Convex Optimization Final Report

Huafeng Fan

May 13, 2019

1 Introduction

Advances in computational power and in algorithmic design have together enabled the understanding of deep insights from large collections of data. The field of Machine Learning aims to study how to learn mathematical models of complex phenomena from these large collections of data. Machine Learning has led to the developments that were previously unimaginable. These developments span the gamut from teaching computers how to recognize handwritten digits to the development of fully autonomous cars on the road. Evidently, Machine Learning has changed the way we think about many problems that we faced, and the field has the potential to lead to even further advancements in many different fields.

Machine Learning algorithms rely on having a large corpus of data from which mathematical models can be learned. In essence, the performance of a Machine Learning algorithm is fundamentally limited by the quality and quantity of the data used for the training of the model. Regardless of how advanced a computational algorithm is, the learned model will not perform well if it is not fed representative and clean data from which it can learn. Therefore, the typical Machine Learning system has high requirements for the quality and precision of the data it uses for training. In the industry, a large fraction of engineering time and effort is spent building and maintaining complex pipelines through which data is obtained, pre-processed, and eventually fed into the Machine Learning system. In this process, most of the data obtained may be noisy and therefore not suitable to be fed into the Machine Learning system, leaving only a small percentage of the data actually usable for model training. The effects of having unclean data and having to discard a majority of data affects each discipline differently. For domains like analyzing images of cats, losing some data may not be as big of a deal because many pictures of cats exist. For other domains like discovering new particles at CERN, there is a dearth of good data that can produce new Physics in that domain, so having access to more clean data quicker can only serve to facilitate further discoveries.

This project aims to study an algorithm that is robust to missing data. This project focuses only on the case of binary classification in the case where the data is not perfectly separable, but similar approaches may be applied to other Machine Learning problems as well. The hope is that by taking into account how uncertain we are about the integrity of our data, we can formulate better algorithms that can take that into account and obtain better results, instead of just assuming that all of the data used for the Machine Learning is perfect. To this end, this project studies the classical binary classification algorithm of Support Vector Machine (SVM) in the context of the appearance of messy and missing data.

2 Related Work

The formulation of the optimization problem and the application of the problem to classification in the presence of missing data is introduced and explained in [1]. This project largely aims to replicate their findings and attempt their approach on different datasets than they used in their paper.

3 The Optimization Problem

This section derives the optimization problem used in the implementation of this project.

3.1 Classic Support Vector Machine Algorithm

In the case where we know the values of all of our training data, one classic algorithm to perform binary classification is the Support Vector Machine algorithm. This algorithm aims to find a hyperplane which separates the input data into two classes. This hyperplane should minimize the margins present between the training data and the hyperplane itself.

Formally, given the training data $\{(x_i, y_i)\}_{i=1}^m$ where $x_i \in \mathbb{R}^n$ are the feature vectors and $y_i \in \{-1, 1\}$ are their corresponding labels, we aim to find a hyperplane $w^T x + b = 0$ such that for a feature vector $p \in \mathbb{R}^n$ whose label is unknown, we can predict its label by looking at the trained model. Specifically, we wish that $w^T p + b \geq 0$ would imply that p's label is 1, and $w^T p + b < 0$ would imply that p's label is -1. This condition can be achieved by solving an optimization problem. One formulation of this is that we wish to solve this optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m u_i \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - u_i (\forall i = 1, \ldots, m) \\ & u_i \geq 0 \qquad\qquad (i = 1, \ldots, m) \\ & ||w||_2 \leq W \end{array} \tag{1}$$

where we have that W is a regularization parameter. The u_i's represent the "slack" we allow for as the training data may not be perfectly separable. Therefore, this optimization problem comes down to minimizing the "slack" allowed from having the data be perfectly separable. Usually, the SVM optimization problem is introduced in a different form which is shown below:

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}||w||_2^2 + C\sum_{i=1}^m u_i \\ \text{subject to} & y_i(w^T x_i + b) \geq 1 - u_i (\forall i = 1, \ldots, m) \\ & u_i \geq 0 \qquad\qquad (\forall i = 1, \ldots, m) \end{array} \tag{2}$$

where C is a regularization parameter. It can be shown that for appropriate values of W and C, these two problems are equivalent.

3.2 Dealing with x_i over a Probability Distribution

Both of the optimization problems above can be used to solve for the separating hyperplane when the training data x_i are exactly known for all i. However, as alluded to in the introduction, this is not always the case in practice. In this subsection, we analyze how to reformulate the Support Vector Machine optimization problem assuming that the data that have access to is in the form $\{(x_i, y_i)\}_{i=1}^m$ where $x_i \sim P_i$ are the feature vectors which each take on values from probability distributions P_i, and $y_i \in \{-1, 1\}$ are their corresponding labels. We will work with equation (1) above. Now, noting that the x_i are taken from a probability distribution and are not exactly known, it makes sense to talk about the achieving some probability that one of the above constraints is satisfied rather than requiring it to be satisfied coming in. Therefore, we obtain this optimization problem:

$$\begin{array}{ll} \text{minimize} & \sum_{i=1}^m u_i \\ \text{subject to} & \Pr\{y_i(w^T x_i + b) \geq 1 - u_i\} \geq k_i (\forall i = 1, \ldots, m) \\ & u_i \geq 0 \qquad\qquad (i = 1, \ldots, m) \\ & ||w||_2 \leq W \end{array} \tag{3}$$

where the $\{k_i\}_{i=1}^n$ are probability thresholds that are chosen prior to performing optimization, i.e. they are not our optimization variables.

3.3 Dealing with x_i over a Normal Distribution

We wish to simplify (3) even further in order to obtain an optimization problem that can be approached with Convex Optimization techniques. In order to do this, we make an assumption about the distribution from which x_i is sampled. For this analysis, we assume that $x_i \sim N(\bar{x_i}, \Sigma_i)$ is sampled from a multivariate Gaussian distribution with mean $\bar{x_i}$ and covariance matrix Σ_i.

Now, we can analyze the second constraint in (2) using this assumption. Since we assumed that $x_i \sim N(\bar{x_i}, \Sigma_i)$, we have that: $z_i = y_i(w^T x_i + b) \sim N(\bar{z_i}, \sigma^2_{z_i})$. where $\bar{z_i} = y_i(w^T \bar{x_i} + b)$ and $\sigma^2_{z_i} = w^T \Sigma_i w$. We know this because of the properties of multivariate Gaussians. Now, we can just normalize and write this constraint in a clearer form relying only on the mean, variance, and the probability threshold k_i chosen in advance:

$$\begin{aligned}
\Pr\{y_i(w^T x_i + b) \geq 1 - u_i\} &= \Pr\{z_i \geq 1 - u_i\} \\
&= \Pr\{\frac{z_i - \bar{z_i}}{\sigma_{z_i}} \geq \frac{1 - u_i - \bar{z_i}}{\sigma_{z_i}}\} \\
&= \phi(\frac{\bar{z_i} + u_i - 1}{\sigma_{z_i}}) \geq k_i \\
&\implies \bar{z_i} \geq \phi^{-1}(k_i)\sigma_{z_i} - u_i + 1 \\
&\implies y_i(w^T \bar{x_i} + b) \geq 1 - u_i + \gamma_i \sigma_{z_i} \\
&\implies y_i(w^T \bar{x_i} + b) \geq 1 - u_i + \gamma_i \sqrt{w^T \Sigma_i w}
\end{aligned}$$

for all $i = 1, \ldots, m$, where

$$\phi(u) := \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{u} \exp(-\frac{s^2}{2}) ds$$

is the CDF of the $N(0, 1)$ probability density function and $\gamma_i := \phi^{-1}(k_i)$ is its inverse. Note that these values can easily be calculated given k_i which are chosen prior to optimization.

Now, we can put this new form of our constraint back into optimization problem (1) to obtain a new optimization problem:

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} u_i \\
\text{subject to} \quad & y_i(w^T \bar{x_i} + b) \geq 1 - u_i + \gamma_i \sqrt{w^T \Sigma_i w} (\forall i = 1, \ldots, m) \\
& u_i \geq 0 \qquad\qquad (i = 1, \ldots, m) \\
& ||w||_2 \leq W
\end{aligned} \tag{4}$$

Note that this problem is similar to (1), with the only differences laid out in using the mean instead of the original known value and adding a term dependent on the covariance and the previously chosen probability threshold k_i.

3.4 Convex Form Assuming Normal Distribution

We can turn (5) in the form of a Convex Optimization problem by noting that all covariance matrices are positive semi-definite and therefore have a unique positive-semidefinite square root. Using that information, we obtain the optimization problem

$$\begin{aligned}
\text{minimize} \quad & \sum_{i=1}^{m} u_i \\
\text{subject to} \quad & y_i(w^T \bar{x_i} + b) \geq 1 - u_i + \gamma_i ||\Sigma_i^{1/2} w||_2 (\forall i = 1, \ldots, m) \\
& u_i \geq 0 \qquad\qquad (i = 1, \ldots, m) \\
& ||w||_2 \leq W
\end{aligned} \tag{5}$$

which can be put in the form of a Second Order Cone Program (SOCP) and can be solved by `cvxpy` or any other convex optimization solver, assuming that $\gamma_i > 0$. Analyzing the values of γ_i that make this problem convex, we get this result. There are three cases for the value of $\gamma_i = \phi^{-1}(k_i)$

1. If $\gamma_i = 0$ or $k_i = 0.5$, then (5) reduces to the original SVM problem (1).
2. If $\gamma_i < 0$ or $k_i < 0.5$, then the second constraint is concave and we have a hard optimization problem which is not convex.
3. If $\gamma_i > 0$ or $k_i > 0.5$, then we have a Convex Optimization problem. Specifically, (5) can be reduced to an SOCP in this case!

4 Building a Binary Classifier which is Robust to Missing Data

In this section, we show how to use the analysis made in the previous section in order to build binary classifier which is hypothesized to be robust to the presence of missing or messy data.

4.1 Convex Problem Assuming both Knowns and Unknowns

In the previous section, we gave the optimization problems in the case where our data is completely known (in equations 1 and 2) as well as in the case where our data is completely unknown but assumed to be sampled from a Normal distribution (in equation 5). In this section, we give the optimization problem assuming that both known and unknown data are present. Let m_a be the number of datapoints for which the values are available, and let m_m be the number of datapoints containing missing values. In this case, it is natural to simply pick the second constraint based on whether the data is known or unknown. If the data is known, use the second constraint in (1), whereas if the data is not known, use the second constraint in (5). Proceeding in that manner, we obtain the following optimization problem.

$$\begin{aligned} \text{minimize} \quad & \sum_{i=1}^{m} u_i \\ \text{subject to} \quad & y_i(w^T x_i + b) \geq 1 - u_i && (i = 1, \dots, m_a) \\ & y_i(w^T \bar{x_i} + b) \geq 1 - u_i + \gamma_i ||\Sigma_i^{1/2} w||_2 && (i = m_a + 1, \dots, m_a + m_m) \\ & u_i \geq 0 && (i = 1, \dots) \\ & ||w||_2 \leq W \end{aligned} \tag{6}$$

Note that this problem is clearly convex. We can estimate $\bar{x_i}$ and Σ_i using the Expectation Minimization (EM) algorithm from our known data assuming that the $x|y_i$ follows a jointly normal distribution with mean μ and covariance Σ. This approach is explained further in the next section.

4.2 An Algorithm for Binary Classification With Missing Data

In the above problem, we are assuming that $x_i \sim N(\bar{x_i}, \Sigma_i)$, and that we know the values of $\bar{x_i}$ and Σ_i. It is not immediately clear how to obtain these values when the input to our Machine Learning algorithm is $\{(x_i, y_i)\}_{i=1}^n$ where some of the x_i are known and some unknown. Here, we give an algorithm for computing $\bar{x_i}$ and Σ from the missing data. This algorithm uses Expectation Minimization to return the mean and the covariances of the unknown points present in our dataset.

Supposing $\{(x_i, y_i)\}_{i=1}^n$ be the data, where for all i, $x_i \in \mathbb{R}^d$ and $y_i \in \{-1, 1\}$,

for $y \in \{-1, 1\}$ **do**

Calculate

$$\mu = \begin{pmatrix} \mu_a \\ \mu_m \end{pmatrix}, \Sigma = \begin{pmatrix} \Sigma_{aa} & \Sigma_{am} \\ \Sigma_{ma} & \Sigma_{mm} \end{pmatrix}$$

where a is the number of components we know and m is the number of components we don't know (i.e. re-order the variables and calculate μ and Σ for every x_i with label y).

while Σ and μ not converged **do**

for every x_m with label y whose values are missing **do**

Update x_m by sampling:

$$x_m | x_a \sim N(\mu_m + \Sigma_{ma}\Sigma_{aa}^{-1}(x_a - \mu_a), \Sigma_{mm} - \Sigma_{ma}\Sigma_{aa}^{-1}\Sigma_{am})$$

end for

Recompute μ and Σ with the updated values using the same formula

end while

end for

return the updated $\{(x_i, y_i)\}_{i=1}^n$ as well as the two μ and Σ values

This algorithm in itself is solving a non-convex problem of maximizing the expectation of our unknown x_i values. As this algorithm is greedily picking the method of steepest descent each time, it will converge to a local minimum.

After performing this algorithm, we will have a new modified dataset. Now, we can use the optimization problem posed in (6) to solve for our parameters. To do this we use the corrected x_i values every time $\bar{x_i}$ appear in the constraints in (6), and the Σ_i for the corresponding class where each value is 0 except for the values involving a missing variable for every time Σ_i appears in the constraints of (6).

4.3 Convex Solver

The algorithm described insection 4.2 was implemented locally using Python 3.6.5 with the appropriate data science libraries, and the optimization problem posed in (6) was tackled using the `pycvx` Convex Optimization framework using the ECOS solver. As the problem posed is a SOCP, any off-the-shelf SOCP solver could be used to solve this problem.

5 Experiments and Results

This section will describe the experiments and results achieved when implementing this approach to binary classification with missing input data.

5.1 Datasets Used

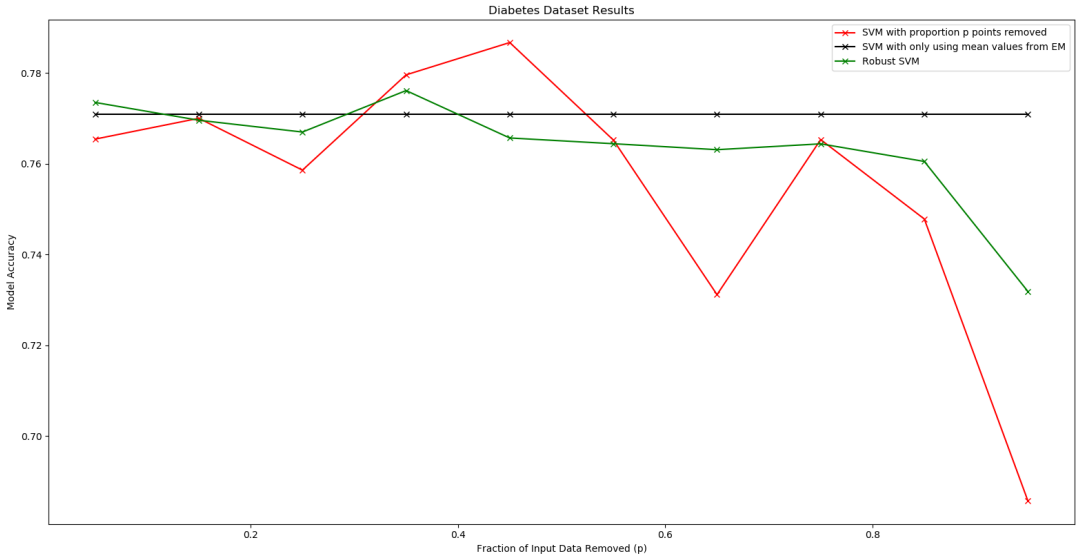
Three datasets were chosen for experiments of this method. The datasets chosen were:

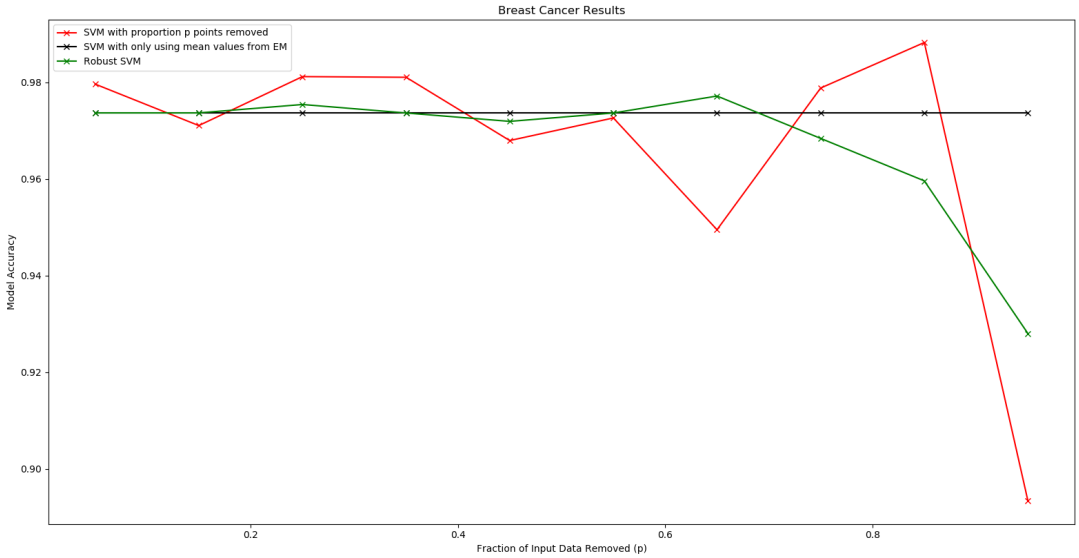
1. The Pima Indian Diabetes Dataset (available at [2])
2. Scikit-learn's Breast Cancer Dataset (available at [3])
3. Scikit-learn's generic classification dataset creator (available at [4])

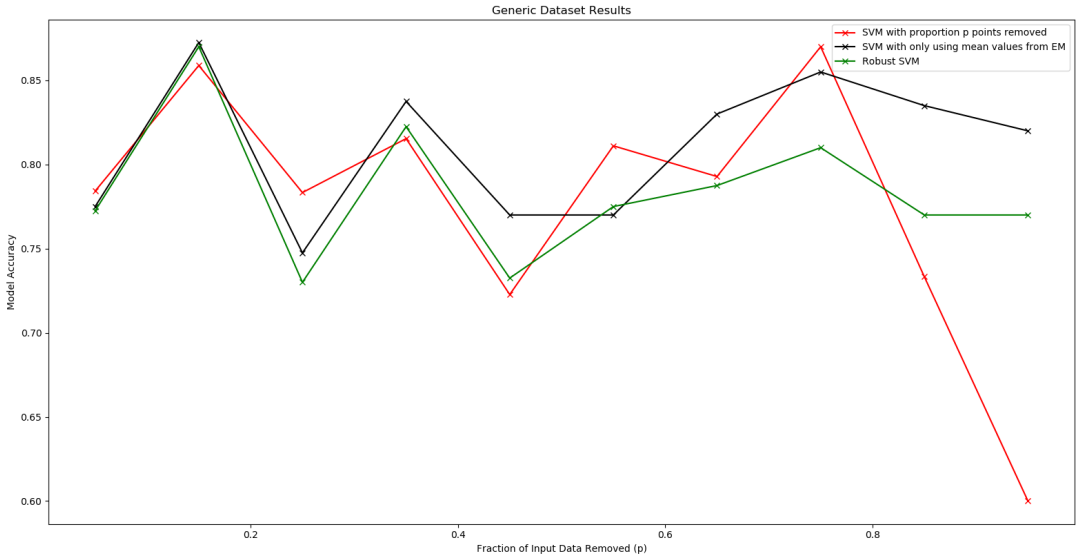
These datasets were chosen because they were publicly available datasets which lend well to performing binary classification on. Additionally, they are datasets which are easily obtainable, so these results could be easily reproduced should the need arise.

5.2 Experiments and Results

The Expectation Maximization algorithm was implemented in Python 3.6.5 along with the typical Data Science packages and frameworks (numpy, matplotlib, etc.). The Convex Optimization problem expressed in (6) was tackled using the cvxpy optimization framework, and experimental results were obtained for the three datasets introduced in the section above. For each dataset, values were randomly erased according to the parameter p, which indicated what ratio of values should be removed from the training data. Various model's performance was recorded as a function of p, and plots of their accuracy are provided below. These accuracy plots were obtained by doing 5-fold cross validation on each of the datasets.







In these graphs, p is plotted along the x axis, which is the ratio of data that is set to be missing in the training dataset. On the y axis, the accuracy of the model is recorded using 5-fold cross validation. The performance of three models were recorded with respect to p for each dataset. The red line tracks the accuracy of regular SVM with just a proportion p of the training sample erased from consideration. The black line tracks the accuracy of regular SVM, but with the Expectation Maximization algorithm given earlier applied too, without taking into account the covariances. The green line tracks the performance of the robust model given in (6).

The experiments suggest that performing Expectation Maximization yields better results than simply omitting data from the input in general. For low p, the robust model tends to perform better than the model that doesn't take into account any of the calculated covariances. For high p, the robust model tends to do worse. We are unsure at this point whether that is due to bugs in the implementation or it is inherent by design.

We can therefore make the conclusion that in the case where most of the data is known, this robust formulation yields to better performance than simple erasure as well as Expectation Maximization without account of covariances.

6 Duality Analysis

In this section, we derive the dual problem and discuss the problem in the context of material covered in class. First, we derive the Lagrangian of (6), which is:

$$\begin{aligned} L(w,b,u,\alpha,\beta,r,s) = &\sum_{i=1}^{m} u_i - \sum_{i=1}^{m} \alpha_i[y_i(w^T x_i + b) - 1 + u_i] - \\ &\sum_{i=m_a+1}^{m} \beta_i \gamma_i ||\Sigma_i^{1/2} w||_2 - \sum_{i=1}^{m} r_i u_i + \\ &s(||w||_2 - W) \end{aligned}$$

where α, β, r, s are the Lagrange multipliers. In this form, it is difficult to find the lagrange dual function of this function. However, by using the identity that for any $x \in \mathbb{R}^n$

$$||x||_2 = \max_{||y||_2 \leq 1} x^T y$$

we can add slack variables $v_1, \ldots, v_{m_a}, \ldots, v_m, v_{m+1}$ with the constraints that $(\forall i)||v_i||_2 \leq 1$ to make the Lagrangian this following form:

$$\begin{aligned} L(w,b,u,\alpha,\beta,r,s,v_1,\ldots,v_{m+1}) = &\sum_{i=1}^{m} u_i - \sum_{i=1}^{m} \alpha_i[y_i(w^T x_i + b) - 1 + u_i] \\ &- \sum_{i=m_a+1}^{m} \beta_i \gamma_i (\Sigma_i^{1/2} w)^T v_i - \sum_{i=1}^{m} r_i u_i \\ &+ s(w^T v_{m+1} - W) \end{aligned}$$

Now, we can find

$$g(\alpha,\beta,r,s,v_1,\ldots,v_{m+1}) = \inf_{w,b,u} L(w,b,u,\alpha,\beta,r,s,v_1,\ldots,v_{m+1})$$

by minimizing over each of the variables. We first start by minimizing b:

$$\frac{\partial L}{\partial b} = 0 \implies \sum_{i=1}^{m} \alpha_i y_i = 0 \tag{7}$$

and then we minimize u:

$$\nabla_u L = 0 \implies 1 - \alpha - r = 0 \tag{8}$$

Now, we minimize w using the results of minimizing over b and u:

$$\nabla_w L = 0 \implies s v_{m+1} = \sum_{i=1}^{m_a} \alpha_i y_i x_i + \sum_{i=m_a+1}^{m} \beta_i \gamma_i (\Sigma_i^{1/2})^T v_i \tag{9}$$

Now, we can plug in (7), (8), (9) into the Lagrangian to obtain:

$$g(\alpha,\beta,s,v_1,\ldots,v_{m+1}) = \sum_{i=1}^{m} u_i - sW \tag{10}$$

Finally, adding in the dual feasibility constraints and constraints achieved during analysis, we obtain the dual problem:

$$\begin{aligned} \text{maximize} \quad & \sum_{i=1}^{m} u_i - sW \\ \text{subject to} \quad & \sum_{i=1}^{m} \alpha_i y_i = 0 \\ & s v_{m+1} = \sum_{i=1}^{m_a} \alpha_i y_i x_i + \sum_{i=m_a+1}^{m} \beta_i \gamma_i (\Sigma_i^{1/2})^T v_i \\ & 1 - \alpha - r = 0 \\ & ||v_i||_2 \leq 1 \\ & \alpha, \beta, s \geq 0 \end{aligned} \tag{11}$$

Although the primal problem looked similar to the original SVM problem except for the added Second Order Cone constraint, the dual problem looks quite different from the dual problem of the original SVM. From this, we see that this problem is more complex than the primal problem, and solving it is harder than simply solving the QP induced by the dual of the original SVM problem.

Now, armed with this dual problem, we can either perform sensitivity analysis or investigate the effects that applying a kernel would do to this classification problem. This discussion is outside the scope of this project and is left to future investigation.

7 Conclusion

Machine Learning models have advanced multiple fields by quite a large margin. However, Machine Learning systems can only be as effective as their data is clean and accurate. In practice, messy data appears extremely often. In this project, we looked at a way to modify the SVM algorithm to take into account the presence of missing data, and formulated a model which can be trained using Convex Optimization, and is shown through experiments to be more robust than simply erasing the data which is missing. In the future, techniques such as this coupled with advancements in other fields will go a long way to ensuring that we are training our Machine Learning models the best we can with the highest quality of data available.

References

[1] Shivaswamy, Pannagadatta K., Chiranjib Bhattacharyya, and Alexander J. Smola. "Second order cone programming approaches for handling missing and uncertain data." Journal of Machine Learning Research 7.Jul (2006): 1283-1314.

[2] https://www.kaggle.com/uciml/pima-indians-diabetes-database

[3] https://bit.ly/2JerQYx

[4] https://bit.ly/2E0aYk0