Support Vector Machine

Chao Long
c3long@ucsd.edu
PID: A53224755

Guorui Yang
guy011@ucsd.edu
PID: A53218067

1 Objective

The purpose of this article is to discuss the basic ideas of support vector machine(SVM), which is among the best supervised learning algorithm. We first introduce the basic intuition of hard-margin SVMs. After that, we move to soft-margin and non-linear kernel part in SVMs. Besides, we focus on convex optimization problems and discuss the primal and dual problems in SVM. The dataset we use is MNIST dataset[1]. We also tried linear SVM and non-linear SVM with Radial basis function(RBF) kernel for classification. And the best accuracy we can get is 94.7% with CVX package in Matlab and 99.17% in Python.

2 Background

SVM is a supervised machine learning algorithm which is widely used in the past twenty years. It uses a technique called the kernel trick to transform your data and then it finds an optimal boundary between the possible outputs based on these transformations. Simply put, it does some extremely complex data transformations, then figures out how to separate your data based on the labels or outputs you've defined. For the pattern recognition part, SVMs have been used for isolating handwritten digit letters, object recgonition, speaker identification, face detection in images and text categorization[2]. For the regression case, SVMs have been applied for time series prediction problems and PET operator inverse problems[3]. In all of this applications, SVMs have shown significant importance in increasing the accuracy and saved training time. In recent years, deep learning is very popular in machine learning area. Deep neural network(DNN), convolutional neural network(CNN) and recurrent neural network(RNN) always show higher accuracies in both regression and classification problems with the development of GPUs and the availability of large amounts of data. However, SVMs are also really important, since it can be easily combined with other learning algorithms like neural networks and boosting. For instance, on the final layer of the CNN, R-CNN adds a SVM that simply classifies whether this is an object[4]. In this project, we focus on two important kinds of SVMs, hard-margin and soft-margin SVMs, and then discuss kernel and convex optimization prolems in SVMs.

3 Method

3.1 Hard Margin SVM

We start with the simplest case: linear machines trained on separable data, where hard margin is considered. The train set is labeled as $\{\mathbf{x}_i, y_i\}$, i = 1,2,...,n, $y_i \in \{1, -1\}$, $\mathbf{x}_i \in R^d$. Here we use {1, -1} corresponding positive examples and negative examples to simplify subsequent formula. So in this case, we suppose there exists some hyperplanes(a separating hyperplane) which separate the examples. The point $\mathbf{x}$ which lies on the hyperplane satisfying $\mathbf{w} \cdot \mathbf{x} + b = 0$, where $\mathbf{w}$ is the normal to the hyperplane. Let d_+, d_-, defined as the "margin" of a separating hyperplane, be the shortest distance from the separating hyperplane to the closest positive and negative examples. For the linearly

29th Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain.

separable case, the support vector algorithm simply looks for the separating hyperplane with largest margin, as shown in Fig. 1.

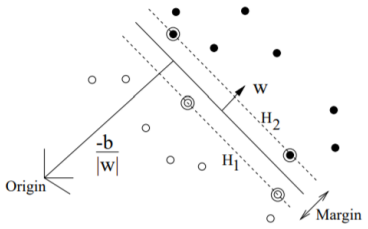

Figure 1: Hard margin for separable data set

According to our assumption, all the training data satisfy the following constraints:

$$\mathbf{w} \cdot \mathbf{x}_i + b \geq +1 \qquad y_i = +1 \tag{1}$$

$$\mathbf{w} \cdot \mathbf{x}_i + b \leq -1 \qquad y_i = -1 \tag{2}$$

Combine above equations, and we can get:

$$y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 \geq 0 \quad \forall i \tag{3}$$

Now consider the points for which the equality in Eq. 1 holds. These points lie on the hyperplane $H_1 : \mathbf{w} \cdot \mathbf{x}_i + b = 1$ with normal **w** and perpendicular distance from the origin $|1 - b| \,/\, \|\mathbf{w}\|$. Similarly, the points satisfy Eq. 2 fall on the hyperplane $H_2 : \mathbf{w} \cdot \mathbf{x}_i + b = -1$. The norm is still **w** but the distance from the origin now is $|-1 - b| \,/\, \|\mathbf{w}\|$. So $d_+ = d_- = 1/\|\mathbf{w}\|$, and the margin is simply $2/\|\mathbf{w}\|$. H_1 and H_2 are parallel to each other and there is no data points between them. Putting these together, we can find the pair of hyperplanes which gives the maximum margin by minimizing $\|\mathbf{w}\|^2$, subject to constraints in Eg. 3.

A typical 2D case is shown in Fig. 1. Note that there are some special points indicated by circles. These points lie either on H_1 or H_2 on the decision hyperplanes.(i.e. the equality in Eg. 3 holds), and their removal would change current solution. They are called support vectors.

Now, We switch to a Lagrangian formulation of this problem. There are two reasons for doing this. First, the constrain in Eq. 3 will be in a form of Lagrangian multipliers. It is easier to handle. Second, after transformation of the problem, the training data will only appear in the form of dot products between vectors. This is a crucial property which will allow us to generalize the procedure to the nonlinear case.

We introduce positive Lagrangian multipliers α_i, i=1,...,l, one for each of the inequality constraints in Eg. 3. Then the Lagrangian is given as:

$$L_P \equiv \frac{1}{2} \|\mathbf{w}\|^2 - \sum_{i=1}^{l} \alpha_i y_i (\mathbf{w} \cdot \mathbf{x}_i + b) + \sum_{i=1}^{l} \alpha_i \tag{4}$$

Now the question becomes to minimize L_P with respect to **w**, b, and simultaneously require that the derivatives of L_P with respect to all the α_i vanish, all subject to the constraints $\alpha_i \geq 0$. Since this is a convex quadratic programming problem, it is equivalent to solve the following dual problem: maximize L_P, subject to the constraints that the gradient of L_P with respect to **w**, b disappear, and also subject to the constraints that $\alpha_i \geq 0$. Requiring that the gradient of L_P with respect to **w** and b vanish give the conditions:

$$\nabla_w L_p = \mathbf{w} - \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i = 0 \tag{5}$$

$$\nabla_b L_p = \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{6}$$

If we combine Eq. 5 and Eq. 6 and plug them back into Eq. 4, we can get dual form below:

$$L_d = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \tag{7}$$

Putting the constraint $\alpha_i \geq 0, i = 1, ..., m$ together, we can get the following dual optimization problems:

$$\begin{aligned} max_\alpha \quad & L_d = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} y_i y_j \alpha_i \alpha_j \mathbf{x}_i \cdot \mathbf{x}_j \\ s.t \quad & \alpha_i \geq 0, \quad i = 1, ..., l \\ & \sum_{i=1}^{l} \alpha_i y_i = 0 \end{aligned}$$

Support vector training therefore can be translated into maximizing L_d with respect to the above constraints. Note that there is a Lagrange multiplier α_i for every training points. And those points with $\alpha_i > 0$ are called support vectors, lying in the hyperplanes H_1 or H_1. All the other training points with $\alpha_i = 0$ are on the side of H_1 or H_2 as the Eq. 3 holds. For this training algorithm, the support vectors are the critical elements of the training set. And those points with $\alpha_i = 0$ can be removed without affecting the result of the separating hyperplane.

3.2 The KKT Conditions

Note that we give the Lagrangian different labels to L_p and L_d arise from the same objective function but with different constraints. In order to solve the above optimization problems, we need to introduce the Karush-Kuhn-Tucker(KKT) Conditions.

For the prime problem above, the KKT conditions may be stated as below[5]:

$$\frac{\partial}{\partial \mathbf{w}} L_p = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i = 0$$

$$\begin{aligned} \frac{\partial}{\partial b} L_p = -\sum_i \alpha_i y_i = 0 \\ y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 \geq 0, \quad i = 1, ..., l \\ \alpha_i \geq 0 \quad \forall i \\ \alpha_i \left(y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) - 1 \right) = 0 \quad \forall i \end{aligned}$$

The KKT conditions are satisfied at the solution of any constrained optimization problems(convex or not). And the SVMs is a convex objective function and for convex problems(if the regularity conditions(a convext objective function, with constraints which give a convex feasible region) hold, meaning $\exists \mathbf{w}, g_i(w) = -y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b \right) + 1 \leq 0$), the KKT conditions are neccessary and sufficient for **w**, b and α to be a solution. Thus solving the SVMs problem is equivalent to finding the a solution to the KKT conditions.

Assuming indeed we solve the dual problems above(finding the optimal α_i). We can use Eq. 5 to go back and find the optimal $\mathbf{w}^*$ as a function of α_i. Considering the support vector lying in the H_1 and H_2 hyperplanes, it is also straightforward to find the optimal value for the intercept term b as:

$$b^* = -\frac{max_{i:y_i=-1} \mathbf{w}^* \cdot \mathbf{x}_i + min_{i:y_i=1} \mathbf{w}^* \cdot \mathbf{x}_i}{2} \tag{8}$$

Then our decision function can be written as:

$$\mathbf{w}^* \cdot \mathbf{x} + b = \left(\sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \right)^T \mathbf{x} + b$$

$$= \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b$$

Hence, if we find the optimal value of α_i, in order to make a prediction, we have to calculate the inner product between **x** and the points in the training set. Moreover, we know that all the α_i will be zero except the support vectors. Thus, we only need to calculate the inner product of **x** and support vectors.

3.3 Soft Margin SVM

When using the above algorithm on non-separable data, there will be no feasible solution. So we need to introduce a further cost for this situation by introducing positive slack variables ξ_i, i = 1,. . . ,l in the constraints[2]. Then the constraints is now :

$$\begin{aligned} \mathbf{w} \cdot \mathbf{x}_i + b &\geq +1 - \xi_i \qquad for \quad y_i = +1 \\ \mathbf{w} \cdot \mathbf{x}_i + b &\leq -1 + \xi_i \qquad for \quad y_i = -1 \\ \xi_i &\geq 0 \quad \forall i \end{aligned} \tag{9}$$

Thus, for an error to occur, the corresponding ξ_i must be larger than unity, so $\sum \xi_i$ is an upper bound on the number of training errors. Hence, it is natural to assign an extra cost for errors by changing the objective function to be minimized from $\|\mathbf{w}\|^2$ to $\|\mathbf{w}\|^2 + C(\sum \xi_i)^2$, where C is a parameter chosen by the user. A large C means to assign a high penalty to errors. And this is a convex problem for any positive integer k. When k = 1, it is desirable to have neither the ξ_i nor the Lagrange multipliers, as below.

$$\begin{aligned} max_\alpha \quad L_d &= \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \\ s.t \quad 0 &\leq \alpha_i \leq C, \quad i = 1, ..., l \\ \sum_{i=1}^{l} &\alpha_i y_i = 0 \end{aligned}$$

The solution is given by

$$\mathbf{w} = \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i \tag{10}$$

where l is the number of support vectors. Thus the only difference from the optimal hyperplane case is that the α_i now have an upper bound of C. This situation is shown in Fig. 2.
We then need the KKT conditions for the primal problem. The primal Lagrangian is

$$L_P = \frac{1}{2} \|\mathbf{w}\|^2 + C \sum \xi_i - \sum \alpha_i \{y_i(\mathbf{w} \cdot \mathbf{x}_i + b) - 1 + \xi_i\} - \sum \mu_i \xi_i \tag{11}$$

where μ_i are the Lagrange multipliers introduced to engorce positivity of the ξ_i. Therefore, the KKT conditions for the primal problem are

$$\frac{\partial L_p}{\partial w} = \mathbf{w} - \sum_{i=1}^{l} \alpha_i y_i \mathbf{x}_i = 0$$

$$\frac{\partial L_p}{\partial b} = -\sum_{i} \alpha y_i = 0$$

$$\frac{\partial L_p}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

$$\xi_i \geq 0$$

$$\alpha_i \geq 0$$

$$\mu_i \geq 0$$

$$y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1 + \xi_i \geq 0$$

$$\alpha_i \left\{y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) - 1 + \xi_i\right\} = 0$$

$$\mu_i \xi_i = 0$$

As before, we can use KKT complementarity conditions to determine the threshold b.

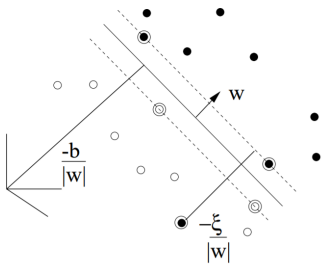

Figure 2: Soft margin - Linear separating hyperplanes for non-separable case

One thing should be pointed out is that the KKT dual-complementarity conditions are:

$$\begin{aligned} \alpha_i = 0 \quad &\Rightarrow \quad y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \geq 1 \\ \alpha_i = C \quad &\Rightarrow \quad y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) \leq 1 \\ 0 < \alpha_i < C \quad &\Rightarrow \quad y_i \left(\mathbf{w} \cdot \mathbf{x}_i + b\right) = 1 \end{aligned}$$

It means that the support vectors are points with $0 < \alpha_i < C$ lying in the hyperplane H1 or H2, and the points with $\alpha_i = C$ are covered with slack variable. For the points with $\alpha_i = 0$, they can be removed without affecting the result of the separating hyperplane.

3.4 Kernel Trick

Now we need to think, what if the decision function is not linear? First, we notice that the only way in which the data appears in the training problem is in the form of dot products, $\mathbf{x}_i \cdot \mathbf{x}_j$. And suppose we first mapped the data to some other Euclidean space H. And the mapping is denoted by Φ as

$$\Phi : \mathbf{R}^d \rightarrow H$$

Then the training algorithm would only depend on the data through dot products in H(i.e. functions of $\Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$). Now if there were a kernel function K such that $K(\mathbf{x}_i, \mathbf{x}_j) = \Phi(\mathbf{x}_i) \cdot \Phi(\mathbf{x}_j)$, we would only need to use K in the training algorithm and we don't even need to know what Φ is exactly. This is the kernel trick. We are still doing a linear separation, but in a different space.
Now an SVM is computing the dot products by the sign of

$$f(x) = \sum_{i=1}^{l} \alpha_i y_i \Phi(\mathbf{s}_i) \cdot \Phi(\mathbf{x}) + b = \sum_{i=1}^{l} \alpha_i y_i K(\mathbf{s}_i, \mathbf{x}) + b \tag{12}$$

where the s_i are the support vectors. So we can avoid computing $\Phi(\mathbf{x})$ explicitly and use $K(\mathbf{s}_i, \mathbf{x})$ to replace it.

Denote the space in which the data lies, L (stands for lower dimension). Note the fact that $\mathbf{w}$ lives in H, there will be no vector in L that maps to $\mathbf{w}$. If there were, $f(x)$ in the Eq. 12 can be computed in one step. It can significantly speed up the test phase of SVMs.

Let's see an example. Suppose $\mathbf{x}, \mathbf{z} \in R^n$, and consider

$$K(\mathbf{x}, \mathbf{z}) = (\mathbf{x} \cdot \mathbf{z})^2 \tag{13}$$

We can also write this as

$$\begin{aligned} K(\mathbf{x}, \mathbf{z}) &= \left(\sum_{i=1}^{n} x_i z_i \right) \left(\sum_{j=1}^{n} x_i z_i \right) \\ &= \sum_{i=1}^{n} \sum_{j=1}^{n} x_i x_j z_i z_j \\ &= \sum_{i,j=1}^{n} (x_i x_j)(z_i z_j) \end{aligned}$$

Thus, the kernel function of $K(\mathbf{x}, \mathbf{z}) = \Phi(\mathbf{x}) \cdot \Phi(\mathbf{z})$, where the feature mapping Φ(Assuming the dimension is n=3) is given by

$$\Phi(\mathbf{x}) = (x_1x_1, x_1x_2, x_1x_3, x_2x_1, x_2x_2, x_2x_3, x_3x_1, x_3x_2, x_3x_3)^T \tag{14}$$

Note that calculating the higher dimension of $\Phi(\mathbf{x})$ requires $O(n^2)$ time, finding $K(\mathbf{x}, \mathbf{z})$ takes only $O(n)$ time, linear in the input attributes, so that it can speed up the training process of SVMs.

A very popular kernel is Gaussian kernel, and corresponds to an infinite dimensional feature mapping Φ, as you can see in the Eq. 15.

$$K(\mathbf{x}, \mathbf{z}) = exp\left(-\frac{\|\mathbf{x} - \mathbf{z}\|^2}{2\sigma^2} \right) \tag{15}$$

4 Dataset

The data set we use is MNIST from `http://yann.lecun.com/exdb/mnist/`. This dataset contains 60000 training examples and 10000 testing samples.Each sample represents a digit ranging from 0 to 9 and is normalized and centered in a gray-scale image with size of 28 by 28 like Figure3.

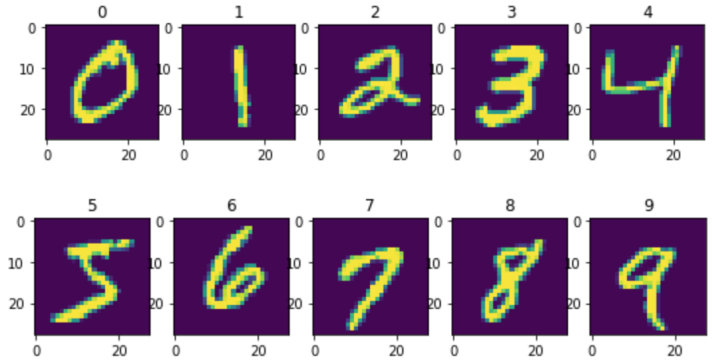

Figure 3: MNIST images from 0 to 9

5 Results

We use MATLAB CVX and Python Sk-learn packages to solve this classification problem. For the MNIST data set, we only consider binary classification. We assign the points with label 0 as class +1 and the others including points with labels 1 to 9 to class -1. Due to the large amount of training time for the RBF kernel in the CVX method, we only choose the first 1000 points for training. But for Python Sk-learn, we choose the first 10000 points. And the result is in Table 1.

Table 1: Result for two methods

	Hard-Margin(%)	Soft-Margin(%)	RBF Kernel(%)
Python-sklearn	98.57	99.17(C=0.005)	90.2(gamma=0.1)
CVX	91.25	94.07(C=10)	83.58(gamma=0.002)

6 Discussion

From the above Table 1, we know both the CVX and Python-Sklearn packages can solve the convex optimization problems in SVMs. We have explored hard-margin, soft margin and RBF kernel in these two methods. In Python-Sklearn, we got the highest accuracy when we use soft-margin SVM. While using CVX method, the convex optimization problem is more straightforward and the best result(94.07%) is also from soft-margin.

We have learnt a lot about SVM during this project. The main difference between hard-margin and soft-margin is how data behaves. Hard-margin SVM assumes that data is very well behaved, and we can find a perfect classifier, which will have zero error on train data. On the contrary, we allow for a little bit of error in soft-margin SVM, and the allowance of softness often gives a better fit. It is also shown in our results. Although the training error will not be 0, but average error over all points is minimized. Both methods assume the classifier is linear, which is sometimes not the case. So we can use RBF kernel to map the data into a higher dimension and get linear separable data.

References

[1] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998.

[2] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[3] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):121–167, 1998.

[4] Ross B. Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. *CoRR*, abs/1311.2524, 2013.

[5] Roger Fletcher. Practical methods of optimization john wiley & sons. *New York*, 80, 1987.