Classification of Linearly Separable and Nonseparable Dataset Using Supporting Vector Machines

Yan Gong A99006702, Thomas J. Dawkins A12447586

***Abstract*— Supporting vector machines (SVMs) have been greatly developed in machine learning in these years. They are very popular and powerful tools for classification and regression problems. There are a lot of ongoing researches about modifying this technique to solve for more problems in more general cases (i.e. non-linear classification). In this project, we will study the theories of SVMs, different implementation based on separable and non-separable datasets and the relationship between the methodology of convex optimization.**

I. INTRODUCTION

Supporting Vector Machines (SVMs) are based on the well-established statistical learning model in Vapnik's books.[1][2] It has been receiving more and more attention in recent decades. Many researching fields, like statistical analysis, pattern recognition and classification, take the advantage of the robust models of SVMs. There are also a lot of ongoing researches on the improvement and modification of SVMs to solve a variety of problems. Joachims declared methods to apply SVMs to solve for large-scale problems.[4] Hsu *et al.* made a profound review on different methods of SVMs to classify for multiple classes.[3] Rebentrost *et al.* proposed the quantum SVMs to classify big data.[5] There are a lot more examples illustrating the nonstopping interests among researchers to discover the use of SVMs models. In this project, we will focus on the basic model of SVMs and learn some variations in its constraints. We will apply the different models into our synthesized data as well as the real data to compare the performance.

II. BASIC MODELS AND APPLICATION

In this section, we demonstrate what convex optimization problem we are solving and the approaches using SVMs. We generate some synthesized data that contain both linearly separable and non-separable sets. We apply the different models to those synthesized data in order to better visualize what each model does and how it works.

A. Hard Margin

Consider two groups of data, S_1 and S_2, in R^n. Suppose they are linearly separable, then the standard way to separate the two data sets is to properly choose a hyperplane:

$$\{x : \beta^T \mathbf{x} = -\beta_0\} \tag{1}$$

that divides the space R^n into two halfspaces. For classification, those x who makes the inner product greater than -β_0 are considered in one set, whereas those makes the inner product less than -β_0 are considered in the other. Then the goal is to find the best vector β and scalar β_0. The formal definition of this problem is as follows:

$$\begin{aligned} \min_{\beta,\beta_0} \quad & ||\beta||_2 \\ \text{subject to} \quad & y_i(\beta^T \mathbf{x}_i + \beta_0) \leq 1,\ i = 1, \ldots, m. \end{aligned}$$

where y_i is the label for each data x_i (with value of 1 or -1, implying which set it belongs to), and m is the total number of the data to be classified. Since we assume that the sets are linearly separable, this is called "Hard Margin" for SVMs because the optimal hyperplane show strictly separates the two sets without having data sitting on the wrong side.

B. Soft Margin

Although the Hard Margin is well established for linearly separable cases, it is more common to encounter the datasets that are not linearly separable. For those which are not separable, the constraints in Hard Margin will always fail for certain portions of data and thus the the problem will be infeasible. In order to compensate this issue, a modified version called Soft Margin is introduced. It basically adds a variable in the objective to release some freedom and another tunable parameter in the constraint to control the released margin. It is formulated as:

$$\begin{aligned} \min_{\beta,\beta_0,\zeta} \quad & ||\beta||_2 + C\sum_i \zeta_i \\ \text{subject to} \quad & y_i(\beta^T \mathbf{x}_i + \beta_0) \geq 1 - \zeta_i \quad \zeta_i \geq 0,\ i = 1, \ldots, m. \end{aligned} \tag{2}$$

where C is tunable, and ζ_i are slack variables. Although we can solve for the optimal solution (β, β_0) using the primal as stated in Eq (2), we can also solve for its dual. Note that in (2) the functions in objective and constraints are all convex functions. First of all, they are both differentiable under (β, β_0, ζ). We can then compute for their Hessian matrix:

$$\nabla^2_{objective} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\nabla^2_{inequality} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

Both of them are PSD matrices. Therefore, by second-order condition they are convex functions. Also, in general two data sets should not have all the data at the margin, thus there should exist at least one data in the sets that can meet the strictly inequality for the constraints. Therefore, we can use sufficient and necessary KKT conditions. As long as the solution satisfies KKT conditions, the strong duality holds and there's zero duality gap between the optimum of the primal and the dual. Therefore, it's equivalent to transform the problem and solve it in dual.

1) Duality for Soft Margin: The Langrangian of the problem in (2) is:

$$\begin{aligned}L(\beta,\beta_0,\zeta,\alpha,\lambda) &= \frac{1}{2}||\beta||^2 + C\sum_i\zeta_i\\ &- \sum_i\alpha_i(y_i(\beta^T\mathbf{x}_i+\beta_0)+\zeta_i-1) - \sum_i\lambda_i\zeta_i\\ &= \frac{1}{2}||\beta||^2 + \sum_i(C-\alpha_i-\lambda_i)\zeta_i - \sum_i\alpha_i y_i(\beta^T\mathbf{x}_i+\beta_0)+\sum_i\alpha_i\end{aligned} \tag{3}$$

Apply KKT Condition (at each i for the optimal points):

$$Gradients:$$

$$\nabla_\beta = \beta - \sum_i\alpha_i y_i\mathbf{x}_i = 0 \tag{4}$$

$$\nabla_{\beta_0} = -\sum_i\alpha_i y_i = 0 \tag{5}$$

$$\nabla_\zeta = C-\alpha-\lambda = 0 \tag{6}$$

$$PrimalFeasibility:$$

$$1-\zeta_i - y_i(\beta^T\mathbf{x}_i+\beta_0) \le 0 \tag{7}$$

$$-\zeta_i \le 0, i=1,2,...,m \tag{8}$$

$$NonnegativeDualVariables: \quad \alpha_i\ge 0, \lambda_i \ge 0 \tag{9}$$

$$ComplimentarySlackness: \quad \alpha_i(y_i(\beta^T\mathbf{x}_i+\beta_0)+\zeta_i-1)=0 \tag{10}$$

$$\lambda_i\zeta_i = 0 \tag{11}$$

Eq(4) yields:

$$\beta^* = \sum_i\alpha_i y_i\mathbf{x}_i \tag{12}$$

which should be our optimal solution for β. Now the problem becomes solving for the optimal α to compute for β^*. Plug this expression back to Eq(3) yields:

$$\begin{aligned}g(\alpha,\lambda) &= inf(L(\beta,\zeta,\beta_0,\alpha,\lambda)\\ &= \frac{1}{2}||\sum_i\alpha_i y_i\mathbf{x}_i||^2 - \sum_i\alpha_i y_i\sum_j\alpha_j y_j\mathbf{x}_j^T\mathbf{x}_i + \sum_i\alpha_i\\ &= -\frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j\langle\mathbf{x}_i,\mathbf{x}_j\rangle + \sum_i\alpha_i\end{aligned} \tag{13}$$

And now our goal becomes to solve for the dual problem, i.e:

$$\begin{aligned}\max_{\alpha,\lambda} \quad & g(\alpha,\lambda)\\ \text{s.t.} \quad & \sum_i\alpha_i y_i = 0\\ & 0\le\alpha_i\le C\end{aligned} \tag{14}$$

After we have *CVX Toolbox* to solve for α^* and λ^*, we can plug it into Eq (12) to obtain β^*.
With constraints from (14) for α to be within 0 and C, combine Eq (6) and (11) we have:

$$(C-\alpha_i)\zeta_i = 0 \tag{15}$$

Therefore, for those α smaller than C, ζ_i has to be zero. Hence we can select any i for which $\alpha_i \neq 0$, eliminate α_i and ζ_i in Eq (10) to solve for β_0:

$$\beta_0^* = \frac{1}{y_i} - \beta^{*T}\mathbf{x}_i \tag{16}$$

Last but not least, for those i that makes α_i=C, plug the β^* and β_0^* back in Eq (10) to solve for ζ_i:

$$\zeta_i = 1 - y_i(\beta^{*T}x_i + \beta_0^*) \tag{17}$$

Now, we have all the variables solved: β^*, β_0^* and ζ^* in primal; α^* and λ^* in dual. The supporting vectors in **x** are those who can make the equality holds in the inequality constraints in (2). Therefore, we are able to extract out the supporting vectors from **x** accordingly and all the other data points are non-influential to our SVM model.

We randomly generate some datasets that are linearly separable and non-separable, and apply the methods mentioned above to illustrate how Hard and Soft Margin SVMs work. All data are in R^2. Figure 1 is a quick demo of the same linearly separable sets that are solved using SVM Hard Margin in primal and dual. It proves that the strong duality holds; that is to say, the optimal solutions are the same.

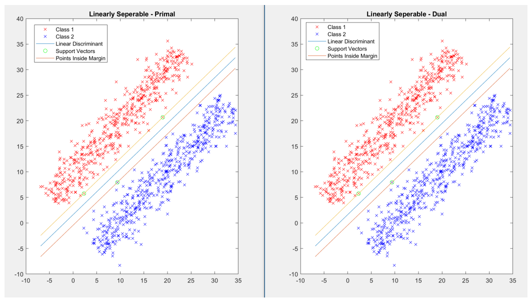


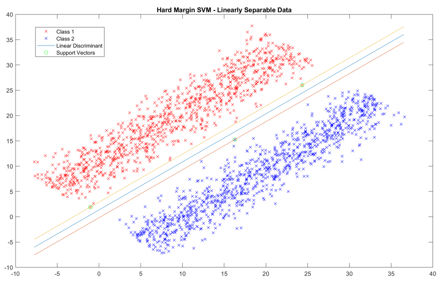
Fig. 1: Linearly Separable Sets using SVM Hard Margin in primal and dual

Figure 2 shows two sets that are clearly separable. The Hard Margin in Fig 2a) is sufficient to solve for the optimal hyperplane as displayed. We also test them using Soft Margin and set different penalty C. As we can see in Fig 2b) and 2c), it also successfully separates the data, but with smaller tolerance for errors on the margin, it regards more data as supporting vectors. Figure 3 shows the datasets that are non-separable. We can around the boundary that some data are interlacing with the other class. In this case, Hard Margin won't work. By Soft Margin with tunable penalty C, it manages to draw the separating hyperplane and denote those supporting vectors at the boundary.

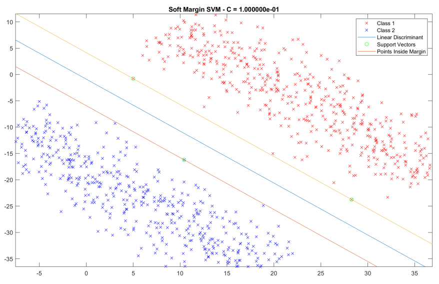
For each method, the training set has 1000 data, and the testing set has 500. Table 1 summarizes the error rates for classifying the testing data under different SVMs models.

	Hard Margin	Soft Margin (C=1)	Soft Margin (C=0.001)
Linearly Sep	0	0	0
Nonlinearly-sep	NA	3.4%	3.4%

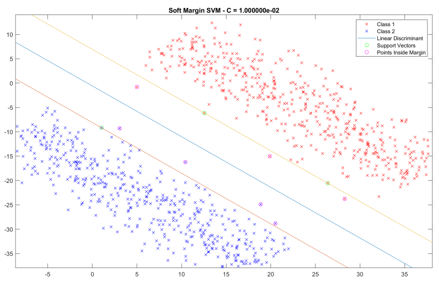
TABLE I: Error Rates on separable and non-separable



(a)

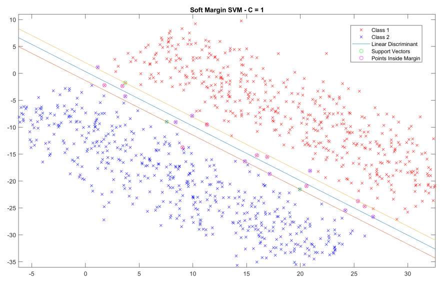


(b)

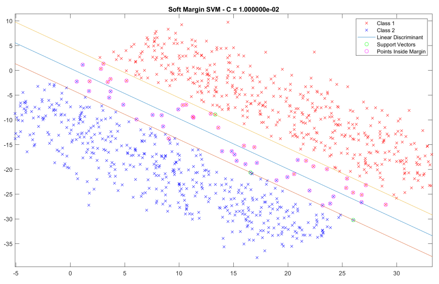


(c)

Fig. 2: Linearly Separable Sets under Hard Margin and Soft Margin. a) Hard Margin SVM, b) Soft Margin SVM with tolerance C=0.1, and c) Soft Margin with C=0.01



(a)



(b)

Fig. 3: Separating the non-linearly separable sets using SVM Soft Margin, with tolerance parameter: a) C=1, b) C=0.01

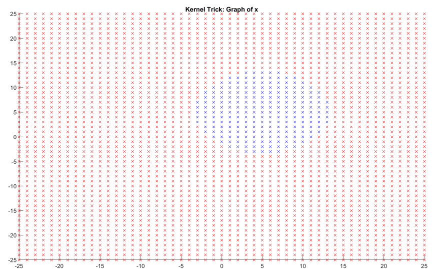


Fig. 4: Synthesized data to show the highly non-separable cases

2) Kernel Trick in Dual: So far We show how the Soft Margin (solved in dual) can compensate the non-separable datasets. However, if the sets are highly non-separable (i.e. two sets are heavily overlapping onto each other), then even the Soft Margin cannot return feasible solutions. Here we generate another example in Figure 4 using Gaussian distribution and set those within a standard deviation to be class 1 and those outside one standard deviation to be class 2. We can conclude intuitively that there can never exist a hyperplane that separates the two sets. Therefore, the Kernel Trick is introduced to transform the original datasets into a higher dimension. The reason behind is that, in another space that has dimension higher than the original, we have more freedom to choose directions for the separating hyperplane and thus it's more likely to make the data separable after transformation. In that case, we need to redefine some terms for solving the dual. Specifically, we need a new definition for the inner product in Eq (13). Since now the dual solutions should also sit in the transform space, a Kernel applied to the data is needed to replace the old one as if that is the new ”inner product” for the transform domain.

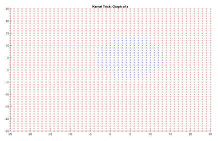
In our experiment, we choose the Kernel to be:

$$K(\mathbf{x}_i, \mathbf{x}_j) = exp(-\frac{||x_i - \mathbf{x}_j||_2^2}{2\sigma^2}) \tag{18}$$

where σ is a tunable variable. While passing all the variables into the dual problem, we only have Eq (13) that is updated:

$$g(\alpha, \lambda) = -\frac{1}{2}\underset{i,j}{\Sigma}\alpha_i\alpha_j y_i y_j * K(\mathbf{x}_i, \mathbf{x}_j) + \underset{i}{\Sigma}\alpha_i \tag{19}$$

Figure 5 shows how the data is non-separable in the original R^2 yet becomes separable after the transformation into R^3. This is a good example of applying the Kernel Trick and being able to draw a hyperplane in the new dimension that separates the datasets.


(a)

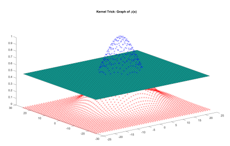
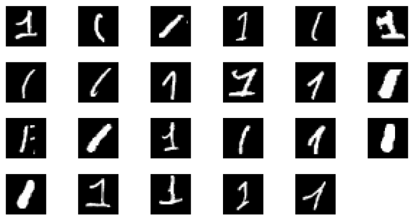

(b)

Fig. 5: a) Non-separable data in 2D, and b) applying the Kernel and find the separating hyperplane in 3D

III. Application to real data

In this section, we apply our implemented algorithms to see how well it classifies some real data. The dataset we choose is called *MNIST*, a well-known database for handwritten digits learning. It consists of 60000 training images and 10000 testing images. Below are the displays of two experiments we run, and the corresponding support vectors for the groups. The supporting vectors should represent the extreme cases for the certain class (i.e. the worst tilted/streched/bold version of that digits).

A. Classifying between 1 and 0


(a)

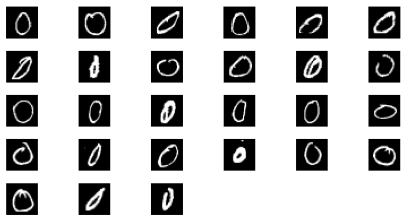
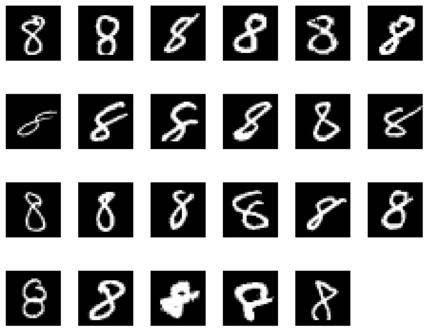

(b)

Fig. 6: Some of the supporting vectors from training images for a) 1, and b) 0. All of them are of size 28x28, despite of the display scaling.

B. Classifying between 0 and 8


(a)

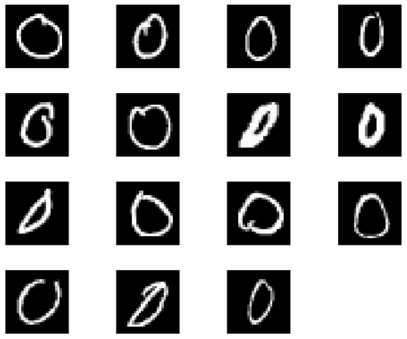

(b)

Fig. 7: Some of the supporting vectors from training images for a) 0, and b) 8.

IV. Conclusions

Support Vector Machine is a convex problem that allows one to obtain a linear discrminant for a binary classifier. This problem can be solved in as the Lagrangian dual problem since strong duality holds. SVM works to maximize the margin between two data sets by orienting a hyperplane. By introducing a slack variable a model can be learned even from data which is not stricly linearly separable in it's home space. When working with non-seperable data a cost is assigned to slack variables which is a parameter of the learning process. The new objective function will be to maximize the marigin while minimizing the slack variables. As shown another feature of SVM is to use the kernel trick in the dual problem. In this way data separable by a non-linear discriminant can be mapped to a space where there is a hyperplane with constant norm to separate the data points. There is no need to know the transformation, but we assume an inner product on the transformed data. In this was we can find decision boundary for data which, in it's native space, is not linearly seperable. As shown this algorithm works well with the MNIST data set. Extension can be made to use SVM for multi-class by learning one vs all boundaries.

References

[1] C. Cortes and V. Vapnik. Support vector networks. Machine Learning, 20:273297, 1995.

[2] C. J. C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, vol. 2, no. 2, pp. 121-167, June 1998.

[3] Hsu, Chih-Wei, and Chih-Jen Lin. "A comparison of methods for multiclass support vector machines." IEEE transactions on Neural Networks 13.2 (2002): 415-425.

[4] Joachims, Thorsten. Making large-scale SVM learning practical. No. 1998, 28. Technical report, SFB 475: Komplexittsreduktion in Multivariaten Datenstrukturen, Universitt Dortmund, 1998.

[5] Rebentrost, Patrick, Masoud Mohseni, and Seth Lloyd. ”Quantum support vector machine for big data classification.” Physical review letters 113.13 (2014): 130503.