The Support Vector Machine

So far we have used a reference assumption that there exists a linear classifier that has a large geometric margin, i.e., whose decision boundary is well separated from all the training images (examples). Such a large margin classifier seems like one we would like to use. Can't we find it more directly? Yes, we can. The classifier is known as the Support Vector Machine or SVM for short.

You could imagine finding the maximum margin linear classifier by first identifying any classifier that correctly classifies all the examples (Figure 2a) and then increasing the geometric margin until the classifier "locks in place" at the point where we cannot increase the margin any further (Figure 2b). The solution is unique.

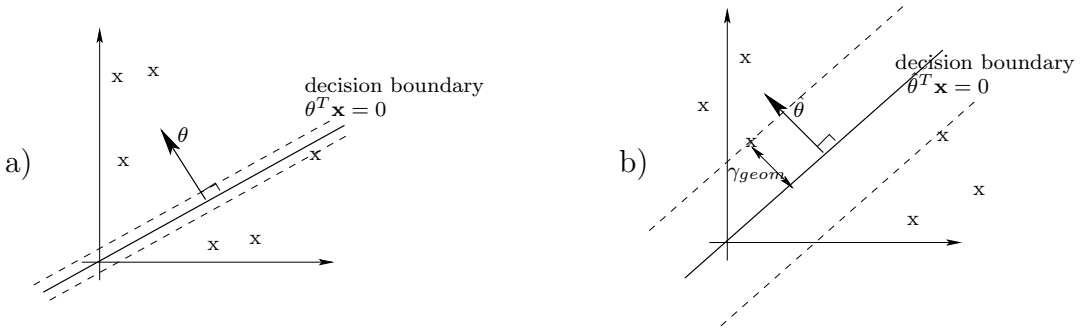

Figure 1: a) A linear classifier with a small geometric margin, b) maximum margin linear classifier.

More formally, we can set up an optimization problem for directly maximizing the geometric margin. We will need the classifier to be correct on all the training examples or $y_t \theta^T \mathbf{x}_t \geq \gamma$ for all $t = 1, \ldots, n$. Subject to these constraints, we would like to maximize $\gamma / \|\theta\|$, i.e., the geometric margin. We can alternatively minimize the inverse $\|\theta\|/\gamma$ or the inverse squared $\frac{1}{2}(\|\theta\|/\gamma)^2$ subject to the same constraints (the factor 1/2 is included merely for later convenience). We then have the following optimization problem for finding $\hat{\theta}$:

$$\text{minimize } \frac{1}{2}\|\theta\|^2/\gamma^2 \text{ subject to } y_t \theta^T \mathbf{x}_t \geq \gamma \text{ for all } t = 1, \ldots, n \tag{1}$$

We can simplify this a bit further by getting rid of γ. Let's first rewrite the optimization problem in a way that highlights how the solution depends (or doesn't depend) on γ:

$$\text{minimize } \frac{1}{2}\|\theta/\gamma\|^2 \text{ subject to } y_t (\theta/\gamma)^T \mathbf{x}_t \geq 1 \text{ for all } t = 1, \ldots, n \tag{2}$$

In other words, our classification problem (the data, setup) only tells us something about the ratio θ/γ, not θ or γ separately. For example, the geometric margin is defined only on the basis of this ratio. Scaling θ by a constant also doesn't change the decision boundary. We can therefore fix $\gamma = 1$ and solve for θ from

$$\text{minimize } \frac{1}{2}\|\theta\|^2 \text{ subject to } y_t\theta^T\mathbf{x}_t \geq 1 \text{ for all } t = 1, \ldots, n \tag{3}$$

This optimization problem is in the standard SVM form and is a *quadratic programming* problem (objective is quadratic in the parameters with linear constraints). The resulting geometric margin is $1/\|\hat{\theta}\|$ where $\hat{\theta}$ is the unique solution to the above problem. The decision boundary (separating hyper-plane) nor the value of the geometric margin were affected by our choice $\gamma = 1$.

General formulation, offset parameter

We will modify the linear classifier here slightly by adding an offset term so that the decision boundary does not have to go through the origin. In other words, the classifier that we consider has the form

$$f(\mathbf{x}; \theta, \theta_0) = \text{sign}(\, \theta^T\mathbf{x} + \theta_0 \,) \tag{4}$$

with parameters θ (normal to the separating hyper-plane) and the offset parameter θ_0, a real number. As before, the equation for the separating hyper-plane is obtained by setting the argument to the sign function to zero or $\theta^T\mathbf{x} + \theta_0 = 0$. This is a general equation for a hyper-plane (line in 2-dim). The additional offset parameter can lead to a classifier with a larger geometric margin. This is illustrated in Figures 2a and 2b. Note that the vector $\hat{\theta}$ corresponding to the maximum margin solution is different in the two figures.

The offset parameter changes the optimization problem only slightly:

$$\text{minimize } \frac{1}{2}\|\theta\|^2 \text{ subject to } y_t(\theta^T\mathbf{x}_t + \theta_0) \geq 1 \text{ for all } t = 1, \ldots, n \tag{5}$$

Note that the offset parameter only appears in the constraints. This is different from simply modifying the linear classifier through origin by feeding it with examples that have an additional constant component, i.e., $\mathbf{x}' = [\mathbf{x}; 1]$. In the above formulation we do not bias in any way where the separating hyper-plane should appear, only that it should maximize the geometric margin.

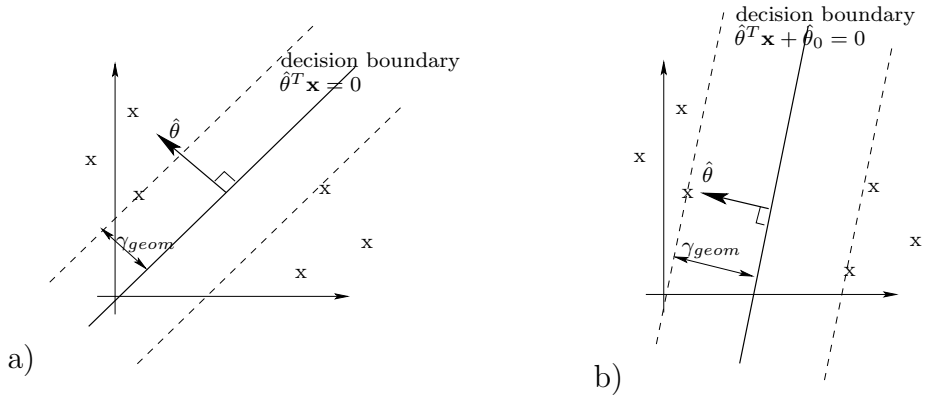

Figure 2: a) Maximum margin linear classifier through origin; b) Maximum margin linear classifier with an offset parameter

Properties of the maximum margin linear classifier

The maximum margin classifier has several very nice properties, and some not so advantageous features.

Benefits. On the positive side, we have already motivated these classifiers based on the perceptron algorithm as the "best reference classifiers". The solution is also unique based on any linearly separable training set. Moreover, drawing the separating boundary as far from the training examples as possible makes it robust to noisy examples (though not noisy labels). The maximum margin linear boundary also has the curious property that the solution depends only on a subset of the examples, those that appear exactly on the margin (the dashed lines parallel to the boundary in the figures). The examples that lie exactly on the margin are called *support vectors* (see Figure 3). The rest of the examples could lie anywhere outside the margin without affecting the solution. We would therefore get the same classifier if we had only received the support vectors as training examples. Is this is a good thing? To answer this question we need a bit more formal (and fair) way of measuring how good a classifier is.

One possible "fair" performance measure evaluated only on the basis of the training examples is *cross-validation*. This is simply a method of retraining the classifier with subsets of training examples and testing it on the remaining held-out (and therefore fair) examples, pretending we had not seen them before. A particular version of this type of procedure is called *leave-one-out cross-validation*. As the name suggests, the procedure is defined as follows: select each training example in turn as the single example to be held-out, train the

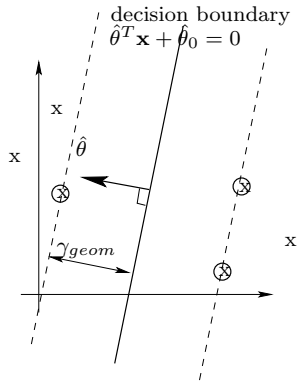

Figure 3: Support vectors (circled) associated the maximum margin linear classifier

classifier on the basis of all the remaining training examples, test the resulting classifier on the held-out example, and count the errors. More precisely, let the superscript '$-i$' denote the parameters we would obtain by finding the maximum margin linear separator without the i^{th} training example. Then

$$\text{leave-one-out CV error} = \frac{1}{n}\sum_{i=1}^{n} \text{Loss}\Big(y_i, f(\mathbf{x}_i; \theta^{-i}, \theta_0^{-i})\Big) \tag{6}$$

where $\text{Loss}(y, y')$ is the zero-one loss. We are effectively trying to gauge how well the classifier would generalize to each training example if it had not been part of the training set. A classifier that has a low leave-one-out cross-validation error is likely to generalize well though it is not guaranteed to do so.

Now, what is the leave-one-out CV error of the maximum margin linear classifier? Well, examples that lie outside the margin would be classified correctly regardless of whether they are part of the training set. Not so for support vectors. They are key to defining the linear separator and thus, if removed from the training set, may be misclassified as a result. We can therefore derive a simple upper bound on the leave-one-out CV error:

$$\text{leave-one-out CV error} \leq \frac{\#\text{ of support vectors}}{n} \tag{7}$$

A small number of support vectors – a sparse solution – is therefore advantageous. This is another argument in favor of the maximum margin linear separator.

Problems. There are problems as well, however. Even a single training example, if mislabeled, can radically change the maximum margin linear classifier. Consider, for example,

what would happen if we switched the label of the top right support vector in Figure 3.

Allowing misclassified examples, relaxation

Labeling errors are common in many practical problems and we should try to mitigate their effect. We typically do not know whether examples are difficult to classify because of labeling errors or because they simply are not linearly separable (there isn't a linear classifier that can classify them correctly). In either case we have to articulate a trade-off between misclassifying a training example and the potential benefit for other examples.

Perhaps the simplest way to permit errors in the maximum margin linear classifier is to introduce "slack" variables for the classification/margin constraints in the optimization problem. In other words, we measure the degree to which each margin constraint is violated and associate a cost for the violation. The costs of violating constraints are minimized together with the norm of the parameter vector. This gives rise to a simple relaxed optimization problem:

$$\text{minimize } \frac{1}{2}\|\theta\|^2 + C\sum_{t=1}^{n}\xi_t \tag{8}$$

$$\text{subject to } \; y_t(\theta^T\mathbf{x}_t + \theta_0) \geq 1 - \xi_t \;\text{ and } \xi_t \geq 0 \;\text{ for all } t = 1, \ldots, n \tag{9}$$

where ξ_t are the slack variables. The margin constraint is violated when we have to set $\xi_t > 0$ for some example. The penalty for this violation is $C\xi_t$ and it is traded-off with the possible gain in minimizing the squared norm of the parameter vector or $\|\theta\|^2$. If we increase the penalty C for margin violations then at some point all $\xi_t = 0$ and we get back the maximum margin linear separator (if possible). On the other hand, for small C many margin constraints can be violated. Note that the relaxed optimization problem specifies a particular *quantitative* trade-off between the norm of the parameter vector and margin violations. It is reasonable to ask whether this is indeed the trade-off we want.

Let's try to understand the setup a little further. For example, what is the resulting margin when some of the constraints are violated? We can still take $1/\|\hat{\theta}\|$ as the geometric margin. This is indeed the geometric margin based on examples for which $\xi_t^* = 0$ where '*' indicates the optimized value. So, is it the case that we get the maximum margin linear classifier for the subset of examples for which $\xi_t^* = 0$? No, we don't. The examples that violate the margin constraints, including those training examples that are actually misclassified (larger violation), do affect the solution. In other words, the parameter vector $\hat{\theta}$ is defined on the basis of examples right at the margin, those that violate the constraint but not enough to

be misclassified, and those that are misclassified. All of these are support vectors in this sense.