Lecture 6

Lecturer: Amir Globerson *Scribe: Yishay Mansour*

6.1 Support Vector Machine (SVM) Classifiers

Classification is one of the most important tasks in machine learning. In previous classes we introduced linear classifiers, and discussed the *perceptron* algorithm for training a linear classifier from labeled data. Here we introduce a different algorithm for classification called *Support Vector Machines (SVM)*. The SVM approach has several advantages. First, it enjoys strong theoretical guarantees in terms of generalization. Second, finding the SVM model corresponds to solving a convex optimization problem, which can be done efficiently. Third, the resulting models are often empirically effective (although current deep learning models will often outperform SVMs). Fourth, SVMs can be generalized to non-linear classification, as we discuss next week.

6.2 Review of Linear Classifiers

6.2.1 Binary classification

In classification problem we seek to map inputs $\boldsymbol{x}$ to a set of labels $\{1, \ldots, K\}$. For example, in handwritten digit recognition problem $\boldsymbol{x}$ would be an image of a digit, and $K = 10$. In what follows we focus on binary classification due to its simplicity. Multiclass extensions are possible, but the principles are similar to the binary case.

Assume we are given N training examples $\{(\boldsymbol{x}_i, y_i)\}_{i=1,N}$, where $\boldsymbol{x}_i \in \mathbb{R}^d$ and $y_i \in \{+1, -1\}$ is the correct label. We would like to learn a classifier $f(\boldsymbol{x}_i)$ such that $f(\boldsymbol{x}_i) \geq 0$ iff $y_i = +1$ and $f(\boldsymbol{x}_i) < 0$ iff $y_i = -1$.

In Figure 6.1 we see an example which is clearly linearly separable. In Figure 6.2 we see a set of points which are not linearly separable (In the next lecture, using kernels, we will see how to deal with the non linearly separable case).

A linear classifier has the form $f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x} + b$, where $\boldsymbol{w} \in \mathbb{R}^d$ and $b \in \mathbb{R}$.[1] An example in 2D is in Figure 6.4 and 3D in Figure 6.5. The parameter $\boldsymbol{w}$ is called the *weights* and is normal to the separator, and the parameter b is called the *bias*.

In previous classes, you learned about the Perceptron algorithm, which can be used to find a separating hyperplane. One shortcoming of the Perceprton is that it may return *any*

[1] We denote $\boldsymbol{w} \cdot \boldsymbol{x} = \sum_{i=1}^{d} x_i w_i$.

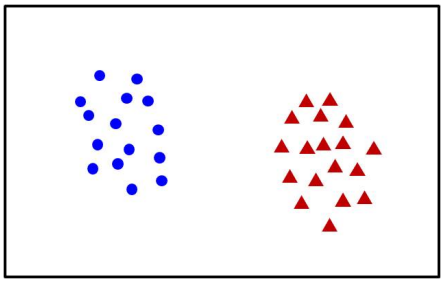

Figure 6.1: Easy linear separable case

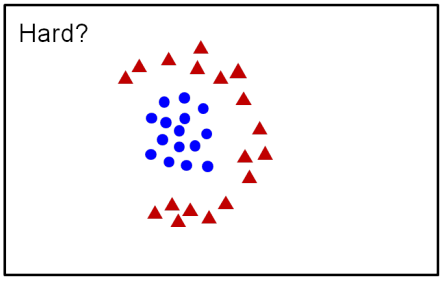

Figure 6.2: Non-linearly separable case.

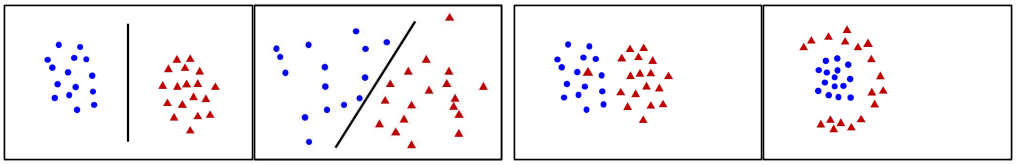

Figure 6.3: More examples of separable and non-separable cases.

hyperplane which is separating. Whereas our intuition is that hyperplanes that separate with a large margin should generalize better. This is precisely the goal of the SVM algorithm.

6.3 Selecting a good linear separator

Figure 6.6 gives various separators for a set of data points. Clearly a separator that has a large error on the data is not desirable. However, there are many consistent linear separators (which have no error on the data). One reasonable criterion is to request a maximum margin. Intuitive reasons are that such a classifier is robust against small perturbations in the data. Also, if we interpret the value of $\boldsymbol{w} \cdot \boldsymbol{x} + b$ as a confidence, it increases the minimal confidence obtained at any example in the data set.

6.3.1 Geometry of Linear Classifiers

Let us first discuss the geometry of the separating hyperplane and its relation to the parameters $\boldsymbol{w}, b$ (see Figure 6.7). The points on the separating hyperplane satisfy $\boldsymbol{w} \cdot \boldsymbol{x} + b = 0$. This implies that if we take two points $\boldsymbol{x}'$ and $\boldsymbol{x}''$ on the hyperplane, we have both $\boldsymbol{w} \cdot \boldsymbol{x}' + b = 0$ and $\boldsymbol{w} \cdot \boldsymbol{x}'' + b = 0$. Therefore $\boldsymbol{w}^T(\boldsymbol{x}' - \boldsymbol{x}'') = 0$, which implies that $\boldsymbol{w}$ is perpendicular to the hyperplane.

The distance between any point $\boldsymbol{x}$ and the separating hyperplane is given by:

$$d(x, \boldsymbol{w}, b) = \frac{|\boldsymbol{w} \cdot \boldsymbol{x} + b|}{\|\boldsymbol{w}\|} \tag{6.1}$$

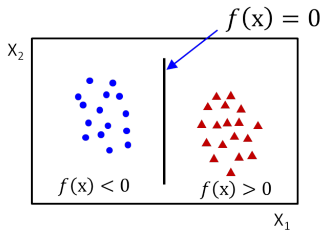

Figure 6.4: A linear separator in 2D

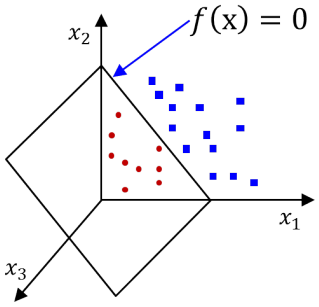

Figure 6.5: A linear separator in 3D

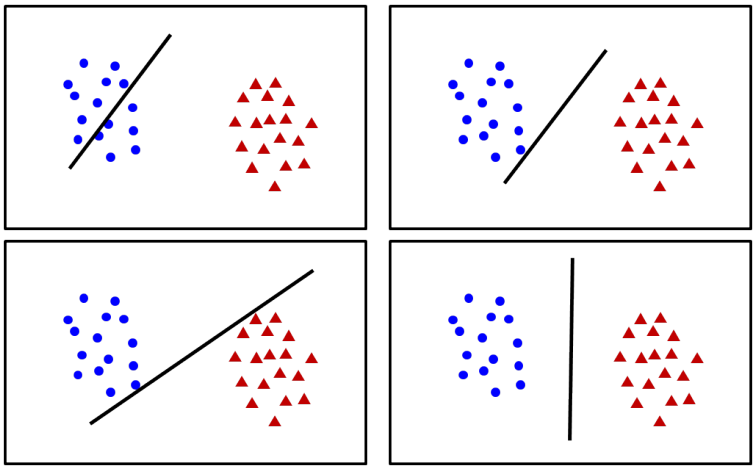

Figure 6.6: Possible linear separators. The bottom right classifier has zero error, and separates with large margin.

See slides for a derivation using Lagrange multipliers (a direct geometric proof is also possible).

Note that the above also implies the distance between the origin and the hyperplane is $\frac{|b|}{\|\boldsymbol{w}\|}$.

In summary, assume that $\|\boldsymbol{w}\| = 1$. Then the separating hyperplane is the hyperplane orthogonal to $\boldsymbol{w}$ which is *shifted* by distance b from the origin.

6.3.2 SVMs in the Separable Case

In this section we assume that the training data is linearly separable. We later turn to the non-separable case.

Our goal in SVM is to find a separator with large margin. By margin we will mean the

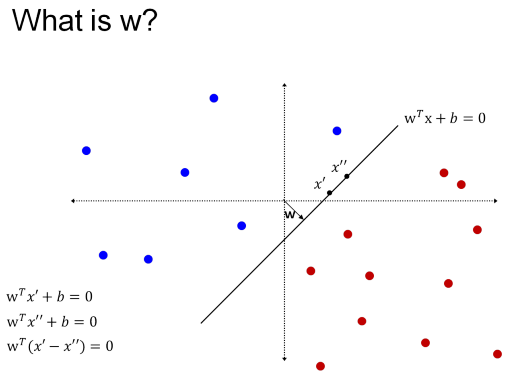

Figure 6.7: What is the w?

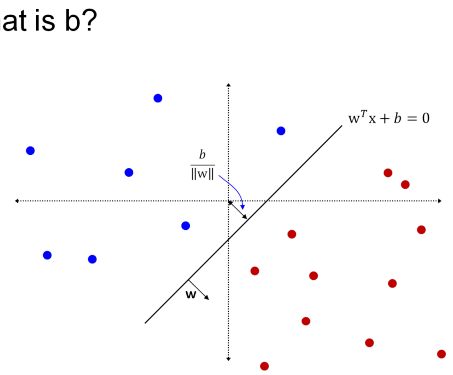

Figure 6.8: What is b?

minimum distance between the separator and any point $\boldsymbol{x}_i$. Thus, the margin $\gamma(\boldsymbol{w}, b)$ is defined by:

$$\gamma(\boldsymbol{w}, b) = \min_i \frac{|\boldsymbol{w} \cdot \boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|} \tag{6.2}$$

where we have used the expression in Equation 6.1 for the distance between $\boldsymbol{x}_i$ and the separator. For example, if there exists even a single point $\boldsymbol{x}_i$ on the separator then the margin will be zero (i.e., the separator is bad).

In SVM, we seek the separator with maximum margin. That is, we want to solve the following problem:

$$\begin{array}{ll} \max\limits_{\boldsymbol{w},b} & \min_i \frac{|\boldsymbol{w} \cdot \boldsymbol{x}_i + b|}{\|\boldsymbol{w}\|} \\ s.t. & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0 \end{array} \tag{6.3}$$

The constraints guarantee that the separator classifies the training data correctly.[2] We next transform this into a problem that is easier to solve. First, note that we can always rescale the parameters $\boldsymbol{w}, b$ to $c\boldsymbol{w}, cb$ for any $c > 0$, and the new parameters will still satisfy the constraints and attain the same objective value.

Since we are free to rescale in this way, we can always rescale any $\boldsymbol{w}, b$ such that:

$$\min_i |\boldsymbol{w} \cdot \boldsymbol{x}_i + b| = 1 \tag{6.4}$$

Thus, without loss of generality we can replace the SVM problem with:

$$\begin{array}{ll} \max\limits_{\boldsymbol{w},b} & \|\boldsymbol{w}\|^{-1} \\ s.t. & y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0 \\ & \min_i |\boldsymbol{w} \cdot \boldsymbol{x}_i + b| = 1 \end{array} \tag{6.5}$$

[2]In principle, the constraints may allow a case where $(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 0$ which gives incorrect classification. However, this will result in zero margin, and therefore the optimization problem will not output such solutions.

Next, note that since $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 0$ then $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = |\boldsymbol{w} \cdot \boldsymbol{x}_i + b|$, and we can replace the above by:

$$\begin{array}{ll} \max\limits_{\boldsymbol{w},b} & \|\boldsymbol{w}\|^{-1} \\ s.t. & \min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1 \end{array} \tag{6.6}$$

Now, note we can require $\min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$, and the solution will still satisfy $\min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1$. This is true, since if $\min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$, we can consider a better solution $c\boldsymbol{w}, cb$ with $0 \leq c < 1$ which would result in $\min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1$, while decreasing $\boldsymbol{w}$ and therefore improving the objective. Thus, we have the following equivalent problem:

$$\begin{array}{ll} \max\limits_{\boldsymbol{w},b} & \|\boldsymbol{w}\|^{-1} \\ s.t. & \min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \end{array} \tag{6.7}$$

The above is equivalent to requiring $\min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1$.

Finally, note that instead of maximizing $\|\boldsymbol{w}\|^{-1}$, we can instead minimize $0.5\|\boldsymbol{w}\|^2$, and obtain the same $\boldsymbol{w}$ (why?). This results in the SVM optimization problem for the separable case:

$$\begin{array}{ll} \min\limits_{\boldsymbol{w},b} & 0.5\|\boldsymbol{w}\|^2 \\ s.t. & \min_i y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) \geq 1 \end{array} \tag{6.8}$$

Support Vectors

Given a separator, we will specifically be interested in the points that are closest to the separating hyperplane. These will be called **Support Vectors**. In our formulation above, the support vectors will have the property that $y_i(\boldsymbol{w} \cdot \boldsymbol{x}_i + b) = 1$.

6.3.3 SVM Optimization

The SVM classifier is the solution to the optimization problem in Equation 6.8. This is a convex optimization problem, since its objective $\|\boldsymbol{w}\|^2$ is a convex function of $\boldsymbol{w}$ and its constraints are linear. Such problems are known as quadratic optimization (QP) problems.

This QP can be solved by standard solvers which take the QP formulation and return the optimal $\boldsymbol{w}, b$. In later lectures we will discuss some specific solution strategies.

Next, we obtain some intuition about the structure of the optimal $\boldsymbol{w}$. We begin by using Lagrange multipliers to solve Equation 6.8. Introduce Lagrange multipliers $\alpha_i \geq 0$ for the i^{th} inequality constraint. The Lagrangian is then:

$$\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0.5\|\boldsymbol{w}\|^2 + \sum_i \alpha_i \left[1 - y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right)\right] \tag{6.9}$$

The optimal $\boldsymbol{w}$ should satisfy $\nabla_{\boldsymbol{w}}\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) = 0$, which yields:

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i \tag{6.10}$$

This is an important result, and is a specific case of the more general *representer theorem*. It says that the optimal $\boldsymbol{w}$ is a linear combination of the $\boldsymbol{x}_i$ (note that the Perceptron classifier also has this property).

Next, we show that α_i are closely related to support vectors (see Section 6.3.2). The KKT conditions imply that if the constraint corresponding to α_i is satisfied with strict inequality, then α_i must be zero. In other words:

$$y_i\left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) > 1 \tag{6.11}$$

implies $\alpha_i = 0$. Thus, any point that is *not* a support vector will have $\alpha_i = 0$. So α_i will be non-zero only for support vectors. In summary, $\boldsymbol{w}$ will be a weighted combination of only support vectors.

The Convex Dual of SVMs

Any convex optimization problem has an equivalent problem called the dual. Equivalence here means that the problems have the same optimal value, and that the solution of one can be used to obtain the solution to the other (the equivalence requires some additional properties, such as the Slater's condition, which are often satisfied in practice). See slides for discussion of duality.

In our case the dual function is given by:

$$\begin{aligned} g(\boldsymbol{\alpha}) &= \min_{\boldsymbol{w},b} \mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha}) \\ &= \min_{\boldsymbol{w},b} 0.5\|\boldsymbol{w}\|^2 + \sum_i \alpha_i \left[1 - y_i\left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right)\right] \end{aligned}$$

Let's see what $g(\boldsymbol{\alpha})$ is. Taking gradient wrt b we have:

$$\sum_i \alpha_i y_i = 0 \tag{6.12}$$

To see why we got this constraint on α_i, note that if $c = \sum_i \alpha_i y_i > 0$ then $b \to -\infty$ will yield $g(\boldsymbol{\alpha}) = -\infty$ (same for $c < 0$). Thus if Equation 6.12 is violated we have $g(\boldsymbol{\alpha}) = -\infty$.

Next, taking gradient wrt $\boldsymbol{w}$ yields Equation 6.15. Substituting this into $\mathcal{L}(\boldsymbol{w}, b, \boldsymbol{\alpha})$ and using the constraint Equation 6.12 we get:

$$g(\boldsymbol{\alpha}) = \begin{cases} \sum_i \alpha_i - 0.5 \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j & \sum_i \alpha_i y_i = 0 \\ -\infty & \sum_i \alpha_i y_i \neq 0 \end{cases} \tag{6.13}$$

This can equivalently be written as the following problem, known as the SVM dual:

$$\begin{array}{ll} \max_{\boldsymbol{\alpha}} & \sum_i \alpha_i - 0.5 \sum_{i,j} \alpha_i \alpha_j y_i y_j \boldsymbol{x}_i \cdot \boldsymbol{x}_j \\ \text{s.}t. & \sum_i \alpha_i y_i = 0 \\ & \alpha_i \geq 0 \end{array} \tag{6.14}$$

The dual is also a convex QP which can be solved using standard solvers. Note that the dual has n variables (like the number of data points), whereas the primal (i.e., the problem with $\boldsymbol{w}$ variables) has $d+1$ variables where d is the dimension of the input $\boldsymbol{x}$.

Recovering $\boldsymbol{w}, b$ from the Dual Solution

The KKT conditions specify the conditions for joint optimality of primal and dual solution. In our case they are:

$$\begin{aligned} \nabla_{\boldsymbol{w}} \mathcal{L}(\boldsymbol{w}, \boldsymbol{\alpha}) &= 0 \\ y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right) &\geq 1 \\ \boldsymbol{\alpha} &\geq 0 \\ \alpha_i \left[1 - y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right)\right] &= 0 \end{aligned}$$

The weight $\boldsymbol{w}$ is easily recovered through the first condition which yields:

$$\boldsymbol{w} = \sum_i \alpha_i y_i \boldsymbol{x}_i \tag{6.15}$$

We can recover b by using the last (complementary slackness) constraint in Equation 6.15. For any support vector where $\alpha_i > 0$ we must have $1 - y_i \left(\boldsymbol{w} \cdot \boldsymbol{x}_i + b\right)$, from which we can solve for b.

A Matrix Formulation of the Dual

We can write the SVM dual as a quadratic program in matrix form. Denote $\mathbf{1}$ a vector of n ones, and $\boldsymbol{y}$ the vector of all y_i labels.

$$\begin{array}{ll} \max\limits_{\boldsymbol{\alpha}} & -\dfrac{1}{2} \boldsymbol{\alpha}^T M \boldsymbol{\alpha} + \mathbf{1}^T \boldsymbol{\alpha} \\ \text{s.t.} & \boldsymbol{y}^T \boldsymbol{\alpha} = 0 \\ & 0 \leq \boldsymbol{\alpha} \end{array}$$

$$M = \begin{pmatrix} y_1 y_1 \boldsymbol{x}_1 \cdot \boldsymbol{x}_1 & y_1 y_2 \boldsymbol{x}_1 \cdot \boldsymbol{x}_2 & \cdots \\ \vdots & \vdots & \vdots \\ y_N y_1 \boldsymbol{x}_N \cdot \boldsymbol{x}_1 & y_N y_2 \boldsymbol{x}_n \cdot \boldsymbol{x}_2 & \cdots \end{pmatrix}$$

We can show that the objective is a concave function of $\boldsymbol{w}$ by showing that the matrix M is positive semi-definite (note that M is a constant matrix that depends only on the inputs). We can rewrite M as $A^T A$ where the i-th column of A is $y_i \boldsymbol{x}_i$. This implies that M is positive semi-definite (note that for any $\boldsymbol{z}$ we have $\boldsymbol{z}^T M \boldsymbol{z} = \boldsymbol{z}^T A^T A \boldsymbol{z} = \|A\boldsymbol{z}\|^2 \geq 0$.)

6.3.4 Generalization Bounds for SVM

Intuitively, the large margin classifier should lead to better generalization than other linear separators. Indeed there is a wealth of theoretical results demonstrating this point. You can read about these in [1, 2].

Margin Based Bounds

Consider the separable case, and assume that you have found a separator with margin γ. Also assume that $\|\boldsymbol{x}\| \leq R$. Then it can be shown (e.g., see [2]) that the generalization error of your classifier will be upper bounded:

$$error(\boldsymbol{w}) \leq \frac{c_1}{m} \frac{R^2}{\gamma^2} + \frac{c_2}{m} \log \frac{m}{\delta} \tag{6.16}$$

for some constants c_1, c_2. The interesting thing to note is that the VC dimension based bounds (for the separable case) are very similar, but have the VC dimension instead of $\frac{R^2}{\gamma^2}$. For linear classifiers the VC dimension will be $d + 1$ where d is the dimension of the input. Thus, we have that the margin determines the *effective* dimension of the classifier, and is independent of the original dimension. This has the important implication that we can even consider classification in infinite dimensional spaces, as long as the margin is large.

Leave One Out (LOO) error bounds

The fact that the classifier $\boldsymbol{w}$ is only specified via the support vector seems to imply that it is *simple* and its generalization error should depend on the number of support vectors. We next make this explicit by using leave one out bounds.

The leave one out error is an approximation to the generalization error of a model. It is often used for model selection (see previous lecture on SRM).

Say you have two hypothesis classes $\mathcal{H}_1$ and $\mathcal{H}_2$ and want to know which one you should learn with. One option is to learn the best model for each, and evaluate it on a holdout set. Now, assume you don't want to *waste* data on a holdout set and want to use most of your labeled data for deciding which model to use.

Denote by S_n the training sample of size n, and by S_n^{-i} the sample with point $(\boldsymbol{x}_i, y_i)$ removed. Denote by $h(S_n^{-i})$ the classifier trained on the sample S_n^{-i}. You can use this classifier to label the point $\boldsymbol{x}_i$. Note that since you did not train on this point, this is a small

generalization experiment. The LOO error is the average error you get by repeating this for all n points. Namely:

$$\hat{R}_{LOO}(S_n) = \frac{1}{n}\sum_{i=1}^{n} I\left[h_{S_n^{-i}}(\boldsymbol{x}_i) \neq y_i\right] \tag{6.17}$$

You can now compare the LOO error you get for training with classes $\mathcal{H}_1$ and $\mathcal{H}_2$, and choose the class that gets the smaller error.

The LOO error is closely related to the true generalization error. Denote by $R(h)$ the generalization error of hypothesis h. Namely:

$$R(h) = E_{(x,y)\sim D} I\left[h(x) \neq y\right] \tag{6.18}$$

Then it is easy to show (see recitation) that:

$$E_{S_n}\left[\hat{R}_{LOO}(S_n)\right] = E_{S_{n-1}}\left[R(h_{S_{n-1}})\right] \tag{6.19}$$

Namely, the expected LOO error equals the expected true generalization error, but where the latter is over samples of size $n-1$.

For SVMs it is easy to relate the LOO error to the number of support vectors. To see this, note that removing a point $\boldsymbol{x}_i$ from the training set, we will only err on $\boldsymbol{x}_i$ if it is a support vector (you can use the relation between α_i and support vectors to show this directly). Thus, denoting $SV(S_n)$ the number of support vectors obtained from training an SVM on S_n, we get:

$$E_{S_{n-1}}\left[R(h_{S_{n-1}})\right] \leq \frac{1}{n} E_{S_n}\left[SV(S_n)\right] \tag{6.20}$$

6.3.5 SVM for Non Separable Data

If the data cannot be separated the constraints of the primal problem Equation 6.8 cannot be met. We therefore introduce variables $\xi_i \geq 0$ that allow violation of the constraint, but try to minimize these violations in addition to the norm of $\boldsymbol{w}$. Formally we replace the "hard" constraints $y_n(\boldsymbol{w}\cdot\boldsymbol{x}_i+b) \geq 1$ by "soft" constraint $y_n(\boldsymbol{w}\cdot\boldsymbol{x}_n+b) \geq 1-\xi_n$ and $\xi_i \geq 0$. For $\xi_i = 0$ we have the required margin constraints. For $\xi_i \in [0,1]$, we have a correct classification, but have a smaller margin than we desire. For $\xi_i > 1$ we have an error.

The new optimization problem is,

$$\begin{aligned} &\min \frac{1}{2}\|\boldsymbol{w}\|_2^2 + C\sum_{i=1}^{n}\xi_i \\ &\text{s.t. } y_n(\boldsymbol{w}\cdot\boldsymbol{x}_i + b) \geq 1 - \xi_i \quad \forall i = 1,\ldots,n \\ &\quad\ \xi_i \geq 0 \end{aligned}$$

A few comments:

- The program is always feasible. We can always set ξ_i large enough and get a feasible solution.
- C is the regularization parameter. Small C allows to ignores constraints more easily. Large C forces to consider the constraints more. At the extreme $C = \infty$ is the hard constraints.
- We still have a unique minimum.
- We need to somehow choose the parameter C. In practice, some form of cross validation is used (e.g., leave one out as above, or more often 5 fold cross-validation).

We can re-derive the dual constraints. The only difference is that now we have $\boldsymbol{\alpha} \leq C$.

Bibliography

[1] S. Shalev-Shwartz and S. Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge University Press, 2014.

[2] J. Shawe-Taylor, P. L. Bartlett, R. C. Williamson, and M. Anthony. Structural risk minimization over data-dependent hierarchies. *IEEE transactions on Information Theory*, 44(5):1926–1940, 1998.