Introduction to Machine Learning	**Fall Semester, 2013**
Lecture 6: November 17	
Lecturer: Lior Wolf	*Scribe: ym*

6.1 Support Vector Machine (SVM) classifiers

The *Support Vector Machine (SVM)* is perhaps the leading machine learning algorithm today for binary classification problems. It is a high quality off-the-shelf classifier, that can adapt to many ML settings. It provides both robustness and uniformity.

We will cover today the following topics:

- Review of linear classifiers
  - Realizability = Linear separability.
  - Revisiting Perceptron
- Support Vector machine (SVM) classifier
  - Wide margin principle
  - Derivation
  - Dual form
  - Slack variables
  - Loss function
  - Multi class item Practical advice

6.2 Review of Linear classifiers

6.2.1 Binary classification

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

In Figure 6.1 we see an example which is clearly linear 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 this specific cylindrical case.)

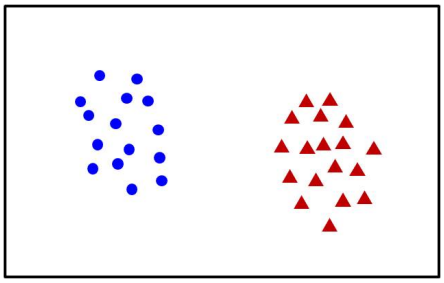

Figure 6.1: Easy linear separable case

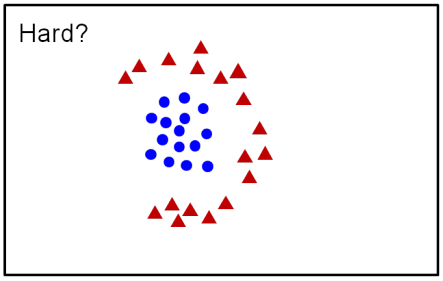

Figure 6.2: Non-linearly separable case.

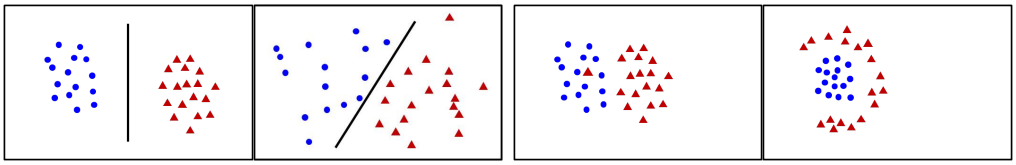

Figure 6.3: More examples of separable and non-separable cases.

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

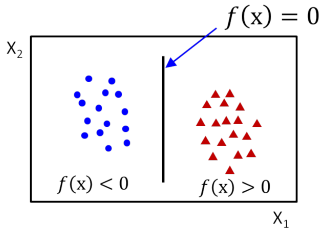

Figure 6.4: A linear separator in 2D

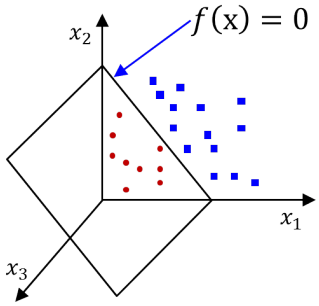

Figure 6.5: A linear separator in 3D

6.2.2 Perceptron - review

Given a linearly separable data (x_i, y_i) the perceptron algorithm finds a linear separator $f(x) = w^t x + b$ which separates the data points. The Perceptron algorithm works as follows:

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

1. Add b to the attributes, and a constant attribute of +1.
2. Initialize $w = 0$.
3. Cycle through the data points (x_i, y_i). For each error update $w = w + y_i x_i$.
4. Terminate when there are no errors

It is clear that the perceptron algorithm only adds or subtracts points. Therefore, at any time $w = \sum_{i=1}^{N} \alpha_i x_i$ for some integers α_i.

While the Perceptron algorithm is guarantee to compute a linear separator, if one exists, it might have a long convergence time. Also, the resulting separator might have a very small margin. (In the recitation of lecture 5 you saw how to modify the Perceptron algorithm to generate a separator with a "good margin".)

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 separator (which have no error on the data). One reasonable criteria 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 $w^t x + b$ as a confidence, it give higher confidence for all examples in the data.

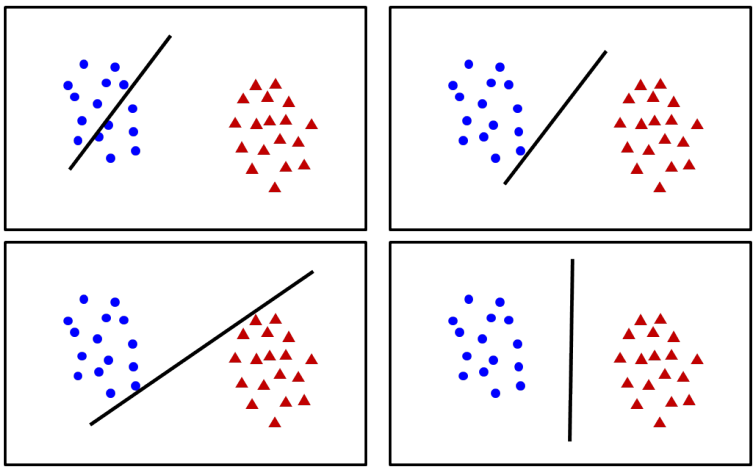

Figure 6.6: Possible linear separators

Let us first discuss the influence of the weight vector w (see Figure 6.7). Any point x on the hyperplane satisfies $w^t x + b = 0$. This implies that if we take two points x' and x'' on

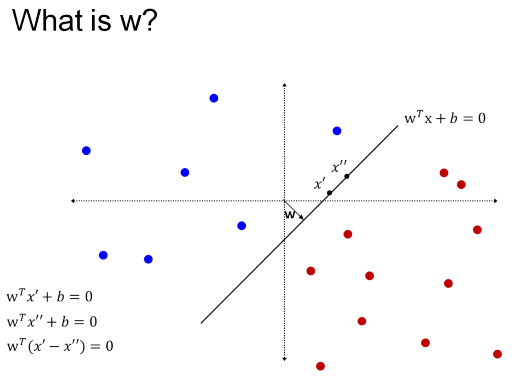

Figure 6.7: What is the w?

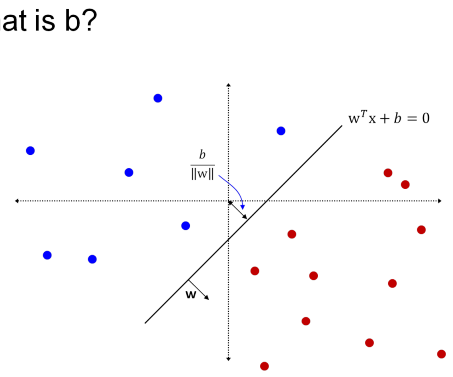

Figure 6.8: What is b?

the hyperplane, we have both $w^t x' + b = 0$ and $w^t x'' + b = 0$. Therefore $w^t(x' - x'') = 0$, which implies that w is perpendicular to plane.

In order to understand the influence of b, consider the distance from the hyperplane to the origin (see Figure 6.8). The distance from the hyperplane to the origin can be computed by the following optimization:

$$\begin{aligned} &\min \|x\| \\ &\text{s.t. } \; w^t x + b = 0 \end{aligned}$$

We can simplify the optimization by replacing $\|x\|$ by $\|x\|^2$, since the maximization will give an identical results. Furthermore, recall that $\|x\|^2 = x^t x$ and we have

$$\begin{aligned} &\min x^t x \\ &\text{s.t. } \; w^t x + b = 0 \end{aligned}$$

Let us now formally solve this simple optimization. We first write the Lagrangian

$$L(x, \lambda) = x^t x - \lambda(w^t x + b)$$

We take the derivative w.r.t x and get

$$\frac{d}{dx} L(x, \lambda) = 2x - \lambda w = 0$$

This implies that $x = (\lambda/2)w$. Now we can use the hyperplane equation to compute λ and get

$$w^t(\frac{\lambda}{2} w) + b = 0$$

This gives

$$\lambda = \frac{-2b}{w^t w}$$

now we can compute x,

$$x = \frac{\lambda}{2} w = \frac{-b}{w^t w} w$$

We can now compute the distance, which is the norm of x,

$$\|x\| = \sqrt{x^t x} = \sqrt{(v)^t (\frac{-b}{w^t w} w)} = \frac{b}{w^t w} \sqrt{w^t w} = \frac{b}{\sqrt{w^t w}} = \frac{b}{\|w\|}$$

Note that our problem is scale invariant, since multiplying w and b by a fixed positive constant c would give the same classification, i.e. $sign(w^t x + b) = sign((cw)^t x + cb)$. We can verify that the distance is also scale invariant, since

$$\frac{cb}{\|cw\|} = \frac{b}{\|w\|}$$

6.3.1 Support vectors and margin

Which points influence the hyperplane that we select? Clearly only the points which are the closest to the decision boundary. Any other point, event if we delete it, will not change the maximum margin hyperplane. We can now use out degree of freedom in selecting the scale. We can set the margin to be 1. Namely, for the positive support vectors x we have $w^t x + b = 1$ and for the negitive support vectors x we have $w^t x + b = -1$. Equivalently, we can require that

$$\min_x |w^t x + b| = 1$$

Once we fixed the scale in the above way, we will show that the maximum margin is exactly $2/\|w\|$. the computation is very similar to the computation of the distance from the origin.

Let x_n be a support vector, and we like to compute its distance from the hyperplane $w^t x + b$. The distance between x_n and any point x on the hyperplane can be decomposed to a vector on the hyperplane and a vector perpendicular to the hyperplane, i.e., in the direction of $\hat{w} = w/\|w\|$. The distance in the direction of $\hat{w}$, of the vector $x_n - x$ is $|\hat{w}(x_n - x)|$. This is done as follows,

$$|\hat{w}^t (x_n - x)| = \frac{1}{\|w\|} |w^t x_n - w^t x| = \frac{1}{\|w\|} |w^t x_n + b - w^t x - b| = \frac{1}{\|w\|} |w^t x_n + b| = \frac{1}{\|w\|}$$

where the third equality uses the fact that $w^t x + b = 0$ and the fourth uses the fact that we scale the points such that $w^t x_n + b = 1$.

This implies that the distance in each direction is $1/\|w\|$ and therefore the margin is $2/\|w\|$.

6.3.2 SVM optimization

We would like to maximize the margin. We can write the following optimization problem:

$$\begin{aligned} &\max \frac{1}{\|w\|} \\ &\text{s.t.} \quad \min_n |w^t x_n + b| = 1 \end{aligned}$$

We would like to turn the optimization to a more conventional form. We can replace $|w^t x_n + b|$ by $y_n(w^t x_n + b)$, which is linear in w and b. Instead of requiring a value of 1 for the closest point x_n, we can require a value of at least 1 to all points. Since the maximization of $1/||w||$ acts to reduce the values of the form $w^t x + b$, for at least one point the inequality will be tight.

We can also replace the maximization of $1/\|w\|$ by the minimization of $0.5\|w\|^2$ and get the following.

$$\begin{aligned} &\min \frac{1}{2} w^t w \\ &\text{s.t.} \quad y_n(w^t x_n + b) \geq 1 \quad \forall n = 1, \ldots, N \end{aligned}$$

Recall that $w \in \mathbb{R}^d$ and $b \in \mathbb{R}$.

This is a constraint optimization and the solution can be derived based on the lagrangian multiplies. Since we have inequality constraints, the optimization can be done through the Karush-Kuhn-Tucker (KKT) conditions. Below, we provide a self-contained explanation that does not require the KKT theorem.

We start by deriving the lagrangian formulation.

$$L(w, b, \alpha) = \frac{1}{2} w^t w - \sum_{n=1}^{N} \alpha_n (y_n(w^t x_n + b) - 1)$$

where we minimize w.r.t. w and b and maximize for $\alpha_n \geq 0$, i.e., $\min_{w,b} \max_\alpha L(w, b, \alpha)$. One can view the role of α_n as controlled by an adversary. If some constraint is violated, namely $y_n(w^t x_n + b) < 1$, then the adversary can set $\alpha_n = \infty$ and $L = \infty$. Therefore, in the solution we will have all the constraints satisfied. In addition, if $y_n(w^t x_n + b) > 1$ the the adversary will set $\alpha_n = 0$. In this case we are simply maximizing $(1/2)w^t w$ subject to the constraints.

We now consider the gradient of L w.r.t. w and b in order to minimize L. In order to do so, there is an implicit assumption that $\min_{w,b} \max_\alpha L(w, b, \alpha) = \max_\alpha \min_{w,b} L(w, b, \alpha)$, which is the case due to a theorem called Slater's condition. This topic is outside the scope of our course.

$$\nabla_w L = w - \sum_{n=1}^{N} \alpha_n y_n x_n = 0 \quad \Longrightarrow \quad w = \sum_{n=1}^{N} \alpha_n y_n x_n$$

This implies that given the solution α we can compute the weights w.

For the derivative w.r.t. b we have,

$$\frac{d}{db} L = -\sum_{n=1}^{N} \alpha_n y_n = 0 \quad \Longrightarrow \quad \sum_{n=1}^{N} \alpha_n y_n = 0$$

This implies a restriction on the α_n.

we can now substitute the two constraints in L,

$$L(\alpha) = -\frac{1}{2} \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_i \alpha_j y_i y_j x_i^t x_j + \sum_{i=1}^{N} \alpha_i$$

where we are maximizing w.r.t. $\alpha_i \geq 0$ and $\sum_{i=1}^{N} \alpha_i y_i = 0$.

We can write this as a quadratic program

$$\begin{aligned} &\min_{\alpha} \alpha^t M \alpha + (-1^t)\alpha \\ &\text{s.t. } y^t \alpha = 0 \\ &\quad 0 \leq \alpha \\ &\quad M = \begin{pmatrix} y_1 y_1 x_1^t x_1 & y_1 y_2 x_1^t x_2 & \dots \\ \vdots & \vdots & \vdots \\ y_N y_1 x_N^t x_1 & y_N y_2 x_n^t x_2 & \cdots \end{pmatrix} \end{aligned}$$

we need to show 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 $y_i x_i$. This implies that M is positive semi-definite (note that for any z we have $z^t M z = z^t A^t A z = \|Az\|^2 \geq 0$.)

We can contrast this with the primal form:

$$\begin{aligned} &\min \frac{1}{2} w^t w \\ &\text{s.t. } y_n (w^t x_n + b) \geq 1 \quad \forall n = 1, \dots, N \end{aligned}$$

Comparing the primal and the dual:

- Variables: The primal has $d + 1$ variables (w and b) and the dual has N variables (α one per example)

- Constrains: The dual has only non-negatively constrains and one equality constraint. The primal has N constraints,one per margin.

Given a solution to the dual α, we can compute the weights using $w = \sum_{i=1}^{N} \alpha_i y_i x_i$.

Recall that due to the adversary role α plays in the optimization, for all n the following equality holds: $\alpha_n(y_n(w^t x_n + b) - 1) = 0$. For any point n for which $\alpha_n > 0$, this implies that $y_n(w^t x_n + b) = 1$. Therefore, this point is a support vector.(Its margin from the hyperplane define by w is minimal.)

We need to compute the parameter b. Given any support vector n, we have that $y_n(w^t x_n + b) = 1$. This implies that

$$b = y_n - w^t x_n$$

For numerical stability, it is better to average multiple support vectors, and not a single one.

Given the dual solution, we can define the decision boundary as

$$f(x) = (\sum_{i=1}^{N} \alpha_i y_i x_i^t)x + b = \sum_{i=1}^{N} \alpha_i y_i (x_i^t x) + b.$$

6.3.3 Leave One Out (LOO) error bound

We will show a rather simple bound for the Leave One Out (LOO). More interesting and sophisticated bounds exists for the SVM, based on the margin and are independent from the dimension size d.

The main observation for the LOO is to notice that any data point which is not a support vector does not influence the solution of the maximum margin. We can give two intuitive proofs of this fact.

Geometric interpretation: All the support vector have the same distance from the hyperplane w. If there is a hyperplane w' that can increase the margin, then we should be able to increase the margin of w (by taking a small step in the direction of w').

Optimization interpretation: We are solving the following problem:

$$\begin{aligned} &\min \frac{1}{2} w^t w \\ &\text{s.t. } y_n(w^t x_n + b) \geq 1 \quad \forall n = 1, \ldots, N \end{aligned}$$

Deleting a constraint can only create a better solution (in our case, smaller $\|w\|$)). If the deleted constraint is not a support vector, we have the same solution. (If it would improve the solution to w', we can take a small step from w to w' and still have a feasible solution in the optimization problem with a lower cost).

The number of mistakes we can make in the LOO is at most the number of support vector. Therefore, the LOO error is at most $\#SV/N$, where $\#SV$ is the number of support vectors.

6.3.4 Non-realizable case

We can add slack variables to support the margin requirement. Formally we replace the "hard" constraints $y_n(w^t x_n + b) \geq 1$ by "soft" constraint $y_n(w^t x_n + b) \geq 1 - \xi_n$ and $\xi_n \geq 0$. For $\xi_n = 0$ we have the required margin constraint. For $\xi_n \in [0, 1]$, we have a correct classification, but have a smaller margin than we desire. For $\xi_n > 1$ we have an error.

The new optimization problem is,

$$\begin{aligned} &\min \frac{1}{2} w^t w + C \sum_{i=1}^{N} \xi_i \\ &\text{s.t. } y_n(w^t x_n + b) \geq 1 + \xi_n \quad \forall n = 1, \ldots, N \\ &\quad \xi_n \geq 0 \end{aligned}$$

A few comments

- The program is always feasible. We can always set ξ_n 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.
- Pain: We have one more parameter to tune. even more confusing is that C relates two quantities which are different (weights magnitude and the errors).

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

Consider the constraint $y_n(w^t x + b) \geq 1 - \xi_n$. Let $f(x) = w^t x + b$, the hypothesis we are using. Then the constraint is $y_n f(x_n) \geq 1 - \xi_n$. When we incorporate the constraint $\xi_n \geq 0$ we have

$$\xi_n = \max\{0, 1 - y_n f(x_n)\}$$

This is essentially the *hinge loss* with margin 1. We can view the hinge loss as upper bounding the classification loss. While it is not the loss we would have developed in an ideal world, the main benefit of the hinge loss is that it is computationally tractable (unlike the classification loss). In addition, a small hinge loss implies a small classification loss. So at least in one direction we have a correct bound.

6.3.5 multi-class SVM

There are a few suggestions how to extend the SVM framework to work with multi-class (beyond the binary classification). However, most of the methods have a limited success.

The popular methods for handling multi-class is a reduction to classification, and then using known classifiers (such as SVM). There are two popular reductions, one vs all and one vs one.

In *one vs all* we build a classifier per class, to predict if a given input belongs to a class k. If multiple predictors claim that the input belongs to their class, we select the one with the highest decision value, i.e., $y(w^t x + b)$.

In *one vs one* we build a quadratic number of classifiers, trying to separate each two classes. Given a point, we run all the classifiers, and select the class that the highest number of classifiers "voted" for.

6.3.6 Practical advice

1. Avoid Matlab implementation. For some reason it is very slow. You can use *libsvm* which can be downloaded from `http://www.csie.ntu.edu.tw/ cjlin/libsvm/` or for the linear case `http://www.csie.ntu.edu.tw/ cjlin/liblinear/`

2. Play with C, this is more an art, and depends on the input.

3. Avoid unbalanced classes. This is especially important for the un-realizable case (which dominates reality)