APPLICATIONS OF THE MULTI-CONSTRAINT MOST PROBABLE EXPLANATION PROBLEM

by

Brij Gulsharan Malhotra

APPROVED BY SUPERVISORY COMMITTEE:

Vibhav Gogate, Chair

Nicholas Ruozzi

Rishabh Iyer

I dedicate my thesis to

my parents and my brother

who have supported me and

who have always believed in me.

APPLICATIONS OF THE MULTI-CONSTRAINT MOST PROBABLE EXPLANATION PROBLEM

by

BRIJ GULSHARAN MALHOTRA, B.TECH

THESIS

Presented to the Faculty of

The University of Texas at Dallas

in Partial Fulfillment

of the Requirements

for the Degree of

MASTER OF SCIENCE IN

COMPUTER SCIENCE

THE UNIVERSITY OF TEXAS AT DALLAS

May 2021

ACKNOWLEDGMENTS

Foremost, I would like to thank my advisor, Professor Vibhav Gogate for the guidance and support that he provided me in the past two semesters. I could not have asked for a better MS thesis experience. I would also like to thank my labmates and colleagues at Professor Gogate's lab - Tahrima Rahman, Sara Rouhani, Shasha Jin, Chiradeep Roy, Vasundhara Komaragiri, and Shivvrat Arya for suggesting last-minute improvements to my defense presentation.

I also thank Mr. Soham Kulkarni, my friend and colleague at the Erik Johnsson School of Engineering and Computer Science, UTD, who spent a lot of time helping me debug certain parts of the code required for this thesis. And lastly, I would like to thank my girlfriend, Ms. Kanka Kulkarni, who deserves a special mention for her support throughout my graduate studies at UTD.

April 2021

APPLICATIONS OF THE MULTI-CONSTRAINT MOST PROBABLE EXPLANATION PROBLEM

Brij Gulsharan Malhotra, MS
The University of Texas at Dallas, 2021

Supervising Professor: Vibhav Gogate, Chair

Probabilistic models are often used in practice to represent and reason about uncertainty. A key reasoning task over them is finding the most probable assignment to all the unobserved variables given observations. This task called the most probable explanation (MPE) task has several applications including finding the most likely (1) topic for a given document, (2) disease given symptoms and (3) price of gold tomorrow given historical gold prices.

In this thesis, we consider a multi-constrained version of the MPE task (MCMPE) which is defined as the task of computing the most probable explanation given a set of user-specified constraints. We demonstrate three possible real-world applications of MCMPE. Our first application is about finding the most probable completion of an image under the constraint that the label assigned to the image is either flipped or remains the same. Our second application is about discovering possible adversarial attacks and involves minimally modifying an image/example such that the decision made by the classifier changes. Our final application is about estimating the robustness of a probabilistic classifier. We propose solving the MCMPE task by encoding it as a mixed linear integer programming task and using an open-source solver such as SCIP for solving the latter. We demonstrate empirically the feasibility and practicality of our proposed approach as well as its applicability on the MNIST digits and the SMS Spam Filtering datasets.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

INTRODUCTION

This thesis is motivated by the following two scenarios that are prevalent in real-world machine learning systems:

1. You build an explainable machine learning system—a system that not only learns from experience and makes decisions but is also able to explain why it arrived at a particular decision—as follows. You first construct a discriminative multi-class classifier from training data that takes as input a full or partial feature vector (a full or partial assignment of values to a set of attributes/features) and outputs a decision (label). Then, you construct a generative probabilistic model for generating explanations; in particular, the model takes a partial feature vector (where values of some features are missing) as input and generates, via probabilistic inference, a most probable completion of the feature vector yielding a full feature vector and an assignment of values to other explanatory variables. At test time, you are given a partial feature vector and you use the two models to provide a label for the feature vector and an explanation that completes the partial vector. However, there is a mismatch between the explanation and the label in that the classifier will assign a different label to the completed or full feature vector if the latter is provided as an input to the classifier.

2. You are given a machine learning classifier that has high accuracy but wants to know if an adversary can easily fool the classifier. Specifically, you want to know if the classifier is *robust* in that it is not easy to force the classifier to make a mistake by changing the values of only a few features. Finally, you also want to know if such attacks can be thwarted if you do not disclose the particulars of the classifier to a malicious actor.

We show that although the issues mentioned in the two scenarios appear to be distinct, they can be addressed by solving a new, unifying probabilistic inference task, which we call

multi-constraint most probable explanation (MCMPE). At a high level, the MCMPE task is a constrained version of the most probable explanation problem (MPE) [11]. Given a probabilistic graphical model (PGM) $\mathcal{M}$ and a set of inequality constraints $\mathcal{C}$, each defined over a set of random variables $\mathbf{X}$, we want to find the most probable assignment to all the variables in $\mathbf{X}$ w.r.t $\mathcal{M}$ such that all constraints in $\mathcal{C}$ are satisfied.

A special case of the MCMPE problem is the recently proposed Constrained Most Probable Explanation (CMPE) problem (see Rouhani et al.([21], [20])). Unlike MCMPE which allows for multiple constraints, the CMPE task allows only one constraint. Rouhani et al. showed that by using graph-based conditioning methods, the CMPE task can be reduced to the multi-choice knapsack problem (MCKP) [10] and can thus be solved efficiently using specialized branch and bound [14] and local search [6] algorithms for MCKP. Rouhani et al. demonstrated via a large-scale experimental evaluation that their specialized methods outperform general-purpose optimization solvers based on mixed-integer linear programming (MILP) [22] by orders of magnitude in some cases. Unfortunately, because of the presence of multiple constraints, MCMPE is much harder than CMPE, and therefore we formulate it as MILP and solve it using off-the-shelf solvers such as SCIP [7] (imported using the Google OR-tools [16]). We leave developing specialized algorithms for solving MCMPE, similar to those developed by Rouhani et al. [21], to future work.

To summarize, the contributions of this thesis are two-fold:

- We define a new optimization task over probabilistic graphical models called multi-constraint most probable explanation.

- We demonstrate how this task can be effectively solved in practice using off-the-shelf mixed-integer linear programming solvers and how it can be used to solve three important real-world application tasks in explainable artificial intelligence (XAI) and robust estimation.

The rest of the thesis is organized as follows: Chapter 2 describes the applications of the MCMPE problem that have been considered in this thesis. Chapter 3 describes experimental results and Chapter 4 concludes the thesis and discusses potential future work.

CHAPTER 2

APPLICATIONS OF THE MCMPE PROBLEM

In this chapter, we define the multi-constraint most probable explanation problem (MCMPE) and describe how three real-world application tasks can be reduced to MCMPE. We begin by formally defining the MCMPE task.

2.1 The Multi-Constraint Most Probable Explanation Problem

Let $\mathbf{X} = \{X_1, \ldots, X_n\}$ be set of discrete random variables or features. Without loss of generality, we assume that each variable X_i takes values from the domain $D_i = \{1, \ldots, d\}$. A log-potential f is a real-valued function from all possible value assignments to a subset of variables $S(f) \subseteq \mathbf{X}$ to $\mathbb{R}$. $S(f)$ is called the scope of f.

A probabilistic graphical model $\mathcal{M}$ is a triple $\langle \mathbf{X}, \mathbf{F}, G \rangle$ where $\mathbf{X}$ is a set of discrete random variables as described above, $\mathbf{F} = \{f_1, \ldots, f_m\}$ is a set of log-potentials and G is an undirected graph called the interaction graph or primal graph. G has a vertex for each variable $X_i \in \mathbf{X}$ and an edge between two vertices if the corresponding variables appear in the scope of a function $f \in \mathbf{F}$. $\mathcal{M}$ represents the following probability distribution:

$$P(\mathbf{x}) = \frac{1}{Z} \exp\left(\sum_{i=1}^{m} f_i(\mathbf{x}_{S(f_i)})\right)$$

where $\mathbf{x}$ is an assignment of values to all variables in the set $\mathbf{X}$ and $\mathbf{x}_{S(f_i)}$ denotes the projection of $\mathbf{x}$ on the scope $S(f_i)$ of f_i. Often we will abuse notation and write $\mathbf{x}_{S(f_i)}$ as $\mathbf{x}$ when the scope of the function is clear from the context.

Given a PGM $\mathcal{M} \equiv (\mathbf{X}, \mathbf{F}, G)$ and a set of sets of log-linear functions $\{\mathbf{F}_1, \ldots, \mathbf{F}_k\}$ such that for each j where $1 \leq j \leq k$ and $f \in \mathbf{F}_j$, the scope of f is a subset of $\mathbf{X}$, namely $S(f) \subseteq \mathbf{X}$, the multi-constraint most probable explanation task is defined mathematically as follows:

$$\begin{aligned} \arg\max_{\mathbf{x}} \quad & \sum_{f \in \mathbf{F}} f(\mathbf{x}) \\ \text{s.t.} \quad & \sum_{f \in \mathbf{F}_j} f(\mathbf{x}) \leq 0;\ \forall j \in \{1, \ldots, k\} \end{aligned} \tag{2.1}$$

The MCMPE task is NP-hard in general and includes the most probable explanation task as a special case.

In the next three sections, we detail three applications of MCMPE: (1) Decision Preserving Most Probable Completions under user-defined constraints; (2) Adversarial Attacks on Probabilistic Models and (3) Estimating Robustness of Classification Algorithms.

2.2 Decision Preserving Most Probable Explanations/Completions under User-Defined Constraints

A major challenge in explainable machine learning is to make robust decisions and generate meaningful explanations in presence of missing data at test time, namely when values of a subset of the features/variables are not observed at test time. The issue is that there may be a mismatch between the decision and explanation (without loss of generality, we define explanation as an assignment of values to all the unobserved variables) in that the full assignment obtained by conjoining the explanation and observations may change the decision. To circumvent this issue, we have to ensure that the explanation preserves the decision. This yields a new task which we call decision preserving most probable explanation.

Formally, let $\mathcal{M} \equiv (\mathbf{X}, \mathbf{F}, G)$ be a PGM such that the variables in the PGM are partitioned into three disjoint subsets: unobserved variables $\mathbf{H}$, observed or evidence variables $\mathbf{E}$ and a decision variable C. At test time, given observation $\mathbf{E} = \mathbf{e}$, let $C = c^*$ be the decision made by $\mathcal{M}$; the decision can be made by solving the following inference task:

$$c^* = \arg\max_{c} \sum_{\mathbf{h}} \exp\left(\sum_{f \in \mathbf{F}} f(c, \mathbf{h}, \mathbf{e}) \right)$$

Given $C = c^*$, the decision preserving most probable explanation task is to generate an assignment $\mathbf{h}^*$ to $\mathbf{H}$ such that $\sum_{f \in \mathbf{F}} f(c^*, \mathbf{h}^*, \mathbf{e}) \geq \sum_{f \in \mathbf{F}} f(c, \mathbf{h}^*, \mathbf{e})$ for all $c \neq c^*$ and the probability is maximized w.r.t. $\mathcal{M}$. Mathematically,

$$\begin{aligned} \arg\max_{\mathbf{h}} \quad & \sum_{f \in \mathbf{F}} f(c^*, \mathbf{h}, \mathbf{e}) \\ \text{s.t.} \quad & \sum_{f \in \mathbf{F}} f(c^*, \mathbf{h}, \mathbf{e}) \geq \sum_{f \in \mathbf{F}} f(c, \mathbf{h}, \mathbf{e}) \; \forall \; c \in \{1, \ldots, d\} \text{ and } c \neq c^* \end{aligned} \tag{2.2}$$

It is easy to see that the task described in Eq. (2.2) is an instance of MCMPE (see Eq. (2.1)). For each value c such that $c \neq c^*$ and $f \in \mathbf{F}$, we can use the function $g_c(\mathbf{h}) = f(c, \mathbf{h}, \mathbf{e}) - f(c^*, \mathbf{h}, \mathbf{e})$ to express the $d - 1$ constraints in Eq. (2.2) in the same form as the ones in Eq. (2.1). If we relax the decision preserving constraint the task simply reduces to the Most Probable Explanation (MPE) task [11].

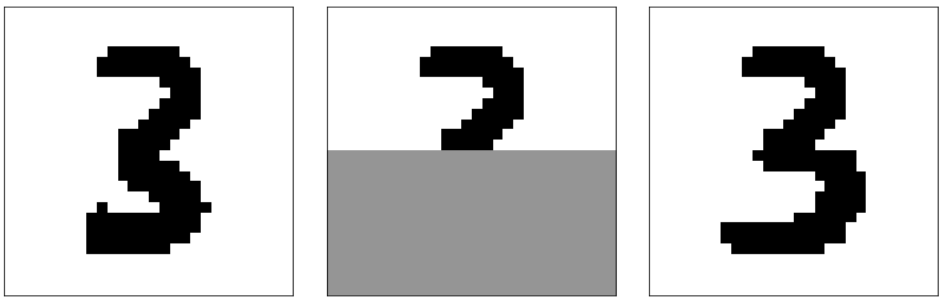
2.2.1 An example of Decision Preserving MPE

Consider that we have trained a Naive Bayes classifier [2] on the MNIST handwritten digit dataset [12], denoted by $\mathbb{D}$, to recognize handwritten digits.[1] A Naive Bayes model is a probabilistic graphical model having the property that all the features (variables) are independent of each other conditioned on the value of the class variable (or the decision variable). The MNIST dataset consists of 60000 images of size 28×28 each and every image contains a handwritten digit with every image categorized using labels as belonging to one of the 10 digits 0–9. The pixels in the images in the original MNIST dataset take values between 0 and 1. We have modified the images in the dataset such that the pixel having values greater than 0 will now have a value equal to 1. Given an incomplete test example $t \in \mathbb{D}$, such as an image whose bottom half is unobserved, if we predict a class $C = c^*$ by summing out the unobserved variables $\mathbf{H}$, we want to assign values to the variables $\mathbf{H} = \mathbf{h}$ such that the

[1] All the classifiers used in this thesis have been trained using the scikit-learn library[15] mostly with their default parameters.

classifier would predict the same class on the complete image as it had made on the incomplete image. Since the Naive Bayes model has a property that the feature variables become independent given the value of the class variable, summing out the unobserved variables is computationally easy.

Figure 2.1 shows an example output of decision preserving MPE for an image belonging to class "3" whose bottom half was intentionally designated as unobserved (shown by graying out the bottom half of the image) and the Naive Bayes classifier made the correct prediction on the incomplete image.

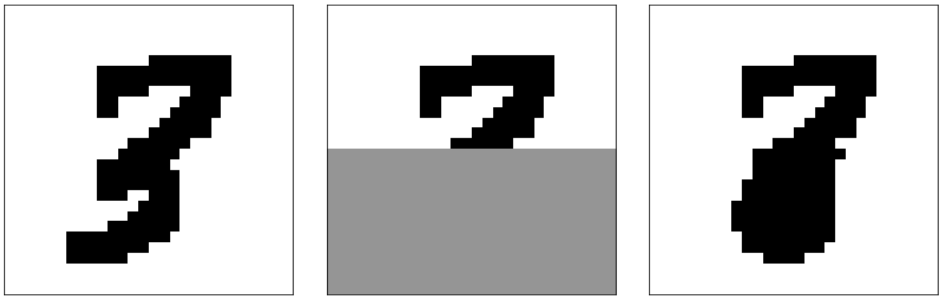


(a) Original Image (b) Incomplete Image (c) Completed Image

Figure 2.1: Decision preserving most probable completions when correct decision is made on the incomplete image by the Naive Bayes classifier.

Similarly, when we show an incomplete image belonging to the same class, and the Naive Bayes classifier makes an incorrect prediction, the values assigned to the unobserved variables are such that the decision is preserved. Figure 2.2 shows one such example of an image belonging to class "3" on which the Naive Bayes model predicted the class "8" and therefore the completions in the bottom half of the image look like an eight.

Thus formulating this task as the MCMPE problem provides us with a practical approach for inspecting whether it is required to observe certain variables to make robust decisions or not. For example, in Figure 2.1, even if we do not observe the bottom half of the image, we are still able to predict it is "3" correctly and when we do not predict the class correctly like in Figure 2.2 we are still completing the image based on what prediction we have made.



(a) Original Image (b) Incomplete Image (c) Completed Image

Figure 2.2: Decision preserving most probable completions when incorrect decision is made on the incomplete image by the Naive Bayes classifier.

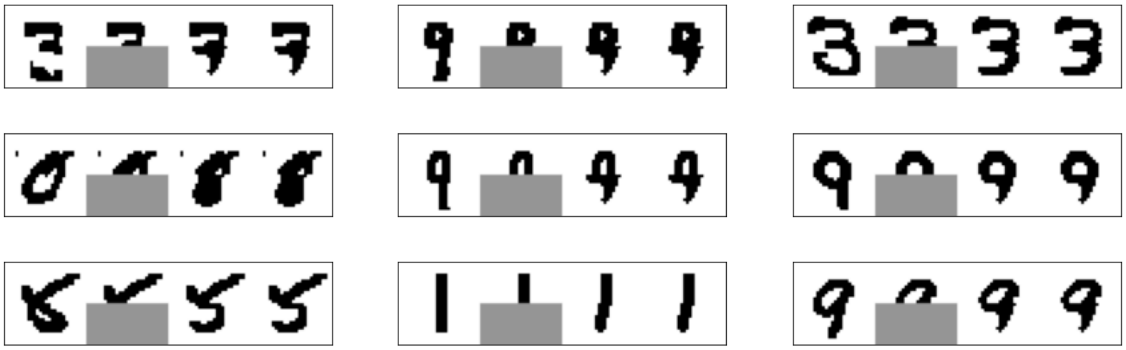


Figure 2.3: Decision preserving most probable completions for images belonging to different classes. First Column in each subfigure is the original image, second column is the incomplete image, third column are the MPE completions and the fourth column are the decision preserving MPE completions

Figure 2.3 shows MPE completions and decision preserving most probable completions for incomplete images belonging to different classes. The incomplete images have their bottom halves intentionally left unobserved. The MPE completions and the decision preserving completions, in case, of the Naive Bayes model[2] usually are the same because it has been observed that MPE completions themselves typically do not violate the decision preserving constraint in the Eq. (2.2).

[2]All the experiments have been performed only on the Naive Bayes model in this section.

2.3 Adversarial Most Probable Explanations

There has been significant research on adversarial attacks on neural networks [8]. The goal of a malicious actor or an adversary is to manipulate the predictions made by machine learning systems. For example, Szegedy et al. [23] have demonstrated a method that uses small perturbation learned by maximizing the network prediction error to change the prediction of the neural network. Brown et al. [4] in their paper have demonstrated a method to create an "Adversarial Patch" which when attached to an image can allow for robust and targeted attacks on a neural network. Both of these methods use backpropagation to learn the perturbations and the patch respectively. In this section, we describe a method that uses the MCMPE problem encoded as a MILP problem to manipulate the image to attack a classifier (namely force the classifier to make a wrong decision) and thus does not require learning/backpropagation.

The Adversarial Most Probable Explanation problem can be viewed as an opposite task to the decision preserving Most Probable Completion problem (2.2). Given a probabilistic graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{F}, G \rangle$, we want to find the most probable assignment to unobserved variables $\mathbf{H}$=$\mathbf{h}$ such that the decision made by the classifier for a test example $\boldsymbol{t} \in \mathbb{D}$ equals a prior user-defined target class $C = c^*$ (out of d possible classes) given evidence $\mathbf{E}$=$\mathbf{e}$.

Formally, given a graphical model $\mathcal{M} = \langle \mathbf{X}, \mathbf{F}, G \rangle$ where $\mathbf{X} = \{C\} \cup \mathbf{E} \cup \mathbf{H}$, we want to solve the following combinatorial optimization problem,

$$\begin{aligned} \arg\max_{\mathbf{h}} \quad & \sum_{f \in F} f(\mathbf{h}, c^*, \mathbf{e}) \\ \text{s.t.} \quad & \sum_{f \in F} f(\mathbf{h}, c^*, \mathbf{e}) \leq \sum_{f \in F} f(\mathbf{h}, c_i, \mathbf{e}) \;\; \forall c_i \in \{1, \ldots, d\} \text{ and} c_i \neq c^* \end{aligned} \tag{2.3}$$

It is easy to see that the task described in Eq. (2.3) is an instance of MCMPE (see Eq. (2.1)).

An example using a text classification dataset

Consider the SMS spam collection dataset [1] which is a text classification dataset. It contains text messages which can belong to one of two classes, namely, spam and ham. Consider a Naive Bayes model which has been trained using the Bag of Words (BOW) model [9]. The goal of an adversary will be to add certain words to a spam message such that the message gets past the spam filtering classifier. In other words, they would want to add certain words to a spam message such that it is classified as "ham" rather than "spam". This problem can be easily formulated as an Adversarial Most Probable Explanation task. We treat the words to be added to the message as unobserved in the test example and the words present in the message as the evidence. Thus, finding the most probable assignment to the unobserved words such that the decision is targeted to be "ham" given the words in the message will lead to the change in the decision from "spam" to "ham" made by the Naive Bayes model.

Figure 2.4 shows an example of what are the most probable words that could be added to a spam message such that the classifier predicts 'ham." The original message has words such as "prize" and "claim" which have a high positive log-odds ratio and that is why the original message will be classified as "spam." Adding words with high negative log-odds ratio compensates for the spam words and the classification changes to "ham."

The Bag of Words model allows us to formulate the problem such that we could add multiple words of the same type. Figure 2.5 shows a comparison of the distribution of words (using word clouds) being selected when experiments were run with two different settings of maximum words of the same type being allowed.

2.3.1 Minimality Constraint

Suppose now we want to influence the decision of the classifier under an additional constraint that the number of unobserved variables that can be manipulated can be at most a user-defined constant k. In other words, we want to minimally manipulate the given example

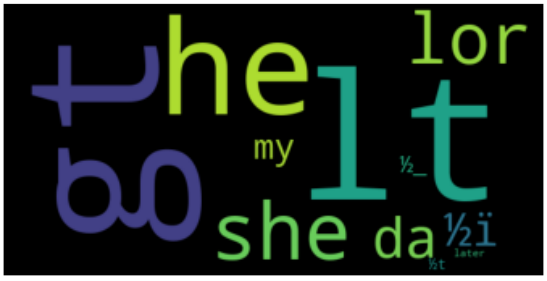
Word	Count	Log-odds ratio
callnumbr	1	0.987
cash	1	2.828
claim	1	**5.551**
have	1	-0.236
number	1	1.642
or	1	0.746
prize	1	**5.278**
to	1	0.149
won	1	-5.03
you	1	-0.877

Add Words

Word	count	Log-odds ratio
da	1	-3.841
gt	1	-4.577
he	1	-4.195
lor	1	-3.903
lt	1	-4.574
she	1	-3.911

Figure 2.4: Changing to ham from spam by adding most probable words

(a) Wordcloud with maximum words of same type set to 1.



(b) Wordcloud with maximum words of same type set to 2.

Figure 2.5: Different distribution of words selected for two different settings.

such that the decision of the generative classifier changes from one class to another. This can be done by adding a constraint on the total number of feature variables taking a particular value.

An example using the MNIST dataset

Consider the task of handwritten digit recognition using the MNIST dataset [12] using the Naive Bayes classifier [2] with pixels taking values 0 (white pixels) and 1 (black pixels). We use the same modification that we used in section 2.2.1. Suppose, we want to minimally manipulate the white pixels to black pixels such that the classifier makes a user-defined target prediction. Therefore, the white pixels form the set of the unobserved variables $\mathbf{H} \subseteq \mathbf{X}$ and the black pixels form the set of evidence variables $\mathbf{E} \subseteq \mathbf{X}$.

This task can be easily formulated as the Adversarial Most Probable Explanation problem with an additional constraint that the number of pixels in the set $\mathbf{H}$ that can take the value 1 (because we will be manipulating white pixels (value 0) to take the value 1 (and thus become black)) is smaller than or equal to a user-defined constant k. In particular, we want to solve the following task,

$$\begin{aligned} \arg\max_{h} \quad & \sum_{f \in F} f(\mathbf{h}, c^*, \mathbf{e}) \\ \text{s.t.} \quad & \sum_{f \in F} f(\mathbf{h}, c^*, \mathbf{e}) \leq \sum_{f \in F} f(\mathbf{h}, c_i, \mathbf{e}) \;\; \forall c_i \in \{1, \ldots, d\} \text{ and } c_i \neq c^* \\ & \sum_{H \in \mathbf{H}} \mathbb{I}_{\mathbf{h}}(H = 1) \leq k \end{aligned} \tag{2.4}$$

For example, Figure 2.6 shows an image of "8" which was minimally manipulated to influence the decision of the Naive Bayes classifier to a target class set to "3." The value of k was set to 7 (thus only 7 white pixels were allowed to flip from white to black).

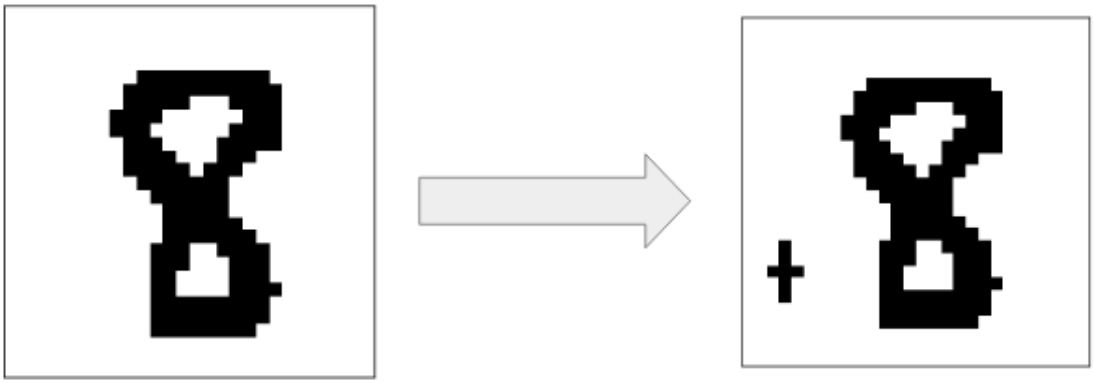


Figure 2.6: Adversarial explanations with minimal manipulation

Soft Constraints

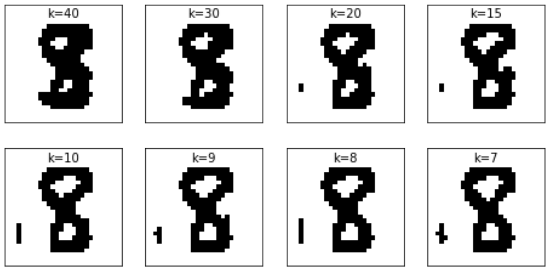
Sometimes setting the value of k too low can render the MILP problem infeasible. To solve this problem, we can replace the hard constraint on the number of pixels in Eq. (2.4) with a soft constraint that introduces a penalty term in the objective function (Lagrange relaxation). This removes the dependability of the feasibility of the problem on the value of k. The modified formulation is as follows

$$\begin{aligned} \arg\max_{\mathbf{h}} \quad & \sum_{f \in F} f(\mathbf{h}, c^*, \mathbf{e}) - \alpha \sum_{H \in \mathbf{H}} \mathbb{I}_{\mathbf{h}}(H = 1) \\ \text{s.t.} \quad & \sum_{f \in F} f(\mathbf{h}, c^*, \mathbf{e}) \leq \sum_{f \in F} f(\mathbf{h}, c_i, \mathbf{e}) \;\; \forall c_i \in \{1, \ldots, d\}, \text{ and } c_i \neq c^* \end{aligned} \tag{2.5}$$

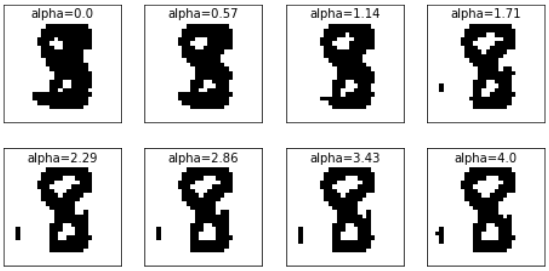
Figure 2.7 shows a comparison between how the optimal solutions change with decreasing the value of k and increasing the value α in the two formulations respectively.

2.3.2 Adversarial Explanantions using Discriminative Classifiers

We do not necessarily need to model the joint probability of the decision variable C in the constraints which means we can also use discriminative classifiers [2] to perform the task of Adversarial Explanations as long as we model the joint probability of the feature variables $\mathbf{X}'$ where $\mathbf{X}' = \mathbf{X} \setminus C$ using a discriminative probabilistic graphical model $\mathcal{M} =$



(a) Effect of decreasing the value of $\mathbf{k}$ on the optimal solution



(b) Effect of increasing the value of α on the optimal solution

Figure 2.7: Comparison of effects of k and α.

$(\mathbf{X}', \mathbf{F}, G)$. Discriminative classifiers allow us to explicitly model the conditional probability of the decision variable given the feature variables i.e., $P(C|\mathbf{X}')$. Assuming that the user-defined target class is c^* out of the d possible classes, Eq. (2.4), therefore, changes as follows

$$\begin{aligned} \max_{\mathbf{h}} \quad & \sum_{f \in F} f(\mathbf{h}, \mathbf{e}) \\ \text{s.t.} \quad & P(C = c^*|\mathbf{h}, \mathbf{e}) \geq P(C = c_i|\mathbf{h}, \mathbf{e}) \ \ \forall c_i \in \{1, \ldots, d\}, \text{ and } c_j \neq c^* \\ & \sum_{H \in \mathbf{H}} I_{\mathbf{h}}(H = 1) \leq k \end{aligned} \tag{2.6}$$

In this thesis, we have used Chow-Liu trees ([5],[11]) to model the joint probability of the variables $\mathbf{X}'$ and in our experiments, we use three different discriminative classifiers to set up the constraints.

Logistic Regression Classifier

Logistic Regression classifier [2] is a linear discriminative classifier that learns a linear discriminant function given labeled dataset $\mathbb{D}$ and the parameters of the function are the weight matrix $W \in \mathbb{R}^{d \times n}$ and a bias vector $b \in \mathbb{R}^{d \times 1}$ where d is the number of classes and n is the number of features in a training example. Given a test example $\mathbf{t} \in \mathbb{D}$, the decision made by the classifier is given by,

$$\arg\max_{d} \quad W_{d \times n} * t_{n \times 1} + b_{d \times 1} \tag{2.7}$$

Therefore, if we want to manipulate the decision such that the decision given by the logistic regression classifier is the target class c^* we would want the value of $col_j(W) \cdot t + b$ to be the maximum among all the classes where j denotes the index of c^*. Therefore, the adversarial completions task with the minimality constraint for the logistic regression classifier given the unobserved variables $\mathbf{H}$ and the evidence variables $\mathbf{E}$=$\mathbf{e}$ is formulated as,

$$\begin{aligned} \arg\max_{\mathbf{h}} \quad & \sum_{f \in F} f(\mathbf{h}, \mathbf{e}) \\ \text{s.t.} \quad & col_j(W) \cdot t + col_j(b) \geq col_i(W) \cdot t + col_i(b) \;\; \forall i \in \{1, \ldots, d\}, \text{and} \;\; i \neq j \\ & \sum_{H \in \mathbf{H}} I_{\mathbf{h}}(H = 1) \leq k \end{aligned} \tag{2.8}$$

Figure 2.8 shows an example of the different outputs for an image belonging to class 2, for each of the remaining classes set as the target class using the soft constraint formulation and the white pixels being treated as unobserved and black pixels forming the evidence variables.

Decision Trees

It is well known that a decision tree can be expressed as a disjunction of conjunctions, namely in disjunctive normal form (DNF), such that the conjunctions are mutually exclusive and exhaustive. In particular, every path in the tree from root to the leaf is a conjunction of test on the feature values and the tree itself is the disjunction of these conjunctions [13].

The decision tree recursively splits the training data at each node based on the feature value being considered at each node. The decision taken at the leaf for a test example $\mathbf{t}$ is the class that has the maximum number of examples at the leaf node. Naturally, in order to influence the decision of the decision tree to a particular target class, we choose the paths consistent with the target class and the value of the evidence variables. Let the set of paths in the decision tree that are consistent with target class c^* be denoted by $\mathbb{P}$ and $|\mathbb{P}| = p$. Let the maximum number of node in the i^{th} path in the set $\mathbb{P}$ be m and each j^{th} feature value be denoted by h_{ij}. We will have to choose exactly one path from the set $\mathbb{P}$ and

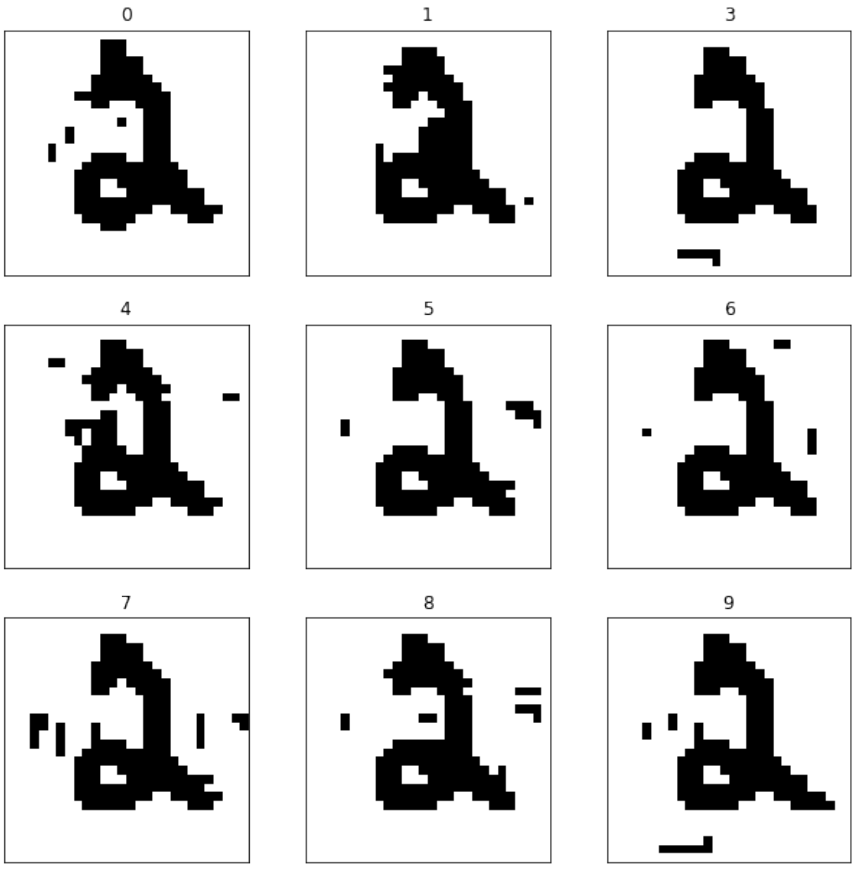


Figure 2.8: Comparison of different outputs for an image belonging to class 2 with the decision being changed to all the other classes for a logistic regression classifier. The soft constraint formulation is being used and the value of α is set to 3

the values assigned to the unobserved variables in the chosen path should be such that the overall sum should at least the number of nodes in the path. Therefore, the formulation of the adversarial explanations task for decision trees is as shown in Eq. (2.9). Note that the formulation applies only to boolean feature variables but can be extended to non-boolean variables using additional terms. Figure 2.9 shows an example of the different outputs of the optimal solution for all the classes set as the target class using the formulation in Eq. (2.9) with the white pixels being treated as unobserved and black pixels forming the set of evidence variables for an image belonging to the class 2.

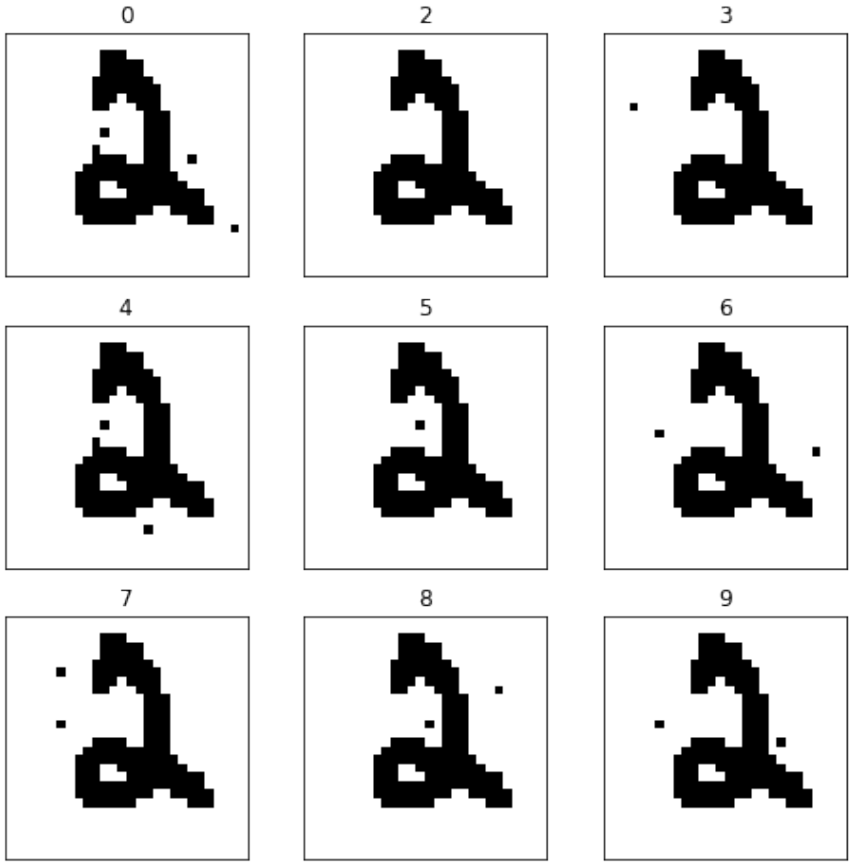


Figure 2.9: Comparison of different outputs for an image belonging to class 2 with the decision being changed to all the other classes for a decision tree. The minimality constraint is not being used.

$$
\begin{aligned}
\arg\max_{\mathbf{h}} \quad & \sum_{f \in F} f(\mathbf{h}, \mathbf{e}) \\
\text{s.t.} \quad & \sum_{j=1}^{m} (1 - \overline{h_{ij}}) + h_{ij} \geq m \times z_i \\
& \sum_{i=1}^{p} z_i = 1, \\
& \sum_{H \in \mathbf{H}} I_{\mathbf{h}}(H = 1) \leq k \\
& z_i \in (0, 1), i = 1, 2, 3, \ldots, p
\end{aligned}
\tag{2.9}
$$

Random Forests

Random Forests [3] is an ensemble of decision trees in which decisions are made via a majority vote. This makes random forests more robust to noise and overfitting as compared to decision trees. We can simply extend the formulation used in the subsection 2.3.2 to random forests. We want at least one more than half the decision trees in the ensemble to make predictions consistent with the target class. Let the number of trees in the ensemble be r. Let the set of consistent paths for e^{th} decision tree in the random forest be $\mathbb{P}_e$ with cardinality p_e. The minimality constraint in the case of random forests is applied to each decision tree independently. The formulation will be modified as follows,

$$
\begin{aligned}
\arg\max_{h} \quad & \sum_{f \in F} f(\mathbf{h}, \mathbf{e}) \\
\text{s.t.} \quad & \sum_{i=1}^{m} (1 - \overline{h_{ij}}) + h_{ij} \geq m \times z_i^{(e)}, \ e \in (1, r) \\
& \sum_{i=1}^{p_e} z_i^{(e)} \leq 1, \ e \in (1, r) \\
& \sum_{e=1}^{r} \sum_{i=1}^{p_e} z_{ei} \geq \left\lfloor \frac{r}{2} \right\rfloor + 1 \\
& \sum_{H \in \mathbf{H}} I_{\mathbf{h}}(H^{(e)} = 1) \leq k, \ e \in (1, r) \\
& z_i^{(e)} \in (0, 1), \ i = 1, 2, 3, ..., p \text{ and } e = 1, 2, 3, ..., r
\end{aligned}
\tag{2.10}
$$

Figure 2.10 shows an example of the different outputs of the optimal solution for each of the remaining classes as the target class using the above formulation and the white pixels being treated as unobserved and black pixels forming the set of evidence variables for an image belonging to the class 2. We can see from Figure 2.9 and Figure 2.10 that we have to change more pixels to change the decisions made by random forests as compared to decision trees.

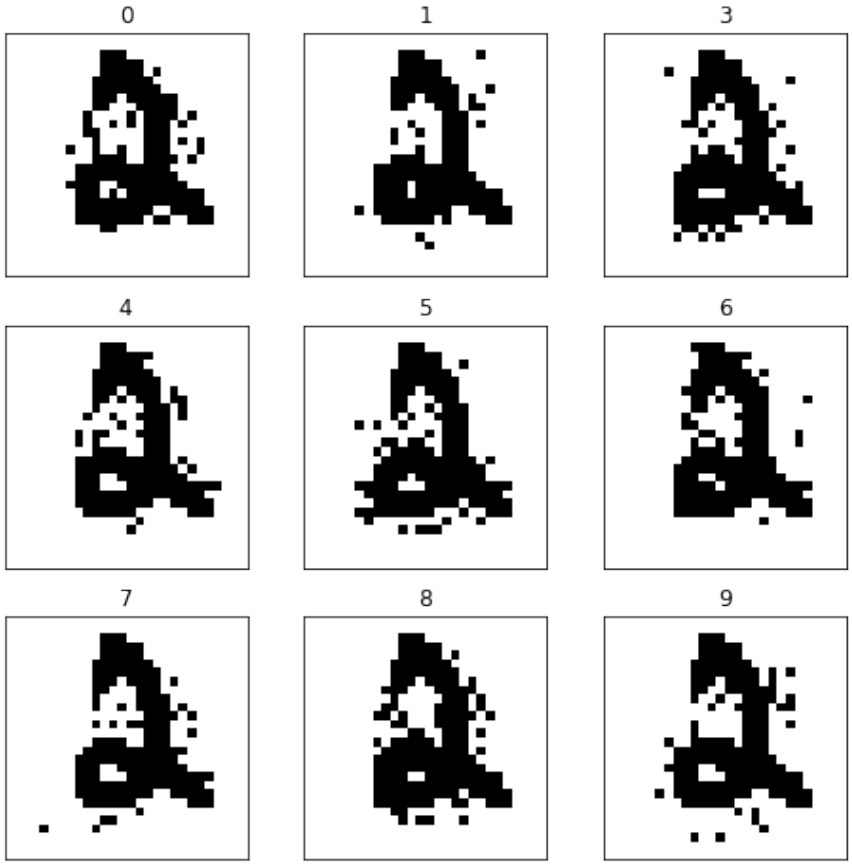


Figure 2.10: Comparison of optimal solutions for an image belonging to class 2 with the decision being changed to all the other classes for a random forest when r=10. The minimality constraint is not being used.

2.4 Estimating Robustness of Different Classifiers

Table 2.1 shows the average percentage of images that were successfully flipped to a different class and the average time required in seconds for 20 randomly chosen images and target class over 10 different runs. The MILP solver was allowed to run for a maximum of 10 seconds. It can be seen from the table that the Decision Tree classifier can be easily influenced even for small values of k while for the Logistic Regression classifier as we increase the value of k the average percentage of images flipped increases. Although the robustness of Random Forests with 10 decision trees compares favorably with that of the Decision Tree, it takes longer for

Table 2.1: Comparing the average percentage of images whose decision was flipped and time required to find the optimal solution(per image) for different classifiers for changing values of k on 20 randomly chosen images and randomly chosen target class for 10 different random trials using the same Chow-Liu tree. The solver was allowed to run for maximum of 10 seconds.

Classifier	Percentage Flipped				Time (in secs)			
k	5	10	20	30	5	10	20	30
Logistic Regression	33.0 ± 10.77	64.5 ± 15.24	93.0 ± 4.58	98.0 ± 2.45	2.44 ± 0.33	3.24 ± 0.53	4.65 ± 0.92	4.92 ± 0.82
Decision Tree	88.0 ± 4.58	89.5 ± 4.72	89.5 ± 4.72	89.5 ± 4.72	0.24 ± 0.04	0.29 ± 0.09	0.26 ± 0.02	0.27 ± 0.03
Random Forests(10)	88.0 ± 6.40	88.5 ± 7.09	88.5 ± 7.09	88.5 ± 7.09	6.37 ± 0.55	7.02 ± 0.70	6.94 ± 0.67	6.97 ± 0.67
Random Forests(50)	14.0 ± 5.83	23.5 ± 7.43	25.5 ± 7.57	26.5 ± 7.43	9.32 ± 0.78	9.63 ± 0.79	9.65 ± 0.76	9.67 ± 0.78

the MILP solver to find optimal/feasible solutions. Random Forest with 50 decision trees is observed to be the most robust because even for higher values of k, the percentage of flipped images is lower.

CHAPTER 3

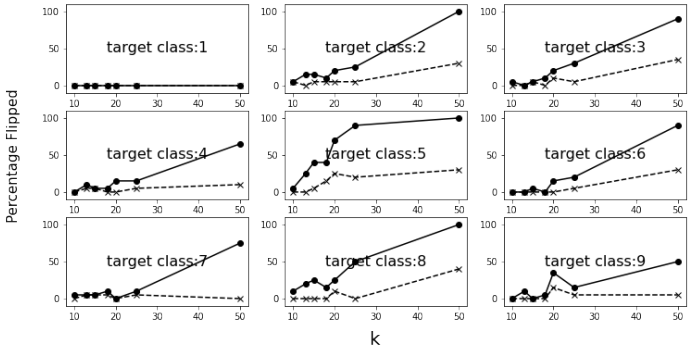
EXPERIMENTAL RESULTS

3.1 Attacking Unknown Classifiers

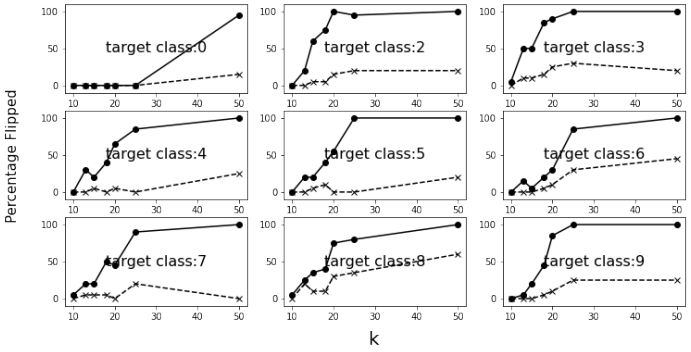
In Chapter 2, we showed how to find adversarial explanations under the assumption that the functional form (type) and parameters of the classifier are known. In this chapter, we consider a harder task: finding adversarial explanations when the functional form (type) or parameters or both, of the classifier, are not known (namely the classifier is unknown). We accomplish this objective using a clone of the unknown classifier.

In the experimental data shown in Figure 3.1, we have shown plots comparing two classifiers. The first classifier is a logistic regression classifier trained on the entire MNIST dataset. Let us call this classifier as LR(a). The second classifier is trained on 1000 images with 100 images from each class with labels generated by predictions from the first logistic regression classifier i.e. LR(a) classifier. Let us call this classifier as LR(b). We count an attack to be successful if we try to flip an image using the LR(b) classifier to a target class and LR(a) classifier also predicts the same target class.

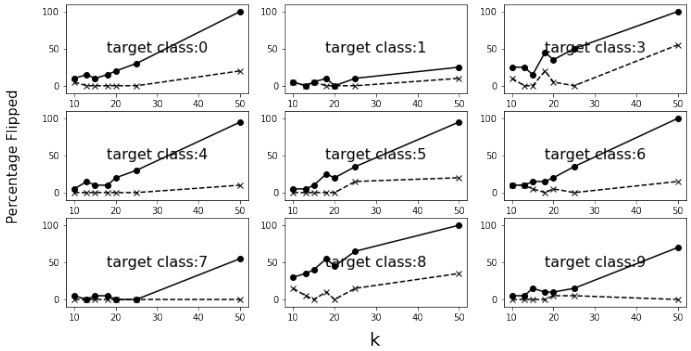
Figure 3.1 shows a comprehensive comparison of percentage of images flipped for 20 different randomly chosen images from each class to be flipped to every other remaining class. Within each plot, solid lines with circle markers show the change in percentage of images flipped with change in the value of k for the LR(a) classifier and the dashed lines with cross markers show the change in percentage of images flipped for the LR(a) classifier using an image manipulated using the LR(b) classifier.



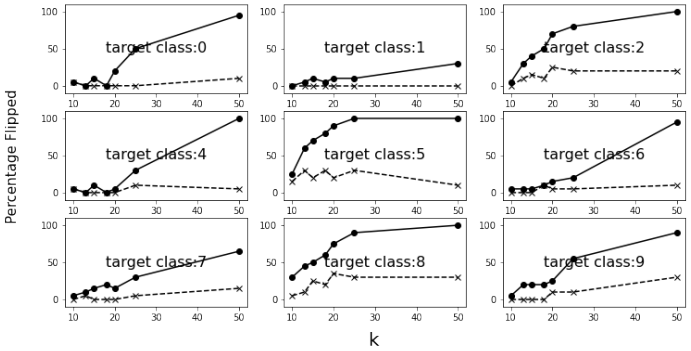
(a) Class 0



(b) Class 1

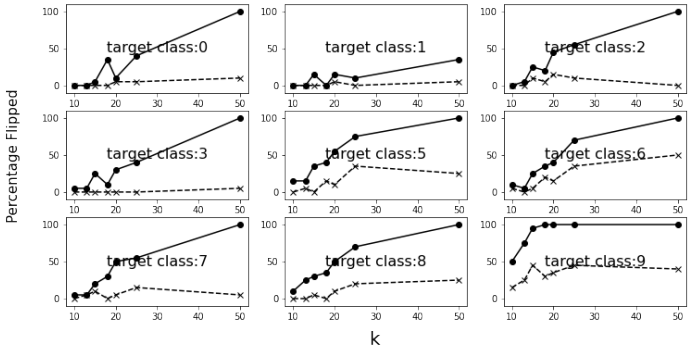


(c) Class 2

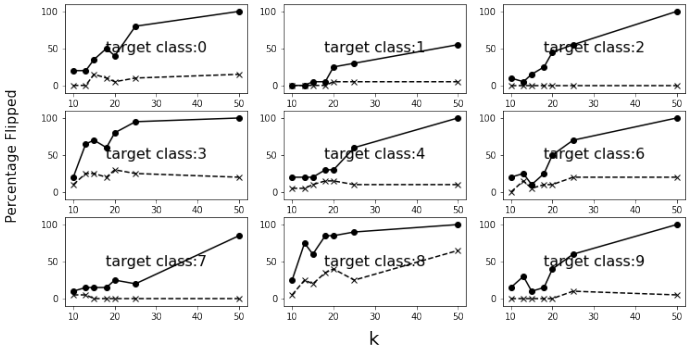


(d) Class 3

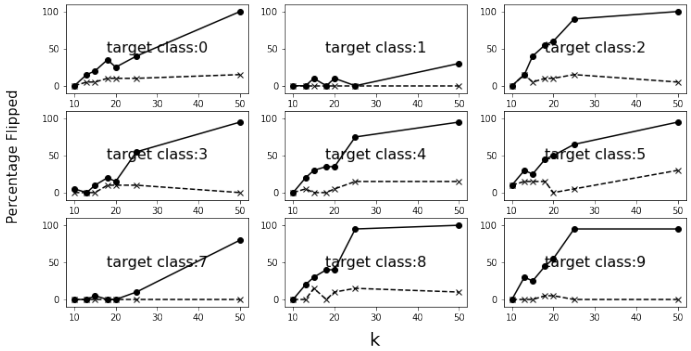
Figure 3.1: Plots showing change in the percentage of images for which we successfully flipped the decision of the two logistic regression classifiers to a randomly chosen target class with increasing k



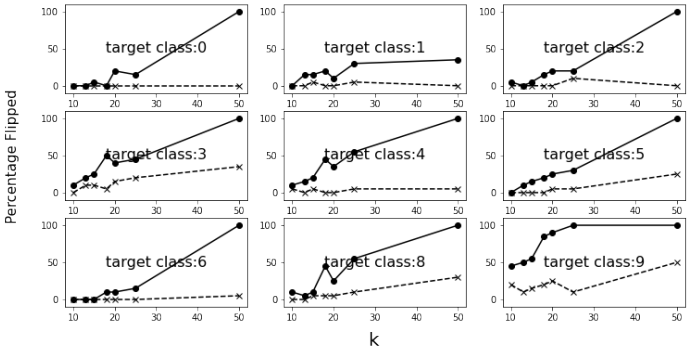
(e) Class 4



(f) Class 5

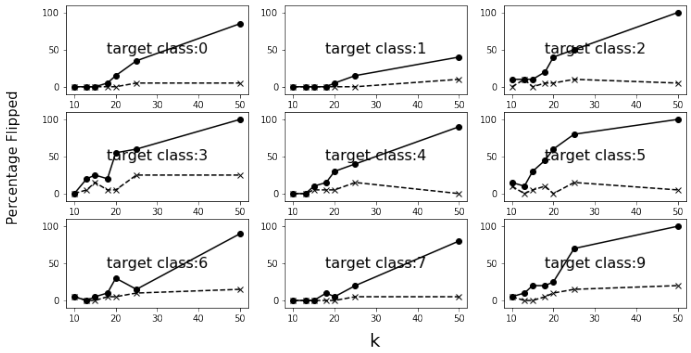


(g) Class 6

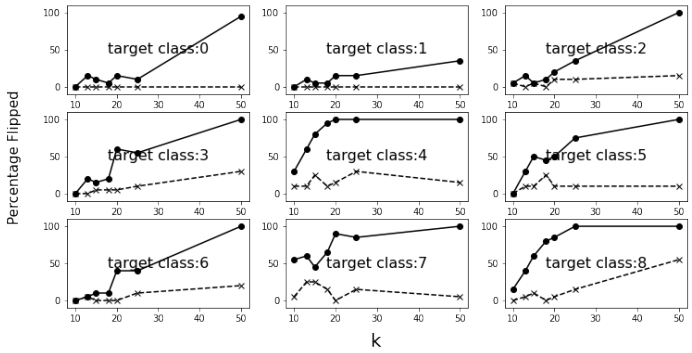


(h) Class 7

Figure 3.1: Plots showing change in the percentage of images for which we successfully flipped the decision of the two logistic regression classifiers to a randomly chosen target class with increasing k



(i) Class 8



(j) Class 9

Figure 3.1: Plots showing change in the percentage of images for which we successfully flipped the decision of the two logistic regression classifiers to a randomly chosen target class with increasing k

CHAPTER 4

CONCLUSION AND FUTURE SCOPE

In this thesis, we have shown that the two distinct tasks of inspecting the effects of missing data on making robust decisions and manipulating the decisions made by machine learning classifiers (when the classifier is either known or unknown) can be formulated as the multi-constraint most probable explanation problem. We can also use the adversarial most probable explanation task to estimate the robustness of different machine learning classifiers which, in turn, can be used for training robust classifiers. The MCMPE problem is NP-hard in general and one way to solve it is to formulate the problem as a mixed-integer linear programming (MILP) problem. This approach works well for sparse graphical models like the Naive Bayes model or Chow-Liu trees and datasets like the MNIST dataset with a small number of random variables. However, as we move to denser graphical models([19], [18], [17]) and datasets with large number of random variables this problem will become harder to solve. Therefore we need specialized algorithms, potentially adapted from the literature on the Knapsack Problem and its variants([10], [21]), to solve the MCMPE problem.

REFERENCES

[1] Almeida, T., J. Gómez Hidalgo, and A. Yamakami. Contributions to the study of sms spam filtering: New collection and results. In *Proceedings of the 2011 ACM Symposium on Document Engineering (DOCENG'11), Mountain View, CA, USA, 2011*. `http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/`.

[2] Alpaydin, E. (2010). *Introduction to Machine Learning* (2nd ed.). The MIT Press.

[3] Breiman, L. (2001, October). Random forests. *Mach. Learn. 45*(1), 5–32. `https://doi.org/10.1023/A:1010933404324`.

[4] Brown, T. B., D. Mané, A. Roy, M. Abadi, and J. Gilmer (2017). Adversarial patch.

[5] Chow, C. and C. Liu (1968). Approximating discrete probability distributions with dependence trees. *IEEE Transactions on Information Theory 14*(3), 462–467.

[6] Crama, Y., A. W. J. Kolen, and E. J. Pesch (1995). *Local search in combinatorial optimization*, pp. 157–174. Berlin, Heidelberg: Springer Berlin Heidelberg. `https://doi.org/10.1007/BFb0027029`.

[7] Gamrath, G., D. Anderson, K. Bestuzheva, W.-K. Chen, L. Eifler, M. Gasse, P. Gemander, A. Gleixner, L. Gottwald, K. Halbig, G. Hendel, C. Hojny, T. Koch, P. Le Bodic, S. J. Maher, F. Matter, M. Miltenberger, E. Mühmer, B. Müller, M. E. Pfetsch, F. Schlösser, F. Serrano, Y. Shinano, C. Tawfik, S. Vigerske, F. Wegscheider, D. Weninger, and J. Witzig (2020, March). The SCIP Optimization Suite 7.0. Technical report, Optimization Online. `http://www.optimization-online.org/DB_HTML/2020/03/7705.html`.

[8] Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

[9] Jurafsky, D. and J. H. Martin (2000). *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition* (1st ed.). USA: Prentice Hall PTR.

[10] Kellerer, H., U. Pferschy, and D. Pisinger (2004). *Knapsack Problems*. Springer, Berlin, Germany.

[11] Koller, D. and N. Friedman (2009). *Probabilistic Graphical Models: Principles and Techniques - Adaptive Computation and Machine Learning*. The MIT Press.

[12] LeCun, Y., C. Cortes, and C. Burges (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: `http://yann.lecun.com/exdb/mnist` 2*.

[13] Mitchell, T. M. (1997). *Machine Learning* (1 ed.). USA: McGraw-Hill, Inc.

[14] Morrison, D. R., S. H. Jacobson, J. J. Sauppe, and E. C. Sewell (2016, February). Branch-and-bound algorithms. *Discret. Optim. 19*(C), 79–102.

[15] Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

[16] Perron, L. and V. Furnon. Or-tools. `https://developers.google.com/optimization/`.

[17] Rahman, T. and V. Gogate (2016). Learning ensembles of cutset networks. In *AAAI conference on Artificial Intelligence*, pp. 3301–3307.

[18] Rahman, T., S. Jin, and V. Gogate (2019). Look ma, no latent variables: Accurate cutset networks via compilation. In K. Chaudhuri and R. Salakhutdinov (Eds.), *Proceedings of the Thirty-Sixth International Conference on Machine Learning*, Volume 97 of *Proceedings of Machine Learning Research*, pp. 5311–5320. PMLR.

[19] Rahman, T., P. Kothalkar, and V. Gogate (2014). Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2014, Nancy, France, September 15-19, 2014. Proceedings, Part II*, pp. 630–645.

[20] Rouhani, S., T. Rahman, and V. Gogate (2018). Algorithms for the nearest assignment problem. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence*, pp. 5096–5102.

[21] Rouhani, S., T. Rahman, and V. Gogate (2020). A novel approach for constrained optimization in graphical models. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin (Eds.), *Advances in Neural Information Processing Systems*, Volume 33, pp. 11949–11960. Curran Associates, Inc. `https://proceedings.neurips.cc/paper/2020/file/8ab9bb97ce35080338be74dc6375e0ed-Paper.pdf`.

[22] Schrijver, A. (1986). *Theory of Linear and Integer Programming*. USA: John Wiley & Sons, Inc.

[23] Szegedy, C., W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus (2013). Intriguing properties of neural networks.

BIOGRAPHICAL SKETCH

Brij G. Malhotra joined The University of Texas at Dallas as a graduate student majoring in the field of Computer Science after working for three years in India after completing his Bachelor's Degree in Electronics and Telecommunication Engineering. He aimed to improve his skills in the field of Machine Learning by choosing the Intelligent systems track at UTD. Enrolling in courses like Machine Learning and Statistical Methods in AI and ML motivated him to pursue research in this domain. This thesis is the result of his efforts in this direction.

CURRICULUM VITAE

Brij G. Malhotra

bxm190007@utdallas.edu

Educational History:

B.Tech, Electronics and Telecommunication Engineering
Vishwakarma Institue of Technology, India, 2016

M.S., Computer Science
The University of Texas at Dallas, U.S.A, 2021(expected)

Employment History:

Software Development Engineer Intern, Amazon, May 2020 - August 2020
Machine Learning Engineer, Digitalmain, July 2018 - June 2019
Machine Learning Consultant, Tweeny Technologies, January 2018 - July 2018
Software Engineer, Pixopal, June 2017 - January 2018
Software Developer, Unway Technologies, September 2016 - June 2017

Past Projects:

Pavement crack detection using Convolutional Neural Networks, 2018
Illegal Activity Recognition in videos using Inverse Reinforcement Learning, 2015-2016

Volunteer Work:

Technical Coordinator, Artificial Intelligence Society, UTD, 2019-2020
Teaching Assistant, Workshop on Real World AI Projects, UTD, 2020