Convex Optimization - Support Vector Machines (SVM)

Brian Wilcox, UC-San Diego, MS, Electrical and Computer Engineering
bpwilcox@ucsd,edu

***Abstract*— Support Vector Machines(SVMs) are a class of supervised learning methods for classifying datasets. SVM algorithms solve for the separating hyperplane which provides the largest margin between the datasets. While a hard-margin SVM classifier requires that datasets be strictly linearly separable, the soft-margin SVM allows for some data points to be misclassified. To handle data which is non-linearly separable, the kernel trick may be used to separate the data in a higher dimensional space. Combining the kernel trick and soft-margin approaches allows for a robust classification of datasets, even when the data is non-separable and non-linear. Due to the convexity of the primal formulation of the SVM and its constraints, the dual function can be derived and shown to efficiently solve for the optimal separating hyperplane.**

I. Objective

The objective of this project is to review the Support Vector Machine algorithm as a convex optimization problem and show how its dual formulation solves for the optimal decision boundary. The use of soft-margin version and the kernel trick will also be shown to form a strong dual form and their strengths will be verified through an algorithm in MATLAB to classify datasets.

II. Background

Classification is a hallmark goal of machine learning. The task of distinguishing the class of a particular piece of data is one that many researchers have spent great effort in solving. Supervised learning, in particular, aims to use the information from expert-labeled data, to generalize a model which will predict the label of a new data point. Methods such as neural networks, decision trees, and k-nearest neighbors are a few of the learning approaches that have been established for classification tasks. Support Vector Machines were introduced in the 1960s as a method of finding the hyperplane which gives the largest margin between the labeled classes on either side of this decision boundary. Even more special was the fact that the problem of finding this hyperplane is convex and dual formulation can be formed to efficiently solve this program. SVMs again re-emerged in the spotlight in the early 1990s when Vapnik introduced the kernel trick for non-linear classification [3]. The kernel trick made a major impact on supervised learning, not only for its ability to work in up to infinite dimensions but for its low cost (and frankly trivial) implementation into the algorithm. Furthermore, its dual formulation results in the same optimal solution via strong duality[4]. The soft margin version of the support vector machine algorithm again introduced a powerful tool for classification by allowing for some datapoints to be misclassified, subject to some penalty. These extended version of the SVM have made it a strong candidate for supervised learning in dataset classification, even in the case of non-linearly separable data [1], which is why SVMs have proven successful in popular datasets such as MNIST. In Section III, the convex optimization problems will be derived alongside their dual formulations. In Section IV, optimization and classification is performed using the CVX Toolbox on data generated in MATLAB to showcase the use of different formulations (hard-margin, soft-margin, kernel trick) of Support Vector Machines.

III. Methods

In this section, the SVM algorithm will be formulated as a primal optimization problem and its dual formulation derived. These derivations will be shown for the hard-margin, soft-margin, and kernel trick versions of the SVM. The goal is to show how by forming the dual optimization problem, we can efficiently solve for the solution to the primal problem even when adding soft constraints and using high-dimensional kernels.

A. Hard-Margin

The basic form of the SVM algorithm is the hard-margin version. *Hard-margin* refers to the strict constraint that the classes in a dataset be linearly separable by a hyperplane. This separating hyperplane is called the *decision boundary* and the decision rule, indicating whether a point x is on either side of the hyperplane defined by normal vector ω and intercept b is:

$$\omega^T x + b \geq 0 \tag{1}$$

For an SVM, we are interested in finding the largest margin between the two datasets, i.e. the supporting hyperplanes which maximizes the distance between the two sets. If we suppose then we have two datasets in $\mathbb{R}^n$, S_1 and S_2, which are linearly separable, we may label every point x_i in the datasets:

$$y_i = 1 \quad for \quad x_i \in S_1 \tag{2}$$

$$y_i = -1 \quad for \quad x_i \in S_2 \tag{3}$$

$$\forall i \in 1...N \tag{4}$$

Thus, we can now express constraints for our margin such that:

$$y_i(\omega^T x_i + b) - 1 \geq 0 \tag{5}$$

This constraint basically means that for any positive sample (point in S_1), the value of the decision rule should be greater than one. Likewise, for negative samples (point in S_2), its

value should be less than -1. Clearly, the margin, or area with no samples allowed, is the area between +1 and -1. We can define the width of this margin as the distance between the supporting hyperplanes of each set

$$y_i(\omega^T x_i + b) - 1 = 0 \tag{6}$$

Hence, for linearly separable data, there will be at least one x_i in S_1 and x_j in S_2 which satisfy this expression for a supporting hyperplane. The width between these supporting hyperplanes then follows:

$$width = (x_i - x_j)^T \frac{\omega}{||\omega||} \tag{7}$$

Note here that the width is a projection of the vector between the two points onto the normal vector of the hyperplane. With some simplification and substitution, we find that

$$width = \frac{2}{||\omega||} \tag{8}$$

Now it is clear that the way to maximize the width of the margin between the two datasets is to minimize $||\omega||$. Without any loss of generality it is equivalent to minimizing $\frac{1}{2}||\omega||^2$, so we can finally form the optimization problem as

$$min_{\omega,b} \frac{1}{2}||\omega||^2 \quad s.t \quad y_i(\omega^T x_i + b) - 1 \geq 0 \tag{9}$$

This forms a quadratic program to solve for ω and b. Notice that the objective function is quadratic, meaning that it is convex, and that its inequality constraints are all affine (and therefore convex). Since the primal optimization problem is convex, and there exists a feasible point in either set in the interior, according to Slater's condition, the duality gap is zero. This means that the optimal solution to the dual problem will be equivalent to the optimal solution of the primal problem. Therefore, to form the dual problem, the Lagrangian of the primal is first formed:

$$L(\omega, b, \lambda) = \frac{1}{2}||\omega||^2 - \sum_{i=1}^{N} \lambda_i (y_i(\omega^T x_i + b) - 1) \tag{10}$$

Then, the gradient of the Lagrangian for ω and b is taken and set to zero to form the following conditions:

$$\nabla_\omega L(\omega, b, \lambda) = \omega - \sum_{i=1}^{N} \lambda_i y_i x_i = 0 \tag{11}$$

$$\Rightarrow \omega = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{12}$$

$$\nabla_b L(\omega, b, \lambda) = \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{13}$$

Replacing (11) and (13) back into (10), now the Lagrangian may be expressed as the dual function:

$$g(\lambda) = \sum_{i=1}^{N} \lambda_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y x_i^T x_j \tag{14}$$

The dual optimization problem is a maximization of the dual function which is concave. Therefore, we can minimize the negative of the dual function to obtain the dual problem:

$$min_\lambda \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y x_i^T x_j - \sum_{i=1}^{N} \lambda_i \tag{15}$$

$$s.t \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{16}$$

$$\lambda_i \geq 0 \tag{17}$$

Notice that this is again a quadratic program, but now it is an optimization in λ and it involves an inner product of the points in the datasets. The fact that the optimization depends on an inner product of data points will be important when we discuss the kernel trick. Now, the dual problem can be solved for and its solution will give the equivalent solution to the primal problem since:

$$\omega^* = \sum_{i=1}^{N} \lambda_i^* y_i x_i \tag{18}$$

We also find a value for b by taking the average of all b_i that satisfy the supporting hyperplane constraints:

$$b^* = \frac{1}{|SV|} \sum_{i \in SV}^{|SV|} (y_i - \sum_{j=1}^{N} (\lambda_i y_j x_j^T x_i)) \tag{19}$$

where here SV is the set of support vectors.

B. Soft-Margin

The main idea behind the *soft margin* is to allow a relaxation on the strict constraint that the data be linearly separable. In other words, some data points are allowed to be misclassified and the optimization problem still find a hyperplane which produces the largest width or gap between the classes. In order to relax this constraint, the slack variables ξ_i are introduced to generate a penalty function in our primal objective function as follows:

$$min_{\omega,b,\xi} \frac{1}{2}||\omega||^2 + C \sum_{i=1}^{N} \xi_i \tag{20}$$

$$s.t \quad -(y_i(\omega^T x_i + b) - 1 + \xi_i) \leq 0 \tag{21}$$

$$\xi_i \geq 0 \tag{22}$$

The penalty function has a factor of C, therefore higher values of C indicate a stronger penalty on misclassified data, while a low C allows for less sensitivity to mistakes. We see that the primal function remains convex (still quadratic) and the constraints remain convex, so strong duality still holds. We can then form the Lagrangian as in the case of the hard-margin version of the SVM as follows:

$$L(\omega, b, \xi, \lambda, \nu) = \frac{1}{2}||\omega||^2 + C \sum_{i=1}^{N} \xi_i - \sum_{i=1}^{N} \lambda_i (y_i(\omega^T x_i + b) - 1 + \xi_i) - \sum_{i=1}^{N} \nu_i \xi_i \tag{23}$$

Here we'll again set the gradients to zero in order to receive the following conditions:

$$\omega = \sum_{i=1}^{N} \lambda_i y_i x_i \tag{24}$$

$$\sum_{i=1}^{N} \lambda_i y_i = 0 \tag{25}$$

$$\lambda_i = C - \nu_i \tag{26}$$

Now, the dual function may be obtained as the infimum (substituting above expressions) of the Lagrangian. Then, the dual problem can be formulated as:

$$min_\lambda \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y x_i^T x_j - \sum_{i=1}^{N} \lambda_i \tag{27}$$

$$s.t \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{28}$$

$$0 \leq \lambda_i \leq C \tag{29}$$

The dual objective function is the same as the hard-margin version, however, λ_i now has an upper bound of the tuning variable C. As mentioned before, C regulates the sensitivity of the SVM to data misclassification. Still, as in the case with the hard-margin, since strong duality holds, it is sufficient to solve the dual problem for the optimal separating hyperplane.

C. Kernel Trick

Though the soft-margin version of the SVM is a great tool to allow a hyperplane to be solved even when the data isn't linearly separable, it still optimizes a for a linear hyperplane. Many datasets are best separated by a non-linear function. The *kernel trick* introduced by Vapnik for support vector machines allows for non-linear classification of data by choosing a non-linear kernel which performs an inner product of a transformed feature vector of the data. A kernel is defined as:

$$k(x, z) = \phi(x)^T \phi(z) \tag{30}$$

where $\phi(x)$ and $\phi(z)$ are functions which map a data point into a higher-dimensional space.There are, of course, many different kernel functions that are common for practical use. Two of the most popular are the radial basis function (Gaussian) kernel and the polynomial kernel:

$$k_{rbf}(x_i, x_j) = e^{-\frac{||x_i - x_j||^2}{2\sigma^2}} \tag{31}$$

$$k_{pol}(x_i, x_j) = (x_i^T x_j + c)^d \tag{32}$$

Different kernels allow for a different kind of nonlinear fit to the data. Note that the hard-margin can be represented by a linear kernel where the kernel is simply the inner product. However,some other properties exist for a kernel function to be considered appropriate or valid, such as that it satisfies Mercer's condition, but a common check is whether the kernel matrix formed by a kernel function of all the data points is positive semidefinite. A particularly convenient consequence of the kernel function being an inner product of high-dimension feature vectors of the input, is that the kernel can simply replace the inner products in the primal and dual optimization problems. This equates to the following form for the dual problem, replacing the inner products in (27) and keeping the soft-margin constraint:

$$min_\lambda \frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \lambda_i \lambda_j y_i y_j k(x_i, x_j) - \sum_{i=1}^{N} \lambda_i \tag{33}$$

$$s.t \quad \sum_{i=1}^{N} \lambda_i y_i = 0 \tag{34}$$

$$0 \leq \lambda_i \leq C \tag{35}$$

Clearly, the problem is the same as the soft-margin version except with the inner products replaced by the kernel function. The problem remain convex in λ with affine constraints, therefore there is zero duality gap and the optimal solution to the primal may be found from the dual problem. To obtain ω and b, substitution follows;

$$\omega^* = \sum_{i=1}^{N} \lambda_i^* y_i \phi(x_i) \tag{36}$$

$$b^* = \frac{1}{|SV|} \sum_{i \in SV}^{|SV|} (y_i - \sum_{j=1}^{N} (\lambda_i y_j \phi(x_j)^T \phi(x_i))) \tag{37}$$

$$\Rightarrow b^* = \frac{1}{|SV|} \sum_{i \in SV}^{|SV|} (y_i - \sum_{j=1}^{N} (\lambda_i y_j k(x_i, x_j))) \tag{38}$$

where again here SV is the set of support vectors.Also, note that we may not have an explicit form for the feature mapping $\phi(x_i)$. Therefore, it may be impossible to obtain an analytical expression for ω. However, the decision function is easily found from the kernel function alone such that:

$$f(x) = \sum_{i=1}^{N} \lambda_i k(x_i, x) + b \tag{39}$$

The $sign(f(x))$ determines the classification label for a point x and the separating hyperplane is found where $f(x)$ is equal to zero. Therefore, it is sufficient to know the kernel function even without having an explicit feature function $\phi(x_i)$.

IV. Results

In this section, results are presented for the hard-margin SVM, soft-margin SVM, and the radial basis function kernel and polynomial kernel. Datasets were generated in MATLAB to be representative of the kind of data appropriate for each SVM version. The CVX toolbox was used to perform the optimization of the dual objective functions given their respective constraints. Figure 1-6 show the decision boundaries (separating hyperplanes in blue) for each of the datasets. In the case of a linear kernel, the margin is shown in a dotted blue line. Data is separated into classes A and B, where class A is represented by red dots and class B

represented by green dots. For the soft-max and kernel methods, data points were intentionally mislabeled to be in the wrong category. Note that the kernel methods also include the soft-margin. Points with a '+' are denoted to represent points that lie on supporting hyperplanes of the sets.

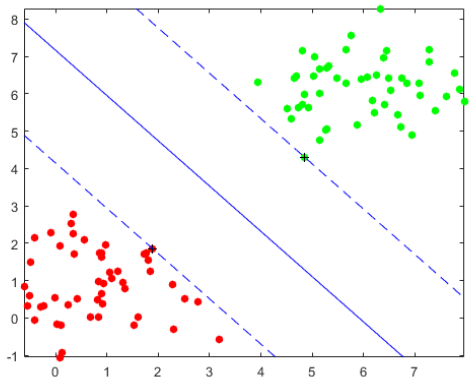

Fig. 1. Decision Boundary for Hard Margin SVM

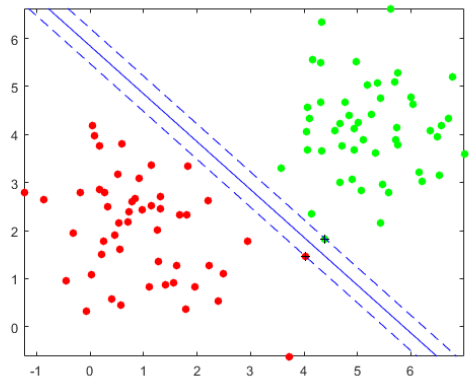

Fig. 2. Decision Boundary for Hard Margin SVM

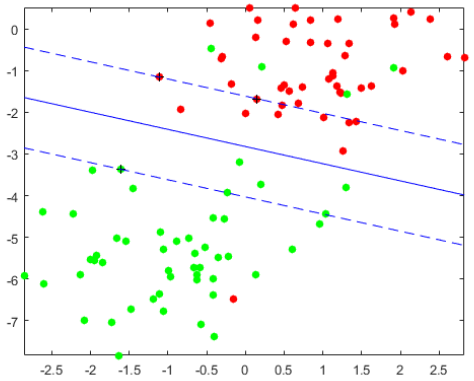

Fig. 3. Decision Boundary for Soft Margin SVM

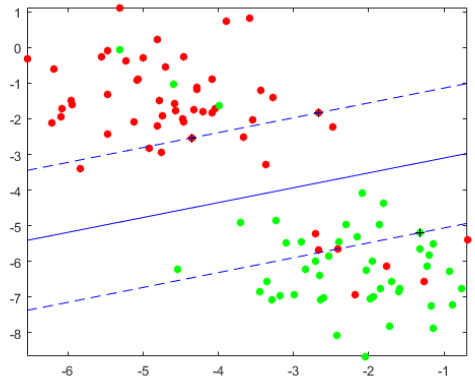

Fig. 4. Decision Boundary for Soft Margin SVM

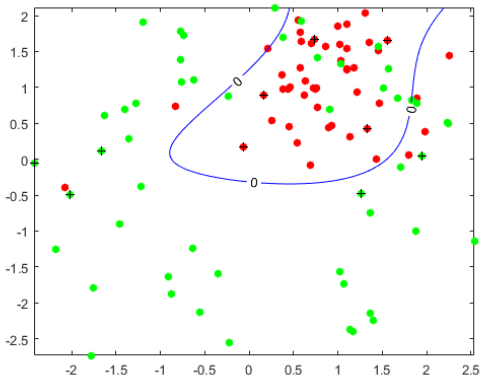

Fig. 5. Decision Boundary for polynomial (d=3) kernel SVM

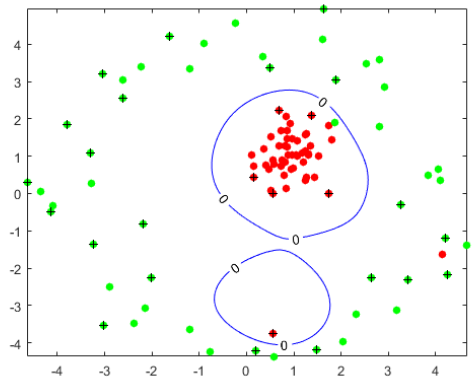

Fig. 6. Decision Boundary for Radial Basis Function SVM

V. Discussion

In Section IV, results were shown for 4 variants on the SVM algorithm under 6 different generated datasets. For the hard-margin constraint on the SVM, it is clear that the SVM needs only to identify the points lying on the supporting hyperplanes in order to derive the optimal separating decision boundary. Thus, all other points not included in the supporting hyperplanes for each set, need not be included i.e. you may remove all interior bounds of the sets. In datasets not shown in this paper, it is evident, as defined in section III.A. that the data sets *must* be linearly separable for the hyperplane to be solved. The soft-margin results showed a robustness to misclassification at a C upper bound of 100. Only the support vector data points are shown with a +, but from the creation of the hyperplane ω, the both the misclassified points as well as the points on the supporting hyperplanes contribute to the decision boundary. Though, the upper bound tuning parameter chose determines the contribution allowed by those mislabels. The kernel trick was very efficient in figures 5 and 6 to find a separating hyperplane ,even with mislabeled data. Since this kernel trick was combined with soft-margin, the mislabeled points contributed to the optimal hyperplane. Overall, the quadratic programs all ran in fast computation time. In order to speed up calculation of the intercepts, decision functions, and normal vector of the separating hyperplanes, as noted in section III, only the nonzero λ_i were selected for their contribution, rather than summing over all points. Though SVMs are indeed a powerful tool in supervised learning for classification, the soft-margin use of a tuning parameter or the kernel parameters are design choices that may need to be confirmed through cross validation to get the most accurate results.

VI. References

References

[1] 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.

[2] I. Steinwart and A. Christmann, “Support Vector Machines”, Springer, 2008

[3] Cortes, C. and Vapnik, V. ”Support-Vector Networks”, Machine Learning, Springer, 1995 20: 273

[4] Boyd, Stephen and Vandenberghe, Lieven, ”Convex Optimizations”, Cambridge University Press, New York, 2004