An Investigation of Support Vector Machine: from the Perspective of Convex Optimization

Mengjie Zhao, Yao Di
Ecole polytechnique fédérale de Lausanne
`{mengjie.zhao, yao.di}@epfl.ch`

Abstract

This document serves in presenting our project of the Convex Optimization and Applications course (2017). Within this project we focus on analyzing the support vector machine (SVM), which is widely used in classification problems of data analytics and other fields. Vanilla SVM is a powerful tool for linear discrimination problem. More importantly, combined with the kernel trick, SVM can be applied in solving non-linear discrimination problem which is more realistic yet harder to deal with. In both cases, convex optimization plays an important role in problem solving. In this work, we firstly provide theoretical analysis of SVM, followed by performance evaluation of applying it on several datasets. In addition, dimension reduction technique such as the principle component analysis (PCA) is introduced to accelerate the speed of problem solving.

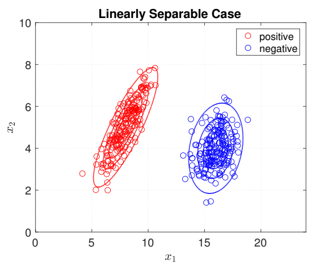
1 Introduction

Support vector machine (SVM) is a binary classification model which is widely used in fields like data analytics and machine learning. Training the SVM is a supervised learning task, which can be formalized as a task of implementing a convex quadratic programming (QP). More advanced frameworks are proposed to further develop SVM such that it now can be applied in multiple-class classification problems. In this work, we focus on applying SVM on binary classification problem.

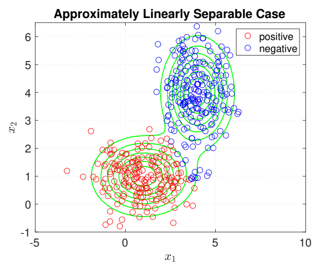
Given a two-class dataset, e.g., demographics of patients and healthy people, a SVM can be trained as a binary classifier to determine if a people was healthy given his/her demographics data. In reality, datasets can have different linearly separabilities. In following contents, we will show how the SVM can be adapted to finish this task using mathematical analysis and experimental results. The key idea is to show that optimization methods play a key role in building and training the SVM.

2 Linear discrimination and SVM

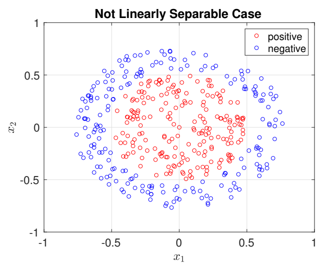
Within this section, we focus on analyzing SVM from a theoretic perspective. We show how the constructing and training of SVM can be formalized and sequentially be solved via using QP.



Linearly Separable



Approxi. Linearly Separable



Not Linearly Separable

Above figures show the three cases that can happen to a two-class dataset[1]. Formally, let

$$\mathcal{T} = \{(x^1, y^1), (x^2, y^2), ..., (x^N, y^N)\}$$

denote the two-class dataset, where $x^i \in \chi = \mathbb{R}^n$ and $y^i \in \mathcal{Y} = \{-1, +1\}$, $i = 1, 2, ..., N$. x^i is the i-th data points in dimension n while y^i is the corresponding label. For visualization, the data points in above figures are generated in two dimensions (x_1, x_2) and are labeled as negative or positive samples.

2.1 Linear separability and margin maximization

Considering the linearly separable case , one can easily find a hyperplane $w^T \cdot x + b = 0$ to separate positive and negative samples without making any error, and the solution is not unique. When the data points are approximately linearly separable, the hyperplane can still be found if the classifier was allowed to make some errors. On the contrary, when the data points are not linearly separable, no hyperplane can be found to classify samples with acceptable accuracy.

When using a hyperplane to distinguish data points in the vector space, it is reasonable to use the distance between a data point and the hyperplane as a measure of confidence of correct classification. Maximizing the confidence of accurate classification means that we want to maximize the margin between two hyperplanes, $w^T \cdot x + b = \pm 1$, that share the same normal vector (w) with the separating hyperplane $w^T \cdot x + b = 0$.

Larger margin between the two hyperplanes results in higher confidence in correct classification, meaning the model is more *robust*. This can be easily understood since we are trying to improve the performance of the worst case scenario.

2.2 Hard margin (for linear separable case)

As described in above part, maximizing the margin between the two hyperplanes $w^T \cdot x + b = \pm 1$ makes the discrimination more robust. This can be formalized as an optimization problem. Due to page limits, we show the distance between the two hyperplanes is $\frac{2}{\|w\|_2}$ in appendix, then the optimization problem is:

$$\begin{aligned} \max_{w,b} \quad & \frac{2}{\|w\|_2} \\ s.t. \quad & w^T \cdot x_j + b \geq 1 \quad j = 1, ..., L^+ \\ & w^T \cdot x_k + b \leq -1 \quad k = 1, ..., L^- \end{aligned} \tag{1}$$

The cost function is equivalent to the convex optimization problem (QP) $\min_{w,b} \frac{1}{2} \|w\|_2^2$ subject to corresponding constraints.

Further, the constraints of the optimization problem could be rewritten in a uniform format by introducing the sign variables.

$$\begin{array}{lll} & y_i(w^T \cdot x_i + b) \geq 1 & i = 1, ..., L \\ where & y_j = 1 & w^T \cdot x_j + b \geq 1, j = 1, ..., L^+ \\ & y_k = -1 & w^T \cdot x_k + b \leq -1, k = 1, ..., L^- \end{array} \tag{2}$$

2.3 Soft margin (for approximately linear separable case)

Support vector machine could also be applied for the case that the data samples cannot be linearly separated. Since the classification error cannot be avoided in this case, we should find a balance between maximizing of distance between separating hyperplanes and minimizing the misclassification error. Therefore, we need to introduce the penalty function to combine the two conditions into a

[1]Artificially generated datasets. Please refer to appendix for detailed data generating process.

single cost function.

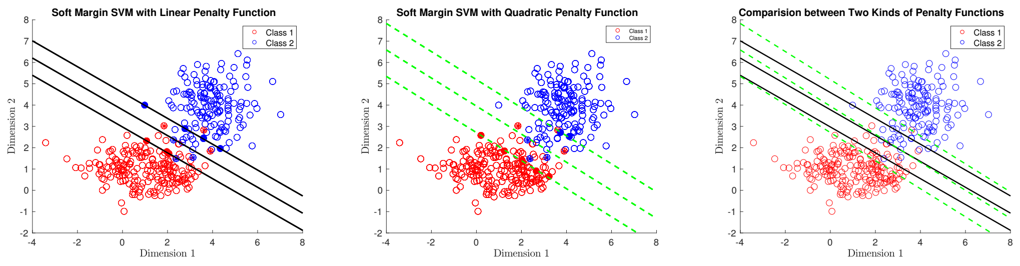
$$
\begin{aligned}
\min_{w,b} \quad & \tfrac{1}{2}\|w\|_2^2 + f_{penalty}(\xi) \\
s.t. \quad & y_i(w^T \cdot x_i + b) \geq 1 \qquad i = 1, ..., L \\
& \qquad \xi_i \geq 0 \qquad\qquad\quad i = 1, ..., L
\end{aligned}
\tag{3}
$$

The penalty could be linearly or quadratically related to the error and the convexity of the optimization problem could be proved. The related MATLAB code for solving the primal problem is attached in the Appendix.

To solve this optimization problem we could apply the KKT conditions on it, the Lagrange dual problem of it could be formulated as.

$$
\begin{aligned}
L(w, b, \xi, \lambda) &= \frac{1}{2}\|w\|_2^2 + c\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\lambda_i[y_i(w^T \cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^{L}\lambda_{L+i}\xi_i \\
&= \sum_{i=1}^{L}\lambda_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L} y_i y_j \lambda_i \lambda_j x_i^T x_j, \; (f_{penalty}(\xi) = c\sum_{i=1}^{L}\xi_i) \\
&= \sum_{i=1}^{L}\lambda_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L} y_i y_j \lambda_i \lambda_j x_i^T x_j - \frac{1}{4c}\sum_{i=1}^{L}\lambda_i^2, \; (f_{penalty}(\xi) = c\sum_{i=1}^{L}\xi_i^2)
\end{aligned}
\tag{4}
$$

The related analysis procedures and MATLAB code for solving the dual problem could also be found in the Appendix.



Linearly Penalty　　Quadratic Penalty　　Comparison

Figure 1: The red circles are data points in class 1 and the blue circles are in class 2. The circles with * inside are the misclassified points. The filled circles are the support vectors.

It could be observed from Figure 1 that there are differences between the classification results with different penalty functions. Compared with the classification line with linear penalty function, the one with quadratic penalty function is rotated clockwise. It could be the result of the distribution of the outliers. The full-size figures could be found in appendix.

According to Figure 1, we could also find that the support vectors just take a small weight of the whole dataset, which is less that 2%. This result meets our expectation since only a few vectors (data points in the high dimension) are functioning in supporting the hyperplanes, resulting in non-zero Lagrangian multipliers. The resting vectors, like vectors correctly classified with high confidence, could have multipliers close to 0, meaning constrains over these vectors is not significant.

3 Kernel method

In previous section, theoretical analysis and experimental results of applying SVM in (approximately) linearly separable problems are demonstrated. In this section, we focus on showing how the kernel trick is employed to make SVM capable to solve non-linear classification problems. Following figure on left shows the distribution of the Banana dataset which is a noisy and artificially created dataset looks like bananas. One can observe that it is not possible to (even approximately) separate the blue and red instances using a line without making acceptable error. We implemented a non-linear SVM with a radial basis function (RBF) kernel and the decision boundaries are shown in figure on the right. The result is much more reasonable than separating with a straight line directly.

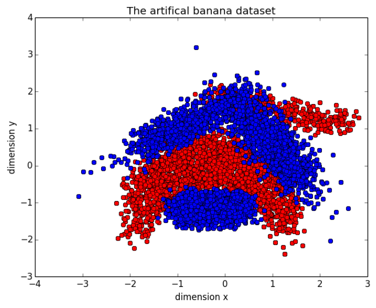


Figure 2: Distribution of banana dataset

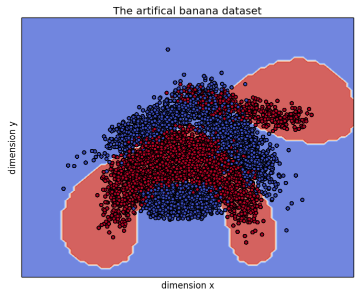


Figure 3: Classification results of SVM

When using kernel method, the idea is to correspond an input space $\mathbb{R}^n$ to a feature space $\mathcal{H}$, such that the classification hyper-surface in $\mathbb{R}^n$ is represented as hyper-plane in $\mathcal{H}$. As a result, the classification task can be resolved by learning a linear SVM in feature space $\mathcal{H}$. In the context of SVM, we can observe that the inner product of $\langle x_i, x_j \rangle$ are involved. Hence, we can use $K(x_i, x_j) = \langle \phi(x_i), \phi(x_j) \rangle$, i.e. the kernel function, to replace $\langle x_i, x_j \rangle$ in (4). In such a case, we can learn a linear SVM in the new feature space using the dataset. When the mapping $\phi(\cdot)$ is non-linear, the kernel SVM we learned is essentially a non-linear classification model. The trick of using kernel method in SVM is that if the kernel function $K(z, x)$ is given, there is no need to explicitly define the feature space $\mathcal{H}$ and mapping $\phi(\cdot)$. The learning procedure in $\mathcal{H}$ is finished implicitly and automatically.

We demonstrate some experimental results of using kernel SVM on some datasets having high dimensionality such as the handwritten digits dataset MNIST. [2] We evaluate the SVM with three different kernels functions, namely linear, polynomial (degree=3) and RBF. The experiments are run over three datasets, namely Abalone Age (AA), Banana and MNIST, and the detailed statistics of these datasets are included in Appendix due to page limit. Following table summaries our experimental results:

Table 1: Accuracy for SVM with different kernels over different datasets

Kernel / Dataset	Linear	Polynomial	RBF
AA	95.33%	95.30%	95.33%
Banana	55.22%	63.65%	90.64%
MNIST	72.80%	43.80%	52.80%

As we can observe, SVM with different kernel functions gives roughly same accuracy on the AA dataset, meaning that the AA dataset has high probability to be linearly separable from some dimensions. For the Banana dataset, SVM with RBF kernel gives best performance, this can be easily observed from previous two figures. For the MNIST dataset, SVM with linear kernel gives the highest accuracy. One interesting observation is that SVM with polynomial kernel gives 43.80% accuracy. This is clearly not acceptable in binary classification problem since the model could keep predicting 1 or 0 but still achieve 50% accuracy.

Now we provide some simple analysis about the support vectors. We define support vector ratio $SVR = \frac{|\mathcal{SV}|}{|\mathcal{T}|}$ where $\mathcal{SV}$ represents the set of support vectors while $\mathcal{T}$ represents the set of all training vectors. Intuitively, higher accuracy results in lower SVR. In following we show the SVR

[2]In all cases, we use 80% of data to train the model and 20% as testing dataset.

performance, and the results meet our expectations. Also, the result is consistent with the analysis in linear separation case.

Table 2: SVR for SVM with different kernels over different datasets

Dataset \ Kernel	Linear	Polynomial	RBF
AA	0.1973	0.1973	0.1975
Banana	0.9100	0.8032	0.2270
MNIST	0.3002	0.4250	0.4275

4 Kernel method + PCA

The data instances in real world can have really high dimension, e.g. 784 for MNIST. High dimensional data can bring computational intractability, especially for big datasets. The principle component analysis (PCA) is applied to find the dimensions that can represent most information (variation) within the data. To implement PCA, we firstly obtain the singular value decomposition of the covariance (or correlation) matrix of the data. After that, we can pick the top several singular values and vectors the do the data reconstruction. A straightforward trade-off here is that PCA can speed up training of SVM since the dimensionality of data is reduced significantly. However, the accuracy will be worse, as information which could be crucial for learning the hyperplanes are missing.

Here we report the accuracy of applying SVM with linear kernel to run on the MNIST dataset since it is the most *intractable* data. In addition, we also add a new Sonar classification dataset (60 dimensions), since AA and Banana are already in low dimension (8 and 2 respectively). We firstly reduce the data to left dimension then begin training. Following table summaries the experimental results:

Table 3: Accuracy after dimension reduction (linear kernel)

Dataset \ Left Dim	15	30	40
Sonar	31.01%	28.57%	35.71%
MNIST	33.20%	36.02%	38.72%

After dimension reduction, the training process is speed up significantly. However, as can be observed that the PCA crashes the model performance and most of them are not acceptable. Hence there is a trade-off between speed and performance. This is a very common phenomenon. For example, the popular deep neural networks achieve remarkable performance yet consume significantly large computational resources and time.

5 Conclusion

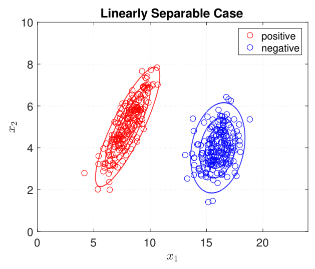
In this work, we analyzed the SVM from the perspective of convex optimization. Theoretical and experimental results are provided in both linearly separable and non-linear separable cases. Dimension reduction technique such as PCA is introduced to leverage the computational intractability. Overall, we believe SVM based on convex optimization is a simple but powerful tool for solving classification problems.

Appendix

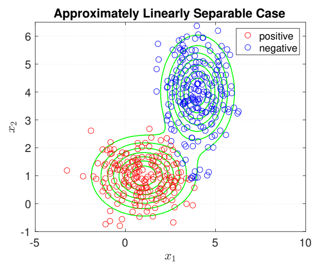
Data generation process (DGP)

Data points in following figures are generated from:

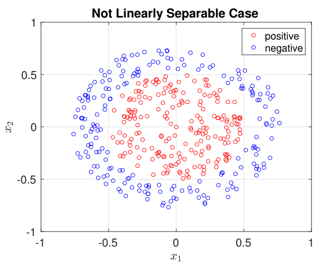
- Two independent two dimensional Gaussian distribution $\mathcal{N}([8,5]^T, \begin{bmatrix}1.3 & 1.1\\1.1 & 1.3\end{bmatrix})$ and $\mathcal{N}([16,4]^T, \begin{bmatrix}1.0 & 0.2\\0.2 & 0.8\end{bmatrix})$.
- Gaussian mixture model of $\mathcal{N}([1,1]^T, \begin{bmatrix}2.0 & 0.0\\0.0 & 0.5\end{bmatrix})$ and $\mathcal{N}([4,4]^T, \begin{bmatrix}1.0 & 0.0\\0.0 & 1.0\end{bmatrix})$.
  It should be noticed that in this case the data points are generated from one distribution, but we artificially label them according to their Euclidean distance between the means of the two Gaussian distributions.
- Random data points generated from two circles with different radius in space.



Linearly Separable



Approxi. Linearly Separable



Not Linearly Separable

Hyperplanes distance calculation

The Euclidean distance between $w^T \cdot x + b = \pm 1$ is $\frac{2}{\|w\|_2}$.

Considering the line $x = wt$, $t \in \mathbb{R}$ that passes through the origin and orthogonal to both two hyperplanes. The line intersect with $w^T \cdot x + b = 1$, we have $(w^T w)t_1 + b = 1$. Similarly, for the other hyperplane we have $(w^T w)t_2 + b = -1$. Hence:

$$\begin{cases} t_1 = \frac{1-b}{w^T w} \\ t_2 = \frac{-1-b}{w^T w} \end{cases} \Rightarrow \quad \|\frac{1-b}{w^T w}w - \frac{-1-b}{w^T w}w\|_2 = \|\frac{2w}{w^T w}\|_2 = \frac{2\|w\|_2}{\|w\|_2^2} = \frac{2}{\|w\|_2}$$

MATLAB code for sovling the primal problem

Listing 1: Soft margin SVM: primal problem

```
function [w, b] = soft_svm_prim(point, class, c, PanaltyType)
% IN
% OUT
[L,d] = size(point);
if PanaltyType == 'Linear'
    cvx_begin
    variables w(d) b xi(L);
        minimize(norm(w,2)/2 + c*sum(xi));
        subject to
            class.* (point*w + b) ≥ 1 - xi;
            xi ≥ 0;
    cvx_end
end
if PanaltyType == 'Quadratic'
    cvx_begin
    variables w(d) b xi(L);
        minimize(norm(w,2)/2 + c*sum(xi));
        subject to
            class.* (point*w + b) ≥ 1 - xi;
            xi ≥ 0;
    cvx_end
end
end
```

MATLAB code for sovling the dual problem

Listing 2: Soft margin SVM: dual problem

```
function [w, b, Lambda] = soft_svm_dual(point, class, c, PanaltyType)
% IN
% OUT
[L,d] = size(point);
H = (point*point') .* (class * class');

if PanaltyType == 'Linear'
    cvx_begin
    variables lambda(L);
        maximize(sum(lambda) - lambda' * H * lambda/ 2);
        subject to
            lambda ≥ 0;
            class' * lambda == 0;
    cvx_end
end

if PanaltyType == 'Quadratic'
    cvx_begin
    variables lambda(L);
        maximize(sum(lambda) - lambda' * H * lambda/ 2  - ...
            sum(lambda.^2) / (4 * c));
        subject to
            lambda ≥ 0;
            class' * lambda == 0;
    cvx_end
end
pos_i = lambda > 1e-6;
w = point' * (lambda .* class);
xi = lambda / (2 * c);
b = mean(class(pos_i) .* (1 - xi(pos_i)) - point(pos_i, :) * w);
```

```
Lambda = [lambda;zeros(L,1)];
end
```

Formalize dual problem for soft-margin SVM

As we observed before, the Lagrangian dual of the function is

$$L(w,b,\xi,\lambda) = \frac{1}{2}\left|w\right|_2^2 + f_{penalty}(\xi) - \sum_{i=1}^{L}\lambda_i[y_i(w^T\cdot x_i + b - 1 + \xi_i)] - \sum_{i=1}^{L}\lambda_{L+i}\xi_i \tag{5}$$

Linear penalty function

If the penalty function is in the linear format, which is

$$f_{penalty}(\xi) = c\sum_{i=1}^{L}\xi_i \tag{6}$$

Since the problem is convex, we could apply the KKT condition for this problem. Thus we have the properties of complementary slackness and stationary.

$$\begin{cases} \frac{\partial L(w,b,\xi,\lambda)}{\partial w} = 0 \\ \frac{\partial L(w,b,\xi,\lambda)}{\partial b} = 0 \\ \frac{\partial L(w,b,\xi,\lambda)}{\partial \xi} = 0 \\ \lambda_i[y_i(w^T\cdot x_i + b - 1 + \xi_i)] = 0 \end{cases} \Rightarrow \begin{cases} \sum_{i=1}^{L}\lambda_i y_i x_i = w \\ \sum_{i=1}^{L}\lambda_i y_i = 0 \\ c - \lambda_i - \lambda_{L+i} = 0 \end{cases} \tag{7}$$

According to Equations 5, 6 and 7, we have

$$\begin{aligned} L(w,b,\xi,\lambda) &= \frac{1}{2}\left|w\right|_2^2 + c\sum_{i=1}^{L}\xi_i - \sum_{i=1}^{L}\lambda_i[y_i(w^T\cdot x_i + b) - 1 + \xi_i] - \sum_{i=1}^{L}\lambda_{L+i}\xi_i \\ &= \frac{1}{2}\left|w\right|_2^2 - \sum_{i=1}^{L}\lambda_i y_i(w^T\cdot x_i + b) + \sum_{i=1}^{L}\lambda_i + \sum_{i=1}^{L}(c - \lambda_i - \lambda_{L+i})\xi_i \\ &= \frac{1}{2}\left|w\right|_2^2 - 0 + \sum_{i=1}^{L}\lambda_i + 0 \\ &= \sum_{i=1}^{L}\lambda_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L}y_i y_j\lambda_i\lambda_j x_i^T x_j \end{aligned} \tag{8}$$

Quadratic penalty function

If the penalty function is in the quadratic format, which is

$$f_{penalty}(\xi) = c\sum_{i=1}^{L}\xi_i^2 \tag{9}$$

Since the problem is convex, we could apply the KKT condition for this problem. Thus we have the properties of complementary slackness and stationary.

$$\begin{cases} \frac{\partial L(w,b,\xi,\lambda)}{\partial w} = 0 \\ \frac{\partial L(w,b,\xi,\lambda)}{\partial b} = 0 \\ \frac{\partial L(w,b,\xi,\lambda)}{\partial \xi} = 0 \\ \lambda_i[y_i(w^T\cdot x_i + b - 1 + \xi_i)] = 0 \end{cases} \Rightarrow \begin{cases} \sum_{i=1}^{L}\lambda_i y_i x_i = w \\ \sum_{i=1}^{L}\lambda_i y_i = 0 \\ 2c\xi_i - \lambda_i - \lambda_{L+i} = 0 \end{cases} \tag{10}$$

According to Equations 5, 9 and 10, we have

$$\begin{aligned}
L(w,b,\xi,\lambda) &= \frac{1}{2}\,|w|_2^2 + c\sum_{i=1}^{L}\xi_i^2 - \sum_{i=1}^{L}\lambda_i[y_i(w^T\cdot x_i+b)-1+\xi_i] - \sum_{i=1}^{L}\lambda_{L+i}\xi_i \\
&= \frac{1}{2}\,|w|_2^2 - \sum_{i=1}^{L}\lambda_i y_i(w^T\cdot x_i+b) + \sum_{i=1}^{L}\lambda_i + \sum_{i=1}^{L}(c\xi_i-\lambda_i-\lambda_{L+i})\xi_i \\
&= \frac{1}{2}\,|w|_2^2 - 0 + \sum_{i=1}^{L}\lambda_i - \frac{1}{4c}\sum_{i=1}^{L}(2c\xi_i)^2 \qquad (11)\\
&= \sum_{i=1}^{L}\lambda_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L}y_iy_j\lambda_i\lambda_j x_i^T x_j - \frac{1}{4c}\sum_{i=1}^{L}(\lambda_i^2+\lambda_{L+i}^2) \\
&\geq \sum_{i=1}^{L}\lambda_i - \frac{1}{2}\sum_{i=1}^{L}\sum_{j=1}^{L}y_iy_j\lambda_i\lambda_j x_i^T x_j - \frac{1}{4c}\sum_{i=1}^{L}\lambda_i^2
\end{aligned}$$

RESULTS FOR SOFT MARGIN SVM

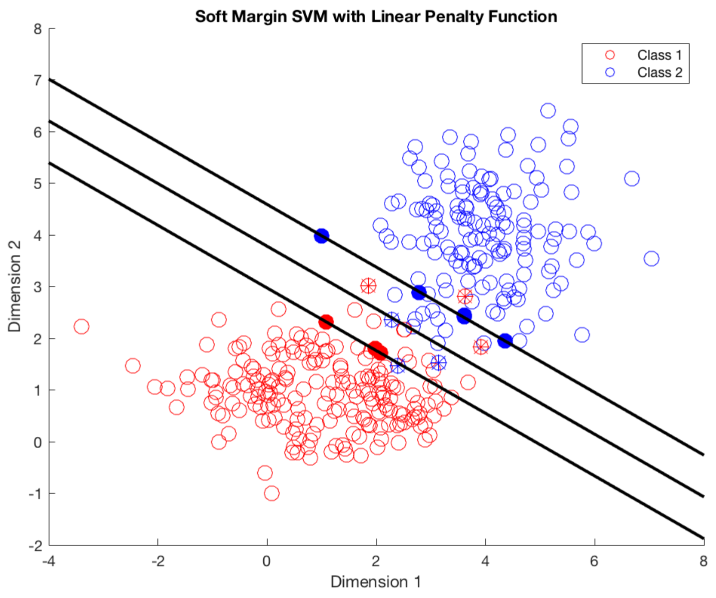


Figure 4: The red circles are data points in class 1 and the blue circles are in class 2. The circles with * inside are the misclassified points. The filled circles are the support vectors.

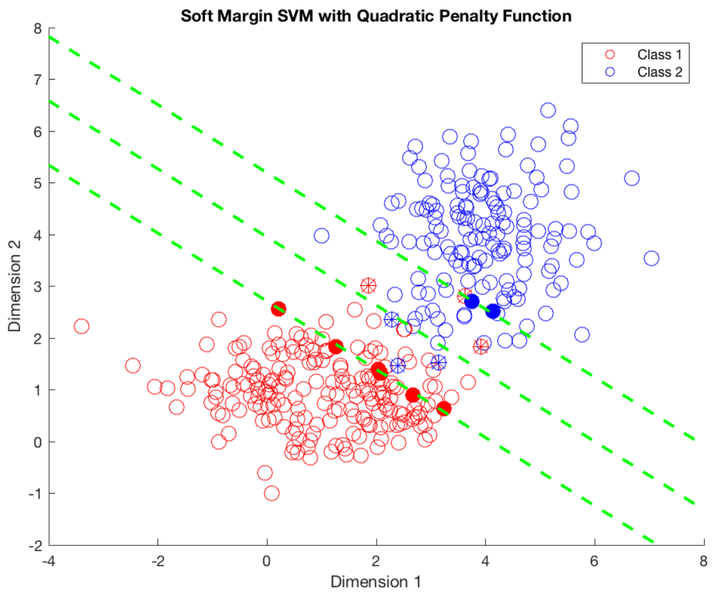


Figure 5: The red circles are data points in class 1 and the blue circles are in class 2. The circles with * inside are the misclassified points. The filled circles are the support vectors.

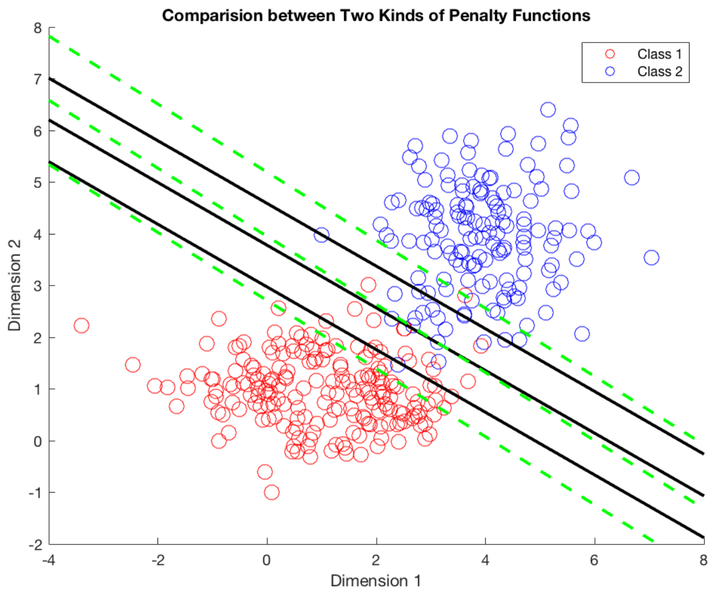


Figure 6: The red circles are data points in class 1 and the blue circles are in class 2. The circles with * inside are the misclassified points. The filled circles are the support vectors.

Dataset description

Within the project, we evaluated the kernel SVM over the following datasets:

- Abalone age data. Predicting the age of abalone from physical measurements. Number of instances is 4177. Number of attributes is 8.

- Banana dataset noisy artificially created dataset. It was created using a mixture of overlapping Gaussian that look like Bananas. Number of instance is 528800. Number of attributes is 2.
- MNIST dataset. Handwritten digits. Number of instances is 6000. Number of attributes is 784.
- Sonar dataset. Sonar signals bounced off a metal cylinder and those bounced off a roughly cylindrical rock. Number of instance is 208. Number of attributes is 60.