node in the tree, and an entry is one if and only if the word of the corresponding index lies in the subtree of the node. To capture the interaction of a non-leaf node i (corresponding to some $S \in \mathcal{O}$), it does not suffice to only delete the corresponding row, because all of its ancestor nodes contain the segment represented by the node as well. To deal with this issue, we compute the distance between the least-squares estimate with the rows corresponding to the node and all of its ancestors deleted, and the least-squares estimate with only the rows corresponding to the ancestors deleted:

$$D_i = d(\hat{\beta}_{(\geq i)}, \hat{\beta}_{(>i)}), \tag{4.4}$$

where $\hat{\beta}_{(\geq i)}, \hat{\beta}_{(>i)}$ denote the estimates with all ancestors, including and excluding node i, deleted. Cook's distance $d(a, b) = a^T X^T X b$ no longer has its statistical meaning here, as the normality assumption of the linear model no longer holds. A natural choice is the Euclidean

distance $d_1(a, b) := \sqrt{a^T b}$, which was also introduced by Cook [63]. One drawback of the Euclidean distance is that it is unable to capture the direction of interaction. When this is an issue, we may use a signed distance: $d_2(a, b) := \sum_i (b_i - a_i)$, which sums up the influence of introducing the extra row on every coefficient of the linear model. We call the score defined by d_1 and d_2 absolute and signed *LS-Tree interaction scores* respectively, as they are constructed from the LS-Tree value.

We propose an iterative algorithm to efficiently compute the interaction of each node on a tree with $n := |\mathcal{O}|$ nodes. As a first step, n model evaluations are performed, one evaluation for each node. For a node i, we denote as $\mathrm{Ch}(i)$ the set of its children, $X_{(\geq i)}$ and $X_{(>i)}$ the data matrices excluding the ancestors of i, further excluding and including i itself respectively, and x_j^T the row corresponding to node j. The interaction score of each $j \in \mathrm{Ch}(i)$ is a function of $\hat{\beta}_{(>j)} - \hat{\beta}_{(\geq j)}$. Denote $A_j = X_{(\geq j)}^T X_{(\geq j)}$. For each non-leaf node j, A_j is of full rank and thus invertible. We show how A_j^{-1} and $\hat{\beta}_{\geq j}$ can be computed from A_i^{-1} and $\hat{\beta}_{\geq i}$. In fact, with an application of the Sherman-Morrison formula [109], we have

$$\begin{aligned}
\hat{\beta}_{(>j)} &= (X_{(\geq j)}^T X_{(\geq j)} + x_j^T x_j)^{-1}(X_{(\geq j)}^T Y_{(\geq j)} + x_j^T Y_j) \\
&= (I - \frac{(X_{(\geq j)}^T X_{(\geq j)})^{-1} x_j x_j^T}{1 + x_j^T (X_{(\geq j)}^T X_{(\geq j)})^{-1} x_j}) \hat{\beta}_{(\geq j)} \\
&\qquad + \frac{(X_{(\geq j)}^T X_{(\geq j)})^{-1} x_j Y_j}{1 + x_j^T (X_{(\geq j)}^T X_{(\geq j)})^{-1} x_j} \\
&= (I - \frac{A_j^{-1} x_j x_j^T}{1 + x_j^T A_j^{-1} x_j}) \hat{\beta}_{(\geq j)} + \frac{A_j^{-1} x_j Y_j}{1 + x_j^T A_j^{-1} x_j}. \qquad (4.5)
\end{aligned}$$

Rearranging the terms in Equation (4.5), we have

$$\hat{\beta}_{(\geq j)} = \hat{\beta}_{(>j)} - A_j^{-1} x_j [Y_j - x_j^T \hat{\beta}_{(>j)}]. \qquad (4.6)$$

With another application of the Sherman-Morrison formula, we have

$$\begin{aligned}
A_j^{-1} &= (X_{(\geq i)}^T X_{(\geq i)} - x_j x_j^T)^{-1} \\
&= A_i^{-1} + \frac{A_i^{-1} x_j x_j^T A_i^{-1}}{1 - x_j^T A_i^{-1} x_j}. \qquad (4.7)
\end{aligned}$$

For leaf nodes, the entry of $\hat{\beta}_{(\geq j)}$ corresponding to j is set to zero, with the remaining entries equal to those of $\hat{\beta}_{(>j)}$. This is a result of the minimal Euclidean norm solution of Problem 4.1, obtained from the pseudoinverse of A_j. Consequently, the (signed) interaction score of a leaf equals the model evaluation on the leaf alone.

We summarize the derivation in Algorithm 1, which traverses the parse tree from root to leaves in a top-down fashion to compute the interaction scores of each node. As the number of nodes in a parse tree is linear in the number of words, Algorithm 1 is of complexity $\mathcal{O}(d^3)$, plus the complexity of parsing the sentence, which is $\mathcal{O}(d)$ in our experiments, and $\mathcal{O}(d)$ model evaluations. Figure 4.1 shows how Algorithm 1 assigns signed interaction scores to a given example.

Algorithm 1 LS-Tree Interaction Detection

Require: Model f.
Require: Sentence x.
Ensure: LS-Tree value; interaction score.
  Find the parse tree $\mathcal{T}$ of x.
  Find the collection of subsets $\wp$ corresponding to the parse tree.
  for each node i in $\mathcal{T}$ **do**
    Compute the model evaluation $v(S)$ for the corresponding subset S.
  end for
  Compute LS-Tree value $\hat{\beta}$ for words via least squares.
  Find the root r of $\mathcal{T}$.
  Recursion(v, $\wp$, r, $(X^TX)^{-1}$, $\hat{\beta}$)

Algorithm 2 Recursion

Require: $v, \wp$, node j, A_i^{-1}, $\hat{\beta}_{(\geq i)}$
  if j is not a leaf **then**
    Compute $A_j^{-1}, \hat{\beta}_{(\geq j)}, D_j$ via Equation (4.7) and Equation (4.6).
    for each child c in of j **do**
      Recursion(v, $\wp$, c, A_j^{-1}, $\hat{\beta}_{(\geq j)}$)
    end for
  else
    Assign D_j with $v(j)$ or $|v(j)|$.
  end if

4.5 Experiments

We carry out experiments to analyze the performance of four different models: Bag of Words (BoW), Word-based Convolutional Neural Network (CNN) [76], bidirectional Long Short-Term Memory network (LSTM) [77], and Bidirectional Encoder Representations from Transformers (BERT) [105], across three sentiment data sets of different sizes: Stanford Sentiment Treebank (SST) [102], IMDB Movie reviews [69] and Yelp reviews Polarity [72]. For an instance with multiple sentences, we parse each sentence separately, and introduce an extra node as the common parent of all roots. Interactions between sentences are not considered in our experiments. The code for replicating the experiments is available anonymously on GitHub: `https://github.com/Anonymous-Alien/LS-tree`.

BoW fits a linear model on the Bag-of-Words features. Both CNN and LSTM use a 300-dimensional GloVe word embedding [110]. The CNN is composed of three 100-dimensional convolutional 1D layers with 3, 4 and 5 kernels respectively, concatenated and fed into a max-pooling layer followed by a hidden dense layer. The LSTM uses a bidirectional LSTM layer with 128 units for each direction. BERT pre-trains a deep bidirectional Transformer [7]

Data Set	Classes	Train Size	Test Size	Avg. Len.	BoW	CNN	LSTM	BERT
SST	2	6,920	872	19.7	0.82%	0.85%	0.85%	0.93%
IMDB	2	25,000	25,000	325.6	0.94%	0.90%	0.88%	0.93%
Yelp	2	560,000	38,000	136.2	0.94%	0.95%	0.96%	0.97%

Table 4.1: Statistics of the three data sets, together with the test accuracy of the four models.

BERT	BoW	Category	Correlation	Depth
Even if you do n't think kissinger's any more guilty of criminal activity than most contemporary statesmen, he'd sure make a courtroom trial great fun to watch.	Even if you don't think kissinger's any more guilty of criminal activity than most contemporary statesmen, he'd sure make a courtroom trial great fun to watch.	Positive	0.173	11
The problem with this film is that it lacks focus.	The problem with this film is that it lacks focus.	Negative	0.939	1
Funny but perilously slight.	Funny but perilously slight.	Positive	0.938	4

Table 4.2: Examples from SST with BERT and BoW. Correlation with linear coefficients and depth of the top node are listed. The top two words ranked by the LS-Tree value, and by the linear coefficients, are colorized.

on a large corpus of text by jointly conditioning on both left and right context in all layers. It has achieved state-of-the-art performance on a large suite of sentence-level and token-level tasks. See Table 4.1 for a summary of data sets and the accuracies of the four models.

We use the Stanford constituency parser [111–114] for all the experiments. It is a transition-based parser that is faster than chart-based parsers yet achieves comparable accuracy, by employing a set of shift-reduce operations and making use of non-local features.

4.5.1 Deviation from linearity

We quantify the deviation of three nonlinear models from a linear model via the proposed LS-Tree value and interaction scores, both for specific instances and on a data set.

The LS-Tree value can be interpreted as supplying the coefficients of the best linear model used to approximate the target model locally for each instance. The correlation between the LS-Tree value and the global linear model with Bag of Words (BoW) features can be used as a measure of nonlinearity of the target model at the instance. Table 4.2 shows three examples in SST, correctly classified by both BERT and BoW. BERT has low and high correlations with linear models at the first and second examples in Table 4.2 respectively. In particular, the top keywords, as ranked by the LS-Tree value, are different between two models.

The average of correlation with BoW across instances can be used as a measure of nonlinearity on a certain data set. The average correlation of BoW, CNN, LSTM and BERT with a linear model is shown in Table 4.3, which indicates that BERT is the most nonlinear model among the four. CNN is more nonlinear than LSTM on IMDB but comparably nonlinear on SST and Yelp.

Correlation alone may not suffice to capture the nonlinearity of a model. For example, the third sentence in Table 4.2 has a relatively high correlation, but the bottom left parse

	BoW	CNN	LSTM	BERT
SST	1.000	0.591	0.580	**0.465**
IMDB	1.000	0.442	0.552	**0.321**
Yelp	1.000	0.683	0.684	**0.476**

Table 4.3: Average correlation of the LS-Tree values with linear coefficients. The average correlation is comparable across different models on the same data set.

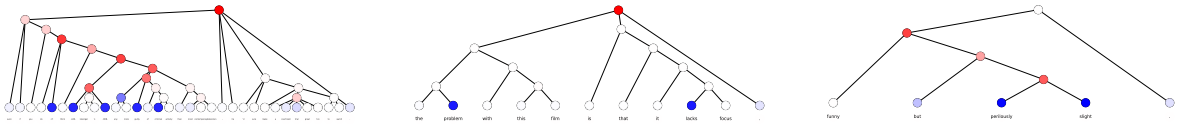
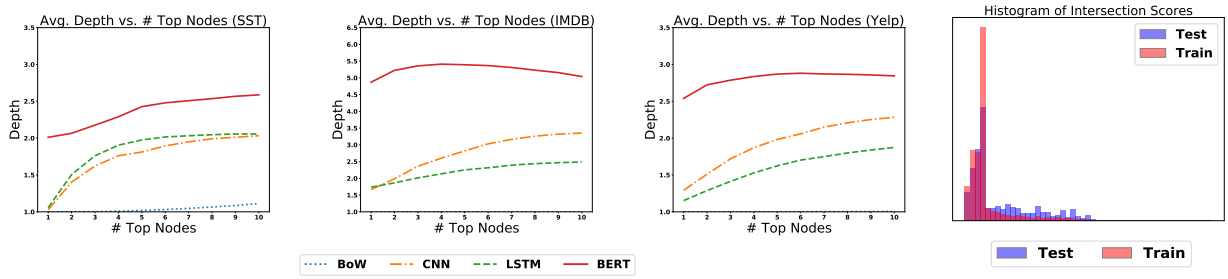


Figure 4.2: Visualization of parse trees of examples in Table 4.2. Nodes are colorized based on the signed interaction scores, red for the direction of positive class, and blue otherwise.



(a) (b)

Figure 4.3: (a) Average depth of top nodes as the number of the selected top nodes varies. (b) The histogram of intersection scores with train and test data for BERT on SST.

tree in Figure 4.2 indicates that the top interaction ranked by the signed interaction score is the node combining "funny" with "but perilously slight." This indicates the BERT model has captured the adversative conjunction, which BoW is not capable of. The ability to capture closer-to-the-top nodes in a parse tree is an indication of nonlinearity of the model. To quantify this ability, we define the depth of a node in the parse tree as the maximum distance of the node from the bottom:

$$\text{Depth}(i) = \begin{cases} 1 + \max_{c \in \text{Ch}(i)} \text{Depth}(c) & \text{if } \text{Ch}(i) \neq \emptyset, \\ 1 & \text{otherwise.} \end{cases}$$

For a linear model, all non-leaf nodes have zero interaction, and thus the top-ranked nodes are of depth 1, until all leaves with positive weights are enumerated. The higher the depth of top-ranked nodes, the more nonlinear a model is at a specific instance.

The average depths of top nodes ranked by interaction scores across instances can be used as a measure of the nonlinearity of the model on that data set. Figure 4.3a compares the average depths across BoW, CNN, LSTM and BERT on the three data sets, with top

Dataset	Model	Avg. Score	not	but	yet	though	although	even though	whereas	except	despite	in spite of
SST	BoW	0.153	0.000(6.318)	0.000(0.079)	0.000(2.005)	0.000(0.865)	0.000(2.222)	0.000(0.000)	-(-)	0.000(4.280)	0.000(3.519)	0.000(0.000)
	CNN	0.634	1.673(4.592)	1.694(1.444)	0.568(0.959)	0.213(0.735)	0.915(0.462)	0.626(0.407)	-(-)	0.948(1.175)	**1.452**(4.270)	**2.119**(1.943)
	LSTM	0.79	**1.746**(2.580)	1.502(0.453)	1.449(2.368)	1.153(1.094)	0.338(0.197)	1.794(0.998)	-(-)	**2.353**(3.835)	1.256(1.818)	0.590(0.624)
	BERT	1.238	1.714(4.383)	**2.148**(1.760)	**1.669**(3.120)	**1.525**(3.268)	**1.741**(3.256)	**1.885**(2.092)	-(-)	1.156(3.331)	1.160(2.998)	0.864(2.352)
IMDB	BoW	0.038	0.000(2.683)	0.000(0.263)	0.000(2.210)	0.000(1.473)	0.000(1.710)	0.000(0.000)	0.000(3.604)	0.000(1.342)	0.000(0.132)	-(-)
	CNN	0.424	1.050(0.819)	**3.442**(0.021)	**1.689**(0.295)	0.922(0.085)	1.036(0.071)	1.175(0.467)	0.469(1.064)	**1.590**(4.067)	0.363(0.434)	-(-)
	LSTM	0.126	0.960(3.087)	2.222(0.524)	1.500(0.238)	0.611(0.087)	0.492(1.270)	0.944(0.683)	**1.222**(3.865)	1.294(4.008)	0.286(0.508)	-(-)
	BERT	1.159	**1.616**(2.057)	3.390(1.800)	1.644(1.152)	**1.371**(2.061)	**1.735**(2.123)	**1.457**(1.557)	0.285(0.430)	1.421(2.060)	**1.518**(2.241)	-(-)
Yelp	BoW	0.035	0.000(8.488)	0.000(1.015)	0.000(3.553)	0.000(1.664)	0.000(1.128)	0.000(0.000)	0.000(0.536)	0.000(0.367)	0.000(1.213)	-(-)
	CNN	0.161	**2.287**(3.467)	**2.454**(0.932)	0.516(0.043)	0.988(0.435)	**0.889**(0.075)	0.789(0.621)	0.286(0.671)	**0.522**(2.529)	0.423(0.889)	-(-)
	LSTM	0.224	2.173(5.950)	1.712(1.676)	**0.988**(2.065)	0.984(1.310)	0.706(1.194)	0.559(0.483)	**1.395**(1.793)	0.344(1.408)	0.514(1.153)	-(-)
	BERT	0.746	1.384(2.106)	2.448(0.658)	0.781(0.184)	**1.336**(0.953)	0.596(0.615)	**1.019**(0.880)	0.095(0.162)	0.331(0.074)	**1.041**(0.414)	-(-)

Table 4.4: Scores with and without parentheses are for nodes containing adversative words alone and their parents where the adversative relation takes place respectively.

Sentence	Meaning	BoW	CNN	LSTM	BERT
... He said he couldn't help. We had to walk **while** the snow blew in our faces. When we were almost there, we saw the shuttle pull out with the smoking shuttle driver in it, driving in the opposite direction, away from us. I can not believe how rude they were.	during the time that	0.000(0.338)	0.781(0.300)	1.761(0.839)	0.062(0.092)
... I ordered a cappuccino. It tasted like milk and no coffee. I was exceptionally disappointed. So **while** the place has a great reputation, even they can screw it up if they don t pay attention to detail, and at this level they should never screw it up. I had a better cup at Martys Market for crying out loud!	whereas (indicating a contrast)	0.000(0.338)	1.142(0.300)	2.155(0.839)	2.167(0.092)
Usually asking the server what is her favorite dish gets you a pretty good recommendation, but in this case, it was crap! The smoked brisket had that discoloration that happens to meat when it's been sitting out for a **while**. And it wasn't even tender!! Am I asking for too much?	a period of time	0.000(0.338)	0.206(0.300)	0.465(0.839)	0.082 (0.092)

Table 4.5: The word "while" in different contexts. Scores with and without parentheses are for nodes containing "while" alone and their parents respectively.

$k = 1, 2, \ldots, 10$ words selected. BoW is used as a baseline whose non-leaf nodes have zero interaction scores. We use the absolute interaction scores here to capture all interactions, no matter whether they are in the same or opposite direction of prediction. BERT is still the most capable of capturing deeper interactions, followed by CNN and LSTM. CNN turns out to be a more nonlinear model than LSTM on Yelp, which was not captured by correlation.

4.5.2 Adversative relations

Adversative words are those which express opposition or contrast. They often play an important role in determining the sentiment of an instance, by reversing the sentiment of a preceding or succeeding word, phrase or statement. We focus on four types of adversative words: negation that reverses the sentiment of a phrase or word (e.g., "not"), adversative coordinating conjunctions that express opposition or contrast between two statements (e.g., "but" and "yet"), subordinating conjunctions indicating adversative relationship (e.g., "though," "although," "even though," and "whereas"), prepositions that precede and govern nouns adversatively (e.g., "except," "despite" and "in spite of").

In most cases, adversative words only function if they interact with their preceding or succeeding companion. In order to verify whether models are able to capture the adversative relationship, we examine the LS-Tree interaction scores of the parent nodes of these words.

We extract all instances that contain any of the above adversative words. Then for each word in an instance, we compute the interaction score of the corresponding node with the word alone, and that of its parent node. A high interaction score on the node with the adversative word alone indicates the model inappropriately attributes to the word itself a negative or positive sentiment. A high interaction score on the parent node indicates the model captures the interaction of the adversative word with its preceding or succeeding component. To compare across different models, we further compute the average interaction score of a generic node across all instances, and report the ratio of average interaction scores of specific nodes to the average score of a generic node for respective models.

Table 4.4 reports the results on three data sets. We observe the ability of capturing adversative relation for different models varies across data sets. BERT takes the lead in capturing adversative relations on SST and IMDB, perhaps with the help of BERT's pre-training process on a large corpus, but CNN and LSTM catch up with BERT on Yelp, which has a larger amount of training data. On the other hand, all models assign a high score on nodes with adversative words alone. This may result from the uneven distribution of adversative words like "not" among the positive and negative classes. An additional observation is that BERT has the highest score for a generic node on average across three data sets, indicating that BERT is the most sensitive to words and interactions on average.

Some words have different meanings in different contexts. It is interesting to investigate whether a model can distinguish the same word under different contexts. The word "while" is such an example. Table 4.5 shows three Yelp reviews that include "while." It can be observed that the scores of the parent nodes of "while" is higher than average when "while" contains an adversative meaning, but lower otherwise. This observation holds across CNN, LSTM and BERT, with the sharpest distinction on BERT.

4.5.3 Detecting overfitting

Overfitting happens when a model captures sampling noise in training data, while failing to capture underlying relationships between the inputs and outputs. Overfitting can be a problem in modern machine learning models like deep neural networks, due to their expressive nature. To mitigate overfitting, one often splits the initial training set into a training and a validation set, and uses the latter to obtain an estimate of the generalization performance [115]. This leads to a waste of training data, depriving the model of potential opportunities to learn from the labelled validation data. We observe that the LS-Tree interaction scores can be used to construct a diagnostic for overfitting, one which is solely computed with unlabelled data.

Figure 4.3b shows the histograms of absolute interaction scores on small subsets of training and test data of SST, for an overfitted BERT model. The scores are more spread out on test data than those on training data. In fact, we have observed that this phenomenon holds

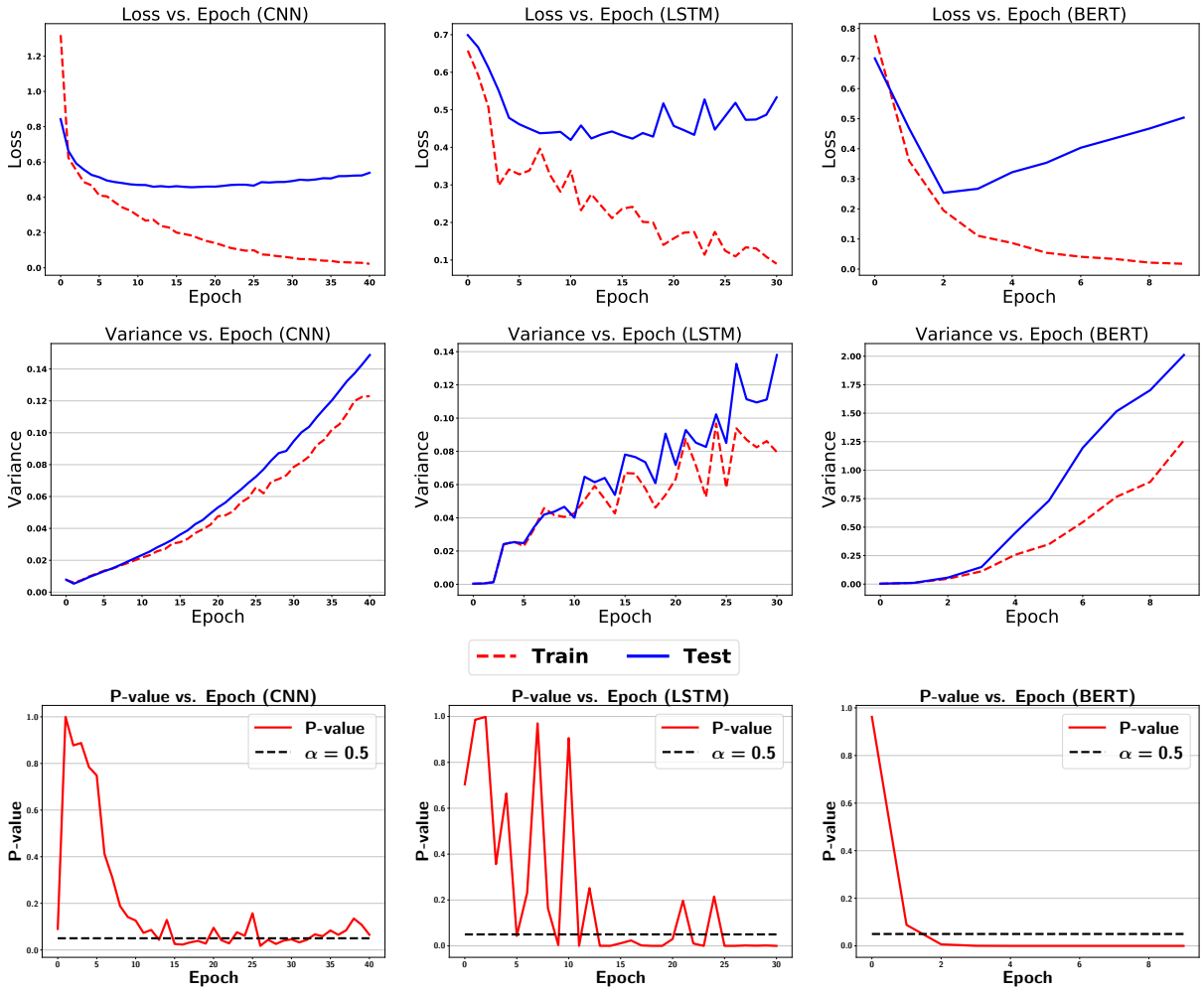


Figure 4.4: The three figures in Line 1 plot training and test loss of CNN, LSTM, BERT respectively. The figures in Line 2 plot the corresponding average variance of interaction scores across instances over training and test sets. The figures in Line 3 show p-values of permutation tests of $50,000$ iterations with 300 randomly selected instances in training and test sets respectively.

true on average across instances for a overfitted model. In particular, interaction scores of test instances have a larger variance on average than those of training instances when the model is overfitted, but comparable otherwise. The observation can also be generalized to other types of neural networks, including CNN and LSTM. We show in Figure 4.4 the average variance on training and test sets for CNN, LSTM and BERT models against training epochs, together with the loss curves. We observe that overfitting occurs when the variances between training and test sets differ.

The observation suggests we may use the difference of average variances of interaction scores between training and test sets as a diagnostic for overfitting. In particular, a per-

mutation test can be carried out under the null hypothesis of equal average variance. The resulting p-values are plotted against the number of training epochs in the third line of Figure 4.4. It can be observed that p-values fall below the significance level of 0.05 when overfitting occurs, which suggests the rejection of the null hypothesis as an early stopping criterion in training.

4.6 Discussion

We have proposed the LS-Tree value as a fundamental quantity for interpreting NLP models. This value leverages a constituency-based parser so that syntactic structure can play a role in determining interpretations. We have also presented an algorithm based on the LS-Tree value for detecting interactions between siblings of a parse tree. To the best of our knowledge, this is the first model-interpretation algorithm to quantify the interaction between words for arbitrary NLP models. We have applied the proposed algorithm to the problem of assessing the nonlinearity of common neural network models and the effect of adversative relations on the models. We have presented a permutation test based on the LS-Tree interaction scores as a diagnostic for overfitting.

Part II

Adversarial Robustness

Chapter 5

Greedy Attack and Gumbel Attack: Generating Adversarial Examples for Discrete Data

We present a probabilistic framework for studying adversarial attacks on discrete data. Based on this framework, we derive a perturbation-based method, *Greedy Attack*, and a scalable learning-based method, *Gumbel Attack*, that illustrate various tradeoffs in the design of attacks. We demonstrate the effectiveness of these methods using both quantitative metrics and human evaluation on various state-of-the-art models for text classification, including a word-based CNN, a character-based CNN and an LSTM. As an example of our results, we show that the accuracy of character-based convolutional networks drops to the level of random selection by modifying only five characters through Greedy Attack.

5.1 Introduction

Robustness to adversarial perturbation has become an extremely important criterion for applications of machine learning in security-sensitive domains such as spam detection [116], fraud detection [117], criminal justice [118], malware detection [119], and financial markets [120]. Although it is not surprising that some small perturbations can change the prediction of an ML model, it is nontrivial to find those perturbations. For instance, random sampling usually cannot find those adversarial examples. Therefore, systematic methods for generating adversarial examples by small perturbations of original input data, also known as "attack," have been developed to operationalize this criterion and to drive the development of more robust learning systems [25, 26, 121].

Most of the work in this area has focused on differentiable models with continuous input spaces [25–27]. In this setting, the proposed attack strategies add a gradient-based perturbation to the original input, resulting in a dramatic decrease in the predictive accuracy of the model. This finding demonstrates the vulnerability of deep neural networks to adversarial

Attack Methods	Training	Efficiency	Success rate	Black-box
Saliency [8]	No	High	Medium	No
Projected FGSM [122]	No	High	Low	No
Delete 1-score [18]	No	Low	High	Yes
DeepWordBug [123]	No	Low	Medium	Yes
Greedy Attack	No	Low	Highest	Yes
Gumbel Attack	Yes	High	Medium	Yes

Table 5.1: Methods comparisons. "Efficiency": computational time and model evaluation times. "Black-box": applicability to black-box models. See Section 5.4 for details.

examples in tasks like image classification and speech recognition.

We focus instead on adversarial attacks on models with discrete input data, such as text data, where each feature of an input sample has a categorical domain. For simplicity, we will limit ourselves to untargeted attack in this chapter. While gradient-based approaches are not directly applicable to this setting, variations of gradient-based approaches have been shown effective in differentiable models. For example, Li et al. [104] proposed to locate the top features with the largest gradient magnitude of their embedding, and Papernot et al. [122] proposed to modify randomly selected features of an input through perturbing each feature by signs of the gradient, and project them onto the closest vector in the embedding space. Dalvi et al. [121] attacked such models by solving a mixed integer linear program. Gao et al. [123] developed scoring functions applicable for sequence data, and proposed to modify characters of the features selected by the scoring functions. Attack methods specifically designed for text data have also been studied recently. Jia and Liang [124] proposed to insert distraction sentences into samples in a human-involved loop to fool a reading comprehension system. Samanta and Mehta [125] added linguistic constraints over the pool of candidate-replacing words. [126] applied a gradient-based technique to attack sequence-to-sequence models.

We propose a systematic probabilistic framework for generating adversarial examples for models with discrete input. The framework is a two-stage process, where the key features to be perturbed are identified in the first stage and are then perturbed in the second stage by values chosen from a dictionary. We present two instantiations of this framework—*Greedy Attack* and *Gumbel Attack*. Greedy Attack evaluates models with single-feature perturbed inputs in two stages, while Gumbel Attack learns a parametric sampling distribution for perturbation. Greedy Attack achieves higher success rate, while Gumbel Attack requires fewer model evaluations, leading to better efficiency in real-time or large-scale attacks. Both attacks assume a score-based threat model. Table 5.1 systematically compares our methods with other methods.

In summary, our contributions in this work are as follows:

- We propose a probabilistic framework for adversarial attacks on models with discrete data.
- We show that Greedy Attack achieves state-of-the-art attack success rates across var-

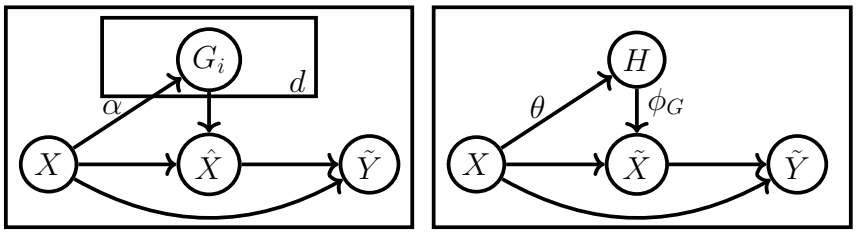


Figure 5.1: The left and right figures show the graphical models of the first and second stage respectively. Parameters α, θ and ϕ_G are specific to Gumbel Attack (Algorithm 4).

ious kinds of models.

- We propose Gumbel Attack as a scalable method with low model-evaluation complexity.
- We observe that character-based models in text classification are particularly vulnerable to adversarial attack.

5.2 Framework

We assume a model in the form of a conditional distribution, $\mathbb{P}_m(Y \mid x)$, for a response Y, supported on a set $\mathcal{Y}$, given a realization of an input random variable $X = x \in \mathbb{W}^d$, where $\mathbb{W} := \{w_0, w_1, \ldots, w_m\}$ is a discrete space such as the dictionary of words or the space of characters. We assume there exists $w_0 \in \mathbb{W}$ that can be taken as a reference point with no contribution to classification. For example, w_0 can be the zero padding in text classification. Let $\tilde{x}$ denote a perturbation of the input variable x. The goal of the adversarial attack is to turn a given input x into $\tilde{x}$ through small perturbations, in such a way that $\tilde{Y} = 1$ given $\tilde{x}$, where $\tilde{Y}$ is the indicator of a successful attack: $\tilde{Y} \mid \tilde{x}, x := \mathbf{1}\{\arg\max_y \mathbb{P}_m(y \mid \tilde{x}) \neq \arg\max_y \mathbb{P}_m(y \mid x)\}$. We restrict the perturbations to k features of x, and approach the problem through two stages. In the first stage, we search for the most important k features of x. In the second stage, we search for values to replace the selected k features:

$$\text{First stage: } \hat{x} = \arg \max_{a \in S_1(x,k)} \mathbb{P}(\tilde{Y} = 1 | a, x), \tag{5.1}$$

$$\text{Second stage: } \tilde{x} = \arg \max_{a \in S_2(\hat{x},x)} \mathbb{P}(\tilde{Y} = 1 | a, x), \tag{5.2}$$

where $S_1(x, k) := \{a \in \mathbb{W}^d \mid a_i \in \{x_i, w_0\} \text{ for all } i, d(a, x) \leq k\}$ is a set containing all the elements that differ from x by at most k positions, with the different features always taking value w_0, and $S_2(\hat{x}, x) := \{a \in \mathbb{W}^d \mid a_i = \hat{x}_i \text{ if } \hat{x}_i = x_i; a_i \in \mathbb{W}' \text{ otherwise}\}$. Here, we denote by $x_i, a_i, \hat{x}_i$ the ith feature of $x, a, \hat{x}$, by $d(a, x)$ the count of features different between a and x, and by $\mathbb{W}' \subseteq \mathbb{W}$ a sub-dictionary of $\mathbb{W}$ chosen by the attacker.

These two objectives are computationally intractable in general. We thus further propose a probabilistic framework to reformulate the objectives into a more tractable objective, as

shown in Figure 5.1. Let G be a random variable in $D_k^d := \{z \in \{0,1\}^d : \sum z_i \leq k\}$, the space of d-dimensional zero-one vectors with at most k ones, and let $\phi : \mathbb{W}^d \times D_k^d \to \mathbb{W}^d$ be a function such that $\phi(x,g)_i = x_i$ if $g_i = 0$ and $\phi(x,g)_i = w_0$ if $g_i = 1$. In the first stage, we let $\hat{X} = \phi(X, G)$ where G is generated from a distribution conditioned on X. We further add a constraint on $\mathbb{P}(G|X)$, by defining k identical random one-hot random variables $G^1, G^2, \ldots, G^k \in D_1^d$ conditioned on X, and letting $G_i := \max_s\{G_i^s\}$, with G_i and G_i^s being the ith entries of the variables G and G^s respectively. We aim to maximize the objective $\mathbb{P}(\tilde{Y} = 1 \mid \hat{X}, X)$ over the distribution of G given X, the probability of successful attack obtained by merely masking features:

$$\max_{\mathbb{P}(G|x)} \mathbb{E}_X[\mathbb{P}(\tilde{Y} = 1 \mid \hat{X}, X)], \text{ s.t. } \hat{X} = \phi(X, G), G^s \overset{i.i.d.}{\sim} \mathbb{P}(\cdot \mid X), \ \tilde{Y} \sim \mathbb{P}(\tilde{Y} \mid \hat{X}, X). \quad (5.3)$$

The categorical distribution $\mathbb{P}(G^s \mid x)$ yields a rank over the d features for a given x. We define $\phi^G : \mathbb{W}^d \to \mathcal{P}_k([d])$ to be the deterministic function that maps an input x to the indices of the top k features based on the rank from $\mathbb{P}(G^s \mid x)$: $\phi^G(x) = \{i_1, \ldots, i_k\}$.

In the second stage, we introduce a new random variable $H = (H^1, \ldots, H^d)$ with each H^i being a one-hot random variable in $D_1^{|\mathbb{W}'|} := \{z \in \{0,1\}^{|\mathbb{W}'|} : \sum z_i = 1\}$. Let $\mathcal{P}_k([d])$ be the set of subsets of $[d]$ of size k. Let $\psi : \mathbb{W}^d \times (D_1^{|\mathbb{W}'|})^d \times \mathcal{P}_k([d]) \to \mathbb{W}^d$ be a function such that $\psi(x, h, \phi^G(x))_i$ is defined to be x_i if $i \notin \phi^G(x)$, and is the value in $\mathbb{W}'$ corresponding to the one-hot vector h_i otherwise. The perturbed input is $\tilde{X} := \psi(X, H, \phi^G(X))$, where H is generated from a distribution conditioned on X. We add a constraint on $\mathbb{P}(H \mid X)$ by requiring $H^1, \ldots, H^d$ to be independent of each other conditioned on X. Our goal is to maximize the objective $\mathbb{P}(\tilde{Y} = 1 \mid \tilde{X}, X)$ over the distribution of H given X:

$$\max_{\mathbb{P}(H|x)} \mathbb{E}_{X,G}[\mathbb{P}(\tilde{Y} = 1|\tilde{X}, X)], \text{ s.t. } \tilde{X} = \psi(X, H, \phi^G(X)), H \sim \mathbb{P}(\cdot|X), \ Y \sim \mathbb{P}(\tilde{Y}|\tilde{X}, X). \quad (5.4)$$

For a given input x, the categorical distribution $\mathbb{P}(H^i \mid x)$ yields a rank over the values in $\mathbb{W}'$ to be chosen for each feature i. The perturbation on x is carried out on the top k features $\phi^G(x) = \{i_1, \ldots, i_k\}$ ranked by $\mathbb{P}(G^s \mid x)$; each chosen feature i_s is assigned the top value in $\mathbb{W}'$ selected by $\mathbb{P}(H^{i_s} \mid x)$.

5.3 Methods

In this section we present two instantiations of our general framework: *Greedy Attack* and *Gumbel Attack*.

5.3.1 Greedy Attack

We motivate Algorithm 3, Greedy attack, as optimizing the lower bounds of Problem (5.3) and Problem (5.4). To solve Problem (5.3), we decompose the objective conditioned on a

single instance x as:

$$\mathbb{E}_{G|X}[\mathbb{P}(\tilde{Y}=1 \mid \hat{X}, X) \mid x] = \sum_{i=1}^{d} \mathbb{P}(G^1 = e_i \mid x)\mathbb{E}_{G^{(1)}|X,G^1}[\mathbb{P}(\tilde{Y}=1 \mid \hat{X}, X) \mid x, e_i],$$

where e_i denote the d-dimensional one-hot vector whose ith component is 1, and $G^{(1)} := (G^2, \ldots, G^k)$. We claim the objective in Problem 5.3 conditioned on a single instance x can be lower bounded as

$$\begin{aligned}
&\max_{\mathbb{P}(G|x)} \mathbb{E}_{G|X}[\mathbb{P}(\tilde{Y}=1 \mid \hat{X}, X) \mid x] \\
\geq &\max_{\mathbb{P}(G^1|x)} \sum_{i=1}^{d} \Big(\mathbb{P}(G^1 = e_i \mid x) \max_{\mathbb{P}(G^{(1)}|x,e_i)} \mathbb{E}[\mathbb{P}(\tilde{Y}=1 \mid \hat{X}, X) \mid x, e_i] \Big) \\
\geq &\max_{\mathbb{P}(G^1|x)} \sum_{i=1}^{d} \mathbb{P}(G^1 = e_i \mid x)\mathbb{P}(\tilde{Y}=1 \mid x_{(i)}). \qquad (5.5)
\end{aligned}$$

In fact, let $\vee$ denote the elementwise maximum of k random vectors. We have

$$\begin{aligned}
\max_{\mathbb{P}(G^{(1)}|x,e_i)} \mathbb{E}[\mathbb{P}(\tilde{Y}=1 \mid \hat{X}, X) \mid x, e_i] &= \max_{\mathbb{P}(G^{(1)}|x,e_i)} \mathbb{E}[\mathbb{P}(\tilde{Y}=1 \mid \phi(X, \vee\{e_i, G^{(1)}\}), X) \mid x, e_i] \\
&\geq \mathbb{P}(\tilde{Y}=1 \mid \phi(x, \vee\{e_i, G^2 = e_i, \ldots, G^k = e_i\})) \\
&= \mathbb{P}(\tilde{Y}=1 \mid x_{(i)}).
\end{aligned}$$

where the first equality follows from the definition of $\hat{X}$, and the inequality follows from degenerating $\mathbb{P}(G^{(1)}|x, e_i)$. The lower bound (5.5) is maximized when

$$\mathbb{P}(G^1 = e_i \mid x) \propto \mathbb{P}(\tilde{Y}=1 \mid x_{(i)}). \qquad (5.6)$$

Similarly, we decompose the objective in Problem (5.4) by conditioning on H^{i_1} and invoking the independence between G and H conditioning on X. Using a similar argument, we arrive at

$$\begin{aligned}
&\max_{\mathbb{P}(H|x,g)} \mathbb{E}_{H|X,G}[\mathbb{P}(\tilde{Y}=1 \mid \tilde{X}, X) \mid x, g] \\
= &\max_{\mathbb{P}(H^{i_1}|x,g)} \sum_{j=1}^{|\mathbb{W}'|} \mathbb{P}(H^{i_1} = e_j \mid x, g) \max_{\mathbb{P}(H^{(i_1)}|x,e_j)} \mathbb{E}_{H^{(i_1)}|X,G,H^{i_1}}[\mathbb{P}(\tilde{Y}=1 \mid \tilde{X}, X) \mid x, e_i] \\
\geq &\max_{\mathbb{P}(H^{i_1}|x,g)} \sum_{j=1}^{|\mathbb{W}'|} \mathbb{P}(H^{i_1} = e_j \mid x, g)\mathbb{P}(\tilde{Y}=1|x_{(i_1 \to w_j)}). \qquad (5.7)
\end{aligned}$$

The lower bound (5.7) is maximized when

$$\mathbb{P}(H^{i_1} = e_j \mid x, g) \propto \mathbb{P}(\tilde{Y}=1 \mid x_{(i_1 \to w_j)}). \qquad (5.8)$$

The same applies to $i_2, \ldots, i_k$. The algorithm Greedy Attack is built up from Equation (5.6) and Equation (5.8) in a straightforward manner. See Algorithm 3 for details.

While Greedy Attack is proposed in its most generic form, it is flexible to incorporate

linguistic coherence for natural language tasks. GloVe [110] is a widely used unsupervised learning algorithm to obtain vector representation of words. The Euclidean distance between two vector embeddings provides an effective method for measuring the semantic similarity of the corresponding words. Throughout experiments, we restrict the candidate pool in the second stage of Greedy Attack to words close to the original word in terms of the Euclidean distance of the corresponding vector embeddings, so as to keep the semantic meaning of the entire sentence.

Algorithm 3 Greedy Attack

Input: Model $\mathbb{P}_m(Y \mid x)$.
Input: Sample $x \in \mathbb{W}^d$.
Input: k, number of features to change.
Input: $\mathbb{W}'$, sub-dictionary.
Output: Modified x.
for $i = 1$ **to** d **do**
  Compute $\mathbb{P}(\tilde{Y}|x_{(i)})$.
end for
$i_1, \ldots, i_k = \mathrm{Top}_k(\mathbb{P}(\tilde{Y}|x_{(i)})_{i=1}^d)$.
for $s = 1$ **to** k **do**
  $x_{i_s} \leftarrow \arg\max_{w \in \mathbb{W}'} \mathbb{P}(\tilde{Y}|x_{(i_s \to w)})$.
end for

Algorithm 4 Gumbel Attack

Input: Model $\mathbb{P}_m(Y \mid x)$.
Input: k, number of features to change.
Input: A data set $\mathcal{D} = \{x_i\}$ (for training).
Input: A data set $\mathcal{D}'$ to be attacked.
Input: $\mathbb{W}'$, sub-dictionary.
Output: Modified data set $\tilde{\mathcal{D}}'$.
Train $\mathbb{P}_\alpha(G|X)$ on $\mathcal{D}$.
Train $\mathbb{P}_\theta(H|X)$ on $\mathcal{D}$ given $\mathbb{P}_\alpha(G|X)$.
for x **in** $\mathcal{D}'$ **do**
  $i_1, \ldots, i_k = \mathrm{Top}_k(\mathbb{P}_\alpha(G|x))$
  for $s = 1$ to k **do**
    $x_{i_s} \leftarrow \arg\max_{w \in \mathbb{W}'} \mathbb{P}_\alpha(H^{i_s}|g, x)$
  end for
  Add the modified x to $\tilde{\mathcal{D}}'$.
end for

5.3.2 Gumbel Attack

Algorithm 3 evaluates the original model $O(d + k \cdot |\mathbb{W}'|)$ times for each sample. In the setting where one would like to carry out the attack over a massive data set $\mathcal{D}'$, Greedy Attack can be infeasible due to the high cost of model evaluations. Assuming that the original model is differentiable and each sample in $\mathcal{D}'$ is generated from a common underlying distribution, an alternative approach to solve Problem (5.3) and Problem (5.4) is to parametrize $\mathbb{P}(G \mid x)$ and $\mathbb{P}(H \mid x)$ and optimize the objectives over the parametric family directly on a training data set from the same distribution before the adversarial attack. An outline of this approach is described in Algorithm 4. We describe the training process in detail below.

In the presence of k categorical random variables in Equation (5.3) and Equation (5.4), direct model evaluation requires summing over d^k terms and $|\mathbb{W}'|^k$ terms respectively. A straightforward approximation scheme is to exploit Equations (5.5) and (5.7), where we

assume the distribution of hidden nodes G and H is well approximated by greedy methods. Nonetheless, this still requires $d + |\mathbb{W}'|^k$ model evaluations for each training sample. Several approximation techniques exist to further reduce the computational burden; e.g., one can take a weighted sum of features parametrized by deterministic functions of X, similar to the soft-attention mechanism [1, 2, 92], and REINFORCE-type algorithms [91]. We instead propose a method based on the "Gumbel trick" [93, 95], combined with the approximation of the objective proposed in Greedy Attack on a small subset of the training data. This achieves better performance with lower variance and higher model evaluation efficiency in our experiments.

The Gumbel trick involves using a Concrete random variable, introduced as a differentiable approximation of a categorical random variable, which has categorical probability $p_1, p_2, \ldots, p_d$ and is encoded as a one-hot vector in $\mathbb{R}^d$. The Concrete random variable C, denoted by $C \sim \text{Concrete}(p_1, p_2, \ldots, p_d)$, is a random vector supported on the relaxed simplex $\Delta_d := \{z \in [0,1]^d : \sum_i z_i = 1\}$, such that $C_i \propto \exp\{(\log p_i + \varepsilon_i)/\tau\}$, where $\tau > 0$ is the tunable temperature, and $\varepsilon_j := -\log(-\log u_i)$, with u_i generated from a standard uniform distribution, defines a Gumbel random variable.

In the first stage, we parametrize $\mathbb{P}(G^s \mid x)$ by its categorical probability $p_\alpha(x)$, where

$$p_\alpha(x) = ((p_\alpha(x))_1, (p_\alpha(x))_2, \ldots, (p_\alpha(x))_d),$$

and approximate G by a random variable U defined from a collection of Concrete random variables:

$$U = (U_1, \ldots, U_d), U_i = \max_{s=1,\ldots,k} \{C_i^s\},$$

where $C^s \overset{i.i.d.}{\sim} \text{Concrete } (p_\alpha(x))$ for $s = 1, \ldots, k$. We write $U = U(\alpha, x, \varepsilon)$ as it is a function of the parameters α, input x and auxiliary random variables ε. The perturbed input $\hat{X} = \phi(X, G)$ is approximated by

$$\hat{X} \approx U \odot X, \text{ with } (U \odot X)_i := (1 - U_i) \cdot X_i + U_i \cdot w_0,$$

where we identify X_i, w_0 and w_j with their corresponding embeddings for notation convenience.

In the second stage, we parametrize $\mathbb{P}(H \mid x)$ by another family $q_\theta(x) = \{(q_\theta)_{ij}, i = 1, \ldots, d; j = 1, \ldots, |\mathbb{W}'|\}$, and approximate each H^i by a Concrete random variable

$$V^i \sim \text{Concrete}\big((q_\theta)_{i1}, \ldots, (q_\theta)_{i|\mathbb{W}'|}\big).$$

The perturbed input $\tilde{X} = \psi(X, H, \phi^G(x))$ is approximated by replacing the i_s feature with a weighted sum of the embeddings of $w \in \mathbb{W}'$ with entries of V^{i_s} as weights, for each i_s in $\phi^G(x)$:

$$\psi(X, H, \phi^G(X)) \approx V \odot_{\phi^G} X,$$

where

$$(V \odot_{\phi^G} X)_i := \begin{cases} \sum_{w_j \in \mathbb{W}'} V_j^i \cdot w_j \text{ if } i \in \phi^G(X), \\ X_i \text{ otherwise.} \end{cases}$$

Data Set	Classes	Train Samples	Test Samples	Average #w	Model	Parameters	Accuracy
IMDB Review [69]	2	25,000	25,000	325.6	WordCNN	351,002	90.1%
AG's News [72]	4	120,000	7,600	278.6	CharCNN	11,337,988	90.09%
Yahoo! Answers [72]	10	1,400,000	60,000	108.4	LSTM	7,146,166	70.84%

Table 5.2: Summary of data sets and models. "Average #w" is the average number of words per sample. "Accuracy" is the model accuracy on test samples.

The final objectives of Gumbel attack on a data set $\mathcal{D}$ become the following:

$$\max_{\alpha} \quad \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \log f(U(\alpha, x, \varepsilon) \odot x),$$

$$\max_{\theta} \quad \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} \log f(V(\theta, x, \varepsilon) \odot_{\phi^G} x),$$

where we define $f(x) := \mathbb{P}(\tilde{Y} = 1 \mid x)$ for notational convenience. Note that ε is an auxiliary random variable independent of the parameters. In the training stage, we can apply stochastic gradient methods directly to optimize the two objectives, where a mini-batch of unlabelled data and auxiliary random variables are jointly sampled to compute a Monte Carlo estimate of the gradient. In the attack stage, one directly perturbs incoming samples from a massive data set $\mathcal{D}'$ based on the trained samplers $\mathbb{P}_\alpha(G|X)$ and $\mathbb{P}_\theta(H|X)$, with no cost on model evaluation. A high-level sketch of the two-stage Gumbel attack is shown in Algorithm 4.

5.4 Experiments

We evaluate the performance of our algorithms in attacking three text classification models, including a convolutional neural network (CNN) and a Long Short-Term Memory (LSTM) network. See Table 5.2 for a summary of the data and models used, and supplementary material for model details. During the adversarial attack, inputs are perturbed at their respective feature levels, and words and characters are units for perturbation for word and character-based models respectively. We compare Greedy Attack and Gumbel Attack with the following methods:

- **Delete-1 Score** [18]: Mask each feature with zero padding, use the decrease in the predicted probability as the score of the feature, and mask the top-k features as unknown.
- **DeepWordBug** [123]: For each feature, compute a linear combination of two scores, with the first score evaluating a feature based on its preceding features, and the second based on its following features. Weights are selected by the user.
- **Projected FGSM** [26, 122]: Perturb a randomly selected subset of k features by replacing the original word w with a w' in the dictionary such that $\|\text{sgn}(\text{emb}(w') - \text{emb}(w)) - \text{sgn}(\nabla f)\|$ is minimized, where $\text{emb}(w)$ is the embedding of w, and ∇f is the gradient of the predicted probability with respect to the original embedding.

- **Saliency** [8, 127]: Select the top k features by the gradient magnitude, defined as the l_1 norm of the gradient with respect to the features' embeddings, and mask them as unknown.
- **Saliency-FGSM**: Select the top k features based on the Saliency map, and replace each of them using projected FGSM.

5.4.1 Word-based models

We use two word-based models: a word-based CNN network [76] and a word-based LSTM network [77]:

- **IMDB with a word-CNN**: We use the Large Movie Review Dataset (IMDB) for sentiment classification [69]. It contains $50,000$ binary labeled movie reviews, with a split of $25,000$ for training and $25,000$ for testing. The word-based CNN model consists of a 50-dimensional word embedding, a 1-D convolutional layer of 250 filters and kernel size 3, a max-pooling and a 250-dimensional dense layer as hidden layers. Both the convolutional and the dense layers are followed by ReLU as nonlinearity, and Dropout [78] as regularization. The model is trained with rmsprop [79] for five epochs. Each review is padded/cut to 400 words. The model achieves accuracy of 90.1% on the test data set.
- **Yahoo! Answers with an LSTM**: We use the ten-category corpus Yahoo! Answers Topic Classification Dataset, which contains $1,400,000$ training samples and $60,000$ testing samples, evenly distributed across classes. Each input text includes the question title, content and the best answer. An LSTM network is used to classify the texts. The network consists of a 300-dimensional randomly-initialized word embedding, a bidirectional LSTM, each with dimension 256, and a dropout layer as hidden layers. The model is trained with rmsprop [79]. It achieved an accuracy of 70.84% on the test data set, close to the state-of-the-art accuracy of 71.2% obtained by character-based CNN [72].

For greedy attack, we try to improve its linguistic coherence by using a candidate pool of similar words for the replacement in the second stage. We first calculate the distribution of Euclidean distance between the corresponding GloVe embeddings for each pair of words. Under the assumption that a closer distance between embeddings implies closer semantic meaning, we limit the candidate pool in the second stage of Greedy Attack to words whose distance to the original word is within 5% quantile of the distribution.

For Gumbel Attack, we use the 500 words with the highest frequencies as the dictionary $\mathbb{W}'$ of replacing words. We parametrize the identifier $p_\alpha(x)$ and perturber $q_\theta(x)$ with the model structure plotted in Figure 5.2, which consists of a local information component and a global information component. The input is initially fed into a common embedding layer and a 100-filter convolutional layer. Then the local component processes the common output through two 50-filter convolutional layers with, and the global component processes

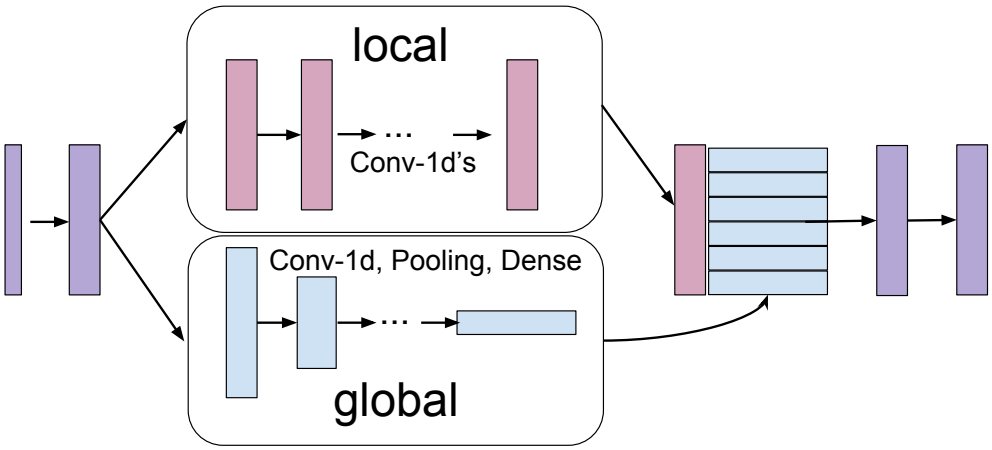


Figure 5.2: Model structure of Gumbel Attack. The same structure is used across three data sets. The input is fed into a common embedding followed by a conv layer. Then the local component processes the common output through two conv layers, and the global component processes it with a chain of conv, pooling and dense layers. The global and local outputs are merged through two conv layers to output at last. See supplementary material for details.

the common output through a max-pooling layer followed by a 100-dimensional dense layer. Then we concatenate the global output to local outputs corresponding to each feature, and process them through a 50-filter convolutional layer, followed by a Dropout layer and a convolutional network with kernel size 1 to output. All previous convolutional layers are of kernel size 3, and ReLU is used as nonlinearity. The identifier and the perturber are trained separately on the training data, by rmsprop [79] with step size 0.001.

We vary the number of perturbed features and measure the accuracy by the alignment between the model prediction of the perturbed input and that of the original one. The same metric was used [123, 125]. The success rate of attack can be defined as the inconsistency with the original model: $1-$ accuracy.

The average accuracy over test samples is shown in Figure 5.3. Greedy Attack performs the best among all methods across both word-based models. Gumbel Attack performs well on IMDB with Word-CNN but has lower success rate than Saliency-Projected FGSM on Yahoo! Answers with LSTM. Examples of successful attacks are shown in Table 5.3.

5.4.2 Character-based models

We carry out experiments on the AG's News corpus with a character-based CNN [72]. The AG's News corpus consists of titles and description fields of $196,000$ news articles from $2,000$ news sources [72]. It is categorized into four classes, each containing $30,000$ training samples and $1,900$ testing samples. The character-based CNN has the same structure as the one proposed in Zhang, Zhao, and LeCun [72]. It consists of six convolutional layers, three max pooling layers, and two dense layers. The alphabet dictionary used is of size 69. The model

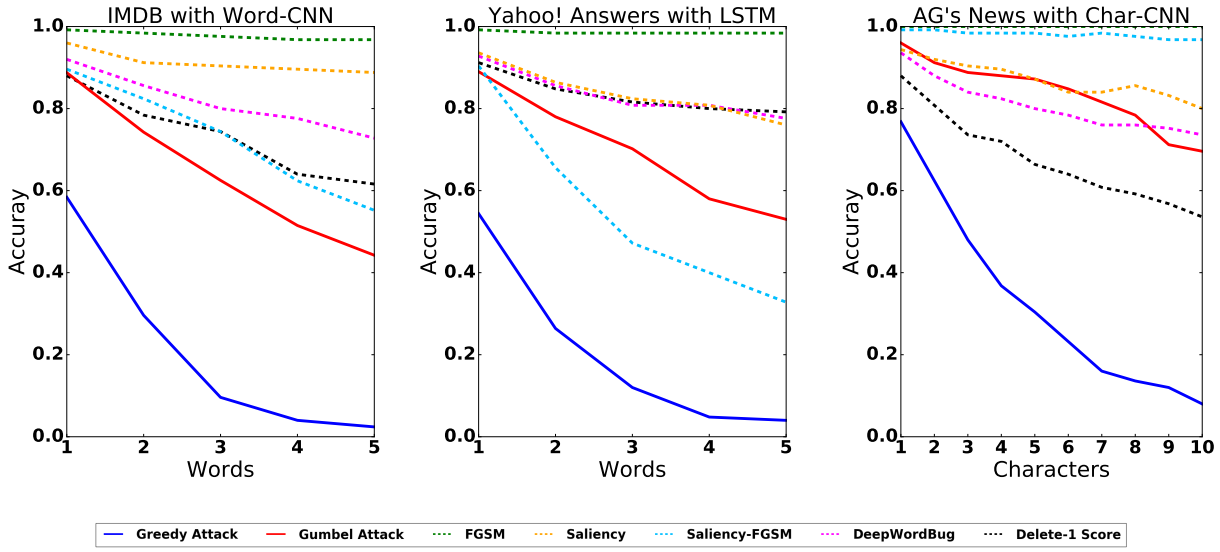


Figure 5.3: The drop in accuracy as the number of perturbed features increases on three data sets.

is trained with SGD with decreasing step size initialized at 0.01 and momentum 0.9. (Details can be found in Zhang, Zhao, and LeCun [72].) The model reaches accuracy of 90.09% on the test data set.

For Greedy Attack, the dictionary for the replacing character $\mathbb{W}'$ is chosen to be the entire alphabet. For Gumbel Attack, the model structure and training are exactly the same as those for word-based models.

Figure 5.3 shows how the alignment of model prediction, given the original data and the perturbed data, changes with the number of characters perturbed by various methods. Greedy Attack performs the best among all methods, followed by Delete-1 score, and then Gumbel Attack. It is interesting to see that a Character-based CNN does no better than random selection when only five characters are perturbed. Examples of successful attacks are shown in Table 5.3.

5.4.3 Efficiency

The efficiency of generating adversarial examples becomes an important factor for large-scale data. We evaluate the clock-time efficiency of various methods. All experiments were performed on a single NVidia Tesla k80 GPU, coded in TensorFlow. Figure 5.4 shows the average clock time for perturbing one sample for various methods. Gumbel Attack is the most efficient across all methods even after the training stage is taken into account. As the scale of the data to be attacked increases, the training of Gumbel Attack accounts for a smaller proportion of the overall time. Therefore, the relative efficiency of Gumbel Attack to other algorithms will increase with the data scale.

Data Set	Class	New Class	Perturbed Texts
IMDB	Negative	Positive	I have read each and every one of Baroness Orczys Scarlet Pimpernel books. Counting this one, I have seen 3 pimpernel movies. The one with Jane Seymour and Anthony Andrews i preferred greatly to this. It goes out of its way for violence and action, occasionally completely violating the spirit of the book. I dont expect movies to stick directly to plots, i gave up being that idealistic long ago, but if an good (excellent) movie of a book has already been made, dont remake it with a tv movie that includes excellent actors and nice costumes, but a barely decent script. Sticking with the 80s version....Rahne
	Positive	Negative	Begotten is black and white distorted images. It looks like it could have come from the nineteenth century. However, the sound is crystal clear, minus the sync and the addition of calm nature sounds.This movie was very critical of the struggles of lives (life). It shows a single mother and child in a violent world that thrives on the innocent. The mother is very oblivious to her surroundings. This leads to lots of torture, pain, and death. You may watch it many times and see different symbolisms, plot devices, and basically what does it mean?.If you appreciate art in movies then you will love it. Otherwise, dont bother.
Yahoo!	Entertainment, Music	Sports	what are some really good dave matthews cup (band) songs ants marching n marching though would probably be my favorite or the first one i would recommend
	Family, Relationships	Health	im diet (bored) so whats a good prank so i can do it on my friends go to their house and dump all the shampoo outta the bottle and replace it with yogurt yup i always wanted to do that let me know how it works out haha
AG's News	Sports	Sci & Tech	DEFOE DRIVES SPURS HOMEJermain Defoe underlined his claims for an improved contract as he inspired Tottenham to a 2_0 win against 10_man Middlesbrough. New sx\\\ Martin Jol, who secured his first win in charge, may have been helped
	Sci & Tech	Business	Oracle Moves To Monthly Patch ScheduleAn alert posted on the company's y)c tite outlined the patches that should be posted to fix numerous security holes in a number of aiplications.
	Business	World	Howard Stern moves radio show to SkriusShopk jock Howard Stern announced Wednesday he's taking his radio show off the public airwaves and over to Sirius satihlhte radio.
	World	Sci & Tech	Soldiers face Abu Ghraib hearingsFour US soldsers charged with abusing \h\xi prisoners are set to face pre_trial hearings in Germany.

Table 5.3: Single-word-perturbed examples of Greedy and Gumbel attacks on IMDB (Word-CNN) and Yahoo! Answers (LSTM), where red words are the replacing words and the blue words are the original words; five-character-perturbed examples of Greedy and Gumbel attacks on AG's News (Char-CNN), where replacing characters are colored with red.

5.4.4 Transferability

An intriguing property of adversarial attack is that examples generated for one model may often fool other methods with different structures [25, 26]. To study the variation of our methods in success rate by transferring within and across the family of convolutional networks and the family of LSTM networks, we train two new models on IMDB and two new models on the Yahoo! Answers respectively. For the IMDB data set, we trained another convolutional network CNN2, differing from the original one by adding more dense layers, and an LSTM that is the same as the one used for the Yahoo! Answers data set. For the Yahoo! Answers

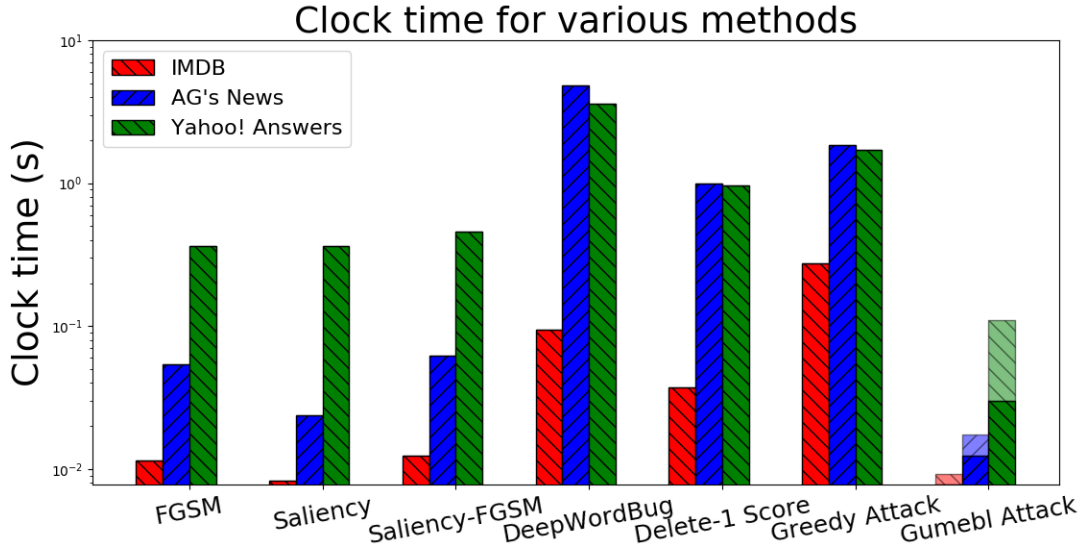


Figure 5.4: The average clock time (on a log scale) of perturbing one input sample for each method. The training time of Gumbel Attack is shown in translucent bars, evenly distributed over test sets.

data set, we train a new LSTM model LSTM2, which is one-directional with 256 memory units, and uses GloVe [110] as a pretrained word embedding. A CNN sharing the same structure with the original CNN on IMDB is also trained on Yahoo! Answers.

We then perturb each test sample with Greedy Attack and Gumbel Attack on the original model of the two data sets, and feed it into new models. The results are shown in Figure 5.5. Greedy Attack achieves comparable success rates for attack on Yahoo! Answers, but suffers a degradation of performance on the IMDB data set. Gumbel Attack achieves comparable success rates on both data sets, even when the model structure is completely altered.

5.4.5 Human evaluation

We address the problem whether small perturbations of adversarial examples in text classification alter human judgment. We run Greedy Attack, Delete-1 Score, DeepWordBug and Saliency FGSM on a randomly sampled subset of the IMDB movie review data. On each instance, we increase the number of words to be perturbed until the prediction of the model changes. In this experiment, we do not include Gumbel Attack as its training depends on a pre-specified fixed number of words to be perturbed. Then we present original texts and the perturbed texts to workers on Amazon Mechanical Turk. Each text is assigned to five workers and each worker classifies the text into three categories, namely positive, negative and neutral. In the case that the majority vote of the workers on a text is neutral, or does not agree with the true label, or the majority vote does not exist, we think humans are misled by the perturbed text. We report accuracy as the average consistency with the truth.

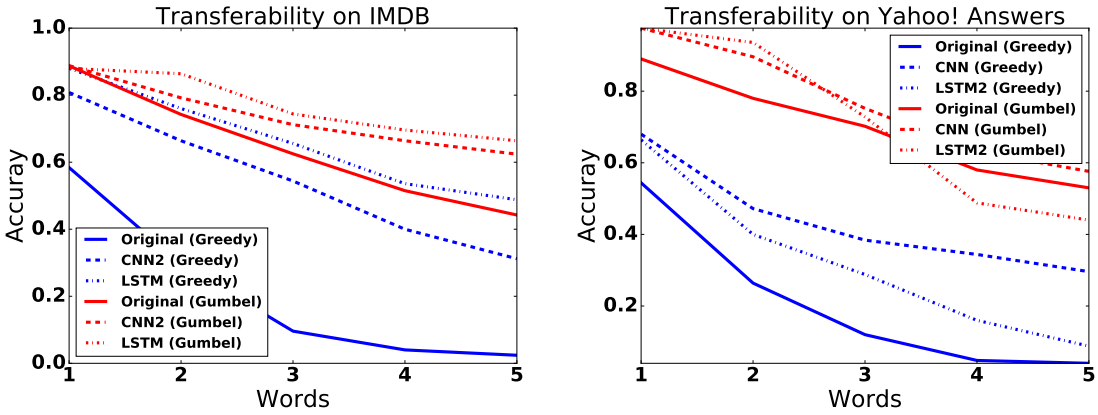


Figure 5.5: Transferability results. Solid lines: accuracy on the original models. Dotted lines: accuracy on the new models.

Algorithm	Human Accuracy	Avg. # of Words Perturbed
Raw	89.0%	0.000
Greedy Attack	**84.4%**	**2.120**
Delete-1 Score	77.1%	18.160
Saliency FGSM	80.7%	9.200
DeepWordBug	81.6%	25.816

Table 5.4: First human evaluation: perturb until success.

The result is reported in Table 5.4. Greedy attack perturbs the least number of words on average. As a result, human is least sensitive to Greedy Attack.

To evaluate the performance of Gumbel Attack, we present the original texts and the perturbed texts on IMDB and AG's News, as generated by Gumbel Attack, to workers on Amazon Mechanical Turk. Gumbel Attack is fixed to perturb two words on IMDB and ten characters on AG's News. The accuracy of neural networks drops by 25% and 46% respectively. For each data set, 200 samples that are successfully attacked by Gumbel Attack are used. On the IMDB movie review data, human accuracy drops from 83.3% on the original samples to 75.8% on adversarial samples from Gumbel Attack. On character-based models, the accuracy of human judgements stays at comparable levels, with 93.3% on the original samples and 91.2% on the perturbed samples

5.5 Discussion

We have proposed a probabilistic framework for generating adversarial examples on discrete data and have created two algorithms to implement it. Greedy Attack improves the state-of-the-art across several widely-used language models, and Gumbel Attack provides a scalable method for real-time generation of adversarial examples. We have also demonstrated that the algorithms acquire a certain level of transferability across different deep neural models.

Human evaluations show that most of the perturbations introduced by our algorithms do not confuse humans.

Chapter 6

HopSkipJumpAttack: A Query-Efficient Decision-Based Attack

The goal of a decision-based adversarial attack on a trained model is to generate adversarial examples based solely on observing output labels returned by the targeted model. We develop HopSkipJumpAttack, a family of algorithms based on a novel estimate of the gradient direction using binary information at the decision boundary. The proposed family includes both untargeted and targeted attacks optimized for ℓ_2 and ℓ_∞ similarity metrics respectively. Theoretical analysis is provided for the proposed algorithms and the gradient direction estimate. Experiments show HopSkipJumpAttack requires significantly fewer model queries than Boundary Attack. It also achieves competitive performance in attacking several widely-used defense mechanisms. Moreover, we show it achieves competitive performance on a wide variety of classifiers in addition to neural networks, including linear models built on top of the Scale Invariant Feature Transform (SIFT) and the Histogram of Oriented Gradients (HOG), tree-based models including gradient boosted trees and random forests, k-nearest neighbors and kernel methods.

6.1 Introduction

In this chapter, we focus on the decision-based threat model introduced in Chapter 1, in which an attacker has access to decisions alone. A widely studied type of the decision-based attack is transfer-based attack. However, transfer-based attack often requires a carefully-designed substitute model, or even access to part of the training data. Moreover, they can be defended against via training on a data set augmented by adversarial examples from multiple static pre-trained models [41]. In recent work, decision-based attacks relying neither on training data nor on the assumption of transferability have been proposed [33, 38, 40], which achieve comparable performance with state-of-the-art white-box attacks such as C&W attack [29]. One limitation, however, is that they require a relatively large number of model queries, rendering it impractical for real-world applications.

It is more realistic to evaluate the vulnerability of a machine learning system under the decision-based attack with a limited budget of model queries. Online image classification platforms often set a limit on the allowed number of queries within a certain time period. For example, the cloud vision API from Google currently allow 1,800 requests per minute. Query inefficiency thus leads to clock-time inefficiency and prevents an attacker from carrying out large-scale attacks. A system may also be set to recognize the behavior of feeding a large number of similar queries within a small amount of time as a fraud, which will automatically filter out query-inefficient decision-based attacks. Last but not least, a smaller query budget directly implies less cost in evaluation and research. Query-efficient algorithms help save the cost of evaluating the robustness of public platforms, which incur a cost for each query made by the attacker. It also helps facilitate research in adversarial vulnerability, as such a decision-based attack which does not require access to model details may be used as a simple and efficient first step in evaluating new defense mechanisms, as we will see in Section 6.6.2.

Another advantage of decision-based attack is its applicability to other models besides neural networks. While adversarial examples for neural networks are attracting growing attention, relatively fewer efforts are put into the evaluation of robustness for other image classifiers, such as linear models built on top of image feature extractors including the Scale Invariant Feature Transform (SIFT) [128] and the Histogram of Oriented Gradients (HOG) [129], tree-based models including decision trees, random forests, and gradient boosted trees, k-nearest neighbors (KNN) and kernel methods. The main obstacle towards attacking these models is that most of these models, except kernel methods, are not differentiable with respect to the input, which makes it impossible to directly apply existing white-box methods designed for neural networks.

In this chapter, we study decision-based attacks for continuous inputs under an optimization framework, and propose a novel family of algorithms for generating both targeted and untargeted adversarial examples that are optimized for minimum distance with respect to either the ℓ_2-distance or ℓ_∞ distance. The family of algorithms are based on a novel asymptotically unbiased estimate of gradient direction at the decision boundary, which solely relies on access to model decisions. The error of the estiamte at the boundary is characterized with a deviation bound. We also discuss ways to control the error when the estimate is used with deviation from the boundary. The family of algorithms is iterative in nature, with each iteration involving three steps: estimation of the gradient direction, step-size search via geometric progression, and boundary search via a binary search. We refer to the algorithm as HopSkipJumpAttack[1]. We establish the convergence of our algorithm under certain mild conditions.

In the experimental section, we demonstrate the superior efficiency of our algorithm over several state-of-the-art decision-based attacks through extensive experiments. Through the evaluation of several defense mechanisms such as defensive distillation, region-based

[1]A hop, skip, and a jump originally referred to an exercise or game involving these movements dating from the early 1700s, but by the mid-1800s it was also being used figuratively for the short distance so covered.

classification, and adversarial training, we suggest our attack can be used as a simple and efficient first step for researchers to evaluate new defense mechanisms. Finally, we show our algorithm works effectively on a wide range of models besides neural networks, without the trouble of accessing model details or adjusting hyper-parameters. The experimental results suggest several widely used classifiers, such as linear models based on SIFT and HOG, tree-based models and kernel methods, can be vulnerable to decision-based attacks.

6.2 Related work

6.2.1 Decision-based attacks

Most related to our work is the Boundary Attack method introduced by Brendel, Rauber, and Bethge [38]. Boundary Attack is an iterative algorithm based on rejective sampling, initialized at an image that lies in the target class. At each step, a perturbation is sampled from a proposal distribution, which reduces the distance of the perturbed image towards the original input. If the perturbed image still lies in the target class, the perturbation is kept. Otherwise, the perturbation is dropped. Boundary Attack achieves performance comparable to state-of-the-art white-box attacks on deep neural networks for image classification. The key obstacle to its practical application is, however, the demand for a large number of model queries. In practice, the required number of model queries for crafting an adversarial example directly determines the level of the threat imposed by a decision-based attack. One source of inefficiency in Boundary Attack is the rejection of perturbations which deviate from the target class. In our algorithm, the perturbations are used for estimation of a gradient direction.

Several other decision-based attacks have been proposed to improve efficiency. Brunner et al. [39] introduced Biased Boundary Attack, which biases the sampling procedure by combining low-frequency random noise with the gradient from a substitute model. Biased Boundary Attack is able to significantly reduce the number of model queries. However, it relies on the transferability between the substitute model and the target model, as with other transfer-based attacks. Our algorithm does not rely on the additional assumption of transferability. Instead, it achieves a significant improvement over Boundary Attack through the exploitation of discarded information into the gradient-direction estimation. Ilyas et al. [33] proposed Limited attack in the label-only setting, which directly performs projected gradient descent by estimating gradients based on novel proxy scores. Cheng et al. [40] introduced Opt attack, which transforms the original problem to a continuous version, and solves the new problem via randomized zeroth-order gradient update. Our algorithm approaches the original problem directly via a novel gradient-direction estimate, leading to improved query efficiency over both Limited Attack and Opt Attack. The majority of model queries in HopSkipJumpAttack come in mini-batches, which also leads to improved clock-time efficiency over Boundary Attack.

6.2.2 Zeroth-order optimization

Zeroth-order optimization refers to the problem of optimizing a function f based only on access to function values $f(x)$, as opposed to gradient values $\nabla f(x)$. Such problems have been extensively studied in the convex optimization and bandit literatures. Flaxman, Kalai, and McMahan [130] studied one-point randomized estimate of gradient for bandit convex optimization. Agarwal et al. [131] and Nesterov and Spokoiny [132] demonstrated that faster convergence can be achieved by using two function evaluations for estimating the gradient. Duchi et al. [133] established optimal rates of convex zeroth-order optimization via mirror descent with two-point gradient estimates. Zeroth-order algorithms have been applied to the generation of adversarial examples under the score-based threat model [32–34]. Subsequent work [134] proposed and analyzed an algorithm based on variance-reduced stochastic gradient estimates.

We formulate decision-based attack as an optimization problem. A core component of our proposed algorithm is a gradient-direction estimate, the design of which is motivated by zeroth-order optimization. However, the problem of decision-based attack is more challenging than zeroth-order optimization, essentially because we only have binary information from output labels of the target model, rather than function values.

6.2.3 Robustness of image classifiers besides neural networks

While the robustness of neural networks has been widely studied, relatively less attention is paid to other machine learning models. Most of the existing work focuses on a specific class of models and requires access to model details. Papernot, McDaniel, and Goodfellow [36] proposed to attack decision trees and KNNs by greedy search over the tree structure and attacking a smoothed variant of nearest neighbor classifiers respectively. Kantchelian, Tygar, and Joseph [135] proposed to craft adversarial examples for tree ensembles via mixed integer programming, which was later improved by Chen et al. [136] via formulating the robustness verification of tree ensembles as a max-clique problem. Wang et al. [137] proposed a quadratic programming formulation for attacking nearest neighbor classifiers. Yang et al. [138] recently proposed a region-based attack that applies to classifiers with convex polyhedra as decision regions, such as tree-based models and nearest neighbor classifiers.

We study the performance of decision-based attack against a wide class of mdoels, including linear models such as logistic regression and support vector machine, kernel support vector machine, k-nearest neighbors (KNN), tree-based models such as random forests, and gradient boosted trees, and feature extracters for natural images, including the Scale Invariant Feature Transform (SIFT) [128], the Histogram of Oriented Gradients (HOG) [129]. To our best knowledge, we are the first to study black-box attack against KNN, random forests, SIFT and HOG features without access to model details or training data.

6.3 An optimization framework

In this section, we describe an optimization framework for finding adversarial instances for an m-ary classification model of the following type. The first component is a *discriminant function* $F : \mathbb{R}^d \to \mathbb{R}^m$ that accepts an input $x \in [0,1]^d$ and produces an output $y \in \Delta_m := \{y \in [0,1]^m \mid \sum_{c=1}^m y_c = 1\}$. The output vector $y = (F_1(x), \ldots, F_m(x))$ can be viewed as a probability distribution over the label set $[m] = \{1, \ldots, m\}$. Based on the function F, the classifier $C : \mathbb{R}^d \to [m]$ assigns input x to the class with maximum probability—that is,

$$C(x) := \arg\max_{c \in [m]} F_c(x).$$

We study adversaries of both the untargeted and targeted varieties. Given some input $x^\star$, the goal of an *untargeted attack* is to change the original classifier decision $c^\star := C(x^\star)$ to any $c \in [m] \backslash \{c^\star\}$, whereas the goal of a *targeted attack* is to change the decision to some pre-specified $c^\dagger \in [m] \backslash \{c^\star\}$. Formally, if we define the function $S_{x^\star} : \mathbb{R}^d \to \mathbb{R}$ via

$$S_{x^\star}(x') := \begin{cases} \max\limits_{c \neq c^\star} F_c(x') - F_{c^\star}(x') & \text{(Untargeted)} \\ F_{c^\dagger}(x') - \max\limits_{c \neq c^\dagger} F_c(x') & \text{(Targeted)} \end{cases} \tag{6.1}$$

then a perturbed image x' is a successful attack if and only if $S_{x^\star}(x') > 0$. The boundary between successful and unsuccessful perturbed images is

$$\mathrm{bd}(S_{x^\star}) := \left\{ z \in [0,1]^d \mid S_{x^\star}(z) = 0 \right\}.$$

As an indicator of successful perturbation, we introduce the Boolean-valued function $\phi_{x^\star} : [0,1]^d \to \{-1, 1\}$ via

$$\phi_{x^\star}(x') := \mathrm{sign}\left(S_{x^\star}(x')\right) = \begin{cases} 1 & \text{if } S_{x^\star}(x') > 0, \\ -1 & \text{otherwise.} \end{cases}$$

This function is accessible in the decision-based setting, as it can be computed by querying the classifier C alone. The goal of an adversarial attack is to generate a perturbed sample x' such that $\phi_{x^\star}(x') = 1$, while keeping x' close to the original sample $x^\star$. This can be formulated as the optimization problem

$$\min_{x'} d(x', x^\star) \quad \text{such that} \quad \phi_{x^\star}(x') = 1, \tag{6.2}$$

where d is a distance function that quantifies similarity. Standard choices of d studied in past work [26, 28, 29] include the usual ℓ_p-norms, for $p \in \{0, 2, \infty\}$.

6.3.1 An iterative algorithm for ℓ_2 distance

Consider the case of the optimization problem (6.2) with the ℓ_2-norm $d(x, x^\star) = \|x - x^\star\|_2$. We first specify an iterative algorithm that is given access to the gradient $\nabla S_{x^\star}$. Given an initial vector x_0 such that $S_{x^\star}(x_0) > 0$ and a stepsize sequence $\{\xi_t\}_{t \geq 0}$, it performs the

update

$$x_{t+1} = \alpha_t x^\star + (1-\alpha_t)\left\{x_t + \xi_t \frac{\nabla S_{x^\star}(x_t)}{\|\nabla S_{x^\star}(x_t)\|_2}\right\}, \tag{6.3}$$

where ξ_t is a positive step size. Here the line search parameter $\alpha_t \in [0,1]$ is chosen such that $S_{x^\star}(x_{t+1}) = 0$—that is, so that the next iterate x_{t+1} lies on the boundary. The motivation for this choice is that our gradient-direction estimate in Section 6.4 is only valid near the boundary.

We now analyze this algorithm with the assumption that we have access to the gradient of $S_{x^\star}$ in the setting of binary classification. We make two assumptions for establishing the convergence of our algorithm.

Assumption A Assume that the function $S_{x^\star}$ is twice differentiable with a locally Lipschitz gradient, meaning that there exists $L > 0$ such that for all $x, y \in \{z : \|z - x^\star\|_2 \leq \|x_0 - x^\star\|_2\}$, we have

$$\|\nabla S_{x^\star}(x) - \nabla S_{x^\star}(y)\|_2 \leq L\|x - y\|_2, \tag{6.4}$$

where x_0 is the initialization point.

Assumption B We assume the gradient is bounded away from zero on the boundary: there exists a positive $c > 0$ such that $\|\nabla S_{x^\star}(z)\| > c$ for any $z \in \mathrm{bd}(S_{x^\star})$.

The above two assumptions hold for several common machine learning models, such as logistic regression, neural networks, and kernel SVMs in binary classification. In experiments, we also show the algorithm works for the setting when the function $S_{x^\star}$ is differentiable almost everywhere, such as tree-based models, KNNs, SIFT and HOG. A theoretical study of its convergence will be delayed to future work.

We analyze the behavior of the updates (6.3) in terms of the angular measure

$$\begin{aligned} r(x_t, x^\star) &:= \cos\angle\left(x_t - x^\star, \nabla S_{x^\star}(x_t)\right) \\ &= \frac{\left\langle x_t - x^\star,\, \nabla S_{x^\star}(x_t)\right\rangle}{\|x_t - x^\star\|_2 \|\nabla S_{x^\star}(x_t)\|_2}, \end{aligned}$$

corresponding to the cosine of the angle between $x_t - x^\star$ and the gradient $\nabla S_{x^\star}(x_t)$. Note that the condition $r(x, x^\star) = 1$ holds if and only if x is a stationary point of the optimization (6.2). The following theorem guarantees that, with a suitable step size, the updates converge to such a stationary point:

Theorem 5. *Under Assumption A and Assumption B, suppose that we compute the updates* (6.3) *with step size* $\xi_t = \|x_t - x^\star\|_2 t^{-q}$ *for some* $q \in \left(\frac{1}{4}, \frac{1}{2}\right)$. *Then there is a universal constant* c *such that*

$$0 \;\leq\; 1 - r(x_t, x^\star) \leq c\,\frac{1}{t^{\frac{1}{2}-q}} \quad \textit{for all iterations } t = 1, 2, \ldots. \tag{6.5}$$

In particular, the algorithm converges to a stationary point of problem (6.2).

Theorem 5 suggests a scheme for choosing the step size in the algorithm that we present in the next section. An experimental evaluation of the proposed scheme is carried out in Section 6.6. The proof of the theorem is constructed by establishing the relationship between

the objective value $d(x_t, x^\star)$ and $r(x_t, x^\star)$, with a second-order Taylor approximation to the boundary. See Section 6.7.1 for details.

6.3.2 Extension to ℓ_∞-distance

We now describe how to extend these updates so as to minimize the ℓ_∞-distance. Consider the ℓ_2-projection of a point x onto the sphere of radius α_t centered at $x^\star$:

$$\Pi^2_{x^\star,\alpha_t}(x) := \underset{\|y-x^\star\|_2 \leq \alpha_t}{\arg\min} \|y - x\|_2 = \alpha_t x^\star + (1-\alpha_t)x. \tag{6.6}$$

In terms of this operator, our ℓ_2-based update (6.3) can be rewritten in the equivalent form

$$x_{t+1} = \Pi^2_{x^\star,\alpha_t}\left(x_t + \xi_t \frac{\nabla S_{x^\star}(x_t)}{\|\nabla S_{x^\star}(x_t)\|_2}\right). \tag{6.7}$$

This perspective allows us to extend the algorithm to other ℓ_p-norms for $p \neq 2$. For instance, in the case $p = \infty$, we can define the ℓ_∞-projection operator $\Pi^\infty_{x^\star,\alpha}$. It performs a per-pixel clip within a neighborhood of $x^\star$, such that the ith entry of $\Pi^\infty_{x^\star,\alpha}(x)$ is

$$\Pi^\infty_{x^\star,\alpha}(x)_i := \max\left\{\min\{x^\star_i, x^\star_i + c\}, x_i - c\right\},$$

where $c := \alpha\|x - x^\star\|_\infty$. We propose the ℓ_∞-version of our algorithm by carrying out the following update iteratively:

$$x_{t+1} = \Pi^\infty_{x^\star,\alpha_t}\big(x_t + \xi_t \text{sign}(\nabla S_{x^\star}(x_t))\big), \tag{6.8}$$

where α_t is chosen such that $S_{x^\star}(x_{t+1}) = 0$, and "sign" returns the element-wise sign of a vector. We use the sign of the gradient for faster convergence in practice, similar to previous work [26, 27, 30].

6.3.3 Connection to existing white-box attacks

In this subsection, we discuss the connection between our proposed updates (6.7) and (6.8), and existing white-box attacks.

Connecting the update (6.7) to penalty methods A common approach to the constrained problem (6.2) is to approximate it with a penalty function formulation [139]:

$$\min_x \frac{1}{2}\|x - x^\star\|_2^2 - \rho S_{x^\star}(x), \tag{6.9}$$

for a carefully chosen penalty parameter ρ. A similar relaxation has been proposed in previous work in adversarial attacks [25, 29].

At the tth step, the update of gradient descent is

$$\begin{aligned} x_{t+1} &= x_t - \alpha_t((x_t - x^\star) - \rho_t \nabla S_{x^\star}(x_t)) \\ &= \alpha_t x^\star + (1-\alpha_t)\Big(x_t + \frac{\rho_t}{1-\alpha_t}\nabla S_{x^\star}(x_t))\Big) \\ &= \Pi^2_{x^\star,\alpha_t}\Big(x_t + \underbrace{\frac{\rho_t \|\nabla S_{x^\star}(x_t)\|_2}{1-\alpha_t}}_{\xi_t} \frac{\nabla S_{x^\star}(x_t)}{\|\nabla S_{x^\star}(x_t)\|_2}\Big), \end{aligned}$$

where we rewrite the gradient update as a weighted average of the original sample $x^\star$ and the t-th update x_t perturbed along the direction of $\nabla S_{x^\star}(x_t)$ with an appropriately chosen size ξ_t. This has exactly the same form as our proposed update, Equation (6.6). However, the hyperparameters α_t and ρ_t in previous work [25, 29] are often chosen via hyperparameter tuning or fixed as constant. Here, we choose choose α_t so that x_{t+1} is at the boundary $S(x_{t+1}) = 0$, which is a requirement to use our proposed gradient estimate to be introduced in Section 6.4.

Connecting the update (6.8) **to projected gradient descent** In the untargeted case, each update of Basic Iterative Method proposed by Kurakin, Goodfellow, and Bengio [27] is of the form

$$x_{t+1} = \mathrm{Clip}_{x^\star,\alpha_t}\{x_t + \xi_t \mathrm{sign}(\nabla_x J(x_t, c_{\text{true}}))\}, \tag{6.10}$$

where J is the cross-entropy loss with respect to the true label $c^\star = C(x^\star)$. The operator $\mathrm{Clip}_{x^\star,\alpha}$ performs per-pixel clip within the α-neighborhood of the corresponding pixel of $x^\star$. As pointed out by Madry et al. [30], the Basic Iterative Method can be interpreted as projected gradient descent in the ℓ_∞-norm.

We observe that the clip operator coincides with the ℓ_∞-projection operator $\Pi^\infty_{x^\star,\alpha}$. The cross entropy loss J is a monotonic function of $S_{x^\star}$, and so introduces only a scaling difference between ∇J and $\nabla S_{x^\star}$. As a consequence, apart from the choice of α_t, each update (6.8) of our algorithm has the same form as the Basic Iterative Method. On the other hand, we need to choose α_t carefully so that x_{t+1} lies at the boundary for gradient-direction estimation.

6.4 A decision-based algorithm based on a novel gradient estimate

We now extend our procedures to the decision-based setting, in which we have access *only* to the Boolean-valued function $\phi_{x^\star}(x) = \mathrm{sign}(S_{x^\star}(x))$—that is, the method cannot observe the underlying discriminant function F or its gradient. In this section, we introduce a gradient-direction estimate based on $\phi_{x^\star}$ when $x_t \in \mathrm{bd}(S_{x^\star})$ (so that $S_{x^\star}(x_t) = 0$ by definition). We proceed to discuss how to approach the boundary. Then we discuss how to control the error of our estimate with a deviation from the boundary. We will summarize the analysis with a decision-based algorithm.

6.4.1 At the boundary

Given an iterate $x_t \in \mathrm{bd}(S_{x^\star})$ we propose to approximate the direction of the gradient $\nabla S_{x^\star}(x_t)$ via the Monte Carlo estimate

$$\widetilde{\nabla S}(x_t, \delta) := \frac{1}{B}\sum_{b=1}^{B} \phi_{x^\star}(x_t + \delta u_b) u_b, \tag{6.11}$$

where $\{u_b\}_{b=1}^B$ are i.i.d. draws from the uniform distribution over the d-dimensional sphere, and δ is small positive parameter. (The dependence of this estimator on the fixed centering point $x^\star$ is omitted for notational simplicity.)

The perturbation parameter δ is necessary, but introduces a form of bias in the estimate. Our first result controls this bias, and shows that $\widetilde{\nabla S}(x_t, \delta)$ is asymptotically unbiased as $\delta \to 0^+$.

Theorem 6. *For a boundary point x_t, suppose that $S_{x^\star}$ has L-Lipschitz gradients in a neighborhood of x_t. Then the cosine of the angle between $\widetilde{\nabla S}(x_t, \delta)$ and $\nabla S_{x^\star}(x_t)$ is bounded as*

$$\cos \angle \Big(\mathbb{E}[\widetilde{\nabla S}(x_t, \delta)], \nabla S_{x^\star}(x_t) \Big) \geq 1 - \frac{9L^2\delta^2 d^2}{8\|\nabla S(x_t)\|_2^2}. \tag{6.12}$$

In particular, we have

$$\lim_{\delta \to 0} \cos \angle \Big(\mathbb{E}[\widetilde{\nabla S}(x_t, \delta)], \nabla S_{x^\star}(x_t) \Big) = 1, \tag{6.13}$$

showing that the estimate is asymptotically unbiased as an estimate of direction.

We remark that Theorem 6 only establishes the asymptotic behavior of the proposed estiamte at the boundary. This also motivates the boundary search step in our algorithm to be discussed in Seciton 6.4.2. The proof of Theorem 6 starts from dividing the unit sphere into three components: the upper cap along the direction of gradient, the lower cap opposite to the direction of gradient, and the annulus in between. The error from the annulus can be bounded when δ is small. See Section 6.7.2 for the proof of this theorem. As will be seen in the sequel, the size of perturbation δ should be chosen proportionally to d^{-1}; see Section 6.4.3 for details.

To characterize the error in the estimate in a more accurate mannar, we provide a tail bound for the deviation of estimate from the direction of gradient. Before that, we introduce the following tail bound which characterizes the deviation of the estimate from its expected value.

Lemma 2. *For any $s > 0$, the Monte Carlo estimate $\widetilde{\nabla S}(x_t, \delta)$ satisfies the following tail bound*

$$\mathbb{P}\Big(\|\widetilde{\nabla S}(x_t, \delta) - \mathbb{E}[\widetilde{\nabla S}(x_t, \delta)]\| \geq s \Big) \leq 2(d+1) \exp\Big(-\frac{Bs^2}{8 + 4s} \Big).$$

The above lemma is proved by a direct application of the Bernstein bound for random matrices [140, 141]. Incorporating the bias with nonzero δ characterized in Theorem 6 into Lemma 2, we have the following result:

Theorem 7. *For a boundary point x_t, suppose that $S_{x^\star}$ has L-Lipschitz gradients in a neighborhood of x_t. Then the following tail bound can be established for the cosine of the angle between $\widetilde{\nabla S}(x_t, \delta)$ and $\nabla S_{x^\star}(x_t)$. For any $a \in (0, 1)$, when δ is chosen such that $\delta < \frac{\|\nabla S_{x^\star}(x_t)\|}{3L(d-1)}\sqrt{\frac{a}{\pi}}$, we have*

$$\mathbb{P}\Big(\cos\angle \Big(\mathbb{E}[\widetilde{\nabla S}(x_t, \delta)], \nabla S_{x^\star}(x_t) \Big) < 1 - a \Big) \leq 2(d+1) \exp\Big\{ - \frac{Ba}{32\pi + 8\sqrt{a\pi}} \Big\}.$$

Remark 1. *Theorem 7 suggests the probability of deviation of the Monte Carlo estimate from the true gradient direction is exponentially small in the size of random directions B, with a linear dimension dependence. The size of perturbation δ should be chosen proportionally to d^{-1}.*

When we only have access to output labels of a model, we replace the true gradient in the update (6.3) by its estimate

$$x_{t+1} = \alpha_t x^\star + (1 - \alpha_t) \left\{ x_t + \xi_t \frac{\widetilde{\nabla S}_{x^\star}(x_t)}{\|\widetilde{\nabla S}_{x^\star}(x_t)\|_2} \right\}. \tag{6.14}$$

By incorporating the tail bound on the estimate into the proof of Theorem 5, we can establish the updates 6.14 converges almost surely. More concretely, we have the following result:

Theorem 8. *Under Assumption A and Assumption B, suppose that we estimate the gradient with batch size $B = \Omega(t^{1-2q})$, and we compute the updates (6.14) with the step size $\xi_t = \|x_t - x^\star\|_2 t^{-q}$ for some $q \in \big(\frac{1}{4}, \frac{1}{2}\big)$. Then there exist universal constants $C, \widetilde{C}, C_\delta$ such that for sufficiently large t, when $\delta_t < C_\delta t^{-0.5+q}$, we have*

$$\mathbb{P}\Big(1 - r(x_t, x^\star) > C \, \frac{1}{t^{\frac{1}{2}-q}} \Big) \geq 1 - 2(d+1) \exp\Big(- \widetilde{C} B t^{-1+2q} \Big). \tag{6.15}$$

In particular, the algorithm converges to a stationary point of problem (6.2) almost surely.

Remark 2. *Theorem 8 suggests the batch size B should be chosen at the scale of $B = \Omega(t^{1-2q})$. It also suggests a trade-off in choosing the size of q. A smaller q would lead to a larger batch size to ensure convergence. On the other hand, x_t converges to a saddle point faster if we increase q towards $\frac{1}{2}$. Intuitively, this is because a more accurate estimate of the direction of gradient is required if we uses a larger step size, and a larger step size results in a faster convergence rate when the estimate is accurate enough.*

The proofs of Lemma 2, Theorem 7 and Theorem 8 are available in Section 6.7.3, 6.7.4 and 6.7.5 respectively.

6.4.2 Approaching the boundary

The proposed estimate (6.11) is only valid at the boundary. We now describe how we approach the boundary via a binary search. Let $\tilde{x}_t$ denote the updated sample before the

operator Π^p_{x,α_t} is applied:

$$\tilde{x}_t := x_t + \xi_t v_t(x_t, \delta_t), \text{ such that} \tag{6.16}$$
$$v_t(x_t, \delta_t) = \begin{cases} \widehat{\nabla S}(x_t, \delta_t)/\|\widehat{\nabla S}(x_t, \delta_t)\|_2, & \text{if } p = 2, \\ \text{sign}(\widehat{\nabla S}(x_t, \delta_t)), & \text{if } p = \infty, \end{cases}$$

where $\widehat{\nabla S}$ will be introduced later in equation (6.20), as a variance-reduced version of $\widetilde{\nabla S}$, and δ_t is the size of perturbation at the t-th step.

We hope $\tilde{x}_t$ is at the opposite side of the boundary to x so that the binary search can be carried out. Therefore, we initialize at $\tilde{x}_0$ at the target side with $\phi_{x^\star}(\tilde{x}_0) = 1$, and set $x_0 := \Pi^p_{x,\alpha_0}(\tilde{x}_0)$, where α_0 is chosen via a binary search between 0 and 1 to approach the boundary, stopped at x_0 lying on the target side with $\phi_{x^\star}(x_0) = 1$. At the t-th iteration, we start at x_t lying at the target side $\phi_{x^\star}(x_t) = 1$. The step size is initialized as

$$\xi_t := \|x_t - x^\star\|_p/\sqrt{t}, \tag{6.17}$$

as suggested by Theorem 5 in the ℓ_2 case, and is decreased by half until $\phi_{x^\star}(\tilde{x}_t) = 1$, which we call *geometric progression* of ξ_t. Having found an appropriate $\tilde{x}_t$, we choose the projection radius α_t via a binary search between 0 and 1 to approach the boundary, which stops at x_{t+1} with $\phi_{x^\star}(x_{t+1}) = 1$. See Algorithm 5 for the complete binary search, where the binary search threshold θ is set to be some small constant.

Algorithm 5 Bin-Search

Require: Samples x', x, with a binary function ϕ, such that $\phi(x') = 1, \phi(x) = 0$, threshold θ, constraint ℓ_p.
Ensure: A sample x'' near the boundary.
  Set $\alpha_l = 0$ and $\alpha_u = 1$.
  while $|\alpha_l - \alpha_u| > \theta$ **do**
    Set $\alpha_m \leftarrow \frac{\alpha_l + \alpha_u}{2}$.
    if $\phi(\Pi_{x,\alpha_m}(x')) = 1$ **then**
      Set $\alpha_u \leftarrow \alpha_m$.
    else
      Set $\alpha_l \leftarrow \alpha_m$.
    end if
  end while
  Output $x'' = \Pi_{x,\alpha_u}(x')$.

6.4.3 Controlling errors of deviations from the boundary

Binary search never places x_{t+1} exactly onto the boundary. We analyze the error of the gradient-direction estimate, and propose two approaches for reducing the error.

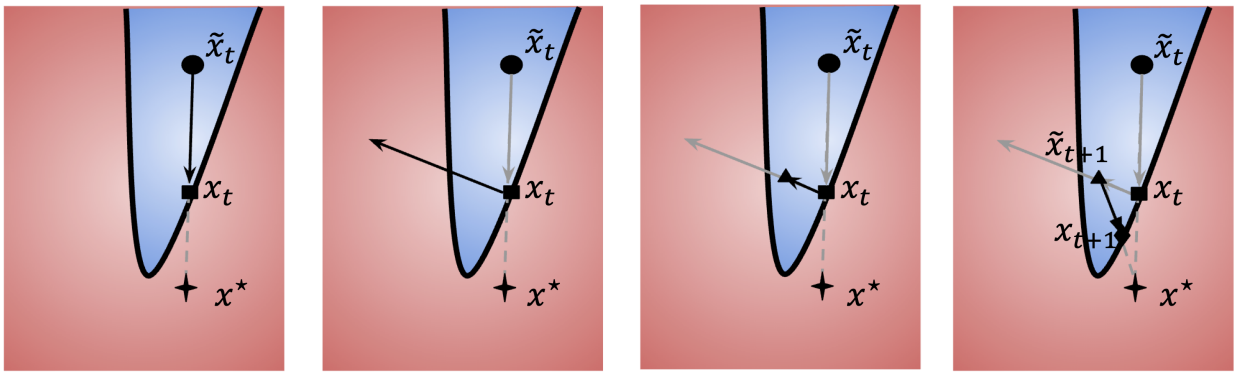


Figure 6.1: Intuitive explanation of HopSkipJumpAttack. (a) Perform a binary search to find the boundary, and then update $\tilde{x}_t \to x_t$. (b) Estimate the gradient at the boundary point x_t. (c) Geometric progression and then update $x_t \to \tilde{x}_{t+1}$. (d) Perform a binary search, and then update $\tilde{x}_{t+1} \to x_{t+1}$.

Appropriate choice of the size of random perturbation First, the size of random perturbation δ_t for estimating the gradient direction is chosen as a function of image size d and the binary search threshold θ. This is different from numerical differentiation, where the optimal choice of δ_t is at the scale of round-off errors (e.g., [142]). Below we characterize the error incurred by a large δ_t as a function of distance between $\tilde{x}_t$ and the boundary, and derive the appropriate choice of ξ_t and δ_t. In fact, with a Taylor approximation of $S_{x^\star}$ at x_t, we have

$$S_{x^\star}(x_t + \delta_t u) = S_{x^\star}(x_t) + \delta_t \big\langle \nabla S_{x^\star}(x_t),\, u \big\rangle + \mathcal{O}(\delta_t^2).$$

At the boundary $S_{x^\star}(x_t) = 0$, the error of gradient approximation scales at $\mathcal{O}(\delta_t^2)$, which is minimized by reducing δ_t to the scale of rooted round-off error. However, the outcome x_t of a finite-step binary search lies close to, but not exactly on the boundary.

When δ_t is small enough such that second-order terms can be omitted, the first-order Taylor approximation implies that $\phi_{x^\star}(x_t + \delta_t u) = -1$ if and only if $x_t + \delta_t u$ lies on the spherical cap $\mathcal{C}$, with

$$\mathcal{C} := \Big\{ u \mid \Big\langle \frac{\nabla S_{x^\star}(x_t)}{\|\nabla S_{x^\star}(x_t)\|_2}, u \Big\rangle < -\delta_t^{-1} \frac{S_{x^\star}(x_t)}{\|\nabla S_{x^\star}(x_t)\|_2} \Big\}.$$

On the other hand, the probability mass of u concentrates on the equator in a high-dimensional sphere, which is characterized by the following inequality [143]:

$$\mathbb{P}(u \in \mathcal{C}) \leq \frac{2}{c} \exp\{-\frac{c^2}{2}\}, \text{where } c = \frac{\sqrt{d-2} S_{x^\star}(x_t)}{\delta_t \|\nabla S_{x^\star}(x_t)\|_2}. \tag{6.18}$$

A Taylor expansion of x_t at $x_t' := \Pi_\partial^2(x_t)$ yields

$$\begin{aligned} S_{x^\star}(x_t) &= \nabla S_{x^\star}(x_t')^T (x_t - x_t') + \mathcal{O}(\|x_t - x_t'\|_2^2) \\ &= \nabla S_{x^\star}(x_t)^T (x_t - x_t') + \mathcal{O}(\|x_t - x_t'\|_2^2). \end{aligned}$$

Algorithm 6 HopSkipJumpAttack

Require: Classifier C, a sample x, constraint ℓ_p, initial batch size B_0, iterations T.
Ensure: Perturbed image x_t.
  Set θ (Equation (6.19)).
  Initialize at $\tilde{x}_0$ with $\phi_{x^\star}(\tilde{x}_0) = 1$.
  Compute $d_0 = \|\tilde{x}_0 - x^\star\|_p$.
  for t in $1, 2, \ldots, T-1$ **do**
    (**Boundary search**)
    $x_t = \text{BIN-SEARCH}(\tilde{x}_{t-1}, x, \theta, \phi_{x^\star}, p)$
    (**Gradient-direction estimation**)
    Sample $B_t = B_0\sqrt{t}$ unit vectors $u_1, \ldots, u_{B_t}$.
    Set δ_t (Equation (6.19)).
    Compute $v_t(x_t, \delta_t)$ (Equation (6.16)).
    (**Step size search**)
    Initialize step size $\xi_t = \|x_t - x^\star\|_p/\sqrt{t}$.
    while $\phi_{x^\star}(x_t + \varepsilon_t v_t) = 0$ **do**
      $\xi_t \leftarrow \xi_t/2$.
    end while
    Set $\tilde{x}_t = x_t + \xi_t v_t$.
    Compute $d_t = \|\tilde{x}_t - x^\star\|_p$.
  end for
  Output $x_t = \text{BIN-SEARCH}(\tilde{x}_{t-1}, x, \theta, \phi_{x^\star}, p)$.

By the Cauchy-Schwarz inequality and the definition of ℓ_2-projection, we have

$$\begin{aligned}
&|\nabla S_{x^\star}(x_t)^T(x_t - x_t')| \\
&\leq \|\nabla S_{x^\star}(x_t)\|_2 \|x_t - \Pi^2_\partial(x_t)\|_2 \\
&\leq \begin{cases} \|\nabla S_{x^\star}(x_t)\|_2 \theta \|\tilde{x}_{t-1} - x^\star\|_p, & \text{if } p = 2, \\ \|\nabla S_{x^\star}(x_t)\|_2 \theta \|\tilde{x}_{t-1} - x^\star\|_p \sqrt{d}, & \text{if } p = \infty. \end{cases}
\end{aligned}$$

This yields

$$c = \mathcal{O}(\frac{d^q \theta \|\tilde{x}_{t-1} - x^\star\|_p}{\delta_t}),$$

where $q = 1 - (1/p)$ is the dual exponent. In order to avoid a loss of accuracy from concentration of measure, we let $\delta_t = d^q \theta \|\tilde{x}_{t-1} - x^\star\|_2$. To make the approximation error independent of dimension d, we set θ at the scale of d^{-q-1}, so that δ_t is proportional to d^{-1}, as suggested by Theorem 6. This leads to a logarithmic dependence on dimension for the number of model queries. In practice, we set

$$\theta = d^{-q-1};\ \delta_t = d^{-1}\|\tilde{x}_{t-1} - x^\star\|_p. \tag{6.19}$$

A baseline for variance reduction in gradient-direction estimation Another source of error comes from the variance of the estimate, where we characterize variance of a random

vector $v \in \mathbb{R}^d$ by the trace of its covariance operator: $\mathrm{Var}(v) := \sum_{i=1}^{d} \mathrm{Var}(v_i)$. When x_t deviates from the boundary and δ_t is not exactly zero, there is an uneven distribution of perturbed samples at the two sides of the boundary:

$$|\mathbb{E}[\phi_{x^\star}(x_t + \delta_t u)]| > 0,$$

as we can see from Equation (6.18). To attempt to control the variance, we introduce a baseline $\overline{\phi_{x^\star}}$ into the estimate:

$$\overline{\phi_{x^\star}} := \frac{1}{B} \sum_{b=1}^{B} \phi_{x^\star}(x_t + \delta u_b),$$

which yields the following estimate:

$$\widehat{\nabla S}(x_t, \delta) := \frac{1}{B-1} \sum_{b=1}^{B} (\phi_{x^\star}(x_t + \delta u_b) - \overline{\phi_{x^\star}}) u_b. \tag{6.20}$$

It can be easily observed that this estimate is equal to the previous estimate in expectation, and thus still asymptotically unbiased at the boundary: When $x_t \in \mathrm{bd}(S_{x^\star})$, we have

$$\cos \angle \left(\mathbb{E}[\widehat{\nabla S}(x_t, \delta)], \nabla S_{x^\star}(x_t) \right) \geq 1 - \frac{9L^2\delta^2 d^2}{8\|\nabla S(x_t)\|_2^2},$$
$$\lim_{\delta \to 0} \cos \angle \left(\mathbb{E}[\widehat{\nabla S}(x_t, \delta)], \nabla S_{x^\star}(x_t) \right) = 1.$$

Moreover, the introduction of the baseline reduces the variance when $\mathbb{E}[\phi_{x^\star}(x_t + \delta u)]$ deviates from zero. In particular, the following theorem shows that whenever $|\mathbb{E}[\phi_{x^\star}(x_t + \delta u)]| = \Omega(B^{-\frac{1}{2}})$, the introduction of a baseline reduces the variance.

Theorem 9. *Defining $\sigma^2 := Var(\phi_{x^\star}(x_t + \delta u)u)$ as the variance of one-point estimate, we have*

$$Var(\widehat{\nabla S}(x_t, \delta)) < Var(\widetilde{\nabla S}(x_t, \delta))(1 - \psi),$$

where

$$\psi = \frac{2}{\sigma^2(B-1)} \left(2B\mathbb{E}[\phi_{x^\star}(x_t + \delta u)]^2 - 1 \right) - \frac{2B-1}{(B-1)^2}.$$

See Section 6.7.6 for the proof. We also present an experimental evaluation of our gradient-direction estimate when the sample deviates from the boundary in Section 6.6, where we show our proposed choice of δ_t and the introduction of baseline yield a performance gain in estimating gradient.

6.4.4 HopSkipJumpAttack

We now combine the above analysis into an iterative algorithm, HopSkipJumpAttack. It is initialized with a sample in the target class for untargeted attack, and with a sample blended with uniform noise that is misclassified for targeted attack. Each iteration of the algorithm has three components. First, the iterate from the last iteration is pushed towards the boundary via a binary search (Algorithm 5). Second, the gradient direction is estimated

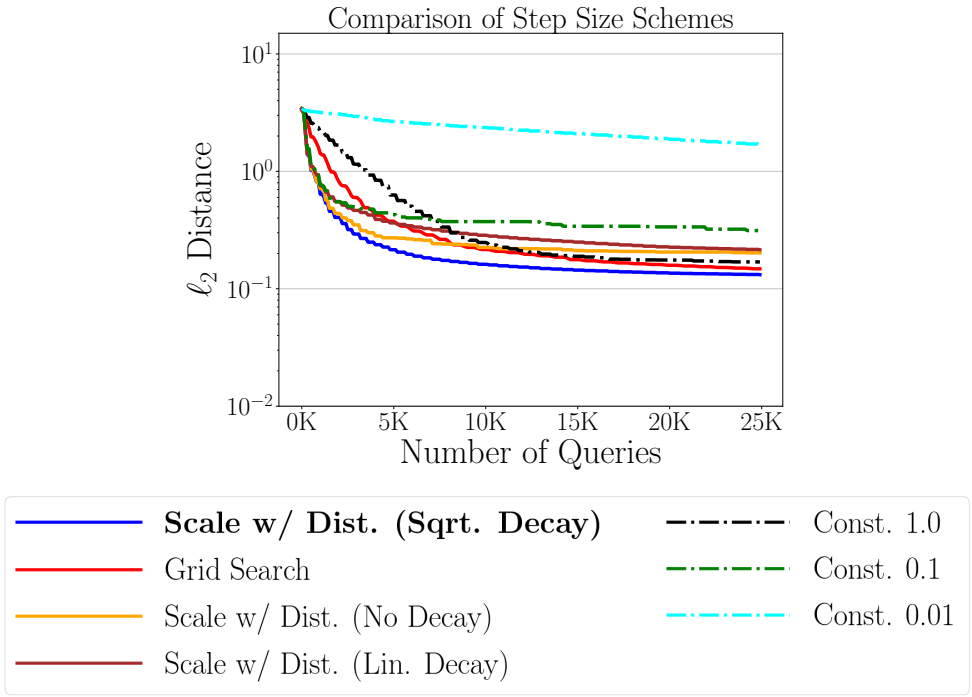


Figure 6.2: Comparison of various choices of step size.

via Equation (6.20). Third, the updating step size along the gradient direction is initialized as Equation (6.17) based on Theorem 5, and is decreased via geometric progression until perturbation becomes successful. The next iteration starts with projecting the perturbed sample back to the boundary again. The complete procedure is summarized in Algorithm 6. Figure 6.1 provides an intuitive visualization of the three steps in ℓ_2. For all experiments, we initialize the batch size at 100 and increase it with $\sqrt{t}$ linearly, so that the variance of the estimate reduces with t. When the input domain is bounded in practice, a clip is performed at each step by default.

6.5 Experiments

In this section, we carry out experimental analysis of HopSkipJumpAttack. We carry out an experimental evaluation on the gradient direction estimate and the scheme of choosing step sizes. We compare the efficiency of HopSkipJumpAttack with several previously proposed decision-based attacks on image classification tasks. In addition, we evaluate the robustness of three defense mechanisms under our attack method. All experiments were carried out on a Tesla K80 GPU, with code for the experiments to be made publically available.

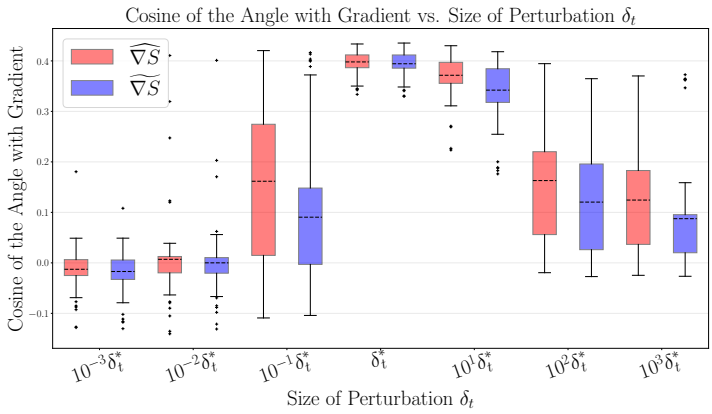


Figure 6.3: Box plots of the cosine of the angle between the proposed estimates and the true gradient.

6.6 Sensitivity analysis

In this section, we carry out experiments to evaluate the hyper-parameters suggested by our theoretical analysis. We use a 20-layer ResNet [144] trained over CIFAR-10 [81]. We run the ℓ_2-optimized HopSkipJumpAttack over a subset of randomly sampled images.

Choice of step size We compare several schemes of choosing step size at each step. The first scheme is suggested by Theorem 5: at the t-th step, we set $\xi_t = \|x_t - x^\star\|_2/\sqrt{t}$, which we call "Scale with Distance (Sqrt. Decay)." We include the other two scales which scale with distance, "Scale with Distance (Linear Decay)" with $\xi_t = \|x_t - x^\star\|_2/t$ and "Scale with Distance (No Decay)" with $\xi_t = \|x_t - x^\star\|_2$. We then include "Grid Search," which searchs step sizes over a log-scale grid, and chooses the step size that best controls the distance with the original sample after projecting the updated sample back to the boundary via binary search. Finally, we include constant stepsizes at $\xi_t = 0.01, 0.1, 1.0$. For all schemes, we always use geometric progression to decrease the step size by half until $\phi_{x^\star}(\tilde{x}_t) = 1$ before the next binary search step.

Figure 6.2 plots the median distance against the number of queries for all schemes. We observe that the scheme suggested by Theorem 5 achieves the best performance in this experiment. Grid search costs extra query budget initially but eventually achieves a comparable convergence rate. When the step size scales with the distance but with inappropriately chosen decay, the algorithm converges slightly slower. The performance of the algorithm suffers from a constant step size.

Choice of perturbation size and introduction of baseline We now study the effectiveness of the proposed perturbation size and baseline for estimating gradient direction when the sample deviates from the boundary. In particular, we focus on the choice of δ_t and the introduction of baseline analyzed in Section 6.4. Gradient direction estimation is carried

Distance	Data	Model	Objective	Model Queries								
				1K			5K			20K		
				BA	Opt	HSJA	BA	Opt	HSJA	BA	Opt	HSJA
ℓ_2	MNIST	CNN	Untargeted	6.14	6.79	**2.46**	5.45	3.76	**1.67**	1.50	2.07	**1.48**
			Targeted	5.41	4.84	**3.26**	5.38	3.90	**2.24**	1.98	2.49	**1.96**
	CIFAR10	ResNet	Untargeted	2.78	2.07	**0.56**	2.34	0.77	**0.21**	0.27	0.29	**0.13**
			Targeted	7.83	8.21	**2.53**	5.91	4.76	**0.41**	0.59	1.06	**0.21**
		DenseNet	Untargeted	2.57	1.78	**0.48**	2.12	0.67	**0.18**	0.21	0.28	**0.12**
			Targeted	7.70	7.65	**1.75**	5.33	3.47	**0.34**	0.35	0.78	**0.19**
	CIFAR100	ResNet	Untargeted	1.34	1.20	**0.20**	1.12	0.41	**0.08**	0.10	0.14	**0.06**
			Targeted	9.30	12.43	**6.12**	7.40	8.34	**0.92**	1.61	4.06	**0.29**
		DenseNet	Untargeted	1.47	1.22	**0.25**	1.23	0.34	**0.11**	0.12	0.13	**0.08**
			Targeted	8.83	11.72	**5.10**	6.76	8.22	**0.75**	0.91	2.89	**0.26**
	ImageNet	ResNet	Untargeted	36.86	33.60	**9.75**	31.95	13.91	**2.30**	2.71	5.26	**0.84**
			Targeted	87.49	84.38	**71.99**	82.91	71.83	**38.79**	40.92	53.78	**10.95**
ℓ_∞	MNIST	CNN	Untargeted	0.788	0.641	**0.235**	0.700	0.587	**0.167**	0.243	0.545	**0.136**
			Targeted	0.567	0.630	**0.298**	0.564	0.514	**0.211**	0.347	0.325	**0.175**
	CIFAR10	ResNet	Untargeted	0.127	0.128	**0.023**	0.105	0.096	**0.008**	0.019	0.073	**0.005**
			Targeted	0.379	0.613	**0.134**	0.289	0.353	**0.028**	0.038	0.339	**0.010**
		DenseNet	Untargeted	0.114	0.119	**0.017**	0.095	0.078	**0.007**	0.017	0.063	**0.004**
			Targeted	0.365	0.629	**0.130**	0.249	0.359	**0.022**	0.025	0.338	**0.008**
	CIFAR100	ResNet	Untargeted	0.061	0.077	**0.009**	0.051	0.055	**0.004**	0.008	0.040	**0.002**
			Targeted	0.409	0.773	**0.242**	0.371	0.472	**0.124**	0.079	0.415	**0.019**
		DenseNet	Untargeted	0.065	0.076	**0.010**	0.055	0.038	**0.005**	0.010	0.030	**0.003**
			Targeted	0.388	0.750	**0.248**	0.314	0.521	**0.096**	0.051	0.474	**0.017**
	ImageNet	ResNet	Untargeted	0.262	0.287	**0.057**	0.234	0.271	**0.017**	0.030	0.248	**0.007**
			Targeted	0.615	0.872	**0.329**	0.596	0.615	**0.219**	0.326	0.486	**0.091**

Table 6.1: Median distance at various model queries. The smaller median distance at a given model query is bold-faced. BA and HSJA stand for Boundary Attack and HopSkipJumpAttack respectively.

out at perturbed images at the ith iteration, for $i = 10, 20, 30, 40, 50, 60$. We use the cosine of the angle between the gradient-direction estimate and the truth gradient of the model as a metric.

Figure 6.3 shows the box plots of two gradient-direction estimates as δ_t varies among $0.01\delta_t^*, 0.1\delta_t^*, \delta_t^*, 10\delta_t^*, 100\delta_t^*$, where $\delta_t^* = 10\sqrt{d}\theta\|\tilde{x}_{t-1} - x^\star\|_2$ is our proposed choice. We observe that our proposed choice of δ_t yields the highest cosine of the angle on average. Also, the baseline in $\widehat{\nabla S}$ further improves the performance, in particular when δ_t is not chosen optimally so that there is severe unevenness in the distribution of perturbed images.

6.6.1 Efficiency evaluation

Baselines We compare HopSkipJumpAttack with three state-of-the-art decision-based attacks: Boundary Attack [38], Limited Attack [33] and Opt Attack [40]. We use the implementation of the three algorithms with the suggested hyper-parameters from the publicly available source code online. Limited Attack is only included under the targeted ℓ_∞ setting, as in Ilyas et al. [33].

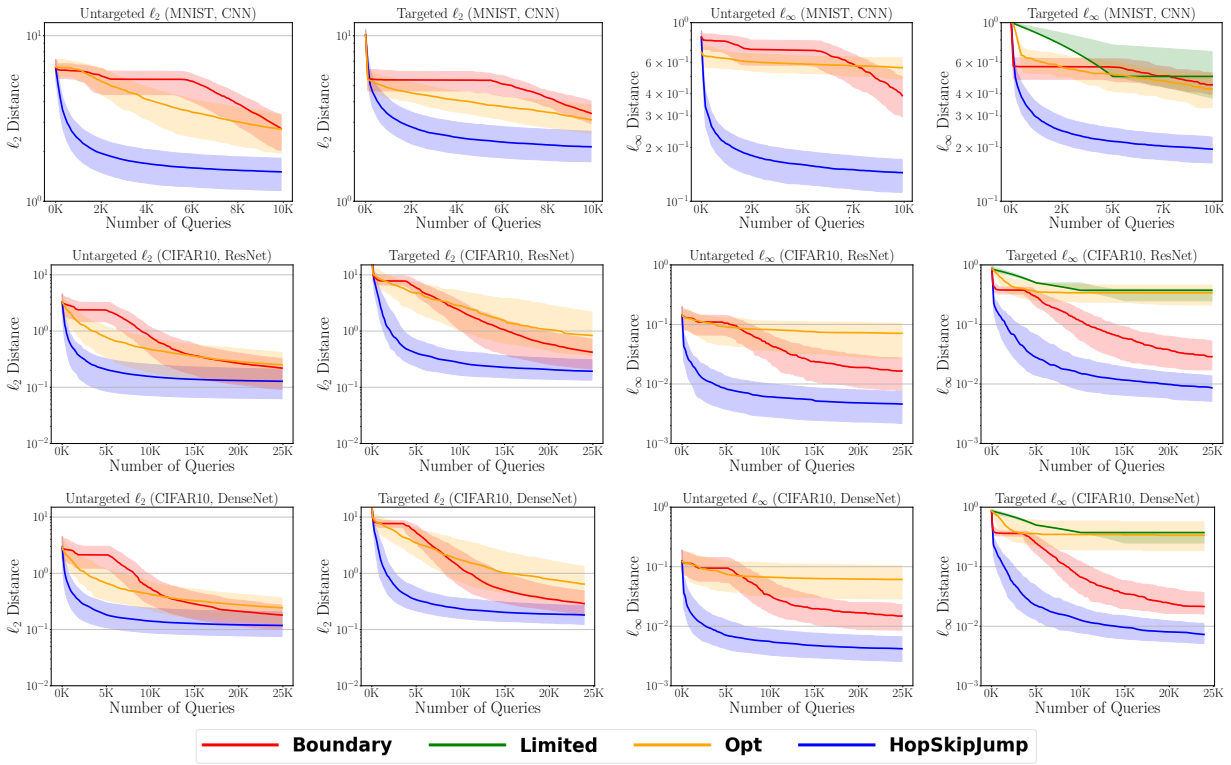


Figure 6.4: Median distance versus number of model queries on MNIST with CNN, and CIFAR-10 with ResNet and DenseNet from top to bottom rows. 1st column: untargeted ℓ_2. 2nd col.: targeted ℓ_2. 3rd col.: untargeted ℓ_∞. 4th col.: targeted ℓ_∞.

Data and models For a comprehensive evaluation of HopSkipJumpAttack, we use a wide range of data and models, with varied image dimensions, data set sizes, complexity levels of task and model structures.

The experiments are carried out over four image data sets: MNIST, CIFAR-10 [81], CIFAR-100 [81], and ImageNet [145] with the standard train/test split [99]. The four data sets have varied image dimensions and class numbers. MNIST contains 70K 28×28 gray-scale images of handwritten digits in the range 0-9. CIFAR-10 and CIFAR-100 are both composed of $32 \times 32 \times 3$ images. CIFAR-10 has 10 classes, with 6K images per class, while CIFAR-100 has 100 classes, with 600 images per class. ImageNet has $1,000$ classes. Images in ImageNet are rescaled to $224 \times 224 \times 3$. For MNIST, CIFAR-10 and CIFAR-100, $1,000$ correctly classified test images are used, which are randomly drawn from the test data set, and evenly distributed across classes. For ImageNet, we use 100 correctly classified test images, evenly distributed among 10 randomly selected classes. The selection scheme follows Metzen et al. [60] for reproducibility.

We also use models of varied structure, from simple to complex. For MNIST, we use a simple convolutional network composed of two convolutional layers followed by a hidden dense layer with 1024 units. Two convolutional layers have $32, 64$ filters respectively, each of which is followed by a max-pooling layer. For both CIFAR-10 and CIFAR-100, we train a

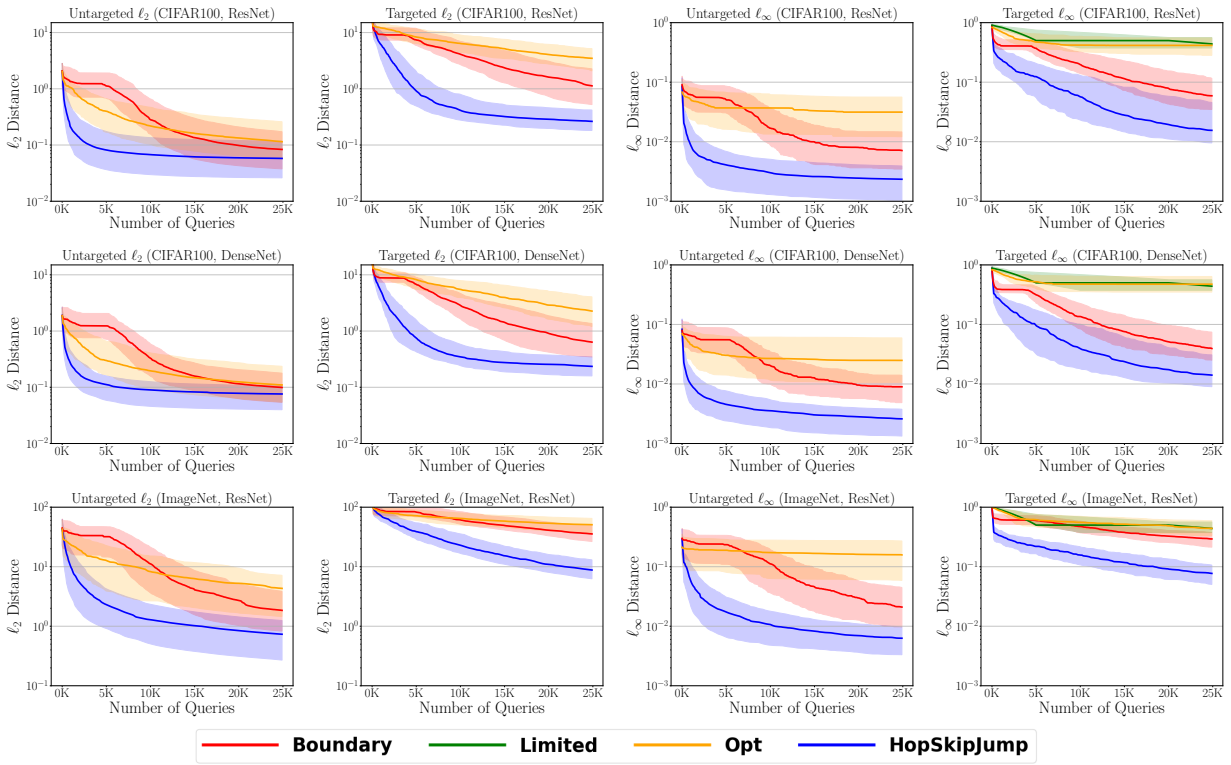


Figure 6.5: Median distance versus number of model queries on CIFAR-100 with ResNet, DenseNet, and ImageNet with ResNet from top to bottom rows. 1st column: untargeted ℓ_2. 2nd col.: targeted ℓ_2. 3rd col.: untargeted ℓ_∞. 4th col.: targeted ℓ_∞.

20-layer ResNet [144] and 121-layer DenseNet [146] respectively, with the canonical network structure [99]. For ImageNet, we use a pre-trained 50-layer ResNet [144]. All models achieve close to state-of-the-art accuracy on the respective data set. All pixels are scaled to be in the range $[0, 1]$. For all experiments, we clip the perturbed image into the input domain $[0, 1]$ for all algorithms by default.

Initialization For untargeted attack, we initialize all attacks by blending an original image with uniform random noise, and increasing the weight of uniform noise gradually until it is misclassified, a procedure which is available on a widely-used python package, Foolbox [147], as the default initialization of Boundary Attack. For targeted attack, the target class is sampled uniformly among the incorrect labels. An image belonging to the target class is randomly sampled from the test set as the initialization. The same target class and a common initialization image are used for all attacks.

Metrics The first metric is the median ℓ_p distance between perturbed and original samples over a subset of test images, which was commonly used in previous work, such as Carlini and Wagner [29]. A version normalized by image dimension was employed by Brendel, Rauber, and Bethge [38] for evaluating Boundary Attack. The ℓ_2 distance can be interpreted in the

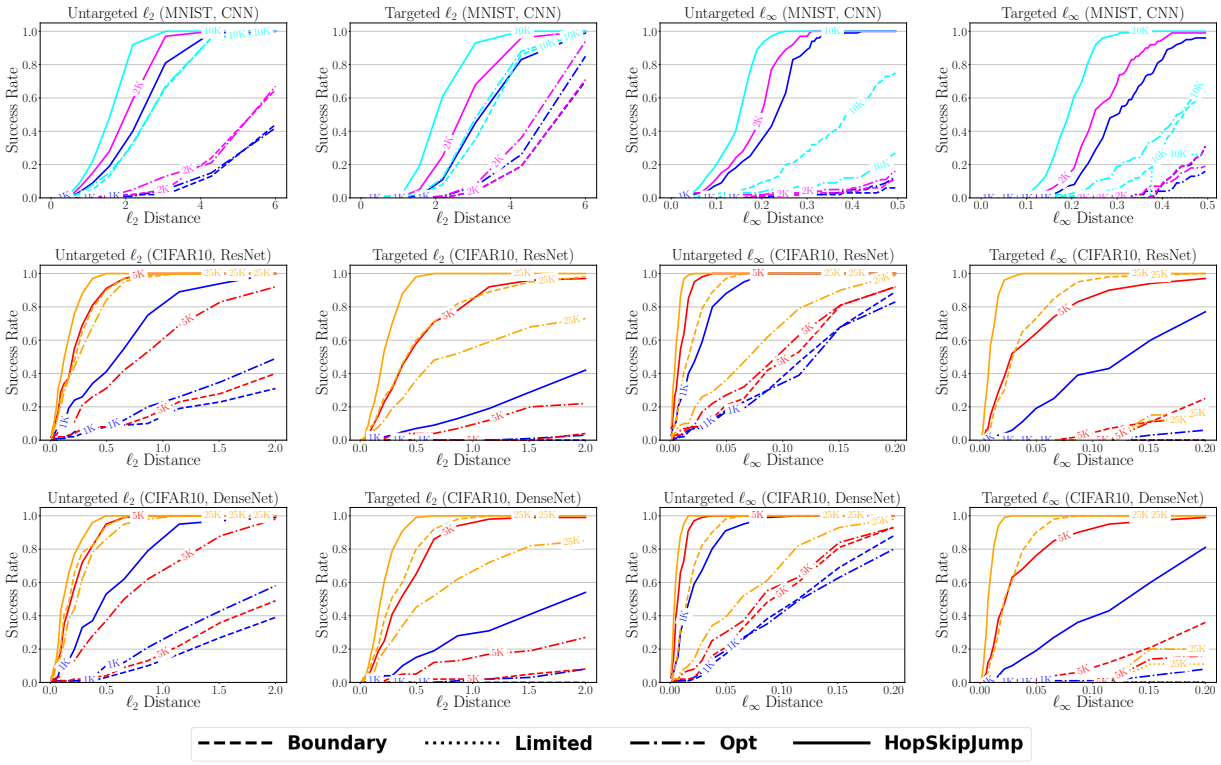


Figure 6.6: Success rate versus distance threshold for MNIST with CNN, and CIFAR-10 with ResNet, DenseNet from top to bottom rows. 1st column: untargeted ℓ_2. 2nd column: targeted ℓ_2. 3rd column: untargeted ℓ_∞. 4th column: targeted ℓ_∞.

following way: Given a byte image of size $h \times w \times 3$, perturbation of size d in ℓ_2 distance on the rescaled input image amounts to perturbation on the original image of $\lceil d/\sqrt{h \times w \times 3} * 255 \rceil$ bits per pixel on average, in the range $[0, 255]$. The perturbation of size d in ℓ_∞ distance amounts to a maximum perturbation of $\lceil 255 \cdot d \rceil$ bits across all pixels on the raw image.

As an alternative metric, we also plot the success rate at various distance thresholds for both algorithms given a limited budget of model queries. An adversarial example is defined a success if the size of perturbation does not exceed a given distance threshold. The success rate can be directly related to the accuracy of a model on perturbed data under a given distance threshold:

$$\text{perturbed acc.} = \text{original acc.} \times (1 - \text{success rate}). \tag{6.21}$$

Throughout the experiments, we limit the maximum budget of queries per image to 25,000, the setting of practical interest, due to limited computational resources.

Results Figure 6.4 and 6.5 show the median distance (on a log scale) against the queries, with the first and third quartiles used as lower and upper error bars. For Boundary, Opt and HopSkipJumpAttack, Table 6.1 summarizes the median distance when the number of queries is fixed at 1,000, 5,000, and 20,000 across all distance types, data, models and objectives.

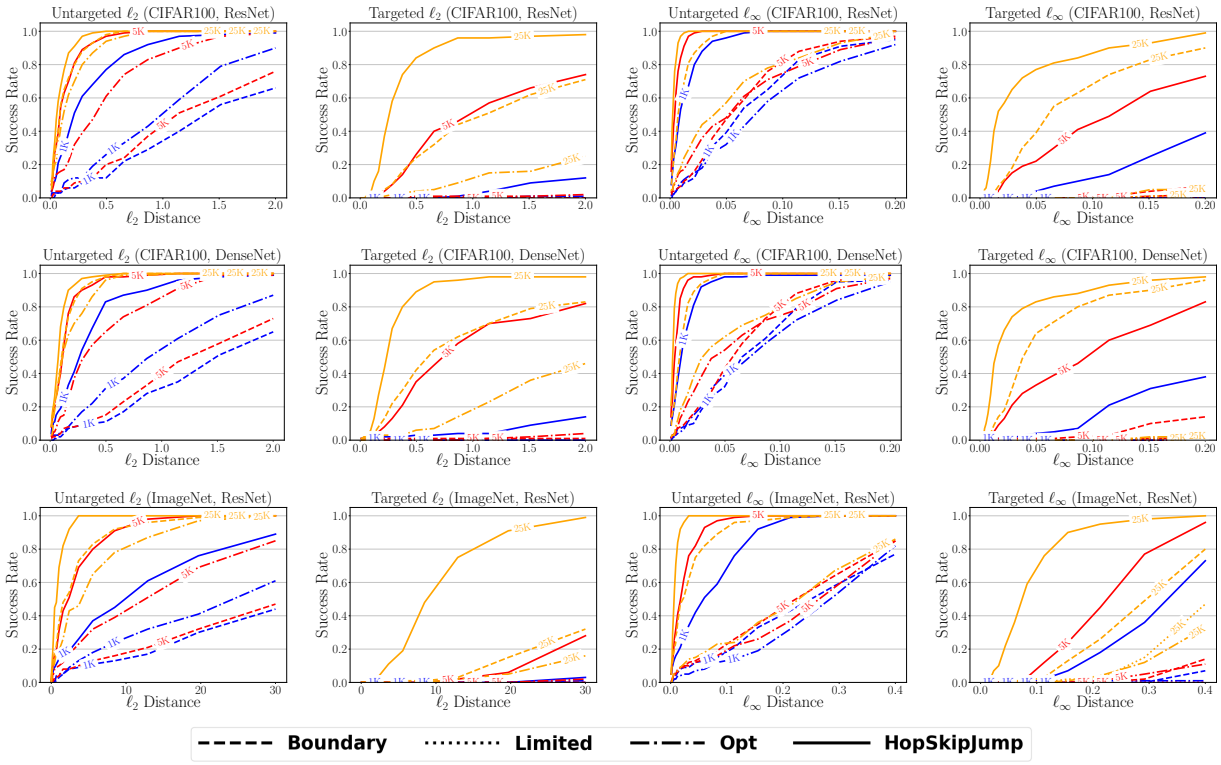


Figure 6.7: Success rate versus distance threshold for CIFAR-100 with ResNet, DenseNet, and ImageNet with ResNet from top to bottom rows. 1st column: untargeted ℓ_2. 2nd column: targeted ℓ_2. 3rd column: untargeted ℓ_∞. 4th column: targeted ℓ_∞.

Figure 6.6 and 6.7 show the success rate against the distance threshold. Figure 6.4 and 6.6 contain results on MNIST with CNN, and CIFAR-10 with ResNet, Denset, subsequently from the top row to the bottom row. Figure 6.5 and 6.7 contain results on CIFAR-100 with ResNet and DenseNet, and ImageNet with ResNet, subsequently from the top row to the bottom row. The four columns are for untargeted ℓ_2, targeted ℓ_2, untargeted ℓ_∞ and targeted ℓ_∞ attacks respectively.

With a limited number of queries, HopSkipJumpAttack is able to craft adversarial examples of a significantly smaller distance with the corresponding original examples across all data sets, followed by Boundary Attack and Opt Attack. As a concrete example, Table 6.1 shows that untargeted ℓ_2-optimized HopSkipJumpAttack achieves a median distance of 0.559 on CIFAR-10 with a ResNet model at $1,000$ queries, which amounts to below $3/255$ per pixel on average. At the same budget of queries, Boundary Attack and Opt Attack only achieve median ℓ_2-distances of 2.78 and 2.07 respectively. The difference in efficiency becomes more significant for ℓ_∞ attacks. As shown in Figure 6.6, under an untargeted ℓ_∞-optimized HopSkipJumpAttack with 1,000 queries, all pixels are within an $8/255$-neighborhood of the original image for around 70% of adversarial examples, a success rate achieved by Boundary Attack only after 20,000 queries.

By comparing the odd and even columns of Figure 6.4-6.7, we can find that targeted

HopSkipJumpAttack takes more queries than the untargeted one to achieve a comparable distance. This phenomenon becomes more explicit on CIFAR-100 and ImageNet, which have more classes. With the same number of queries, there is an order-of-magnitude difference in median distance between untargeted and targeted attacks (Figure 6.4 and 6.5). For ℓ_2-optimized HopSkipJumpAttack, while the untargeted version is able to craft adversarial images by perturbing 4 bits per pixel on average within 1,000 queries for $70\% - 90\%$ of images in CIFAR-10 and CIFAR-100, the targeted counterpart takes 2,000-5,000 queries. The other attacks fail to achieve a comparable performance even with 25,000 queries. On ImageNet, untargeted ℓ_2-optimized HopSkipJumpAttack is able to fool the model with a perturbation of size 6 bits per pixel on average for close to 50% of images with $1,000$ queries; untargeted ℓ_∞-optimized HopSkipJumpAttack controls the maximum perturbation across all pixels within 16 bits for 50% images within $1,000$ queries. The targeted Boundary Attack is not able to control the perturbation size to such a small scale until after around $25,000$ queries. On the one hand, the larger query budget requirement results from a strictly more powerful formulation of targeted attack than untargeted attack. On the other hand, this is also because we initialize targeted HopSkipJumpAttack from an arbitrary image in the target class. The algorithm may be trapped in a bad local minimum with such an initialization. Future work can address systematic approaches to better initialization.

As a comparison between data sets and models, we see that adversarial images often have a larger distance to their corresponding original images on MNIST than on CIFAR-10 and CIFAR-100, which has also been observed in previous work (e.g., [29]). This might be because it is more difficult to fool a model on simpler tasks. On the other hand, HopSkipJumpAttack also converges in a fewer number of queries on MNIST, as is shown in Figure 6.4. It does not converge even after $25,000$ queries on ImageNet. We conjecture the query budget is related to the input dimension, and the smoothness of decision boundary. We also observe the difference in model structure does not have a large influence on decision-based algorithms, if the training algorithm and the data set keep the same. For ResNet and DenseNet trained on a common data set, a decision-based algorithm achieves comparable performance in crafting adversarial examples, although DenseNet has a more complex structure than ResNet.

As a comparison with state-of-the-art white-box targeted attacks, C&W attack [29] achieves an average ℓ_2-distance of 0.33 on CIFAR-10, and BIM [27] achieves an average ℓ_∞-distance of 0.014 on CIFAR-10. Targeted HopSkipJumpAttack achieves a comparable distance with 5K-10K model queries on CIFAR-10, without access to model details. On ImageNet, targeted C&W attack and BIM achieve an ℓ_2-distance of 0.96 and an ℓ_∞-distance of 0.01 respectively. Untargeted HopSkipJumpAttack achieves a comparable performance with $10,000 - 15,000$ queries. The targeted version is not able to perform comparably as targeted white-box attacks when the budget of queries is limited within $25,000$.

Visualized trajectories of HopSkipJumpAttack optimized for ℓ_2 distances along varied queries on CIFAR10 and ImageNet can be found in Figure 6.8. On CIFAR-10, we observe untargeted adversarial examples can be crafted within around 500 queries; targeted HopSkipJumpAttack is capable of crafting human indistinguishable targeted adversarial examples within around $1,000 - 2,000$ queries. On ImageNet, untargeted HopSkipJumpAttack is

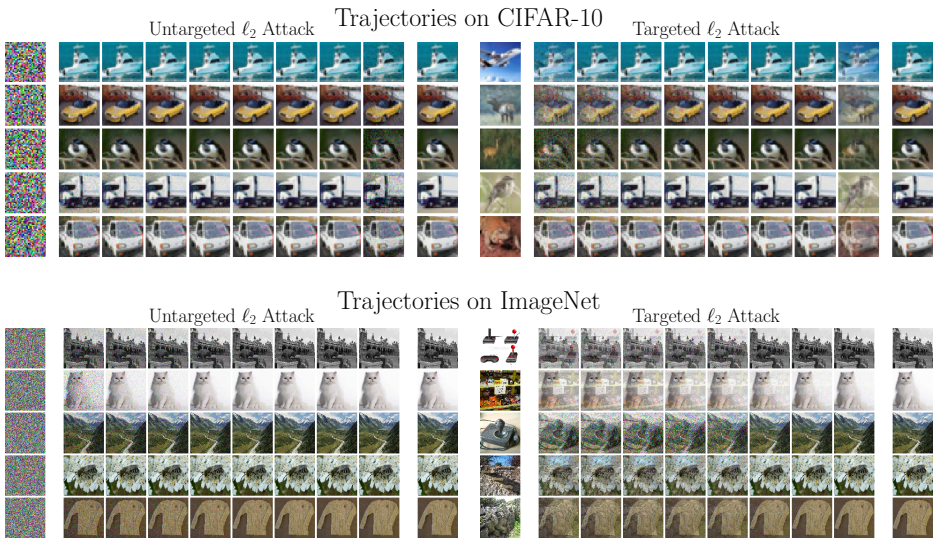


Figure 6.8: Visualized trajectories of HopSkipJumpAttack for optimizing ℓ_2 distance on randomly selected images in CIFAR-10 and ImageNet. 1st column: initialization (after blended with original images). 2nd-9th columns: images at 100, 200, 500, 1K, 2K, 5K, 10K, 25K model queries. 10th column: original images.

able to craft good adversarial examples with $1,000$ queries, while targeted HopSkipJumpAttack takes $10,000 - 20,000$ queries.

6.6.2 Defense mechanisms under decision-based attacks

We investigate the robustness of various defense mechanisms under decision-based attacks.

Defense mechanisms Three defense mechanisms are evaluated: defensive distillation, region-based classification, and adversarial training. Defensive distillation [44], a form of gradient masking [37], trains a second model to predict the output probabilities of an existing model of the same structure. We use the implementaion provided by Carlini and Wagner [29] for defensive distillation. The second defense, region-based classification, belongs to a wide family of mechanisms which add test-time randomness to the inputs or the model, causing the gradients to be randomized [148]. Multiple variants have been proposed to randomize the gradients [47, 149–152]. We adopt the implementation in Cao and Gong [149] with suggested noise levels. Given a trained base model, region-based classification samples points from the hypercube centered at the input image, predicts the label for each sampled point with the base model, and then takes a majority vote to output the label. Adversarial training [26, 27, 30, 41] is known to be one of the most effective defense mechanisms against adversarial perturbation [148, 153]. We evaluate a publicly available model trained through a robust optimization method proposed by Madry et al. [30].

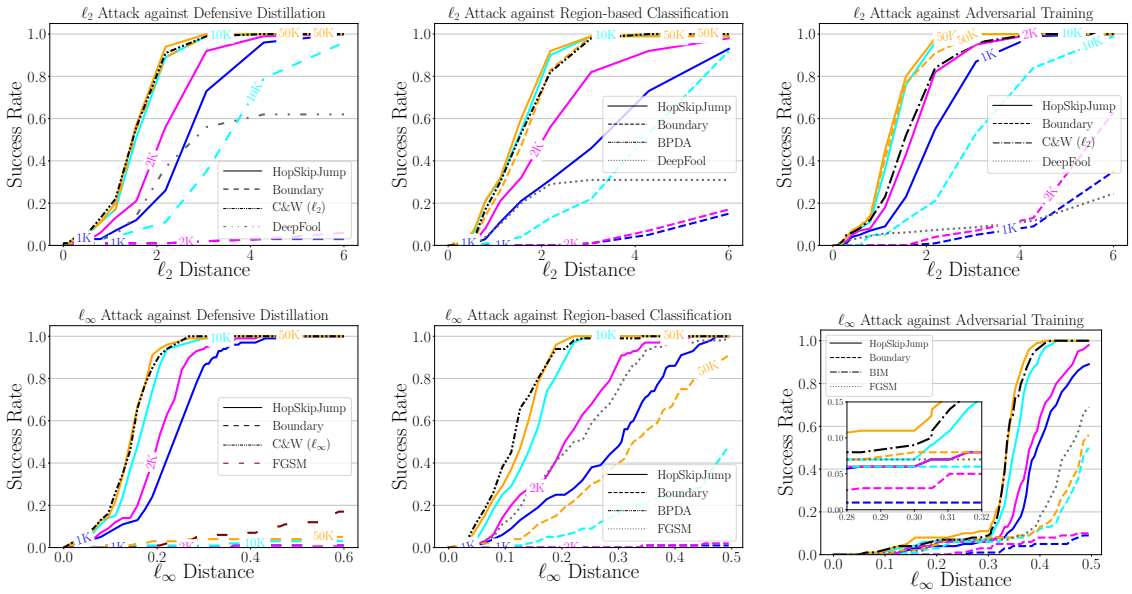


Figure 6.9: Success rate versus distance threshold for a distilled model, a region-based classifier and an adversarially trained model on MNIST. Blue, magenta, cyan and orange lines are used for HopSkipJumpAttack and Boundary Attack at the budget of 1K, 2K, 10K and 50K respectively. Different attacks are plotted with different line styles. An amplified figure is included near the critical ℓ_∞-distance of 0.3 for adversarial training.

Baselines We compare our algorithm with state-of-the-art attack algorithms that require access to gradients, including C&W Attack [29], DeepFool [31] for minimizing ℓ_2-distance, and FGSM [26], and BIM [30, 154] for minimizing ℓ_∞-distance. For region-based classification, the gradient of the base classifier is taken with respect to the original input.

We further include methods designed specifically for the defense mechanisms under threat. For defensive distillation, we include the ℓ_∞-optimized C&W Attack [29]. For region-based classification, we include backward pass differentiable approximation (BPDA) [148], which calculates the gradient of the model at a randomized input to replace the gradient at the original input in C&W Attack and BIM. All of these methods assume access to model details or even defense mechanisms, which is a stronger threat model than the one required for decision-based attacks. We also include Boundary Attack as a decision-based baseline.

For HopSkipJumpAttack and Boundary Attack, we include the success rate at three different scales of query budget: 2K, 10K and 50K, so as to evaluate our method both with limited queries and a sufficient number of queries. We find the convergence of HopSkipJumpAttack becomes unstable on region-based classification, resulting from the difficulty of locating the boundary in the binary search step when uncertainty is increased near the boundary. Thus, we increase the binary search threshold to 0.01 to resolve this issue.

Results Figure 6.9 shows the success rate of various attacks at different distance thresholds for the three defense mechanisms. On all of the three defenses, HopSkipJumpAttack

Data	Model							
	LR	L-SVM	K-SVM	KNN	RF	GBT	SIFT	HOG
MNIST	0.926	0.918	0.945	0.971 (3)	0.947	0.967	0.880	0.899
b-CIFAR10	0.606	0.576	0.626	0.601 (50)	0.610	0.683	0.625	0.660
CIFAR10	0.405	0.389	N/A	N/A	0.355	0.502	0.349	0.495

Table 6.2: Accuracy of different models on natural test images for the three data sets.

demonstrates similar or superior performance compared to state-of-the-art white-box attacks with sufficient model queries. Even with only 1K-2K model queries, it also achieves acceptable performance, although worse than the best white-box attacks. With sufficient queries, Boundary Attack achieves a comparable performance under the ℓ_2-distance metric. But it is not able to generate any adversarial examples when the number of queries is limited to $1,000$. We think this is because the strength of our batch gradient direction estimate over the random walk step in Boundary Attack becomes more explicit when there is uncertainty or non-smoothness near the decision boundary. We also observe that Boundary Attack does not work in optimizing the ℓ_∞-distance metric for adversarial examples, making it difficult to evaluate defenses designed for ℓ_∞ distance, such as adversarial training proposed by Madry et al. [30].

On a distilled model, when the ℓ_∞-distance is thresholded at 0.3, a perturbation size proposed by Madry et al. [30] to measure adversarial robustness, HopSkipJumpAttack achieves success rates of 86% and 99% with 1K and 50K queries respectively. At an ℓ_2-distance of 3.0, the success rate is 91% with 2K queries. HopSkipJumpAttack achieves a comparable performance with C&W attack under both distance metrics with 10K-50K queries. Also, gradient masking [37] by defensive distillation does not have a large influence on the query efficiency of HopSkipJumpAttack, indicating that the gradient direction estimate is robust under the setting where the model does not have useful gradients for certain white-box attacks.

On region-based classification, with 2K queries, HopSkipJumpAttack achieves success rates of 82% and 93% at the same ℓ_∞- and ℓ_2-distance thresholds respectively. With 10K-50K queries, it is able to achieve a comparable performance to BPDA, a white-box attack tailored to such defense mechanisms. On the other hand, we observe that HopSkipJumpAttack converges slightly slower on region-based classification than itself on ordinary models, which is because stochasticity near the boundary may prevent binary search in HopSkipJumpAttack from locating the boundary accurately.

On an adversarially trained model, HopSkipJumpAttack achieves a success rate of 11.0% with 50K queries when the ℓ_∞-distance is thresholded at 0.3. As a comparison, BIM has a success rate of 7.4% at the given distance threshold. The success rate of ℓ_∞-HopSkipJumpAttack transfers to an accuracy of 87.58% on adversarially perturbed data, close to the state-of-the-art performance achieved by white-box attacks.[2] With 1K queries, HopSkipJumpAttack also achieves comparable performance to BIM and C&W attack.

[2]See `https://github.com/MadryLab/mnist_challenge`

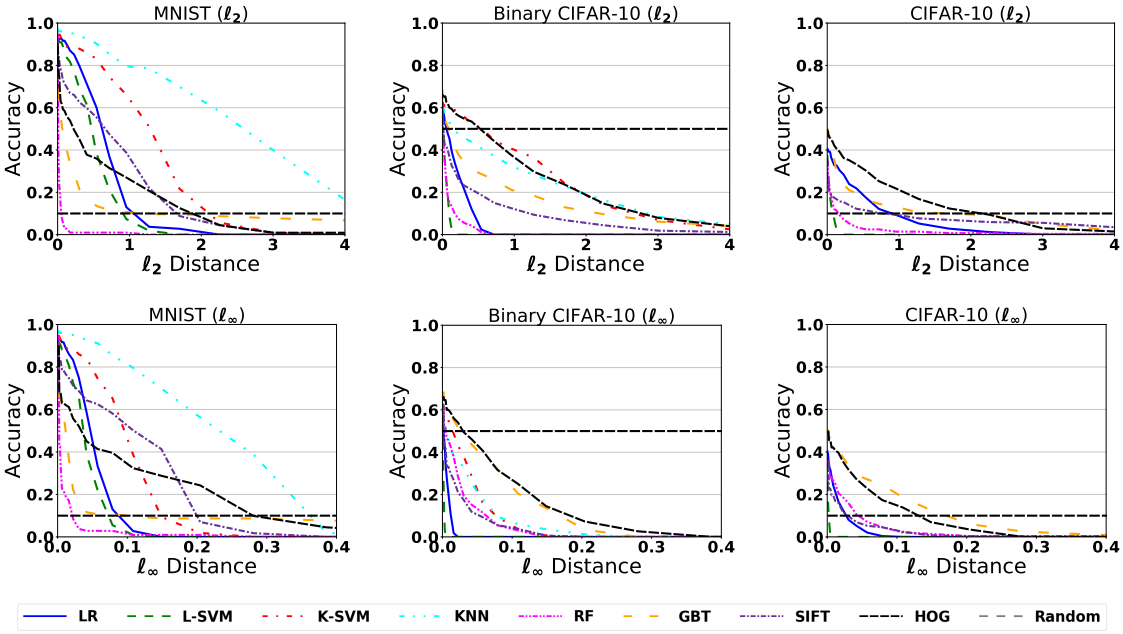


Figure 6.10: Accuracy of various models on adversarial examples at different distance thresholds. The accuracy of random guess is shown with the dotted gray line.

6.6.3 Robustness of other common image classifiers under decision-based attack

We study the robustness of several other common image classifiers, including logistic regression (LR), linear support vector machine (L-SVM), kernel support vector machine (K-SVM), k-nearest neighbors (KNN), random forest (RF), gradient boosted trees (GBT), scale invariant feature transform (SIFT), and histogram of oriented gradients (HOG). Among them, LR, L-SSVM, and K-SVM are differentiable with respect to the input, while KNN, RF, GBT, SIFT, and HOG are nondifferentiable with respect to the input. LR, L-SVM, K-SVM, KNN, and RF are implemented in scikit-learn; GBT is implemented in LightGBM, and SIFT and HOG are implemented in OpenCV.

Logistic Regression (LR): The logistic regression constructs a linear model between raw pixels and the log-odds of output classes. ℓ_2 regularization is added to the linear coefficients, with the penalty parameter chosen to be 1.

Linear Support Vector Machine (L-SVM): The linear SVM is a linear model of raw pixels trained with a hinge loss plus an ℓ_2 regularization on the linear coefficients. We choose the penalty parameter to be 10^{-5}, and use multiple one-vs-rest classifiers for multi-class data.

Kernel Support Vector Machine (K-SVM): Kernel SVM replaces the dot product between linear coefficients and raw pixels by a kernel function, and fits a nonlinear model with a hinge loss plus an ℓ_2 regularization by applying the kernel trick. We use the RBF kernel, choose the penalty parameter to be 1, and use multiple one-vs-rest classifiers for multi-class data.

Distance	Data	Objective	Model							
			LR	L-SVM	K-SVM	KNN	RF	GBT	SIFT	HOG
ℓ_2	MNIST	Untargeted	0.63	0.48	1.35	2.79	0.01	0.08	1.69	0.30
		Targeted	1.09	0.83	2.08	3.19	0.05	1.34	2.42	1.36
	b-CIFAR10	Targeted	0.20	0.05	1.43	1.09	0.04	0.31	0.15	1.16
	CIFAR10	Untargeted	0.44	0.03	N/A	N/A	0.06	0.18	0.10	0.77
		Targeted	1.17	0.10	N/A	N/A	0.19	2.63	1.56	2.31
ℓ_∞	MNIST	Untargeted	0.047	0.039	0.099	0.263	0.002	0.014	0.155	0.043
		Targeted	0.087	0.064	0.161	0.304	0.029	0.152	0.218	0.249
	b-CIFAR10	Targeted	0.005	0.001	0.039	0.054	0.022	0.075	0.010	0.075
	CIFAR10	Untargeted	0.013	0.001	N/A	N/A	0.029	0.073	0.010	0.049
		Targeted	0.034	0.003	N/A	N/A	0.096	0.155	0.088	0.151

Table 6.3: Median distance between adversarial examples and the corresponding original examples for different image classifiers, distance types, data sets, and objectives.

K-Nearest Neighbors (KNN): The k-nearest neighbors algorithm classifies an input by a plurality vote of the k nearest neighbors of a given input, in terms of the ℓ_2 distance in the raw pixel space. The number of neighbors k is chosen by cross validation, which is 3 for MNIST and 50 for binary CIFAR-10.

Random Forests (RF): The algorithm for training random forests applies bootstrap aggregating to tree learners. We include ten trees in the forest, and use the Gini impurity as split criterion. For each split, $\sqrt{d}$ randomly selected features are used.

Gradient Boosted Trees (GBT): Gradient boosting trees take each decision tree as a weak learner, and combine multiple trees iteratively into a single strong learner. We use a learning rate of 0.05 to boost 100 trees. For each tree, we set the minimum number of samples in each leaf to be 10, use 70% of randomly selected d features, and use 70% randomly sampled data.

Scale Invariant Feature Transform (SIFT): The scale-invariant feature transform (SIFT) detect and describe local features in images by extracting keypoints and compute its descriptors. K-means clustering with $K = 2,000$ is performed for keypoint features of all the training images. A histogram is built for the frequency of occurrences of cluster centers for each image. A linear SVC with penalty parameter 1 is fit on histogram representations of the training data.

Histogram of Oriented Gradients (HOG): The histogram of oriented gradients counts occurrences of gradient orientation in localized portions of an image. A linear SVC with penalty parameter 1 is fit on histogram representations of the training data.

We carry out the experiments on three data sets, MNIST, binary CIFAR-10 and CIFAR-10. For binary CIFAR-10, we reduce classification on CIFAR-10 to a binary classification problem by keeping images of cat and dog alone. We include the binary CIFAR-10 because fitting K-SVM and KNN on CIFAR-10 causes memory error on a c5.xlarge node with 8GB memory. Table 6.2 shows the accuracy of various classifiers on the three data sets.

HopSkipJumpAttack is applied to the generation of adversarial examples. Both untar-

Figure 6.11: Pairs of original and corresponding adversarial images under targeted ℓ_2 attack. Attacks are against LR, L-SVM, K-SVM, KNN, RF, GBT, SIFT and HOG respectively from the top row to the bottom row. The leftmost five columns are on MNIST, the middle five columns are on binary CIFAR-10, and the rightmost five columns are on CIFAR-10.

geted and targeted attacks with ℓ_2 and ℓ_∞ constraints are carried out. For untargeted attack, we run HopSkipJumpAttack for 27 iterations across all models and data sets, which amount to a budget of $10,000$ model queries. For targeted attack, we initialize from three different images at the target class when attacking a single image, so as to prevent the algorithm from being trapped in a bad local minimum. From each initialization, we run HopSkipJumpAttack for 45 iterations, which amount to a budget of $20,000$ model queries. The smallest perturbation among all initializations is reported.

The accuracy of various models on adversarial examples in the test set at various distance thresholds for the three data sets is plotted in Figure 6.10. We observe that most of the image classifiers being tested are vulnerable to decision-based attack. On MNIST, the left column of Figure 6.10 shows that all models except KNN performs comparably or worse than random guess (10%) under ℓ_2 perturbation of size 2, or ℓ_∞ perturbation of size 0.3. On binary CIFAR-10, all models achieve an accuracy worse than random guess (50%) at ℓ_2 perturbation of size 1, or ℓ_∞ perturbation of size 0.03 (approximately 8/255). On CIFAR-10, all models except GBT and HOG achieve an accuracy worse than random guess (50%) at ℓ_2 perturbation of size 1, or ℓ_∞ perturbation of size 0.03 (approximately 8/255). KNN, GBT and HOG are the most robust against HopSkipJumpAttack among all image classifiers.

We also report the median distance between adversarial and original images in Table 6.3 as another metric. Similar to the observation made previously, we find all models are vulnerable to untargeted decision-based attack. On the other hand, several models yield a certain level of robustness against targeted decision-based attack. KNN requires an ℓ_∞ perturbation of size 0.304 on average, or an ℓ_2 perturbation of size 3.19 on MNIST, and GBT and HOG requires a median ℓ_∞ distance of size 0.155 and 0.151, or a median ℓ_2 distance of size 2.63 and 2.31 for targeted perturbation on CIFAR-10.

The visualization of adversarial examples for different models under targeted ℓ_2 attack are shown in Figure 6.11. On MNIST, differences between original and adversarial images against KNN (the fourth row) and HOG (the bottom row) can sometimes be observed by human. As an example, we can clearly see the digit "4" on the fourth row has been perturbed towards "9". We may also observe noisy pixels on adversarial examples for GBT for binary

CIFAR-10 and CIFAR-10. Under most circumstances, humans are able to tell the true label of an adversarial image generated by HopSkipJumpAttack.

While HopSkipJumpAttack is able to generate human-indistinguishable adversarial examples on these models, we also note that convergence of our algorithm cannot be directly established by Theorem 8 for KNN, RF, GBT, SIFT, and HOG, because of their non-differentiability with respect to the input. A meaningful future direction is to characterize the property of our algorithm theoretically on these models.

Given the observation that KNN and HOG are comparably more robust to adversarial perturbation, it is also worth investigating whether they can be combined into state-of-the-art image classifiers such as deep neural networks, so as to improve their performance under adversarial perturbation.

6.7 Proofs

For notational simplicity, we use the shorthand $S \equiv S_{x^\star}$ throughout the proofs.

6.7.1 Proof of Theorem 5

Proof. For notational simplicity, let us denote $\tau_t := \xi_t / \|\nabla S(x_t)\|_2$, so that the update (6.3) at iterate t can be rewritten as

$$x_{t+1} = \alpha_t x^\star + (1 - \alpha_t)(x_t + \tau_t \nabla S(x_t)). \tag{6.22}$$

Recalling our step size choice $\xi_t = \eta_t \|x_t - x^\star\|$ with $\eta_t := t^{-q}$, we have $\tau_t = \eta_t \frac{\|x_t - x^\star\|}{\|\nabla S(x_t)\|}$.

The squared distance ratio is

$$\frac{\|x_{t+1} - x^\star\|_2^2}{\|x_t - x^\star\|_2^2} = \frac{\|(1-\alpha)(\tau_t \nabla S(x_t) + x_t - x^\star)\|_2^2}{\|x_t - x^\star\|_2^2}. \tag{6.23}$$

By a second-order Taylor series, we have

$$0 = \left\langle \nabla S(x_t),\, x_{t+1} - x_t \right\rangle + \frac{1}{2}(x_{t+1} - x_t)^T H_t (x_{t+1} - x_t), \tag{6.24}$$

where $H_t = \nabla^2 S(\beta x_{t+1} + (1-\beta) x_t)$ for some $\beta \in [0, 1]$. Plugging equation (6.22) into equation (6.24) yields

$$\left\langle \nabla S(x_t),\, -\alpha v_t + \tau_t \nabla S(x_t) \right\rangle + \frac{1}{2}(-\alpha v_t + \tau_t \nabla S(x_t))^T H_t (-\alpha v_t + \tau_t \nabla S(x_t)) = 0, \tag{6.25}$$

where we define $v_t := x_t - x^\star + \tau_t \nabla S(x_t)$. This can be rewritten as a quadratic equation with respect to α:

$$v_t^T H_t v_t \alpha^2 - 2 \nabla S(x_t)^T (I + \tau_t H_t) v_t \alpha + \nabla S(x_t)^T (\tau_t^2 H_t + 2\tau_t I) \nabla S(x_t) = 0. \tag{6.26}$$

Solving for α yields

$$\alpha = \frac{\nabla S(x_t)^T(\tau_t^2 H_t + 2\tau_t I)\nabla S(x_t)}{2\nabla S(x_t)^T(I+\tau_t H_t)v_t} \cdot \frac{2}{1+\sqrt{1-\frac{v_t^T H_t v_t \nabla S(x_t)^T(\tau_t^2 H_t + 2\tau_t I)\nabla S(x_t)}{(\nabla S(x_t)^T(I+\tau_t H_t)v_t)^2}}} \tag{6.27}$$

$$\geq \frac{\nabla S(x_t)^T(\tau_t^2 H_t + 2\tau_t I)\nabla S(x_t)}{2\nabla S(x_t)^T(I+\tau_t H_t)v_t}. \tag{6.28}$$

In order to simplify the notation, define $\nabla_t := \nabla S(x_t)$ and $d_t := x_t - x^\star$, which leads to

$$\alpha \geq \frac{\nabla_t^T(\frac{1}{2}\tau_t^2 H + \tau_t I)\nabla_t}{\nabla_t^T(I+\tau_t H)(d_t + \tau_t \nabla_t)}.$$

Hence, we have

$$\begin{aligned}
(1-\alpha)^2 &\leq \Big(1 - \frac{\nabla_t^T(\frac{1}{2}\tau_t^2 H_t + \tau_t I)\nabla_t}{\nabla_t^T(I+\tau_t H_t)(d_t+\tau_t\nabla_t)}\Big)^2 \\
&= \Big(\frac{\frac{1}{2}\tau_t^2\nabla_t^T H_t\nabla_t + \nabla_t^T d_t + \tau_t\nabla_t^T H_t d_t}{\tau_t\nabla_t^T\nabla_t + \tau_t^2\nabla_t^T H_t\nabla_t + \nabla_t^T d_t + \tau_t\nabla_t^T H_t d_t}\Big)^2 \\
&\leq \Big(\frac{\frac{1}{2}\tau_t^2 L\|\nabla_t\|^2 + \nabla_t^T d_t + \tau_t L\|d_t\|\|\nabla_t\|}{(\tau_t + \frac{1}{2}\tau_t^2 L)\|\nabla_t\|^2 + \nabla_t^T d_t + \tau_t L\|d_t\|\|\nabla_t\|}\Big)^2 \\
&= \Big(\frac{r_t + (\frac{1}{2}\eta_t^2 + \eta_t)L\frac{\|d_t\|_2}{\|\nabla_t\|_2}}{\eta_t + r_t + (\frac{1}{2}\eta_t^2+\eta_t)L\frac{\|d_t\|_2}{\|\nabla_t\|_2}}\Big)^2 \\
&\leq \Big(\frac{r_t + \eta_t\cdot\frac{3}{2}L\frac{\|d_t\|_2}{\|\nabla_t\|_2}}{r_t + \eta_t\cdot(1+\frac{3}{2}L\frac{\|d_t\|_2}{\|\nabla_t\|_2})}\Big)^2.
\end{aligned}$$

where

$$r_t = \frac{\langle x_t - x^\star, \nabla S(x_t)\rangle}{\|x_t - x^\star\|_2\|\nabla S(x_t)\|_2} = \frac{\langle d_t, \nabla_t\rangle}{\|d_t\|_2\|\nabla_t\|_2}. \tag{6.29}$$

Let $\kappa_t := \frac{3}{2}L\frac{\|d_t\|_2}{\|\nabla_t\|_2}$. Then we have κ_t is bounded:

$$\kappa_t \leq \frac{3}{2}L\frac{\|x_0 - x^\star\|_2}{c}. \tag{6.30}$$

Equation (6.23) and the bound on $(1-\alpha)^2$ yield

$$\frac{\|x_{t+1} - x^\star\|_2^2}{\|x_t - x^\star\|_2^2} = (1-\alpha)^2\cdot\Big(\frac{\tau_t^2\|\nabla S(x_t)\|^2 + 2\tau_t\langle\nabla S(x_t), x_t - x^\star\rangle}{\|x_t - x^\star\|^2} + 1\Big) \tag{6.31}$$

$$= (1-\alpha)^2\cdot(\eta_t^2 + 2\eta_t r_t + 1) \tag{6.32}$$

$$\leq \Big(\frac{r_t + \eta_t\kappa_t}{r_t + \eta_t(1+\kappa_t)}\Big)^2\cdot(\eta_t^2 + 2\eta_t r_t + 1). \tag{6.33}$$

We will show the convergence of r_t to 1 in two steps. In the first step, we show $\lambda_t := \frac{\eta_t}{r_t} \to 0$ by contradiction. In the second step, we establish the convergence rate of r_t to 1 based on the result of the first step.

Assume there exists a subsequence λ_{t_i} of λ_t that is bounded away from 0, such that $\lambda_{t_i} > c_1$ for some constant c_1. (Note that we always have $r_t > 0$ as x_t is on the target side of the boundary for any t.)

Define $\theta_t := \left(\frac{r_t+\eta_t\kappa_t}{r_t+\eta_t(1+\kappa_t)}\right)^2 \cdot (\eta_t^2 + 2\eta_t r_t + 1)$. Then we have

$$\theta_{t_i} \leq \Big(\frac{1+c_1\kappa_t}{1+c_1(1+\kappa_t)}\Big)^2 \cdot (\eta_{t_i}^2 + 2\eta_{t_i} + 1). \tag{6.34}$$

As $\eta_{t_i} \to 0$, there exists a constant $c_2 > 0$ such that $\theta_{t_i} < 1 - c_2$ for i large enough.

Because θ_t is an increasing function of r_t, we have

$$\theta_t \leq \Big(\frac{1+\eta_t\kappa_t}{1+\eta_t(1+\kappa_t)}\Big)^2 \cdot (\eta_t^2 + 2\eta_t + 1) \tag{6.35}$$

$$= \frac{1 + 2\eta_t + 2\kappa_t\eta_t + \mathcal{O}(\eta_t^2)}{1 + 2\eta_t + 2\kappa_t\eta_t + \mathcal{O}(\eta_t^2)} \tag{6.36}$$

$$= 1 + \mathcal{O}(t^{-2q}). \tag{6.37}$$

The product of $1 + t^{-2q}$ over t from 1 to ∞ is finite when $q > \frac{1}{4}$:

$$\Pi_{t=1}^{\infty}(1 + t^{-2q}) = \exp\{\sum_{t=1}^{\infty} \log(1 + t^{-2q})\} \tag{6.38}$$

$$\leq \exp\{\sum_{t=1}^{\infty} t^{-2q}\} < \infty. \tag{6.39}$$

Therefore, we have

$$\Pi_{t=1}^{\infty}\theta_t \leq \Pi_{i=1}^{\infty}\theta_{t_i} \cdot \Pi_{t=1}^{\infty}(1 + \mathcal{O}(t^{-2q})) = 0, \tag{6.40}$$

which implies x_t converges to $x^\star$, a contradiction. Therefore, $\lambda_t \to 0$ as $t \to \infty$.

Now we can establish the convergence of r_t to 1. We expand θ_t as

$$\theta_t = \frac{(1 + 2\lambda_t\kappa_t + \lambda_t^2\kappa_t^2)(\eta_t^2 + 2\eta_t r_t + 1)}{1 + 2\lambda_t(1+\kappa_t) + \lambda_t^2(1+\kappa_t)^2} \tag{6.41}$$

$$= \frac{1 + 2\lambda_t\kappa_t + \lambda_t^2\kappa_t^2 + 2\lambda_t r_t^2}{1 + 2\lambda_t\kappa_t + \lambda_t^2(\kappa_t+1)^2 + 2\lambda_t} + \frac{\eta_t^2(4\kappa_t + (1+\lambda_t\kappa_t)^2 + 2\lambda_t\kappa_t^2)}{1 + 2\lambda_t\kappa_t + \lambda_t^2(\kappa_t+1)^2 + 2\lambda_t}$$

$$\leq \frac{1 + 2\lambda_t\kappa_t + \lambda_t^2\kappa_t^2 + 2\lambda_t r_t^2}{1 + 2\lambda_t\kappa_t + \lambda_t^2\kappa_t^2 + 2\lambda_t} + \eta_t^2(4\kappa_t + (1+\lambda_t\kappa_t)^2 + 2\lambda_t\kappa_t^2)$$

$$\leq 1 - \frac{2\lambda_t(1 - r_t^2)}{1 + 2\lambda_t\kappa_t + \lambda_t^2\kappa_t^2 + 2\lambda_t} + \eta_t^2(4\kappa_t + 2) \tag{6.42}$$

$$\leq 1 - \lambda_t(1 - r_t^2) + \eta_t^2(4\kappa_t + 2), \tag{6.43}$$

where Inequality (6.42) holds for large enough t. As the product of θ_t over t is positive, we have

$$\sum_{t=1}^{\infty} \log\theta_t = \log\Pi_{t=1}^{\infty}\theta_t > -\infty. \tag{6.44}$$

Given that $\eta_t^2(4\kappa_t + 2) = \Theta(t^{-2q})$, Equation (6.44) is equivalent to

$$\sum_{t=1}^{\infty} \lambda_t(1 - r_t^2) < \infty, \tag{6.45}$$

which implies $\lambda_t(1 - r_t^2) = o(t^{-\frac{1}{2}})$.

As $r_t \leq 1$ and $\eta_t = t^{-q}$, we have $\lambda_t = \mathcal{O}(t^{-q})$ and hence we have $1 - r_t = o(t^{-\frac{1}{2}+q})$. □

6.7.2 Proof of Theorem 6

The idea of the proof is to divide the unit sphere into three components: the upper cap along the direction of gradient, the lower cap opposite to the direction of gradient, and the annulus in between.

Proof. Let u be a random vector uniformly distributed on the sphere. By Taylor's theorem, for any $\delta \in (0, 1)$, we have

$$S(x_t + \delta u) = \delta \nabla S(x_t)^T u + \frac{1}{2}\delta^2 u^T \nabla^2 S(x')u. \tag{6.46}$$

for some x' on the line between x_t and $x_t + \delta u$, where we have made use of the fact that $S(x_t) = 0$. As the function S has Lipschitz gradients, we can bound the second-order term as

$$|\frac{1}{2}\delta^2 u^T \nabla^2 S(x')u| \leq \frac{1}{2}L\delta^2. \tag{6.47}$$

Let $w := \frac{1}{2}L\delta$. By the Taylor expansion and the bound on the second order term by eigenvalues, when $\nabla S(x_t)^T u > w$, we have

$$\begin{aligned} S(x_t + \delta u) &\geq \delta \nabla S(x_t)^T u + \frac{1}{2}\delta^2 u^T \nabla^2 S(x')u \\ &\geq \delta(\nabla S(x_t)^T u - \frac{1}{2}L\delta) > 0. \end{aligned}$$

Similarly, we have $S(x_t + \delta u) < 0$ when $\nabla S(x_t)^T u < -w$. Therefore, we have

$$\phi_x(x_t + \delta u) = \begin{cases} 1 \text{ if } \nabla S(x_t)^T u > w, \\ -1 \text{ if } \nabla S(x_t)^T u < -w. \end{cases}$$

We expand the vector $\nabla S(x_t)$ to an orthogonal bases in $\mathbb{R}^d$: $v_1 = \nabla S(x_t)/\|\nabla S(x_t)\|_2, v_2, \ldots, v_d$. The random vector u can be expressed as

$$u = \sum_{i=1}^{d} \beta_i v_i,$$

where β is uniformly distributed on the sphere. Denote the upper cap as $E_1 := \{\nabla S(x_t)^T u > w\}$, the annulus as $E_2 := \{|\nabla S(x_t)^T u| < w\}$, and the lower cap as $E_3 := \{\nabla S(x_t)^T u < -w\}$. Let $p := \mathbb{P}(E_2)$ be the probability of event E_2. Thus we have $\mathbb{P}(E_1) = \mathbb{P}(E_3) = (1 - p)/2$. By symmetry, for any $i \neq 1$, we have

$$\mathbb{E}[\beta_i \mid E_1] = \mathbb{E}[\beta_i \mid E_3] = 0.$$

Therefore, the expected value of the estimator is

$$\begin{aligned}\mathbb{E}[\phi_x(x_t+\delta u)u] &= \big(\mathbb{E}[\phi_x(x_t+\delta u)u \mid E_1] + \mathbb{E}[\phi_x(x_t+\delta u)u \mid E_3]\big)\cdot(1-p)/2 + \\ &\quad \mathbb{E}[\phi_x(x_t+\delta u)u \mid E_2]\cdot p \\ &= (\mathbb{E}[\sum_{i=1}^{d}\beta_i v_i \mid E_1] + \mathbb{E}[-\sum_{i=1}^{d}\beta_i v_i \mid E_3])\cdot(1-p)/2 + \mathbb{E}[\phi_x(x_t+\delta u)u \mid E_2]\cdot p \\ &= \big(\mathbb{E}[\beta_1 v_1 \mid E_1] + \mathbb{E}[-\beta_1 v_1 \mid E_3]\big)\cdot(1-p)/2 + \mathbb{E}[\phi_x(x_t+\delta u)u \mid E_2]\cdot p \\ &= p\cdot\big(\mathbb{E}[\phi_x(x_t+\delta u)u \mid E_2] - \frac{1}{2}\mathbb{E}[\beta_1 v_1 \mid E_1] - \frac{1}{2}\mathbb{E}[-\beta_1 v_1 \mid E_3]\big) + \\ &\quad \mathbb{E}[\beta_1 v_1 \mid E_1] + \mathbb{E}[-\beta_1 v_1 \mid E_3]\end{aligned}$$

Exploiting the above derivation, we bound the difference between $\mathbb{E}[|\beta_1|v_1] = \frac{\mathbb{E}|\beta_1|}{\|\nabla S(x_t)\|_2}\nabla S(x_t)$ and $\mathbb{E}[\phi_x(x_t+\delta u)u]$. In fact, we have

$$\|\mathbb{E}[\phi_x(x_t+\delta u)u] - \mathbb{E}[|\beta_1|v_1]\|_2 \leq p\Big\|\mathbb{E}[\phi_x(x_t+\delta u)u \mid E_2] - \frac{1}{2}\mathbb{E}[\beta_1 v_1 \mid E_1] - \frac{1}{2}\mathbb{E}[-\beta_1 v_1 \mid E_3]\Big\|_2 \tag{6.48}$$

$$\begin{aligned}&\quad + \Big\|\mathbb{E}[|\beta_1|v_1 \mid E_1] + \mathbb{E}[|\beta_1|v_1 \mid E_3] - \mathbb{E}[|\beta_1|v_1]\Big\|_2 \\ &\leq 2p + p = 3p,\end{aligned} \tag{6.49}$$

which yields

$$\cos\angle\big(\mathbb{E}[\phi_x(x_t+\delta u)u], \nabla S(x_t)\big) \geq 1 - \frac{1}{2}\Big(\frac{3p}{\mathbb{E}|\beta_1|}\Big)^2. \tag{6.50}$$

We can bound p by observing that $\langle\frac{\nabla S(x_t)}{\|\nabla S(x_t)\|_2}, u\rangle^2$ is a Beta distribution $\mathcal{B}(\frac{1}{2}, \frac{d-1}{2})$:

$$\begin{aligned}p &= \mathbb{P}(|\nabla S(x_t)^T u| < w) \\ &= \mathbb{P}\Big(\langle\frac{\nabla S(x_t)}{\|\nabla S(x_t)\|_2}, u\rangle^2 \leq \frac{w^2}{\|\nabla S(x_t)\|_2^2}\Big) \\ &= \int_0^{\frac{w^2}{\|\nabla S(x_t)\|_2^2}} \frac{x^{-\frac{1}{2}}(1-x)^{\frac{d-3}{2}}}{\mathcal{B}(\frac{1}{2}, \frac{d-1}{2})}dx \\ &\leq \frac{2w}{\mathcal{B}(\frac{1}{2}, \frac{d-1}{2})\|\nabla S(x_t)\|_2}.\end{aligned}$$

Plugging into Equation (6.50), we get

$$\begin{aligned}
&\cos\angle\left(\mathbb{E}[\phi_x(x_t+\delta u)u], \nabla S(x_t)\right) \\
\geq\ & 1 - \frac{18w^2}{(\mathbb{E}|\beta_1|)^2\mathcal{B}(\frac{1}{2}, \frac{d-1}{2})^2\|\nabla S(x_t)\|_2^2} \\
\geq\ & 1 - \frac{9L^2\delta^2}{2(2/(d-1))^2\|\nabla S(x_t)\|_2^2} \\
=\ & 1 - \frac{9L^2\delta^2(d-1)^2}{8\|\nabla S(x_t)\|_2^2}
\end{aligned}$$

We also observe that

$$\mathbb{E}\widetilde{\nabla S}(x_t, \delta) = \mathbb{E}[\phi_x(x_t+\delta u)u].$$

As a consequence, we have established

$$\cos\angle\left(\mathbb{E}[\widetilde{\nabla S}(x_t, \delta)], \nabla S(x_t)\right) \geq 1 - \frac{9L^2\delta^2(d-1)^2}{8\|\nabla S(x_t)\|_2^2}.$$

Taking $\delta \to 0$, we get

$$\lim_{\delta\to 0} \cos\angle\left(\mathbb{E}[\widetilde{\nabla S}(x_t, \delta)], \nabla S(x_t)\right) = 1.$$

□

6.7.3 Proof of Lemma 2

Proof. Recall that $\widetilde{\nabla S}(x_t, \delta) = \frac{1}{B}\sum_{b=1}^{B} \phi_{x^\star}(x_t+\delta u_b)u_b$, where u_b's are i.i.d. uniformly distributed unit random vectors. Define $v_b := \phi_{x^\star}(x_t+\delta u_b)u_b - \mathbb{E}[\widetilde{\nabla S}(x_t, \delta)]$. We have that $\{v_b\}_{b=1}^B$ is an i.i.d. sequence of zero-mean random vectors such that

$$\begin{aligned}
\|v_b\|_2 &\leq \|\phi_{x^\star}(x_t+\delta u_b)u_b\|_2 + \|\mathbb{E}[\widetilde{\nabla S}(x_t, \delta)]\|_2 \\
&\leq 1 + 1 = 2.
\end{aligned}$$

The covariance matrix Σ of v_b is

$$\begin{aligned}
\Sigma &= \mathbb{E}[v_b v_b^T] - \mathbb{E}[v_b]\mathbb{E}[v_b]^T \\
&= \mathbb{E}[u_b u_b^T] - \mathbb{E}[v_b]\mathbb{E}[v_b]^T \\
&\preceq \mathbf{I}_d,
\end{aligned}$$

where $\mathbf{I}_d$ is the identity matrix, and $A \preceq B$ for two matrices A, B if and only if $B - A$ is positive semi-definite. By applying Bernstein's inequality for random matrices [140] to the $(d+1)$-dimensional symmetric matrices

$$\begin{bmatrix} 0, v_b^T \\ v_b, \mathbf{0}_d \end{bmatrix},$$

we can obtain the following Bernstein-type inequality on random variable $\|\sum_{b=1}^{B} v_b\|_2$ (See Exercise 6.13 of Wainwright [141])

$$\mathbb{P}\Big(\|\frac{1}{B}\sum_{b=1}^{B} v_b\| \geq s\Big) \leq 2(d+1)\exp\Big(-\frac{Bs^2}{8+4s}\Big). \tag{6.51}$$

That is,

$$\mathbb{P}\Big(\|\widetilde{\nabla S}(x_t,\delta) - \mathbb{E}[\widetilde{\nabla S}(x_t,\delta)]\| \geq s\Big) \leq 2(d+1)\exp\Big(-\frac{Bs^2}{8+4s}\Big). \tag{6.52}$$

□

6.7.4 Proof of Theorem 7

Proof. For notational simplicity, we denote $\widetilde{\nabla}_t := \widetilde{\nabla S}(x_t,\delta)$ and $\nabla_t = \nabla S(x_t)$. By definition, we have

$$\cos\angle\Big(\widetilde{\nabla}_t, \nabla S_{x^\star}(x_t)\Big) = 1 - \frac{1}{2}\Big\|\frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|}\Big\|^2. \tag{6.53}$$

By repeated application of triangle inequality, we have for any constant k,

$$\begin{aligned}
\Big\|\frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|}\Big\| &= \frac{1}{\|\nabla_t\|}\Big\|\widetilde{\nabla}_t\frac{\|\nabla_t\|}{\|\widetilde{\nabla}_t\|} - \nabla_t\Big\| \\
&\leq \frac{1}{\|\nabla_t\|}\Big(\Big\|k\widetilde{\nabla}_t - \nabla_t\Big\| + \Big\|\widetilde{\nabla}_t\frac{\|\nabla_t\|}{\|\widetilde{\nabla}_t\|} - k\widetilde{\nabla}_t\Big\|\Big) \\
&= \frac{1}{\|\nabla_t\|}\Big(\Big\|k\widetilde{\nabla}_t - \nabla_t\Big\| + \Big|\|\nabla_t\| - \|k\widetilde{\nabla}_t\|\Big|\Big) \\
&\leq \frac{2}{\|\nabla_t\|}\Big\|k\widetilde{\nabla}_t - \nabla_t\Big\| \\
&\leq \frac{2}{\|\nabla_t\|}\Big(k\Big\|\widetilde{\nabla}_t - \mathbb{E}[\widetilde{\nabla}_t]\Big\| + \Big\|k\mathbb{E}[\widetilde{\nabla}_t] - \nabla_t\Big\|\Big)
\end{aligned} \tag{6.54}$$

In the proof of Theorem 6 (Equation (6.49)), we have established that

$$\Big\|\mathbb{E}[\widetilde{\nabla}_t] - \sqrt{\frac{2}{\pi}}\frac{\nabla_t}{\|\nabla_t\|}\Big\| \leq \frac{3L\delta(d-1)}{2\|\nabla_t\|}.$$

As a result, letting $k := \sqrt{\frac{\pi}{2}}\|\nabla_t\|$, we get

$$\Big\|\frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|}\Big\| \leq \sqrt{2\pi}\Big\|\widetilde{\nabla}_t - \mathbb{E}[\widetilde{\nabla}_t]\Big\| + \sqrt{\frac{\pi}{2}}\frac{3L\delta(d-1)}{\|\nabla_t\|}. \tag{6.55}$$

Combining Equation (6.53) and Inequality (6.55), we have

$$\begin{aligned}
&\mathbb{P}\Big(\cos\angle \Big(\mathbb{E}[\widetilde{\nabla S}(x_t,\delta)], \nabla S_{x^\star}(x_t)\Big) < 1-a\Big) \\
&= \mathbb{P}\Big(\Big\| \frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|}\Big\| > \sqrt{2a}\Big) \\
&\leq \mathbb{P}\Big(\sqrt{2\pi}\Big\|\widetilde{\nabla}_t - \mathbb{E}[\widetilde{\nabla}_t]\Big\| + \sqrt{\frac{\pi}{2}}\frac{3L\delta(d-1)}{\|\nabla_t\|} > \sqrt{2a}\Big) \\
&\leq 2(d+1)\exp\Big\{ -\frac{B\big(\sqrt{\frac{a}{\pi}} - \frac{3L\delta(d-1)}{2\|\nabla_t\|}\big)^2}{8 + 4\big(\sqrt{\frac{a}{\pi}} - \frac{3L\delta(d-1)}{2\|\nabla_t\|}\big)}\Big\},
\end{aligned}$$

where the last inequality follows from Lemma 2 when $\frac{3L\delta(d-1)}{2\|\nabla_t\|} < \sqrt{\frac{a}{\pi}}$. When $\delta < \frac{\|\nabla_t\|}{3L(d-1)}\sqrt{\frac{a}{\pi}}$, we have $\frac{3L\delta(d-1)}{2\|\nabla_t\|} < \frac{1}{2}\sqrt{\frac{a}{\pi}}$, and therefore

$$\mathbb{P}\Big(\cos\angle \Big(\mathbb{E}[\widetilde{\nabla S}(x_t,\delta)], \nabla S_{x^\star}(x_t)\Big) < 1-a\Big) \leq 2(d+1)\exp\Big\{ -\frac{Ba}{32\pi + 8\sqrt{a\pi}}\Big\}.$$

□

6.7.5 Proof of Theorem 8

Proof. For notational simplicity, let us denote $\tau_t := \xi_t / \|\nabla S(x_t)\|_2$ and $u_t := \frac{\|\nabla S(x_t)\|}{\|\widehat{\nabla S}(x_t)\|}\widehat{\nabla S}(x_t)$, so that the update (6.3) at iterate t can be rewritten as

$$x_{t+1} = \alpha_t x^\star + (1-\alpha_t)(x_t + \tau_t u_t). \tag{6.56}$$

Recalling our step size choice $\xi_t = \eta_t \|x_t - x^\star\|$ with $\eta_t := t^{-q}$, we have $\tau_t = \eta_t \frac{\|x_t - x^\star\|}{\|\nabla S(x_t)\|}$. The squared distance ratio is

$$\frac{\|x_{t+1} - x^\star\|_2^2}{\|x_t - x^\star\|_2^2} = \frac{\|(1-\alpha)(\tau_t u_t + x_t - x^\star)\|_2^2}{\|x_t - x^\star\|_2^2}. \tag{6.57}$$

By a second-order Taylor series, we have

$$0 = \big\langle \nabla S(x_t),\, x_{t+1} - x_t \big\rangle + \frac{1}{2}(x_{t+1} - x_t)^T H_t (x_{t+1} - x_t), \tag{6.58}$$

where $H_t = \nabla^2 S(\beta x_{t+1} + (1-\beta)x_t)$ for some $\beta \in [0,1]$. Plugging equation (6.56) into equation (6.58) yields

$$\big\langle \nabla S(x_t),\, -\alpha v_t + \tau_t u_t \big\rangle + \frac{1}{2}(-\alpha v_t + \tau_t u_t)^T H_t (-\alpha v_t + \tau_t u_t) = 0, \tag{6.59}$$

where we define $v_t := x_t - x^\star + \tau_t u_t$. This can be rewritten as a quadratic equation with respect to α:

$$v_t^T H_t v_t \alpha^2 - 2(\tau_t u_t^T H_t v_t + \nabla S(x_t)^T v_t)\alpha + (\tau_t^2 u_t^T H_t u_t + 2\tau_t \nabla S(x_t)^T u_t) = 0. \tag{6.60}$$

Solving for α yields

$$\alpha = \frac{0.5\tau_t^2 u_t^T H_t u_t + \tau_t \nabla S(x_t)^T u_t}{\nabla S(x_t)^T v_t + \tau_t u_t^T H_t v_t} \cdot \frac{2}{1 \pm \sqrt{1 - \frac{v_t^T H_t v_t (\tau_t^2 u_t^T H_t u_t + 2\tau_t \nabla S(x_t)^T u_t) \nabla S(x_t)}{(\tau_t u_t^T H_t v_t + \nabla S(x_t)^T v_t)}}} \tag{6.61}$$

$$\geq \frac{0.5\tau_t^2 u_t^T H_t u_t + \tau_t \nabla S(x_t)^T u_t}{\nabla S(x_t)^T v_t + \tau_t u_t^T H_t v_t}. \tag{6.62}$$

The above inequality is true for large enough t when $\big\langle u_t, \nabla S(x_t) \big\rangle > 0$, which is true for small enough $\delta_t > 0$ based on the definition of $\widehat{\nabla S}$. In order to simplify the notation, define $\nabla_t := \nabla S(x_t)$, $\widetilde{\nabla}_t := \widehat{\nabla S}(x_t, \delta_t)$, and $d_t := x_t - x^\star$, which leads to

$$\alpha \geq \frac{0.5\tau_t^2 u_t^T H_t u_t + \tau_t \nabla_t^T u_t}{\nabla_t^T v_t + \tau_t u_t^T H_t v_t}.$$

Define the error $\varepsilon_t := u_t - \nabla S(x_t)$. We have

$$\begin{aligned}
(1-\alpha)^2 &= \Big(\frac{\frac{1}{2}\tau_t^2 u_t^T H_t u_t + \nabla_t^T d_t + \tau_t u_t^T H_t d_t}{\tau_t \nabla_t^T u_t + \tau_t^2 u_t^T H_t u_t + \nabla_t^T d_t + \tau_t u_t^T H_t d_t} \Big)^2 \\
&\leq \Big(\frac{\frac{1}{2}\tau_t^2 L \|\nabla_t\|^2 + \nabla_t^T d_t + \tau_t L \|d_t\| \|\nabla_t\|}{(\tau_t + \frac{1}{2}\tau_t^2 L)\|\nabla_t\|^2 + \nabla_t^T d_t + \tau_t L \|d_t\| \|\nabla_t\| + \tau_t \|\nabla_t\| \nabla_t^T \varepsilon_t} \Big)^2 \\
&= \Big(\frac{r_t + (\frac{1}{2}\eta_t^2 + \eta_t) L \frac{\|d_t\|}{\|\nabla_t\|}}{\eta_t + r_t + (\frac{1}{2}\eta_t^2 + \eta_t) L \frac{\|d_t\|}{\|\nabla_t\|} + \eta_t \big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_t \big\rangle} \Big)^2 \\
&\leq \Big(\frac{r_t + \eta_t \cdot \frac{3}{2} L \frac{\|d_t\|}{\|\nabla_t\|}}{r_t + \eta_t \cdot (1 + \frac{3}{2} L \frac{\|d_t\|}{\|\nabla_t\|}) + \eta_t \big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_t \big\rangle} \Big)^2.
\end{aligned}$$

where

$$r_t = \frac{\langle x_t - x^\star, \nabla S(x_t) \rangle}{\|x_t - x^\star\|_2 \|\nabla S(x_t)\|_2} = \frac{\langle d_t, \nabla_t \rangle}{\|d_t\|_2 \|\nabla_t\|_2}. \tag{6.63}$$

Let $\kappa_t := \frac{3}{2} L \frac{\|d_t\|_2}{\|\nabla_t\|_2}$. Then we have κ_t is bounded:

$$\kappa_t \leq \frac{3}{2} L \frac{\|x_0 - x^\star\|_2}{c}. \tag{6.64}$$

Equation (6.57) and the bound on $(1-\alpha)^2$ yield

$$\begin{aligned}
\frac{\|x_{t+1} - x^\star\|_2^2}{\|x_t - x^\star\|_2^2} &= (1-\alpha)^2 \cdot \big(\frac{\tau_t^2 \|\nabla_t\|^2 + 2\tau_t \langle u_t, d_t \rangle}{\|d_t\|^2} + 1 \big) \\
&= (1-\alpha)^2 \cdot (\eta_t^2 + 2\eta_t r_t + 1 + 2\eta_t \big\langle \varepsilon_t, \frac{d_t}{\|d_t\|} \big\rangle) \\
&\leq \Big(\frac{r_t + \eta_t \kappa_t}{r_t + \eta_t(1 + \kappa_t) + \eta_t \big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_t \big\rangle} \Big)^2 \cdot (\eta_t^2 + 2\eta_t r_t + 1 + 2\eta_t \big\langle \varepsilon_t, \frac{d_t}{\|d_t\|} \big\rangle).
\end{aligned} \tag{6.65}$$

By Cauchy Inequality, we have

$$\Big|\Big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_t \Big\rangle\Big| \le \|\varepsilon_t\| = \|\nabla_t\| \cdot \Big\| \frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|} \Big\|, \text{ and}$$

$$\Big|\Big\langle \frac{d_t}{\|d_t\|}, \varepsilon_t \Big\rangle\Big| \le \|\varepsilon_t\| = \|\nabla_t\| \cdot \Big\| \frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|} \Big\|$$

In the proof of Theorem 7, we have established that for any $a > 0$, when $\delta_t < \frac{a\|\nabla_t\|}{3L(d-1)\sqrt{2\pi}}$, we have

$$\mathbb{P}\Big(\Big\| \frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|} - \frac{\nabla_t}{\|\nabla_t\|} \Big\| > a \Big) < \exp\Big\{ - \frac{Ba^2}{64\pi + 8a\sqrt{2\pi}} \Big\}. \tag{6.66}$$

We will show the bound the deviation of r_t from 1 in two steps. In the first step, we show $\lambda_t := \frac{\eta_t}{r_t} \to 0$ by contradiction. In the second step, we establish the deviation bound based on the result of the first step.

Assume there exists a subsequence λ_{t_i} of λ_t that is bounded away from 0, such that $\lambda_{t_i} > c_1$ for some constant c_1. (Note that we always have $r_t > 0$ as x_t is on the target side of the boundary for any t.) Define $\theta_t := \Big(\frac{r_t + \eta_t \kappa_t}{r_t + \eta_t(1+\kappa_t) + \eta_t \Big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_t \Big\rangle} \Big)^2 \cdot (\eta_t^2 + 2\eta_t r_t + 1 + 2\eta_t \big\langle \varepsilon_t, \frac{d_t}{\|d_t\|} \big\rangle)$. Then we have

$$\begin{aligned}
\theta_{t_i} &= \Big(\frac{1 + \lambda_{t_i}\kappa_{t_i}}{1 + \lambda_{t_i}(1+\kappa_{t_i}) + \lambda_{t_i} \big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_{t_i} \big\rangle} \Big)^2 \cdot (\eta_{t_i}^2 + 2\eta_{t_i} r_{t_i} + 1 + 2\eta_{t_i} \big\langle \varepsilon_{t_i}, \frac{d_{t_i}}{\|d_{t_i}\|} \big\rangle) \\
&\le C \cdot \Big(\frac{1 + \lambda_{t_i}\kappa_{t_i}}{1 + \lambda_{t_i}(1+\kappa_{t_i}) + \lambda_{t_i} \big\langle \frac{\nabla_t}{\|\nabla_t\|}, \varepsilon_{t_i} \big\rangle} \Big)^2 \\
&\le C \cdot \Big(\frac{1 + c_1\kappa_{t_i}}{1 + c_1(1+\kappa_{t_i}) - c_1\|\varepsilon_{t_i}\|} \Big)^2
\end{aligned}$$

for some fixed constant C. By Equation (6.66) and $B = \sqrt{t}$, there exists some constant t_0, such that for any $t_i > t_0$,

$$\mathbb{P}\Big(\|\varepsilon_{t_i}\| < \frac{1}{2} \Big) > 0. \tag{6.67}$$

When $\|\varepsilon_{t_i}\| < \frac{1}{2}$, we have

$$\theta_{t_i} \le C \cdot \Big(\frac{1 + c_1\kappa_{t_i}}{1 + c_1(0.5 + \kappa_{t_i})} \Big)^2 < 1 - c_2, \text{ for some } c_2 \in (0, 1).$$

Moreover, for large enough t, we always have $\theta_t \le 1 + Ct^{-2q}$ for some given constant C. By the independence between ε_{t_i} with $i = 1, 2, \ldots$, we have that for any small constant $s > 0$, there exists T such that with nonzero probability, we have

$$\prod_{t=1}^{T} \theta_t \le \prod_{t_i < T} \theta_{t_i} \cdot \prod_{t=1}^{T} (1 + Ct^{-2q}) \le \prod_{t_i < T} (1 - c_2) \cdot \prod_{t=1}^{T} (1 + Ct^{-2q}) < s,$$

which implies x_t converges to $x^\star$, a contradiction. Therefore, $\lambda_t \to 0$ as $t \to \infty$.

Now we can establish the deviation bound. We expand θ_t as

$$\begin{aligned}\theta_t &= \frac{(1+2\lambda_t\kappa_t+\lambda_t^2\kappa_t^2)(\eta_t^2+2\eta_t r_t+1+2\eta_t\langle\varepsilon_t,\,\frac{d_t}{\|d_t\|}\rangle)}{1+2\lambda_t(1+\kappa_t)+\lambda_t^2(1+\kappa_t)^2+\lambda_t^2\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle^2+2\lambda_t\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle(1+\lambda_t(1+\kappa_t))}\\ &= \frac{1+2\lambda_t(r_t^2+\kappa_t+r_t\langle\varepsilon_t,\,\frac{d_t}{\|d_t\|}\rangle)+o(\eta_t)+\mathcal{O}(\lambda_t^2)}{1+2\lambda_t(1+\kappa_t+\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle)+\mathcal{O}(\lambda_t^2)}\\ &= 1-\frac{2\lambda_t(1-r_t^2+(\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle-r_t\langle\varepsilon_t,\,\frac{d_t}{\|d_t\|}\rangle)+o(\eta_t)+\mathcal{O}(\lambda_t^2)}{1+2\lambda_t(1+\kappa_t+\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle)}\end{aligned}$$

As the product of θ_t over t is positive, we have

$$\sum_{t=1}^{\infty}\log\theta_t=\log\Pi_{t=1}^{\infty}\theta_t>-\infty. \tag{6.68}$$

Equation (6.68) implies there exists a positive constant C, such that

$$1-\theta_t\le C\cdot t^{-\frac{1}{2}}. \tag{6.69}$$

Let the event E be

$$E:=\Big\{\Big|\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle-r_t\langle\varepsilon_t,\,\frac{d_t}{\|d_t\|}\rangle)\Big|>\frac{1}{4}C\cdot t^{-0.5+q}\Big\}\cup\Big\{\langle\frac{\nabla_t}{\|\nabla_t\|},\,\varepsilon_t\rangle<-1\Big\}$$

By Equation (6.66), for large t, when $\delta_t<\frac{Ct^{-0.5+q}}{24L(d-1)\sqrt{2\pi}}$, we have

$$\begin{aligned}\mathbb{P}\Big(E\Big)&\le\mathbb{P}\Big(2\Big\|\frac{\widetilde{\nabla}_t}{\|\widetilde{\nabla}_t\|}-\frac{\nabla_t}{\|\nabla_t\|}\Big\|>\frac{Ct^{-0.5+q}}{4\|\nabla_t\|}\Big)\\ &\le 2(d+1)\exp\Big\{-\frac{B\big(\frac{Ct^{-0.5+q}}{8\|\nabla_t\|}\big)^2}{64\pi+8\sqrt{2\pi}\big(\frac{Ct^{-0.5+q}}{8\|\nabla_t\|}\big)}\Big\}\\ &= 2(d+1)\exp\Big\{-B\cdot\frac{C^2t^{-1+2q}}{64\|\nabla_t\|^2(64\pi+\sqrt{2\pi}Ct^{-0.5+1}/\|\nabla_t\|)}\Big\}\\ &\le 2(d+1)\exp\Big(-\widetilde{C}Bt^{-1+2q}\Big),\end{aligned}$$

for some fixed constant $\widetilde{C}$. When E is false, we have

$$\theta_t\le 1-2\lambda_t(1-r_t^2-\frac{1}{4}C\cdot t^{-0.5+q}). \tag{6.70}$$

Inequality (6.69) and Inequality (6.70) jointly yield that $r_t\ge 1-\frac{1}{4}C\cdot t^{-0.5+q}$ with probability at least $1-2(d+1)\exp\Big(-\widetilde{C}Bt^{-1+2q}\Big)$. □

6.7.6 Proof of Theorem 9

Proof. For notational simplicity, we denote $\xi_b:=\phi_x(x_t+\delta u_b)$, and $\bar{\xi}=\frac{1}{B}\sum_{b=1}^{B}\xi_b=\overline{\phi_x}$. We use ξ,u to denote i.i.d. copies of ξ_b and u_b respectively. The variance of the estimate with

the baseline is

$$
\begin{aligned}
&\mathrm{Var}(\widehat{\nabla S}(x_t,\delta)) \\
&= \mathbb{E}\Big\langle \widehat{\nabla S}(x_t,\delta) - \mathbb{E}\widehat{\nabla S}(x_t,\delta), \widehat{\nabla S}(x_t,\delta) - \mathbb{E}\widehat{\nabla S}(x_t,\delta)\Big\rangle \\
&= \frac{1}{(B-1)^2}\Big\langle \sum_{b=1}^{B}\xi_b u_b - \frac{B-1}{B}\mathbb{E}[\xi u] - \bar{\xi}u_b, \sum_{b=1}^{B}\xi_b u_b - \frac{B-1}{B}\mathbb{E}[\xi u] - \bar{\xi}u_b\Big\rangle \\
&= \frac{1}{(B-1)^2}\sum_{a,b=1}^{B}\Big\langle \xi_a u_a - \mathbb{E}[\xi u] - (\bar{\xi}u_a - \frac{1}{B}\mathbb{E}[\xi u]), \xi_b u_b - \mathbb{E}[\xi u] - (\bar{\xi}u_b - \frac{1}{B}\mathbb{E}[\xi u])\Big\rangle
\end{aligned}
$$

When $a \neq b$, the summand can be simplified by independence of u_a, u_b and independence of $\xi_a u_a, \xi_b u_b$. In fact,

$$
\begin{aligned}
&\mathbb{E}\Big\langle \xi_a u_a - \mathbb{E}[\xi u] - (\bar{\xi}u_a - \frac{1}{B}\mathbb{E}[\xi u]), \xi_b u_b - \mathbb{E}[\xi u] - (\bar{\xi}u_b - \frac{1}{B}\mathbb{E}[\xi u])\Big\rangle \\
&= -2\mathbb{E}\Big\langle \xi_a u_a - \mathbb{E}[\xi u], \bar{\xi}u_b - \frac{1}{B}\mathbb{E}[\xi u]\Big\rangle + \mathbb{E}\Big\langle \bar{\xi}u_a - \frac{1}{B}\mathbb{E}[\xi u], \bar{\xi}u_b - \frac{1}{B}\mathbb{E}[\xi u]\Big\rangle \\
&= -2\mathbb{E}\Big\langle \xi_a u_a - \mathbb{E}[\xi u], \frac{\xi_a+\xi_b}{B}u_b - \frac{1}{B}\mathbb{E}[\xi u]\Big\rangle + \\
&\quad \mathbb{E}\Big\langle \frac{\xi_a+\xi_b}{B}u_a - \frac{1}{B}\mathbb{E}[\xi u], \frac{\xi_a+\xi_b}{B}u_b - \frac{1}{B}\mathbb{E}[\xi u]\Big\rangle \\
&= -\frac{2}{B}\mathbb{E}\xi_a^2\Big\langle u_a, u_b\Big\rangle - 2\mathbb{E}\Big\langle \xi_a u_a - \mathbb{E}[\xi u], \frac{\xi_b}{B}u_b - \frac{1}{B}\mathbb{E}[\xi u]\Big\rangle + \\
&\quad \mathbb{E}[\frac{(\xi_a+\xi_b)^2}{B^2}\langle u_a, u_b\rangle] - \frac{2}{B}\mathbb{E}[\xi u]\mathbb{E}[\frac{\xi_b}{B}u_b] + \frac{1}{B^2}\|\mathbb{E}[\xi u]\|_2^2 \\
&= 0 + 0 + (\frac{2}{B^2} - \frac{2}{B^2} + \frac{1}{B^2})\|\mathbb{E}[\xi u]\|_2^2 \\
&= \frac{1}{B^2}\|\mathbb{E}[\xi u]\|_2^2.
\end{aligned}
$$

When $a = b$, each summand becomes:

$$
\begin{aligned}
&\mathbb{E}\Big\|\xi_a u_a - \mathbb{E}[\xi u] - (\bar{\xi}u_a - \frac{1}{B}\mathbb{E}[\xi u])\Big\|_2^2 \\
&= \mathbb{E}\Big\|\xi_a u_a - \mathbb{E}[\xi u]\Big\|_2^2 - 2\mathbb{E}\Big\langle \xi_a u_a - \mathbb{E}[\xi u], \bar{\xi}u_a - \frac{1}{B}\mathbb{E}[\xi u]\Big\rangle + \\
&\quad \mathbb{E}\|\bar{\xi}u_a - \frac{1}{B}\mathbb{E}[\xi u]\|_2^2 \\
&= \mathbb{E}\Big\|\xi_a u_a - \mathbb{E}[\xi u]\Big\|_2^2 - 2\mathbb{E}\Big\langle \xi_a u_a, \bar{\xi}u_a\Big\rangle + \frac{2}{B}\|\mathbb{E}[\xi u]\|_2^2 + \mathbb{E}\|\bar{\xi}u_a\|_2^2 - \\
&\quad \frac{2}{B}\langle \mathbb{E}[\bar{\xi}u_a], \mathbb{E}[\xi u]\rangle + \frac{1}{B^2}\|\mathbb{E}[\xi u]\|_2^2 \\
&= \mathbb{E}\Big\|\xi_a u_a - \mathbb{E}[\xi u]\Big\|_2^2 - 2\mathbb{E}[\bar{\xi}\xi_a] + \mathbb{E}\bar{\xi}^2 + \frac{2B-1}{B^2}\|\mathbb{E}[\xi u]\|^2.
\end{aligned}
$$

Therefore, the variance can be written as

$$\begin{aligned}\mathrm{Var}(\widehat{\nabla S}(x_t,\delta)) &= \frac{1}{(B-1)^2}\sum_{a=1}^{B}\Big(\mathbb{E}\Big\|\xi_a u_a - \mathbb{E}[\xi u]\Big\|_2^2 - 2\mathbb{E}[\bar{\xi}\xi_a] + \mathbb{E}\bar{\xi}^2 + \\ &\qquad (\frac{2}{B} - \frac{1}{B^2})\|\mathbb{E}[\xi u]\|^2\Big) + \frac{\|\mathbb{E}\xi u\|_2^2}{B(B-1)} \\ &= \frac{B^2\mathrm{Var}(\widetilde{\nabla S}(x_t,\delta))}{(B-1)^2} - \frac{B\mathbb{E}[\bar{\xi}^2]}{(B-1)^2} + \frac{(3B-2)\|\mathbb{E}[\xi u]\|_2^2}{B(B-1)^2} \\ &\leq \frac{B^2\mathrm{Var}(\widetilde{\nabla S}(x_t,\delta))}{(B-1)^2} - \frac{B\mathbb{E}[\bar{\xi}^2]}{(B-1)^2} + \frac{3B-2}{B(B-1)^2}. \end{aligned} \tag{6.71}$$

The middle term can be expanded as

$$\begin{aligned}&-\frac{B}{(B-1)^2}\mathbb{E}[\bar{\xi}^2] \\ &= -\frac{B}{(B-1)^2}\sum_{a,b=1}^{B}\mathbb{E}[\xi_a\xi_b] \\ &= -\frac{1}{(B-1)^2 B}(B\cdot 1 + B(B-1)\cdot(2\mathbb{E}\xi - 1)^2) \\ &= -\frac{1}{(B-1)^2} - \frac{4}{B-1}(\mathbb{E}\xi - \frac{1}{2})^2)\end{aligned}$$

Plugging into Equation (6.71), we get

$$\begin{aligned}\mathrm{Var}(\widehat{\nabla S}(x_t,\delta)) &\leq \frac{B^2}{(B-1)^2}\mathrm{Var}(\widetilde{\nabla S}(x_t,\delta)) + \frac{2}{B(B-1)}(1 - 2B(\mathbb{E}[\xi] - \frac{1}{2})^2) \\ &= \mathrm{Var}(\widetilde{\nabla S}(x_t,\delta))\Big\{1 + \frac{2B-1}{(B-1)^2} - \frac{2}{\sigma^2(B-1)}\big(2B(\mathbb{E}[\xi] - \frac{1}{2})^2 - 1\big)\Big\}.\end{aligned}$$

When $\mathbb{E}[\xi]$ satisfies $(\mathbb{E}[\xi] - \frac{1}{2})^2 > \frac{1}{2B}(1 + \frac{2B-1}{2B-2}\sigma^2)$, we have

$$\frac{2B-1}{(B-1)^2} < \frac{2}{\sigma^2(B-1)}(2B(\mathbb{E}[\xi] - \frac{1}{2})^2 - 1),$$

which implies $\mathrm{Var}(\widehat{\nabla S}(x_t,\delta)) < \mathrm{Var}(\widetilde{\nabla S}(x_t,\delta))$.

□

6.8 Discussion

We have proposed a family of query-efficient algorithms based on a novel gradient-direction estimate, HopSkipJumpAttack, for decision-based generation of adversarial examples, which is capable of optimizing ℓ_2 and ℓ_∞-distances for both targeted and untargeted attacks. Convergence analysis has been carried out given access to the gradient. We have also provided analysis for the error of our Monte Carlo estimate of gradient direction, which comes from

three sources: bias at the boundary for a nonzero perturbation size, bias of deviation from the boundary, and variance. Theoretical analysis has provided insights for selecting the step size and the perturbation size, which leads to a hyperparameter-free algorithm. We have also carried out extensive experiments, showing HopSkipJumpAttack compares favorably to several state-of-the-art decision-based attack methods in query efficiency. It also achieves competitive performance on several defense mechanisms. We also show HopSkipJumpAttack successfully attacks several common image classifiers in addition to neural networks, including non-differentiable ones such as k-nearest neighbors, tree ensembles, SIFT and HOG.

Given the fact that HopSkipJumpAttack is able to craft a human-indistinguishable adversarial example within a realistic budget of queries, it becomes important for the community to consider the real-world impact of decision-based threat models. We have also demonstrated that HopSkipJumpAttack is able to achieve comparable or even superior performance to state-of-the-art white-box attacks on several defense mechanisms, under a much weaker threat model. In particular, masked gradients, stochastic gradients, and non-differentiability are not barriers to our algorithm. Because of its effectiveness, efficiency, and applicability to non-differentiable models, we suggest future research on adversarial defenses may evaluate the designed mechanism against HopSkipJumpAttack as a first step.

One limitation of all existing decision-based algorithms, including HopSkipJumpAttack, is that they require evaluation of the target model near the boundary. They may not work effectively by limiting the queries near the boundary. We have also observed that it still takes tens of thousands of model queries for HopSkipJumpAttack to craft imperceptible adversarial examples with a target class on ImageNet, which has a relatively large image size. Future work may seek the combination of HopSkipJumpAttack with transferred attack to resolve these issues.

Part III

Intersection

Chapter 7

ML-LOO: Detecting Adversarial Examples with Feature Attribution

Deep neural networks obtain state-of-the-art performance on a series of tasks. However, they are easily fooled by adding a small adversarial perturbation to the input. The perturbation is often imperceptible to humans on image data. We observe a significant difference in feature attributions between adversarially crafted examples and original examples. Based on this observation, we introduce a new framework to detect adversarial examples through thresholding a scale estimate of feature attribution scores. Furthermore, we extend our method to include multi-layer feature attributions in order to tackle attacks that have mixed confidence levels. As demonstrated in extensive experiments, our method achieves superior performances in distinguishing adversarial examples from popular attack methods on a variety of real data sets compared to state-of-the-art detection methods. In particular, our method is able to detect adversarial examples of mixed confidence levels, and transfer between different attacking methods. We also show that our method achieves competitive performance even when the attacker has complete access to the detector.

7.1 Introduction

Deep neural networks have achieved state-of-the-art performance on a variety of tasks, including image classification, object detection, speech recognition and machine translation. However, they have been shown to be vulnerable to adversarial examples. This incurs a security risk when DNNs are applied to sensitive areas such as finance, medicine, criminal justice and transportation. Adversarial examples are inputs to machine learning models that an attacker constructs intentionally to fool the model [155]. Szegedy et al. observed that a visually indistinguishable perturbation in pixel space to the original image can alter the prediction of a neural network. Later, a series of papers [26–39] designed more sophisticated methods for the worst-case perturbation within a restricted set, often a small L_p ball with $p = 0, 2, \infty$.

While a line of work tries to explain why adversarial examples exist [26, 156–158], a comprehensive analysis of underlying reasons has not yet been attained, largely because deep neural networks have complex functional forms such that mathematical characterizations are difficult to obtain. On the other hand, there has been a growing interest in developing tools for tackling the black-box nature of neural networks, among which feature attribution is a widely studied approach [8–10, 12, 14–18, 20, 24, 67, 104]. Given a predictive model, such a method outputs, for each instance to which the model is applied, a vector of importance scores associated with the underlying features. Feature attribution has been used to improve transparency and fairness of machine learning models [14, 17].

In this chapter, we investigate the application of feature attribution to detecting adversarial examples. In particular, we observe that the feature attribution map of an adversarial example near the boundary always differs from that of the corresponding original example. A motivating example is shown in Figure 7.1, which demonstrates images in CIFAR-10 to be fed into a residual neural network and the corresponding feature attribution from Leave-One-Out (LOO) [18]. The latter interprets decisions from a neural model by observing the effects on the model of erasing each pixel of input before and after the worst-case perturbation by a C&W attack. While the perturbation on the original image is visually imperceptible, the feature attribution is altered drastically. We further observe that the difference can be summarized by simple statistics that characterize *feature disagreement*, which are capable of distinguishing adversarial examples from natural images. We conjecture that this is because adversarial attacks tend to perturb samples into an unstable region on the decision surface.

The above observation led to an effective method for detecting adversarial examples near the decision boundary. On the other hand, there also exist adversarial examples in which the model has high confidence [29]. Previous work has observed that several state-of-the-art detection methods are vulnerable to such attacks [148, 159]. However, we observe an interesting phenomenon: middle layers of neural networks still contain information on uncertainty even for high-confidence adversarial examples. Based on this observation, we generalize our method to incorporate multi-layer feature attribution, where attribution scores for intermediate layers are computed without incurring extra model queries.

In numerical experiments, our method achieves superior performance in detecting adversarial examples generated from popular attack methods on MNIST, CIFAR-10 and CIFAR-100 among state-of-the-art detection methods. The proposed method is also capable of detecting mixed-confidence adversarial examples, transferring between adversarial examples of different confidence levels, and adversarial examples generated by various types of attacks. We further show that the proposed method performs competitively under the setting where the attacker has complete access to the detector.

7.2 Related Work

In this section, we review related work in feature attribution, adversarial attack, adversarial defense and detection.

Feature attribution A variety of methods have been proposed to assign feature attribution scores. For each specific instance where the model is applied, an attribution method assigns an importance score for each feature, by approximating the target model via a linear model locally around the instance. One popular class of methods assumes the differentiability of the model, and propagates the prediction to features through gradients. Examples include direct use of gradient (Saliency Map) [8], Layer-wise Relevance Propagation (LWRP) [9] and its improved version DeepLIFT [10], and Integrated Gradients [12].

Another class is perturbation-based and thus model-agnostic. Given an instance, multiple perturbed samples are generated by masking different groups of features with a pre-specified reference value. The feature attribution of the instance is computed according to the prediction scores of a model on these samples. Popular perturbation based methods include Leave-One-Out [18, 160], LIME [14] and KernelSHAP [15].

It has been observed in Ghorbani, Abid, and Zou that gradient-based feature attribution maps are sensitive to small perturbations. Adversarial attack to feature attribution is designed to characterize the fragility. On the contrary, robustness of an attribution method has been observed on a robust model. In fact, Yeh et al. observed that gradient based explanations of an adversarially trained network are less sensitive, and Chalasani et al. established theoretical results for the robustness of attribution map on an adversarially trained logistic regression. These observations indicate that the sensitivity of a feature attribution might be rooted in the sensitivity of the model, instead of the attribution method. This motivates the detection of adversarial examples via attribution methods.

Adversarial attack Adversarial attacks try to alter, with minimal perturbation, the prediction of an original instance from a given model, which leads to adversarial examples. Adversarial examples can be categorized as targeted or untargeted, depending on whether the goal is to classify the perturbed instance into a given target class or an arbitrary class different from the correct one. Attacks also differ by the type of distance they use to characterize minimal perturbation. ℓ_∞, ℓ_0, and ℓ_2 distances are the most commonly used distances. Fast Gradient Sign Method (FGSM) by Goodfellow, Shlens, and Szegedy is an efficient method to minimize the ℓ_∞ distance. Kurakin, Goodfellow, and Bengio and Madry et al. proposed ℓ_∞-PGD (BIM), an iterative version of FGSM, which achieves a higher success rate with a smaller size of perturbation. DeepFool presented by Moosavi-Dezfooli, Fawzi, and Frossard minimizes ℓ_2 distance through an iterative linearization procedure. Carlini and Wagner proposed effective algorithms to generate adversarial examples for each of the three distances. In particular, Carlini and Wagner proposed a loss function that is capable of controlling the confidence level of adversarial examples. The Jacobian-based Saliency Map Attack (JSMA) by [28] is a greedy method for perturbation with ℓ_0 metric. Recently, several black-box adversarial attacks that solely depend on probability scores or decisions have been introduced. Chen et al. and Ilyas et al., Ilyas, Engstrom, and Madry introduced score-based methods using zeroth-order gradient estimation to craft adversarial examples. Brendel, Rauber, and Bethge introduced Boundary Attack, as a black-box method to minimize the ℓ_2 distance,

that does not need access to gradient information and relies solely on the model decision. We demonstrate in our experiments that our method is capable of detecting adversarial examples generated by these attacks, regardless of the distance, confidence level, or whether the gradient information is used.

Adversarial defense and detection To improve the robustness of neural networks, various approaches have been proposed to defend against adversarial attacks, including adversarial training [26, 27, 30, 41, 42], distributional smoothing [43], defensive distillation [44], generative models [45], feature squeezing [46], randomized models [47–49], and verifiable defense [50, 51]. These defenses often involve modifications in the training process of a model, which often require higher computational or sample complexity [53], and lead to loss of accuracy [52].

Complimentary to the previous defending techniques, an alternative line of work focuses on screening out adversarial examples in the test stage without touching the training of the original model. Data transformations such as PCA have been used to extract features from the input and layers of neural networks for adversarial detection [54–56]. Alternative neural networks are used to classify adversarial and original images [58–60]. Feinman et al. proposed to use kernel density estimate (KD) and Bayesian uncertainty (BU) in hidden layers of the neural network for detection. Ma et al. observed Local Intrinsic Dimension (LID) of hidden-layer outputs differ between the original and adversarial examples. Lee et al. obtained the class conditional Gaussian distributions with respect to lower-level and upper-level features of the deep neural network under Gaussian discriminant analysis, which result in a confidence score based on the Mahalanobis distance (MAHA), followed by a logistic regression model on the confidence scores to detect adversarial examples. Through vast experiments, we show that our method achieves comparable or superior performance than these detection methods across various attacks. Furthermore, we show that our method achieves competitive performance for attacks with a varied confidence level, a setting where the other detection methods fail to work [148, 159].

Most related to our work, Tao et al. proposed to identify neurons critical for individual attributes to detect adversarial examples, but their method is restricted to models in face recognition. Instead, our method is applicable across different types of image data. Zhang et al. proposed to identify adversarial perturbations by training a neural network on the saliency map of inputs. However, their method depends on additional neural networks, which are vulnerable to white-box attacks when attackers perturb the image to fool the original model and the new neural network simultaneously. As we will show in experiments, our method achieves competitive performance under white-box attacks.

7.3 Adversarial detection with feature attribution

We motivate our method by an observation on feature attribution with and without adversarial perturbation. Then we discuss metrics to quantify the dispersion in attribution.

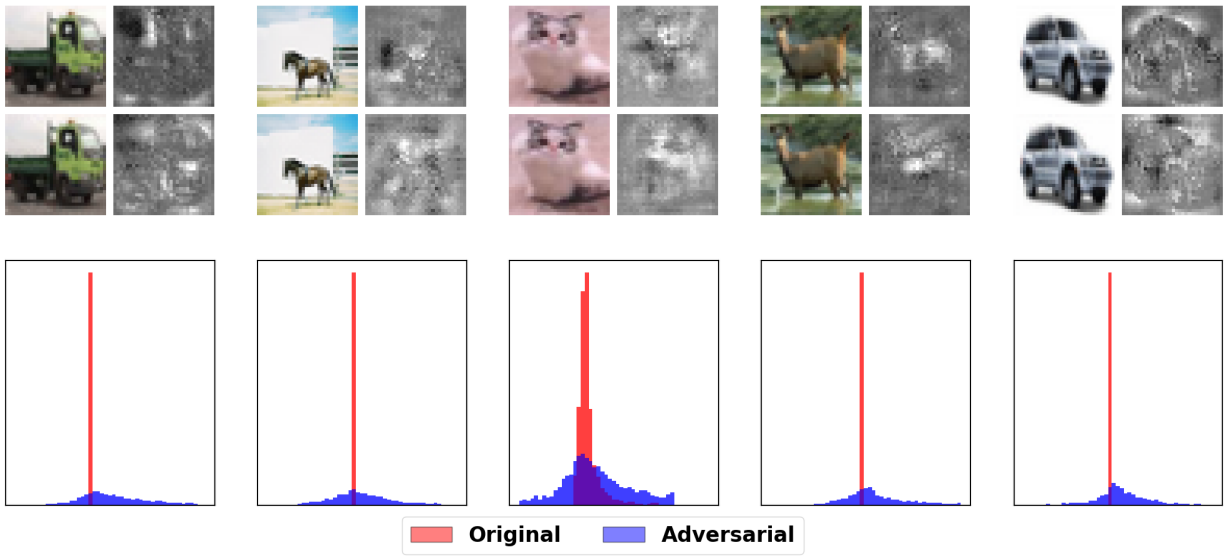


Figure 7.1: The first row shows the original CIFAR-10 examples and their corresponding feature attributions. The second row shows the adversarial examples and their corresponding feature attributions. The third row plots the histograms of the original and adversarial feature attributions.

Finally, we extend our method to the multi-layer version for detecting adversarial examples with mixed confidence levels.

7.3.1 Feature attribution before and after perturbation

Assume that the model is a function $f : \mathbb{R}^d \to [0,1]^C$ which maps an image x of dimension $d = h \times w \times c$ to a probability vector $f(x)$ of dimension C, where C is the number of classes. A feature attribution method ϕ maps an input image $x \in \mathbb{R}^d$ to an attribution vector of the same shape as the image: $\phi(x) \in \mathbb{R}^d$, such that the i-th dimension of $\phi(x)$ is the contribution of feature i in the prediction of the model on the specific image x. We suppress the dependence of ϕ on the model f for notational convenience. We focus on the leave-One-Out (LOO) method [18, 160] throughout the paper, which assigns to each feature the reduction in the probability of the selected class when the feature in consideration is masked by some reference value, e.g. 0. Denoting the example with the i-th feature masked by 0 as $x_{(i)}$, LOO defines ϕ as

$$\phi(x)_i := f(x)_c - f(x_{(i)})_c, \text{ where } c = \arg\max_{j \in C} f(x)_j. \tag{7.1}$$

Adversarial attacks aim to change the prediction of a model with minimal perturbation of a sample, so that human is not able to detect the difference between an original image x and its perturbed version x'. Yet we observed that ϕ is sensitive to the small difference between x and x'. Figure 7.1 shows the attribution maps $\phi(x), \phi(x')$ with the original image x and its adversarially perturbed counterpart x' by C&W attack. Even with human eyes,

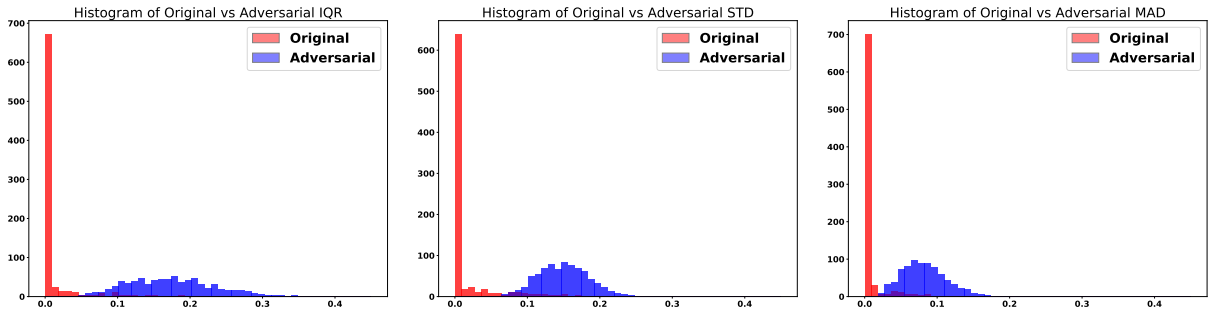


Figure 7.2: Histogram of dispersion measures

we can observe an explicit difference in the attribution maps of the original and adversarial images. In particular, adversarial images have a larger dispersion in its importance scores, as demonstrated in Figure 7.1. We comment here that our proposed framework of adversarial detection via feature attribution is generic to popular feature attribution methods, such as Integrated Gradients. LOO achieves the best performance among all attribution methods across different data sets.

7.3.2 Quantify the dispersion in feature attribution maps

Motivated by the apparent differences in the distributions of importance scores between the original and adversarial images, as demonstrated in Figure 7.1, we propose to use measures of statistical dispersion in feature attribution to detect adversarial examples. In particular, we tried standard deviation (STD), median absolute deviation (MAD), which is the median of absolute differences between entries and their median, and interquartile range (IQR), which is the difference between the 75th percentile and the 25th percentile among all entries of $\phi(x) \in \mathbb{R}^d$:

$$\text{IQR}(\phi(x)) = Q_{\phi(x)}(0.75) - Q_{\phi(x)}(0.25),$$
$$\text{where } Q_{\phi(x)}(p) := \min\{\beta : \frac{\#\{i : \phi(x)_i < \beta\}}{d} \geq p\}.$$

We observe there is a larger dispersion, which we call *feature disagreement*, between feature contribution to a model for an adversarially perturbed image. The difference is universal across different images. Figure 7.2 compares the histograms of these three dispersion measures of feature attributions for ResNet on natural test images from CIFAR-10 with those on adversarially perturbed images, where the adversarial perturbation is carried out by C&W Attack. We can see there is a significant difference in the distributions of STD, MAD and IQR between natural and adversarial images. A majority of adversarially perturbed images have a larger dispersion in feature attribution than an arbitrary natural image, besides the corresponding original images. We propose to distinguish adversarial images from natural images by thresholding the IQR of feature attribution maps. While all the three measures yield competitive performance for adversarial detection, we stick to IQR for the rest of the paper, which is robust and has a slightly superior performance among the three.

7.3.3 Extension to multi-layer LOO: detection of attacks with mixed confidence levels

Carlini and Wagner proposed the following objective to generate adversarial images with small ℓ_2 perturbation.

$$\min_w \|x' - x\|_2 + \alpha \max\{F(x)_{y_{\text{true}}} - \max_{j \neq y_{\text{true}}} F(x')_j + c, 0\}, \tag{7.2}$$

where $x' = 0.5(\tanh(w) + 1)$, F maps an image to logits, $y_{\text{true}} = \arg\max F(x)$ is the original label, and c is a hyperparameter for tuning confidence. Adversarial images with high confidence can be obtained by assigning a large value to c. The loss can be modified to generate ℓ_∞ constrained perturbation at different confidence levels as well [30]. Recently, Lu, Chen, and Yu and Athalye, Carlini, and Wagner observed that LID has a poor performance when faced with adversarial examples at various confidence scales. In our experiments, a similar phenomenon is observed for several other state-of-the-art detection methods, including KD+BU and MAHA, as is shown in Figure 7.4. This suggests that characterization of adversarial examples in related work may only hold true for adversarial examples near the decision boundary. IQR of feature attribution map, unfortunately, suffers from the same problem.

To detect adversarial images with mixed confidence levels, we generalize our method to capture dispersion of feature attributions beyond the output layer of the model. For an adversarial example within a small neighborhood of its original example in the pixel space but achieving a high confidence at the output layer in a different class from the original one, the feature representation deviates away from that of its original example gradually along the layers. Thus, we expect neurons of middle layers contain uncertainty that can be captured by a feature attribution map. We denote the map from input to an arbitrary neuron n of an intermediate layer of the model by $f_n : \mathbb{R}^d \to \mathbb{R}$. The feature attribution of neuron n is defined as $\phi_{f_n}(x) : \mathbb{R}^d \to \mathbb{R}^d$, such that the i-th entry quantifies the contribution of feature i to neuron n. For Leave-One-Out (LOO), we have

$$\phi_{f_n}(x)_i = f_n(x) - f_n(x_{(i)}).$$

To coordinate the scale difference between different neurons, we fit a logistic regression for the dispersion of feature attribution from different neurons on a hold-out training set to distinguish adversarial images from original images. The multi-layer extension of our method is called 'ML-LOO'.

7.4 Experiments

We present an experimental evaluation of ML-LOO, and compare our method with several state-of-the-art detection methods. Then we consider the setting where attacks have different confidence levels. We further evaluate the transferability of various detection methods on an unknown attack. Finally, we evaluate the performance of our method under the white-box attacker who knows the existence of our detector.

Data	Model	Metric	Attacks											
			C&W				ℓ_∞-PGD				FGSM			
			KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO
MNIST	CNN	AUC	0.893	1.000	0.957	**1.000**	0.766	0.902	0.736	**1.000**	0.744	0.780	0.967	**1.000**
		TPR (FPR@0.01)	0.23	**0.99**	0.94	0.98	0.09	0.32	0.01	**0.99**	0.01	0.09	0.54	**0.99**
		TPR (FPR@0.05)	0.46	**0.99**	0.94	0.98	0.28	0.58	0.12	**0.99**	0.15	0.23	0.92	**0.99**
		TPR (FPR@0.10)	0.55	**0.99**	0.94	0.98	0.34	0.72	0.29	**0.99**	0.24	0.40	0.94	**0.99**
CIFAR10	ResNet	AUC	0.623	0.990	0.962	**0.995**	0.834	0.970	0.958	**0.999**	0.673	0.972	0.770	**0.997**
		TPR (FPR@0.01)	0.01	0.55	0.57	**0.86**	0.54	0.52	0.41	**0.96**	0.04	0.29	0.04	**0.82**
		TPR (FPR@0.05)	0.09	**0.98**	0.95	**0.98**	0.61	0.85	0.86	**0.98**	0.20	0.82	0.16	**0.99**
		TPR (FPR@0.10)	0.22	**0.99**	0.95	**0.99**	0.62	0.91	0.91	**0.98**	0.29	0.93	0.38	**0.99**
	DenseNet	AUC	0.679	0.958	0.966	**0.977**	0.955	0.952	0.768	**0.997**	0.790	0.706	0.829	**1.000**
		TPR (FPR@0.01)	0.06	0.30	**0.48**	0.33	0.69	0.51	0.03	**0.99**	0.17	0.04	0.00	**0.99**
		TPR (FPR@0.05)	0.13	0.79	**0.91**	0.84	0.74	0.84	0.23	**0.99**	0.28	0.12	0.29	**0.99**
		TPR (FPR@0.10)	0.22	0.91	0.94	**0.98**	0.80	0.88	0.31	**0.99**	0.41	0.23	0.51	**0.99**
CIFAR100	ResNet	AUC	0.637	0.717	0.945	**0.967**	0.855	0.984	0.966	**0.999**	0.773	0.985	0.875	**1.000**
		TPR (FPR@0.01)	0.07	0.00	0.00	**0.33**	0.59	0.69	0.48	**0.94**	0.39	0.48	0.12	**0.99**
		TPR (FPR@0.05)	0.16	0.01	0.52	**0.70**	0.61	0.94	0.82	**0.99**	0.49	0.89	0.43	**0.99**
		TPR (FPR@0.10)	0.29	0.01	0.80	**0.92**	0.64	0.96	0.92	**0.99**	0.56	0.99	0.57	**0.99**
	DenseNet	AUC	0.567	0.727	0.916	**0.958**	0.549	0.732	0.947	**0.971**	0.577	0.751	0.951	**0.974**
		TPR (FPR@0.01)	0.02	**0.07**	0.00	**0.07**	0.01	0.00	0.00	**0.21**	0.01	0.01	0.00	**0.31**
		TPR (FPR@0.05)	0.17	0.15	0.61	**0.66**	0.14	0.01	0.70	**0.75**	0.17	0.06	0.77	**0.81**
		TPR (FPR@0.10)	0.22	0.26	0.84	**0.88**	0.20	0.04	0.91	**0.96**	0.23	0.18	0.93	**0.94**

Data	Model	Metric	Attacks											
			JSMA				DeepFool				Boundary			
			KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO
MNIST	CNN	AUC	0.886	**1.000**	0.976	**1.000**	0.901	**1.000**	0.869	**1.000**	0.905	**1.000**	0.991	**1.000**
		TPR (FPR@0.01)	0.30	**1.00**	0.87	0.99	0.32	**1.00**	0.04	**1.00**	0.32	**1.00**	0.79	**1.00**
		TPR (FPR@0.05)	0.46	**1.00**	0.94	**1.00**	0.43	**1.00**	0.36	**1.00**	0.45	**1.00**	0.98	**1.00**
		TPR (FPR@0.10)	0.51	**1.00**	0.95	**1.00**	0.57	**1.00**	0.59	**1.00**	0.55	**1.00**	0.98	**1.00**
CIFAR10	ResNet	AUC	0.614	**0.986**	0.941	0.981	0.618	0.990	0.981	**0.994**	0.676	0.990	0.967	**0.997**
		TPR (FPR@0.01)	0.01	**0.49**	0.45	0.46	0.01	0.57	0.60	**0.89**	0.03	0.64	0.60	**0.92**
		TPR (FPR@0.05)	0.10	**0.98**	0.87	0.82	0.10	**0.99**	0.96	0.96	0.20	**0.99**	0.94	**0.99**
		TPR (FPR@0.10)	0.21	**0.99**	0.90	**0.99**	0.24	**0.99**	0.96	**0.99**	0.38	**0.99**	0.94	**0.99**
	DenseNet	AUC	0.645	0.937	0.947	**0.964**	0.646	0.976	**0.977**	0.976	0.700	**0.983**	0.981	0.980
		TPR (FPR@0.01)	0.04	0.14	**0.41**	0.12	0.03	0.34	**0.51**	0.24	0.05	0.58	**0.62**	0.31
		TPR (FPR@0.05)	0.10	0.67	0.68	**0.72**	0.09	0.90	**0.95**	0.82	0.12	**0.93**	0.91	0.89
		TPR (FPR@0.10)	0.18	0.86	0.88	**0.96**	0.17	**0.98**	0.97	**0.98**	0.23	**0.98**	0.96	**0.98**
CIFAR100	ResNet	AUC	0.600	0.740	0.907	**0.964**	0.610	0.714	0.953	**0.970**	0.635	0.732	0.956	**0.972**
		TPR (FPR@0.01)	0.00	0.01	0.00	**0.42**	0.06	0.00	0.00	**0.41**	0.07	0.01	0.00	**0.49**
		TPR (FPR@0.05)	0.12	0.14	0.49	**0.70**	0.14	0.01	0.56	**0.74**	0.16	0.07	0.61	**0.78**
		TPR (FPR@0.10)	0.27	0.24	0.77	**0.91**	0.29	0.01	0.87	**0.94**	0.30	0.15	**0.94**	0.93
	DenseNet	AUC	0.567	0.727	0.916	**0.958**	0.549	0.732	0.947	**0.971**	0.577	0.751	0.951	**0.974**
		TPR (FPR@0.01)	0.02	**0.07**	0.00	**0.07**	0.01	0.00	0.00	**0.21**	0.01	0.01	0.00	**0.31**
		TPR (FPR@0.05)	0.17	0.15	0.61	**0.66**	0.14	0.01	0.70	**0.75**	0.17	0.06	0.77	**0.81**
		TPR (FPR@0.10)	0.22	0.26	0.84	**0.88**	0.20	0.04	0.91	**0.96**	0.23	0.18	0.93	**0.94**

Table 7.1: Performance of detection methods on different data sets, models and attack methods.

7.4.1 Known attacks

We compare our method with state-of-the-art detection algorithms including LID [62], Mahalanobis (MAHA) [57], and KD+BU [61], on three data sets: MNIST, CIFAR-10 and CIFAR-100, with the standard train/test split [99]. We used a convolutional network composed of 32-filter convolutional layers followed by a hidden dense layer with 1024 units for MNIST. Each convolutional layer was followed by a max-pooling layer. For both CIFAR-10 and CIFAR-100, we trained a 20-layer ResNet [144] and 121-layer DenseNet [146] respectively. For each data set, we generated 2,000 adversarial examples from correctly classified test images by each attacking method. Among them, 1,000 adversarial images with the corresponding 1,000 natural images were used for the training process of LID, Mahalanobis and our method. Results are reported for the other 1,000 adversarial images with the corresponding natural images. We consider the following attacking methods, grouped by the

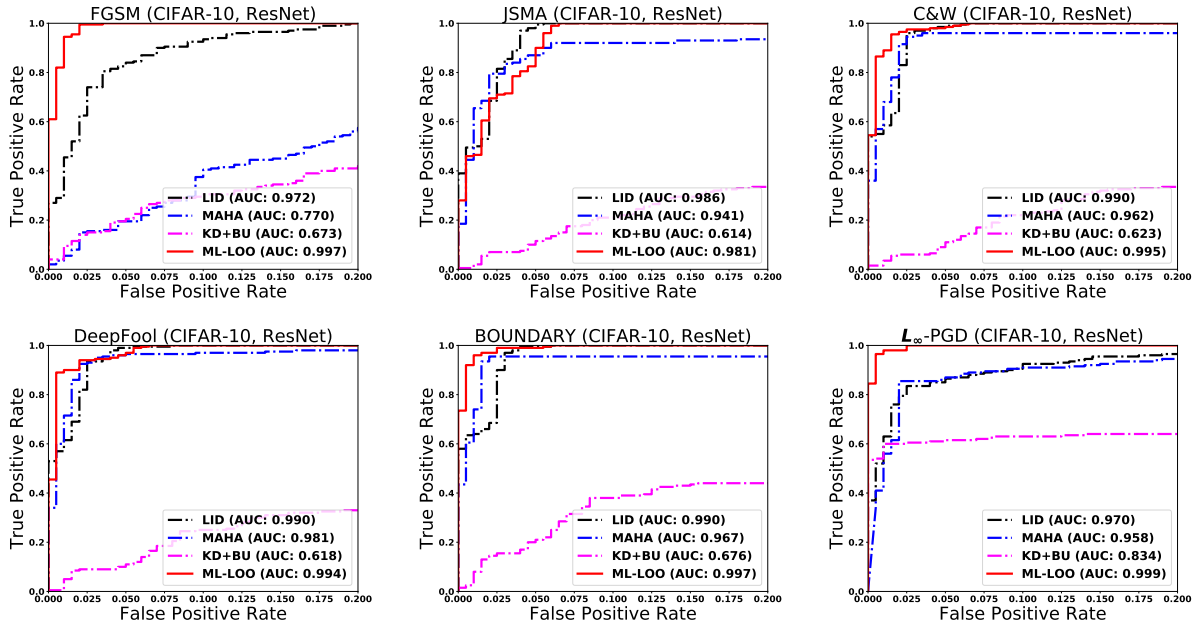


Figure 7.3: ROC curves of detection methods on CIFAR-10 with ResNet

norms they are optimized for:

- L_∞: FGSM [26], L_∞-PGD [27, 30].
- ℓ_2: C&W [29], DeepFool [31], Boundary Attack [38].
- ℓ_0: JSMA [28].

Let true positive rate (TPR) be the proportion of adversarial images classified as adversarial, and false positive rate (FPR) be the proportion of natural images classified as adversarial. We report area under the curve (AUC) of the ROC curve as the performance evaluation as well as the true positive rates by thresholding FPR at 0.01,0.05 and 0.1, as it is practical to keep misclassified natural images at a low proportion.

The results are reported in Table 7.1, and the ROC curves on CIFAR-10 with ResNet are shown in Figure 7.3. ML-LOO shows superior performance over the other three detection methods across different data sets, models for all attacks optimized for ℓ_2 and ℓ_∞ distances. By controlling FPR at 0.1, our method is able to find over 95% adversarial examples generated by most existing attacks.

7.4.2 Attacks with varied confidence levels

Lu, Chen, and Yu and Athalye, Carlini, and Wagner observed that LID fails when the confidence level of adversarial examples generated from C&W attack varies. We consider adversarial images with varied confidence levels for both ℓ_2 and ℓ_∞ attacks. We use C&W attack for optimizing ℓ_2 distance, and adjust the confidence hyperparameter c in Equation (7.2)

Data	Model	Metric	Attacks											
			C&W MIX				C&W LC				C&W HC			
			KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO
CIFAR10	ResNet	AUC	0.620	0.649	0.640	**0.840**	0.623	0.445	0.641	**0.711**	0.829	0.816	0.966	**0.988**
		TPR (FPR@0.01)	0.04	0.01	0.03	**0.25**	0.01	0.00	0.01	**0.12**	0.52	0.23	0.51	**0.87**
		TPR (FPR@0.05)	0.17	0.06	0.14	**0.42**	0.09	0.06	0.10	**0.21**	0.59	0.43	0.90	**0.94**
		TPR (FPR@0.10)	0.28	0.19	0.21	**0.59**	0.22	0.11	0.16	**0.34**	0.60	0.62	0.93	**0.97**
Data	Model	Metric	Attacks											
			ℓ_∞-PGD-MIX				ℓ_∞-PGD-LC				ℓ_∞-PGD-HC			
			KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO
CIFAR10	ResNet	AUC	0.753	0.812	0.813	**0.953**	0.606	0.578	0.578	**0.767**	0.834	0.935	0.962	**0.996**
		TPR (FPR@0.01)	0.20	0.10	0.11	**0.60**	0.01	0.01	0.01	**0.09**	0.54	0.26	0.46	**0.89**
		TPR (FPR@0.05)	0.37	0.36	0.45	**0.77**	0.12	0.07	0.04	**0.23**	0.61	0.67	0.89	**0.98**
		TPR (FPR@0.10)	0.46	0.41	0.56	**0.84**	0.25	0.17	0.12	**0.33**	0.62	0.85	0.91	**0.99**

Table 7.2: Top: Performance of detection methods trained with C&W-MIX and tested on C&W-LC, C&W-HC and C&W-MIX. Bottom: Performance of detection methods trained with ℓ_∞-PGD-MIX and tested on ℓ_∞-PGD-LC, ℓ_∞-PGD-HC and ℓ_∞-PGD-MIX.

to achieve mixed confidence levels. To achieve adversarial examples optimized for ℓ_∞ distance, we use ℓ_∞-PGD for optimizing ℓ_∞ distance, and vary the constraint ε for different confidence levels.

C&W Attack for optimizing ℓ_2 distance We consider three settings for C&W attack, low-confidence (C&W-LC), mixed-confidence (C&W-MIX) and high-confidence (C&W-HC). We set the confidence parameter $c = 0$ for C&W-LC and $c = 50$ for C&W-HC. For mixed-confidence C&W attack, we generate adversarial images from C&W attack with the confidence parameter in Equation (7.2) randomly selected from $\{1, 3, 5, \cdots, 29\}$ when generating an adversarial image, so that the distribution of confidence levels for adversarial images is comparable with that of original images. The confidence levels of images under the three settings, along with confidence levels of original images are shown in Figure 7.4. The confidence level in Figure 7.4 is defined as $-\log(1-p)$, where p is the probability score of the predicted class.

We carried out the experiments on ResNet trained on CIFAR-10 using $1,000$ adversarial images generated from the mixed-confidence C&W attack, together with the corresponding original images, as the training data for LID, Mahalanobis, KD+BU, and our method. We test the detection methods on a different set of original and adversarial images generated from three versions: low-confidence C&W attack ($c = 0$), high-confidence C&W attack ($c = 50$), and the mixed-confidence C&W attack. Table 7.2 (Top) and Figure 7.4 (Left) show TPRs at different FPR thresholds, AUC, and the ROC curve. Mahalanobis, LID and KD+BU fail to detect adversarial examples of mixed-confidence effectively, while our method performs consistently better for adversarial images across the three settings.

L_∞-PGD for optimizing ℓ_∞ distance L_∞-PGD [30], also named as BIM [27], searches for adversarial examples by iteratively updating the original image with the following:

$$x_{N+1} = \text{Clip}_{x,\varepsilon}\{x_N + \alpha \text{sign}(\nabla_X J(x_N, y_{\text{true}}))\}, \tag{7.3}$$

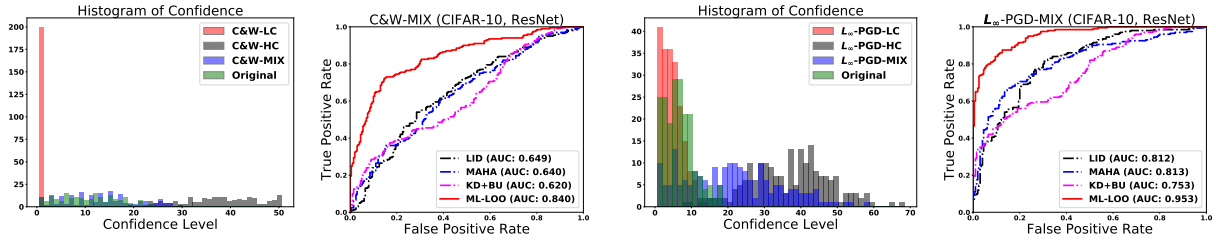


Figure 7.4: The left two figures plot the histogram of confidence levels of C&W-LC, C&W-HC, and C&W-MIX, and the ROC curves of detection methods under C&W-MIX attack. The right two figures plot the histogram of confidence levels of ℓ_∞-PGD-LC, ℓ_∞-PGD-HC, and ℓ_∞-PGD-MIX, and the ROC curves of detection methods under ℓ_∞-PGD-MIX attack.

where y_{true} is the original class, J is the cross-entropy loss, and Clip operator clips an image elementwise to an ε-neighborhood. For mixed-confidence ℓ_∞-PGD attack, we generated adversarial images from ℓ_∞-PGD with different confidence levels by randomly selecting the constraint ε in Equation (7.3) from $\{1, 2, 3, 4, 5, 6, 7, 8\}/255$. The confidence levels of images from mixed-confidence ℓ_∞-PGD attack are shown in Figure 7.4.

We used $1,000$ adversarial images generated from the mixed-confidence ℓ_∞-PGD, together with their corresponding original images, as the training data for all detection methods. We report the results on adversarial images generated from three versions: high-confidence ℓ_∞-PGD ($\varepsilon = 0.03$), low-confidence ℓ_∞-PGD ($\varepsilon = 0.005$), and the mixed-confidence ℓ_∞-PGD that is used to generate the training data. The corresponding original images are different from the training images. Table 7.2 (Bottom) and Figure 7.4 (Right) show TPRs at different FPR thresholds, AUC, and the ROC curve. Mahalanobis, LID and KD+BU fail to detect adversarial examples of mixed-confidence effectively, while our method performs significantly better across the three settings.

7.4.3 Transferability

In this experiment, we evaluate the transferability of different methods by training detection methods on adversarial examples generated from one attacking method and carry out the evaluation on adversarial examples generated from different attacking methods. We trained all methods on adversarial examples generated by C&W attack and carried out the evaluation on adversarial examples generated by the rest of the attacking methods.

Experiments are carried out on MNIST, CIFAR-10, and CIFAR-100 data sets. AUC and TPRs at different FPR thresholds are reported in Table 7.3. All methods trained on C&W attack are capable of detecting adversarial examples generated from an unknown attack, even when the optimized distance is ℓ_∞, or the attack is not gradient-based. The same phenomenon has been observed in Lee et al. as well. This indicates attacks might share some common features. Our method yields a slightly higher AUC consistently, and has a significantly higher TPR when FPRs are controlled to be small.

Data	Model	Metric	Attacks											
			ℓ_∞-PGD				DeepFool				FGSM			
			KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO
CIFAR10	ResNet	AUC	0.753	0.763	0.818	**0.879**	0.618	0.990	0.962	**0.992**	0.673	0.610	0.730	**0.796**
		TPR (FPR@0.01)	0.20	0.08	0.14	**0.21**	0.01	0.56	0.61	**0.72**	0.04	**0.07**	0.06	0.04
		TPR (FPR@0.05)	0.37	0.35	0.45	**0.48**	0.10	0.96	0.94	**0.96**	0.20	0.17	**0.22**	0.14
		TPR (FPR@0.10)	0.46	0.45	0.60	**0.65**	0.24	0.98	0.94	**0.99**	0.29	0.23	0.34	**0.37**

Data	Model	Metric	Attacks							
			JSAM				Boundary			
			KD+BU	LID	MAHA	ML-LOO	KD+BU	LID	MAHA	ML-LOO
CIFAR10	ResNet	AUC	0.614	0.984	0.957	**0.984**	0.676	0.991	0.964	**0.994**
		TPR (FPR@0.01)	0.01	0.43	0.44	**0.45**	0.03	0.56	0.60	**0.82**
		TPR (FPR@0.05)	0.10	**0.93**	0.91	0.91	0.20	**0.99**	0.95	0.97
		TPR (FPR@0.10)	0.21	0.98	0.94	**0.99**	0.38	**0.99**	0.95	**0.99**

Table 7.3: Performance of detection methods trained with C&W and transferred to ℓ_∞-PGD, FGSM, JSMA, Boundary and DeepFool.

7.4.4 White-box evaluation

The previous experiments are carried out in a "gray-box" threat model, where the attacker has access to the model details such as gradients, but does not have access to the design of the detector. The "white-box" setting assumes a stronger threat model, where an attacker knows exactly how our detector is constructed and its parameters. Such a setting is often missing in previous study of adversarial detection. Previous work such as LID and KD+BU has been shown to fail under this setting [148, 153]. We evaluate the performance of ML-LOO in this setting.

We carried out the white-box attack on CIFAR-10 with the ResNet. The attacker aims to optimize the following objective

$$\min_w L(x') = \|x' - x\|^2 + c_1 L_{\text{C\&W}}(x') + c_2 L_{\text{DET}}(x'),$$

where $x' = 0.5(\tanh(w) + 1)$, the C&W loss $L_{\text{C\&W}} = \max\{F(x)_{y_{\text{true}}}, 0\}$, and L_{DET} aims at controlling the statistic used by the detector, which will be defined differently under different scenarios below. For each image, we increase c_2 gradually until adversarial images cannot be detected (at FPR=0.05) at all. For each c_2, c_1 is selected via binary search. The loss is minimized with Adam [83]. Under this scheme, the generated examples are expected to fool the detector, and we aim to check whether it fools the original model at an acceptable perturbation size as well. We will evaluate three variants of ML-LOO, including the simplest output layer standard deviation (SD) thresholding, ML-LOO (SD), and ML-LOO (IQR). We measure the performance by the success rate on the original model, the detector, and the rate of fooling both simultaneously. We also report the average ℓ_2 distance between original and successful adversarial images. The results are summarized in Table 7.4.

In the first scenario, we evaluate the robustness of the single-layer variate of ML-LOO (SD) to demonstrate the power of our attacker design, which only thresholds the SD of the probability of the predicted class alone. We define the detector loss as $L^1_{\text{DET}} := \max\{\text{SD}(\phi(x')) - \tau, 0\}$, which penalizes $\text{SD}(\phi(x'))$ over attribution scores if it is larger than τ. The threshold τ is chosen to keep the FPR at 0.05 when detecting adversarial examples generated by gray-box C&W attack from natural images. When LOO over all pixels is intractable, we sample pixels to estimate the SD. The attacker always fool the detector. However, the success rate

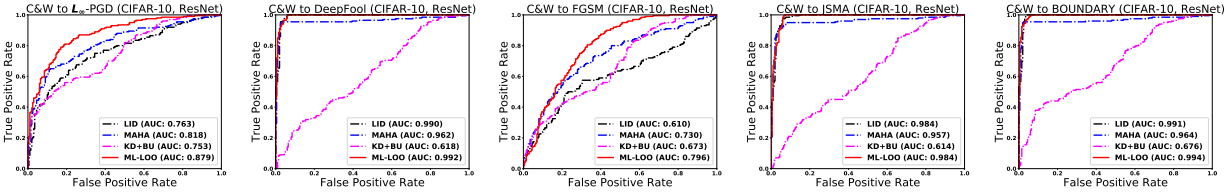


Figure 7.5: Transferability of detection methods trained with C&W attack and tested on ℓ_∞-PGD, FGSM, JSMA, Boundary and DeepFool.

Detector	None	SD	ML-LOO(SD)	ML-LOO(IQR)
Succ. rate on Model	100%	93%	52%	58%
Succ. rate on Detector	N/A	100%	100%	78%
Succ. rate on Both	100%	93%	52%	**36%**
Avg. ℓ_2 distance	0.31	0.43	**1.23**	1.07

Table 7.4: Performance under the white-box attacks.

of fooling the original model decreases from 100% to 93%, and the average distance increases from 0.31 to 0.43.

In the second scenario, We use ML-LOO (with SD) as the detector (a Logistic Regression (LR) applied to SD of multi-layer feature attributions). The white-box attack in the first setting fails to fool this detector completely. Therefore, we define the detector loss as

$$L^2_{\text{DET}} := \max\{\sum_n w_n \text{SD}(\phi_{f_n}(x)) - \tau, 0\},$$

where n loops over neurons of selected layers, $\phi_{f_n}(x)$ is the attribution score at neuron n, and w_n is the corresponding learned coefficients. The threshold τ is still chosen at FPR $=$ 0.05. The success rate of fooling the model decreases from 100% to 52%, and the average distance increases to 1.23. In the third scenario, we evaluate ML-LOO (IQR), which is non-differentiable. However, there is an approximately linear relationship between SD and IQR under the normality assumption [166], which suggests that we can apply the same detector loss L^2_{DET} with the transformed threshold at FPR $= 0.05$, and the transformed coefficients learned in ML-LOO (IQR). The attacker achieves worse performance. In particular, only 78% generated images fool the detector, with a 58% success rate of fooling the model (over all the examples) and an average distance of 1.07.

We observe that ML-LOO (IQR) achieves competitive performance even under the strongest white-box threat model. The white-box attack fails to fool the model and the detector simultaneously for 64% of test images. The average size of successful perturbations also increases by over three times. We expect ML-LOO works better under a white-box attack for adversarially trained models with larger certified radii.

7.5 Discussion

In this chapter, we introduce a new framework to detect adversarial examples with multi-layer feature attribution, by capturing the scaling difference of feature attribution scores between the original and adversarial examples. We show that our detection method outperforms other state-of-the-art methods in detecting various kinds of attacks. It also displays strong performance in detecting adversarial examples of varied confidence levels, in detecting transferred examples from other attacks, and when an attacker has complete access to the detector.

Chapter 8

Future Directions

In this thesis, we have discussed a range of different problems in model interpretation and adversarial robustness, such as the design of query-efficient algorithms and the leverage of prior knowledge in model-agnostic model interpretation, the generation of adversarial examples for discrete data, the query-efficient decision-based generation of adversarial examples, and the application of model interpretation to detecting adversarial examples. There are many problems that still remain open in these two fields.

One critical problem is to quantify the uncertainty in the feature importance scores in model interpretation. Current approaches yield a vector of importance scores with the underlying features for each instance. Due to the limitation of computational budget, there can be uncertainty in the procedure of producing these scores for many existing methods. Thus, it may be more realistic to have an interval estimate of the truth values. One potential solution is to associate a hypothesis test with each feature, and seek existing procedures in statistics (e.g., [167]) to provide error rate control and quantify uncertainty in feature attribution. When there is a graph structure in the data, such as a parse tree in linguistic data, the framework built in Ramdas et al. [168] can be exploited for false discovery rate control.

Another problem is about the trade-off between different axiomatic frameworks of a faithful interpretation. This directly leads to a critical question: which framework should one use in practice? The answer might vary from application to application. The choice not only depends on underlying axioms [see, e.g., 22, 169], but may also go beyond philosophical preferences. The ease of implementation and the existence of an efficient and accurate approximation of true scores can all be important factors. A detailed discussion on such issues is needed for popularizing tools in model interpretation within the research community and beyond.

It is also important to investigate the application of tools in model interpretation to areas beyond detection of adversarial examples. A potential direction is on privacy-preserving machine learning. In privacy-sensitive fields such as medical science, a critical criterion of a trained model is whether it preserves the privacy of training data [170]. The statistic constructed in Chapter 7 may be used a tool to distinguish whether a data point has appeared

in the training data for models which do not preserve privacy.

One limitation of all existing decision-based algorithms for generating adversarial examples, including HopSkipJumpAttack, is that they require evaluations of the target model near the boundary. A potential defense mechanism against these algorithms is to limit the queries near the boundary. We have also observed that it still takes tens of thousands of model queries for HopSkipJumpAttack to craft imperceptible adversarial examples with a target class on ImageNet, which has a relatively large image size. Future work may seek the combination of HopSkipJumpAttack with transferred attack to resolve these issues.

A potential exciting application of decision-based algorithms is to evaluate the robustness of human vision systems. With a carefully-designed interface, HopSkipJumpAttack may be used to craft "adversarial examples" for human eyes. However, it requires a careful control in the range of temporal exposures and the number of similar images within a short time period, so as to disrupt short-term memory that may lead to dependence between consecutive decisions on similar images. Such a study may provide a deeper understanding in the human vision system and its similarity and difference with current machine learning algorithms.

The methods developed in this thesis may also provide some insights for other areas. The framework developed in Chapter 3 might be adapted for learning generative models with discrete inputs. The framework of the Banzhaf value for a game where there is a graph structure among players in Chapter 4 may be used as a mathematical model for some problems in economy. The algorithm developed in Chapter 6 essentially projects an arbitrary point onto a manifold embedded in Euclidean space, without access to the gradient of the manifold. Thus it may have a broader range of applications, such as serving as a retraction map for optimization on manifolds.

Bibliography

[1] Jimmy Ba, Volodymyr Mnih, and Koray Kavukcuoglu. "Multiple Object Recognition with Visual Attention". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

[2] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio. "Show, attend and tell: Neural image caption generation with visual attention". In: *International Conference on Machine Learning*. 2015, pp. 2048–2057.

[3] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. "DRAW: A recurrent neural network for image generation". In: *arXiv preprint arXiv:1502.04623* (2015).

[4] Kan Chen, Jiang Wang, Liang-Chieh Chen, Haoyuan Gao, Wei Xu, and Ram Nevatia. "ABC-CNN: An attention based convolutional neural network for visual question answering". In: *arXiv preprint arXiv:1511.05960* (2015).

[5] Zichao Yang, Xiaodong He, Jianfeng Gao, Li Deng, and Alex Smola. "Stacked attention networks for image question answering". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 21–29.

[6] Huijuan Xu and Kate Saenko. "Ask, attend and answer: Exploring question-guided spatial attention for visual question answering". In: *European Conference on Computer Vision*. Springer. 2016, pp. 451–466.

[7] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008.

[8] K. Simonyan, A. Vedaldi, and A. Zisserman. "Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2014.

[9] Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. "On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation". In: *PloS One* 10.7 (2015), e0130140.

[10] Avanti Shrikumar, Peyton Greenside, and Anshul Kundaje. "Learning Important Features Through Propagating Activation Differences". In: *ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, June 2017, pp. 3145–3153.

[11] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. "Visualizing and Understanding Recurrent Networks". In: *International Conference on Learning Representations*. 2016.

[12] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic Attribution for Deep Networks". In: *International Conference on Machine Learning*. 2017, pp. 3319–3328.

[13] F. Godin, K. Demuynck, J. Dambre, W. De Neve, and T. Demeester. "Explaining Character-Aware Neural Networks for Word-Level Prediction: Do They Discover Linguistic Rules?" In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL. Brussels, Belgium, 2018, pp. 3275–3284.

[14] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "Why should I trust you?: Explaining the predictions of any classifier". In: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM. 2016, pp. 1135–1144.

[15] Scott M Lundberg and Su-In Lee. "A Unified Approach to Interpreting Model Predictions". In: *Advances in Neural Information Processing Systems 30*. Ed. by I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett. Curran Associates, Inc., 2017, pp. 4765–4774. URL: `http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf`.

[16] Erik Štrumbelj and Igor Kononenko. "An Efficient Explanation of Individual Classifications using Game Theory". In: *Journal of Machine Learning Research* 11 (2010), pp. 1–18.

[17] Anupam Datta, Shayak Sen, and Yair Zick. "Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems". In: *Security and Privacy (SP), 2016 IEEE Symposium on*. IEEE. 2016, pp. 598–617.

[18] Jiwei Li, Will Monroe, and Dan Jurafsky. "Understanding neural networks through representation erasure". In: *arXiv preprint arXiv:1612.08220* (2016).

[19] Jianbo Chen, Le Song, Martin Wainwright, and Michael Jordan. "Learning to Explain: An Information-Theoretic Perspective on Model Interpretation". In: *International Conference on Machine Learning*. 2018, pp. 882–891.

[20] Jianbo Chen, Le Song, Martin J. Wainwright, and Michael I. Jordan. "L-Shapley and C-Shapley: Efficient Model Interpretation for Structured Data". In: *International Conference on Learning Representations*. 2019.

[21] Jianbo Chen and Michael I Jordan. "LS-Tree: Model Interpretation When the Data Are Linguistic". In: *arXiv preprint arXiv:1902.04187* (2019).

[22] Lloyd S Shapley. "A value for n-person games". In: *Contributions to the Theory of Games* 2.28 (1953), pp. 307–317.

[23] John F Banzhaf III. "Weighted voting doesn't work: A mathematical analysis". In: *Rutgers L. Rev.* 19 (1964), p. 317.

[24] Zachary C Lipton. "The mythos of model interpretability". In: *arXiv preprint arXiv:1606.03490* (2016).

[25] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. "Intriguing properties of neural networks". In: *International Conference on Learning Representations*. 2014.

[26] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. "Explaining and harnessing adversarial examples". In: *Proceedings of the International Conference on Learning Representations*. 2015.

[27] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. "Adversarial machine learning at scale". In: *International Conference on Learning Representations*. 2017.

[28] Nicolas Papernot, Patrick McDaniel, Somesh Jha, Matt Fredrikson, Z Berkay Celik, and Ananthram Swami. "The limitations of deep learning in adversarial settings". In: *2016 IEEE European Symposium on Security and Privacy*. IEEE. 2016, pp. 372–387.

[29] Nicholas Carlini and David Wagner. "Towards evaluating the robustness of neural networks". In: *2017 IEEE Symposium on Security and Privacy*. IEEE. 2017, pp. 39–57.

[30] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. "Towards Deep Learning Models Resistant to Adversarial Attacks". In: *International Conference on Learning Representations*. 2018.

[31] Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. "Deepfool: a simple and accurate method to fool deep neural networks". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 2574–2582.

[32] Pin-Yu Chen, Huan Zhang, Yash Sharma, Jinfeng Yi, and Cho-Jui Hsieh. "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM. 2017, pp. 15–26.

[33] Andrew Ilyas, Logan Engstrom, Anish Athalye, and Jessy Lin. "Black-box Adversarial Attacks with Limited Queries and Information". In: *International Conference on Machine Learning*. 2018, pp. 2142–2151.

[34] Andrew Ilyas, Logan Engstrom, and Aleksander Madry. "Prior Convictions: Black-box Adversarial Attacks with Bandits and Priors". In: *International Conference on Learning Representations*. 2019.

[35] Yanpei Liu, Xinyun Chen, Chang Liu, and Dawn Song. "Delving into transferable adversarial examples and black-box attacks". In: *Proceedings of the International Conference on Learning Representations*. 2017.

[36] Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples". In: *arXiv preprint arXiv:1605.07277* (2016).

[37] Nicolas Papernot, Patrick McDaniel, Ian Goodfellow, Somesh Jha, Z Berkay Celik, and Ananthram Swami. "Practical black-box attacks against machine learning". In: *Proceedings of the 2017 ACM on Asia Conference on Computer and Communications Security*. ACM. 2017, pp. 506–519.

[38] Wieland Brendel, Jonas Rauber, and Matthias Bethge. "Decision-Based Adversarial Attacks: Reliable Attacks Against Black-Box Machine Learning Models". In: *International Conference on Learning Representations*. 2018.

[39] Thomas Brunner, Frederik Diehl, Michael Truong Le, and Alois Knoll. "Guessing Smart: Biased Sampling for Efficient Black-Box Adversarial Attacks". In: *arXiv preprint arXiv:1812.09803* (2018).

[40] Minhao Cheng, Thong Le, Pin-Yu Chen, Huan Zhang, JinFeng Yi, and Cho-Jui Hsieh. "Query-Efficient Hard-label Black-box Attack: An Optimization-based Approach". In: *International Conference on Learning Representations*. 2019.

[41] Florian Tramèr, Alexey Kurakin, Nicolas Papernot, Ian Goodfellow, Dan Boneh, and Patrick McDaniel. "Ensemble Adversarial Training: Attacks and Defenses". In: *International Conference on Learning Representations*. 2018.

[42] Xuanqing Liu and Cho-Jui Hsieh. "Rob-GAN: Generator, Discriminator, and Adversarial Attacker". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2019.

[43] Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, Ken Nakae, and Shin Ishii. "Distributional smoothing with virtual adversarial training". In: *arXiv preprint arXiv:1507.00677* (2015).

[44] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. "Distillation as a defense to adversarial perturbations against deep neural networks". In: *2016 IEEE Symposium on Security and Privacy*. IEEE. 2016, pp. 582–597.

[45] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. "PixelDefend: Leveraging Generative Models to Understand and Defend against Adversarial Examples". In: *International Conference on Learning Representations*. 2018.

[46] Weilin Xu, David Evans, and Yanjun Qi. "Feature Squeezing: Detecting Adversarial Examples in Deep Neural Networks". In: *25th Annual Network and Distributed System Security Symposium, NDSS 2018, San Diego, California, USA, February 18-21, 2018*. 2018.

[47] Xuanqing Liu, Minhao Cheng, Huan Zhang, and Cho-Jui Hsieh. "Towards robust neural networks via random self-ensemble". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 369–385.

[48] Mathias Lecuyer, Vaggelis Atlidakis, Roxana Geambasu, Daniel Hsu, and Suman Jana. "Certified robustness to adversarial examples with differential privacy". In: *IEEE Symposium on Security and Privacy*. 2019.

[49] Xuanqing Liu, Yao Li, Chongruo Wu, and Cho-Jui Hsieh. "Adv-BNN: Improved Adversarial Defense through Robust Bayesian Neural Network". In: *International Conference on Learning Representations*. 2019.

[50] Eric Wong and J Zico Kolter. "Provable defenses against adversarial examples via the convex outer adversarial polytope". In: *International Conference on Machine Learning*. 2018.

[51] Krishnamurthy Dvijotham, Sven Gowal, Robert Stanforth, Relja Arandjelovic, Brendan O'Donoghue, Jonathan Uesato, and Pushmeet Kohli. "Training verified learners with learned verifiers". In: *arXiv preprint arXiv:1805.10265* (2018).

[52] Dimitris Tsipras, Shibani Santurkar, Logan Engstrom, Alexander Turner, and Aleksander Madry. "There is no free lunch in adversarial robustness (but there are unexpected benefits)". In: *arXiv preprint arXiv:1805.12152* (2018).

[53] Ludwig Schmidt, Shibani Santurkar, Dimitris Tsipras, Kunal Talwar, and Aleksander Madry. "Adversarially robust generalization requires more data". In: *Advances in Neural Information Processing Systems*. 2018, pp. 5019–5031.

[54] Xin Li and Fuxin Li. "Adversarial examples detection in deep networks with convolutional filter statistics". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 5764–5772.

[55] Arjun Nitin Bhagoji, Daniel Cullina, Chawin Sitawarin, and Prateek Mittal. "Enhancing robustness of machine learning systems via data transformations". In: *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*. IEEE. 2018, pp. 1–5.

[56] Dan Hendrycks and Kevin Gimpel. "Early methods for detecting adversarial images". In: *International Conference on Learning Representations*. 2017.

[57] Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. "A simple unified framework for detecting out-of-distribution samples and adversarial attacks". In: *Advances in Neural Information Processing Systems*. 2018, pp. 7167–7177.

[58] Kathrin Grosse, Praveen Manoharan, Nicolas Papernot, Michael Backes, and Patrick McDaniel. "On the (statistical) detection of adversarial examples". In: *arXiv preprint arXiv:1702.06280* (2017).

[59] Zhitao Gong, Wenlu Wang, and Wei-Shinn Ku. "Adversarial and clean data are not twins". In: *arXiv preprint arXiv:1704.04960* (2017).

[60] Jan Hendrik Metzen, Tim Genewein, Volker Fischer, and Bastian Bischoff. "On detecting adversarial perturbations". In: *International Conference on Learning Representations*. 2017.

[61] Reuben Feinman, Ryan R Curtin, Saurabh Shintre, and Andrew B Gardner. "Detecting adversarial samples from artifacts". In: *arXiv preprint arXiv:1703.00410* (2017).

[62] Xingjun Ma, Bo Li, Yisen Wang, Sarah M. Erfani, Sudanthi Wijewickrema, Grant Schoenebeck, Michael E. Houle, Dawn Song, and James Bailey. "Characterizing Adversarial Subspaces Using Local Intrinsic Dimensionality". In: *International Conference on Learning Representations*. 2018.

[63] R Dennis Cook. "Detection of influential observation in linear regression". In: *Technometrics* 19.1 (1977), pp. 15–18.

[64] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. "Greedy attack and Gumbel attack: Generating adversarial examples for discrete data". In: *arXiv preprint arXiv:1805.12316* (2018).

[65] Jianbo Chen, Michael I Jordan, and Martin J Wainwright. "HopSkipJumpAttack: a query-efficient decision-based adversarial attack". In: *arXiv preprint arXiv:1904.02144* (2019).

[66] Puyudi Yang, Jianbo Chen, Cho-Jui Hsieh, Jane-Ling Wang, and Michael I Jordan. "ML-LOO: Detecting Adversarial Examples with Feature Attribution". In: *arXiv preprint arXiv:1906.03499* (2019).

[67] David Baehrens, Timon Schroeter, Stefan Harmeling, Motoaki Kawanabe, Katja Hansen, and Klaus-Robert Müller. "How to explain individual classification decisions". In: *Journal of Machine Learning Research* 11 (2010), pp. 1803–1831.

[68] Roger B Myerson. "Graphs and cooperation in games". In: *Mathematics of Operations Research* 2.3 (1977), pp. 225–229.

[69] Andrew L Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. "Learning word vectors for sentiment analysis". In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2011, pp. 142–150.

[70] H Peyton Young. "Monotonic solutions of cooperative games". In: *International Journal of Game Theory* 14.2 (1985), pp. 65–72.

[71] Thomas M Cover and Joy A Thomas. *Elements of Information Theory*. John Wiley & Sons, 2012.

[72] Xiang Zhang, Junbo Zhao, and Yann LeCun. "Character-level convolutional networks for text classification". In: *Advances in Neural Information Processing Systems*. 2015, pp. 649–657.

[73] Andrej Karpathy, Justin Johnson, and Li Fei-Fei. "Visualizing and understanding recurrent networks". In: *arXiv preprint arXiv:1506.02078* (2015).

[74] Hendrik Strobelt, Sebastian Gehrmann, Hanspeter Pfister, and Alexander M Rush. "Lstmvis: A tool for visual analysis of hidden state dynamics in recurrent neural networks". In: *IEEE transactions on visualization and computer graphics* 24.1 (2018), pp. 667–676.

[75] W James Murdoch and Arthur Szlam. "Automatic rule extraction from long short term memory networks". In: *arXiv preprint arXiv:1702.02540* (2017).

[76] Yoon Kim. "Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1746–1751.

[77] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780.

[78] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: A simple way to prevent neural networks from overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[79] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. *Neural Networks for Machine Learning-Lecture 6a-Overview of mini-batch gradient descent*.

[80] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[81] Alex Krizhevsky. "Learning multiple layers of features from tiny images". In: (2009).

[82] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in Neural Information Processing Systems*. 2012, pp. 1097–1105.

[83] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *Proceedings of the International Conference on Learning Representations (ICLR)*. 2015.

[84] Maurice Kendall. *Rank Correlation Methods*. 4th ed. Charles Griffin, London, 1975.

[85] Yunlong Jiao and Jean-Philippe Vert. "The Kendall and Mallows kernels for permutations". In: *IEEE transactions on pattern analysis and machine intelligence* 40.7 (2018), pp. 1755–1769.

[86] Isabelle Guyon and André Elisseeff. "An introduction to variable and feature selection". In: *Journal of machine learning research* 3.Mar (2003), pp. 1157–1182.

[87] J.T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller. "Striving for Simplicity: The All Convolutional Net". In: *ICLR (workshop track)*. 2015. URL: `http://cia.informatik.uni-freiburg.de/Publications/2015/DB15a`.

[88] Pieter-Jan Kindermans, Kristof Schütt, Klaus-Robert Müller, and Sven Dähne. "Investigating the influence of noise and distractors on the interpretation of neural networks". In: *arXiv preprint arXiv:1611.07270* (2016).

[89] Shuyang Gao, Greg Ver Steeg, and Aram Galstyan. "Variational Information Maximization for Feature Selection". In: *Advances in Neural Information Processing Systems*. 2016, pp. 487–495.

[90] Hanchuan Peng, Fuhui Long, and Chris Ding. "Feature selection based on mutual information criteria of max-dependency, max-relevance, and min-redundancy". In: *IEEE Transactions on pattern analysis and machine intelligence* 27.8 (2005), pp. 1226–1238.

[91] Ronald J Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine Learning* 8.3-4 (1992), pp. 229–256.

[92] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. "Neural Machine Translation by Jointly Learning to Align and Translate". In: *arXiv e-prints* abs/1409.0473 (Sept. 2014).

[93] Eric Jang, Shixiang Gu, and Ben Poole. "Categorical reparameterization with gumbel-softmax". In: *stat* 1050 (2017), p. 1.

[94] Chris J Maddison, Daniel Tarlow, and Tom Minka. "A* sampling". In: *Advances in Neural Information Processing Systems*. 2014, pp. 3086–3094.

[95] Chris J Maddison, Andriy Mnih, and Yee Whye Teh. "The concrete distribution: A continuous relaxation of discrete random variables". In: *arXiv preprint arXiv:1611.00712* (2016).

[96] Colin Raffel, Thang Luong, Peter J Liu, Ron J Weiss, and Douglas Eck. "Online and linear-time attention by enforcing monotonic alignments". In: *arXiv preprint arXiv:1704.00784* (2017).

[97] Jianbo Chen, Mitchell Stern, Martin J Wainwright, and Michael I Jordan. "Kernel Feature Selection via Conditional Covariance Minimization". In: *Advances in Neural Information Processing Systems 30*. 2017, pp. 6949–6958.

[98] Ye Zhang and Byron Wallace. "A Sensitivity Analysis of (and Practitioners' Guide to) Convolutional Neural Networks for Sentence Classification". In: *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Vol. 1. 2017, pp. 253–263.

[99] François Chollet et al. *Keras*. `https://github.com/keras-team/keras`. 2015.

[100] Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. "A hierarchical neural autoencoder for paragraphs and documents". In: *arXiv preprint arXiv:1506.01057* (2015).

[101] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality". In: *Advances in Neural Information Processing Systems*. 2013, pp. 3111–3119.

[102] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Ng, and Christopher Potts. "Recursive deep models for semantic compositionality over a sentiment treebank". In: *Proceedings of the 2013 conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL. 2013, pp. 1631–1642.

[103] Tao Lei, Regina Barzilay, and Tommi Jaakkola. "Rationalizing Neural Predictions". In: *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL. 2016, pp. 107–117.

[104] Jiwei Li, Xinlei Chen, Eduard Hovy, and Dan Jurafsky. "Visualizing and Understanding Neural Models in NLP". In: *Proceedings of NAACL-HLT*. 2016, pp. 681–691.

[105] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[106] Andrzej S Nowak. "On an axiomatization of the Banzhaf value without the additivity axiom". In: *International Journal of Game Theory* 26.1 (1997), pp. 137–141.

[107] Peter L Hammer and Ron Holzman. "Approximations of pseudo-Boolean functions; applications to game theory". In: *Zeitschrift für Operations Research* 36.1 (1992), pp. 3–21.

[108] Ilya Katsev. "The Least Square Values for Games with Restricted Cooperation". In: *Game Theory and Management*. 2011, p. 117.

[109] Jack Sherman and Winifred J Morrison. "Adjustment of an inverse matrix corresponding to a change in one element of a given matrix". In: *The Annals of Mathematical Statistics* 21.1 (1950), pp. 124–127.

[110] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on Empirical Methods in Natural Language Processing (EMNLP)*. ACL. 2014, pp. 1532–1543.

[111] Yoav Goldberg and Joakim Nivre. "A dynamic oracle for arc-eager dependency parsing". In: *Proceedings of COLING 2012* (2012), pp. 959–976.

[112] Kenji Sagae and Alon Lavie. "A classifier-based parser with linear run-time complexity". In: *Proceedings of the Ninth International Workshop on Parsing Technology*. ACL. 2005, pp. 125–132.

[113] Yue Zhang and Stephen Clark. "Transition-based parsing of the Chinese treebank using a global discriminative model". In: *Proceedings of the 11th International Conference on Parsing Technologies*. ACL. 2009, pp. 162–171.

[114] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. "Fast and accurate shift-reduce constituent parsing". In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vol. 1. 2013, pp. 434–443.

[115] Selmer C Larson. "The shrinkage of the coefficient of multiple correlation." In: *Journal of Educational Psychology* 22.1 (1931), p. 45.

[116] Gianluca Stringhini, Christopher Kruegel, and Giovanni Vigna. "Detecting spammers on social networks". In: *Proceedings of the 26th annual computer security applications conference*. ACM. 2010, pp. 1–9.

[117] Sushmito Ghosh and Douglas L Reilly. "Credit card fraud detection with a neural-network". In: *System Sciences, 1994. Proceedings of the Twenty-Seventh Hawaii International Conference on*. Vol. 3. IEEE. 1994, pp. 621–630.

[118] Richard A Berk and Justin Bleich. "Statistical procedures for forecasting criminal behavior". In: *Criminology & Public Policy* 12.3 (2013), pp. 513–544.

[119] J Zico Kolter and Marcus A Maloof. "Learning to detect and classify malicious executables in the wild". In: *Journal of Machine Learning Research* 7.Dec (2006), pp. 2721–2744.

[120] David West. "Neural network credit scoring models". In: *Computers & Operations Research* 27.11-12 (2000), pp. 1131–1152.

[121] Nilesh Dalvi, Pedro Domingos, Sumit Sanghai, Deepak Verma, et al. "Adversarial classification". In: *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM. 2004, pp. 99–108.

[122] Nicolas Papernot, Patrick McDaniel, Ananthram Swami, and Richard Harang. "Crafting adversarial input sequences for recurrent neural networks". In: *Military Communications Conference, MILCOM 2016-2016 IEEE*. IEEE. 2016, pp. 49–54.

[123] Ji Gao, Jack Lanchantin, Mary Lou Soffa, and Yanjun Qi. "Black-box Generation of Adversarial Text Sequences to Evade Deep Learning Classifiers". In: *arXiv preprint arXiv:1801.04354* (2018).

[124] Robin Jia and Percy Liang. "Adversarial Examples for Evaluating Reading Comprehension Systems". In: *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*. 2017, pp. 2021–2031.

[125] Suranjana Samanta and Sameep Mehta. "Towards Crafting Text Adversarial Samples". In: *arXiv preprint arXiv:1707.02812* (2017).

[126] Minhao Cheng, Jinfeng Yi, Huan Zhang, Pin-Yu Chen, and Cho-Jui Hsieh. "Seq2sick: Evaluating the robustness of sequence-to-sequence models with adversarial examples". In: *arXiv preprint arXiv:1803.01128* (2018).

[127] Bin Liang, Hongcheng Li, Miaoqiang Su, Pan Bian, Xirong Li, and Wenchang Shi. "Deep Text Classification Can be Fooled". In: *arXiv preprint arXiv:1704.08006* (2017).

[128] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60.2 (2004), pp. 91–110.

[129] N Danal. "Histgram of Oriented Gradients for Human Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2005, pp. 886–893.

[130] Abraham D Flaxman, Adam Tauman Kalai, and H Brendan McMahan. "Online convex optimization in the bandit setting: gradient descent without a gradient". In: *Proceedings of the Sixteenth Annual ACM-SIAM Symposium on Discrete Algorithms*. SIAM. 2005, pp. 385–394.

[131] Alekh Agarwal, Dean P Foster, Daniel J Hsu, Sham M Kakade, and Alexander Rakhlin. "Stochastic convex optimization with bandit feedback". In: *Advances in Neural Information Processing Systems*. 2011, pp. 1035–1043.

[132] Yurii Nesterov and Vladimir Spokoiny. "Random gradient-free minimization of convex functions". In: *Foundations of Computational Mathematics* 17.2 (2017), pp. 527–566.

[133] John C Duchi, Michael I Jordan, Martin J Wainwright, and Andre Wibisono. "Optimal rates for zero-order convex optimization: The power of two function evaluations". In: *IEEE Transactions on Information Theory* 61.5 (2015), pp. 2788–2806.

[134] Sijia Liu, Bhavya Kailkhura, Pin-Yu Chen, Paishun Ting, Shiyu Chang, and Lisa Amini. "Zeroth-order stochastic variance reduction for nonconvex optimization". In: *Advances in Neural Information Processing Systems*. 2018, pp. 3731–3741.

[135] Alex Kantchelian, JD Tygar, and Anthony Joseph. "Evasion and hardening of tree ensemble classifiers". In: *International Conference on Machine Learning*. 2016, pp. 2387–2396.

[136] Hongge Chen, Huan Zhang, Duane Boning, and Cho-Jui Hsieh. "Robust Decision Trees Against Adversarial Examples". In: *International Conference on Machine Learning*. 2019, pp. 1122–1131.

[137] Lu Wang, Xuanqing Liu, Jinfeng Yi, Zhi-Hua Zhou, and Cho-Jui Hsieh. "Evaluating the Robustness of Nearest Neighbor Classifiers: A Primal-Dual Perspective". In: *arXiv preprint arXiv:1906.03972* (2019).

[138] Yao-Yuan Yang, Cyrus Rashtchian, Yizhen Wang, and Kamalika Chaudhuri. "Adversarial Examples for Non-Parametric Methods: Attacks, Defenses and Large Sample Limits". In: *arXiv preprint arXiv:1906.03310* (2019).

[139] Jan A Snyman. *Practical Mathematical Optimization*. Springer, 2005.

[140] Joel A Tropp. "User-friendly tail bounds for sums of random matrices". In: *Foundations of computational mathematics* 12.4 (2012), pp. 389–434.

[141] Martin J Wainwright. *High-Dimensional Statistics: A Non-Asymptotic Viewpoint*. Vol. 48. Cambridge University Press, 2019.

[142] David Kincaid, David Ronald Kincaid, and Elliott Ward Cheney. *Numerical Analysis: Mathematics of Scientific Computing*. Vol. 2. American Mathematical Soc., 2009.

[143] Michel Ledoux. *The Concentration of Measure Phenomenon*. 89. American Mathematical Soc., 2001.

[144] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Identity mappings in deep residual networks". In: *European conference on computer vision*. Springer. 2016, pp. 630–645.

[145] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. "ImageNet: A Large-Scale Hierarchical Image Database". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2009.

[146] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[147] Jonas Rauber, Wieland Brendel, and Matthias Bethge. "Foolbox: A Python toolbox to benchmark the robustness of machine learning models". In: *arXiv preprint arXiv:1707.04131* (2017).

[148] Anish Athalye, Nicholas Carlini, and David Wagner. "Obfuscated Gradients Give a False Sense of Security: Circumventing Defenses to Adversarial Examples". In: *International Conference on Machine Learning*. 2018, pp. 274–283.

[149] Xiaoyu Cao and Neil Zhenqiang Gong. "Mitigating evasion attacks to deep neural networks via region-based classification". In: *Proceedings of the 33rd Annual Computer Security Applications Conference*. ACM. 2017, pp. 278–287.

[150] Guneet S. Dhillon, Kamyar Azizzadenesheli, Jeremy D. Bernstein, Jean Kossaifi, Aran Khanna, Zachary C. Lipton, and Animashree Anandkumar. "Stochastic activation pruning for robust adversarial defense". In: *International Conference on Learning Representations*. 2018.

[151] Jeremy Cohen, Elan Rosenfeld, and Zico Kolter. "Certified Adversarial Robustness via Randomized Smoothing". In: *International Conference on Machine Learning*. 2019, pp. 1310–1320.

[152] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. "Mitigating Adversarial Effects Through Randomization". In: *International Conference on Learning Representations*. 2018.

[153] Nicholas Carlini and David Wagner. "Adversarial examples are not easily detected: Bypassing ten detection methods". In: *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*. ACM. 2017, pp. 3–14.

[154] Alexey Kurakin, Ian J Goodfellow, and Samy Bengio. "Adversarial Examples in the Physical World". In: *Artificial Intelligence Safety and Security*. Chapman and Hall/CRC, 2018, pp. 99–112.

[155] Ian Goodfellow, Nicolas Papernot, Sandy Huang, Yan Duan, and Peter Abbeel. "Attacking Machine Learning with Adversarial Examples". In: *Open AI Blog* (2017).

[156] Thomas Tanay and Lewis Griffin. "A boundary tilting persepective on the phenomenon of adversarial examples". In: *arXiv preprint arXiv:1608.07690* (2016).

[157] Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. "Analysis of classifiers' robustness to adversarial perturbations". In: *Machine Learning* 107.3 (2018), pp. 481–508.

[158] Alhussein Fawzi, Seyed-Mohsen Moosavi-Dezfooli, and Pascal Frossard. "Robustness of classifiers: from adversarial to random noise". In: *Advances in Neural Information Processing Systems*. 2016, pp. 1632–1640.

[159] Pei-Hsuan Lu, Pin-Yu Chen, and Chia-Mu Yu. "On the limitation of local intrinsic dimensionality for characterizing the subspaces of adversarial examples". In: *arXiv preprint arXiv:1803.09638* (2018).

[160] Matthew D Zeiler and Rob Fergus. "Visualizing and understanding convolutional networks". In: *European conference on computer vision*. Springer. 2014, pp. 818–833.

[161] Amirata Ghorbani, Abubakar Abid, and James Zou. "Interpretation of neural networks is fragile". In: *arXiv preprint arXiv:1710.10547* (2017).

[162] Chih-Kuan Yeh, Cheng-Yu Hsieh, Arun Sai Suggala, David Inouye, and Pradeep Ravikumar. "How Sensitive are Sensitivity-Based Explanations?" In: *arXiv preprint arXiv:1901.09392* (2019).

[163] Prasad Chalasani, Somesh Jha, Aravind Sadagopan, and Xi Wu. "Adversarial Learning and Explainability in Structured Datasets". In: *arXiv preprint arXiv:1810.06583* (2018).

[164] Guanhong Tao, Shiqing Ma, Yingqi Liu, and Xiangyu Zhang. "Attacks meet interpretability: Attribute-steered detection of adversarial samples". In: *Advances in Neural Information Processing Systems*. 2018, pp. 7728–7739.

[165] Chiliang Zhang, Zuochang Ye, Yan Wang, and Zhimou Yang. "Detecting Adversarial Perturbations with Saliency". In: *2018 IEEE 3rd International Conference on Signal and Image Processing (ICSIP)*. IEEE. 2018, pp. 271–275.

[166] JP Royston. "Algorithm AS 177: Expected normal order statistics (exact and approximate)". In: *Journal of the royal statistical society. Series C (Applied statistics)* 31.2 (1982), pp. 161–165.

[167] Yoav Benjamini and Yosef Hochberg. "Controlling the false discovery rate: a practical and powerful approach to multiple testing". In: *Journal of the Royal statistical society: series B (Methodological)* 57.1 (1995), pp. 289–300.

[168] Aaditya Ramdas, Jianbo Chen, Martin J Wainwright, and Michael I Jordan. "A sequential algorithm for false discovery rate control on directed acyclic graphs". In: *Biometrika* 106.1 (Jan. 2019), pp. 69–86. ISSN: 0006-3444. DOI: 10.1093/biomet/asy066.

[169] Pradeep Dubey and Lloyd S Shapley. "Mathematical properties of the Banzhaf power index". In: *Mathematics of Operations Research* 4.2 (1979), pp. 99–131.

[170] Rakesh Agrawal and Ramakrishnan Srikant. "Privacy-preserving data mining". In: *ACM Sigmod Record*. Vol. 29. 2. ACM. 2000, pp. 439–450.