Binary Support Vector Machines for Classification

Carlo Tomasi

October 11, 2021

Any linear binary classifier separates the space $\mathbb{R}^d$ of all data points $\mathbf{x}_n$ into two half-spaces, ideally splitting data points with positive label $y_n = 1$ from those with negative label[1] $y_n = -1$. Of course, the joint model $p(\mathbf{x}, y)$ from which the data are drawn may not be linearly separable. As a consequence, linear classifiers are typically formulated so as to allow for some misclassification.

Consider placing a decision boundary (a hyperplane in $\mathbb{R}^d$) where it splits as much of the data as possible correctly into the two classes. After this choice is made, there is often still quite a bit of freedom on the exact position and orientation of the boundary: Especially when the dimensionality is large, it is often possible to translate and rotate the hyperplane somewhat, without changing the label the classifier assigns to any of the training data. Placing the boundary in just the right place, in the face of this freedom, is important, and drives the generalization performance of the classifier. In other words, moving the hyperplane around a bit may not change the error rate on the training set, but it may change that on previously unseen data.

A logistic-regression classifier chooses where to place the decision boundary somewhat arbitrarily, as its loss function penalizes *every* training sample in the training set T according to how close it is to the separating hyperplane, and on which side of it. However, once the boundary splits T as well as possible, the points in T that end up being close to the boundary are the only data that really matter when deciding where exactly the boundary should be. Points that are very far away from the boundary should have little or no say. This is particularly important in the presence of *outliers*, which, loosely speaking, are rare data samples where $\mathbf{x}$ is distant from more typical values. Figure 1 illustrates that even a single outlier can shift the boundary of a logistic-regression classifier quite a bit, and this seems undesirable, since the exact position of the boundary affects the output of the classifier only for points that are near it.

A different family of classifiers, called *Support Vector Machines* (SVMs), still uses a separating hyperplane as the decision boundary. Thus SVMs, in their simplest form, are linear classifiers as well. However, after the hyperplane is placed so that it separates the data in T as well as possible, it then "pays attention" only to training samples that are either incorrectly classified or correctly classified but very close to the hyperplane.

To achieve this effect, the loss function used in SVMs insists that the decision boundary has as wide a *margin* on both of its sides as possible. A margin is a slab of space (that is, a hyperplane with some thickness) that has as few training samples as possible. By leaving this nearly empty space around the decision boundary, SVMs reduce the probability that a previously unseen sample falls close to this boundary, and therefore possibly on the wrong side of it.

[1]The specific choice of label values is conceptually irrelevant. For support vector machines, labeling negative samples with -1 rather than 0 simplifies the math considerably.

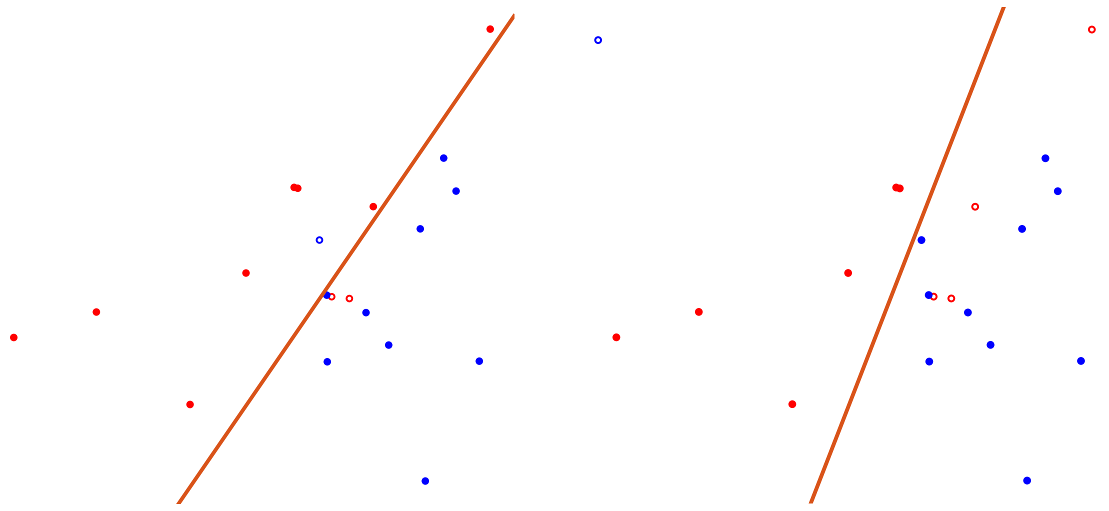

Figure 1: The color of dots in these two diagrams denotes true class membership, and hollow dots are samples that the logistic-regression classifier (whose decision boundary is the orange line) misclassifies. The training sets in the two diagrams are identical, except for a single data sample added to the one on the right, close to the top-left corner. This distant sample is enough to shift the decision boundary quite a bit.

As a result, SVMs tend to generalize quite well, at the cost of solving a more involved optimization problem. Specifically, learning an SVM turns out to be a strictly convex quadratic program with a unique solution. Fairly efficient algorithms exist for these optimization problems, but they do take longer than solving an unconstrained quadratic minimization problem of the type that logistic-regression classifiers do.

The superiority of margin-based classifiers over logistic regression classifiers is in large part theoretical when the training set has a nontrivial size. For large training sets, the performance of logistic-regression classifiers is quite similar to that of SVMs in practice. Instead, what made SVMs popular is that they make it possible to go well beyond linear classification boundaries through what are called *kernels*, which allow the decision boundary to be of nearly arbitrary complexity, as we will see in a later note.

The next two sections discuss the geometry of high-margin binary classifiers and the attending loss and risk function. Minimizing the empirical risk thus defined leads to what are called *soft SVMs*, to distinguish them from the "hard" version, which makes the unrealistic assumption that training data are linearly separable.

1 The Geometry of High-Margin Classifiers

Separating Hyperplane Consider a binary classification problem with data space $X = \mathbb{R}^d$ and label space $Y = \{-1, 1\}$. A hyperplane in X can be represented by an equation of the form

$$\mathbf{n}^T\mathbf{x} + c = 0 \quad \text{with} \quad \|\mathbf{n}\| = 1 \; , \tag{1}$$

where $\mathbf{n}$ is a unit vector in $\mathbb{R}^d$ and c is a scalar. The decision rule

$$\hat{y} = h(\mathbf{x}) = \text{sign}(\mathbf{n}^T\mathbf{x} + c) \tag{2}$$

classifies the sample $(\mathbf{x}, y)$ correctly (that is, $\hat{y} = y$) when[2]

$$\mathbf{n}^T\mathbf{x} + c \geq 0 \quad \text{if } y = 1$$

and

$$\mathbf{n}^T\mathbf{x} + c \leq 0 \quad \text{if } y = -1 \quad .$$

These two inequalities can be written compactly as follows:

$$y(\mathbf{n}^T\mathbf{x} + c) \geq 0 \tag{3}$$

since $y \in \{1, -1\}$.

Margin The *margin* of a sample $(\mathbf{x}, y) \in X \times Y$ with respect to the separating hyperplane with parameters $\mathbf{v} = (\mathbf{n}, c)$ is the real value

$$\mu_{\mathbf{v}}(\mathbf{x}, y) \overset{\text{def}}{=} y\,(\mathbf{n}^T\mathbf{x} + c)\ .$$

Thus, the margin of a sample is the left-hand side of the inequality 3, that is, the slack by which the inequality is satisfied. A high-margin sample is well inside the correct decision region of the classifier. Similarly, a sample with a large negative margin is well inside the wrong region.

Geometrically, for a correctly classified sample, for which therefore $\hat{y} = y$, the margin is the distance of the data point from the separating hyperplane, as we saw when we discussed the geometry of linear classifiers. For an incorrectly classified sample, the margin is the negative of this distance. Thus, the margin of a sample is its signed distance from the decision boundary, which we called $\Delta(\mathbf{x})$ in that discussion.

The *margin* $\mu_{\mathbf{v}}(T)$ *of a training set*

$$T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

is the smallest margin of any of its samples:

$$\mu_{\mathbf{v}}(T) \overset{\text{def}}{=} \min_{(\mathbf{x},y)\in T} \mu_{\mathbf{v}}(\mathbf{x}, y)$$

and the hyperplane with parameters $\mathbf{v}$ is said to *linearly separate* training set T if

$$\mu_{\mathbf{v}}(T) > 0\ .$$

The training set T is *linearly separable* if there exists a hyperplane that linearly separates it.

[2]We deem the classification of a sample *on* the decision boundary to be correct regardless of its true label.

2 Loss and Risk

The Hinge Loss A separating hyperplane for which the given training set T has a large (positive) margin is likely to generalize well. This is because to the extent that the training set is representative of the true distribution of the data, if few training samples fall close to the boundary then few samples will fall close to it even at inference time, and are therefore less likely to be misclassified.

However, a training set T may not be linearly separable at all, let alone with a large margin: No matter where one places a separating hyperplane, some of the samples in T may end up falling on the wrong side of it. The SVM loss function is defined to encourage the training algorithm to seek separating hyperplanes that achieve large margins for most of the data and as few negative margins as possible on the training set.

Specifically, consider some strictly positive *reference margin*[3] $\mu^* > 0$, and define the *hinge loss*

$$\ell_{\mathbf{v}}(\mathbf{x}, y) = \frac{1}{\mu^*} \max\{0, \mu^* - \mu_{\mathbf{v}}(\mathbf{x}, y)\} \; .$$

Training samples with

$$\mu_{\mathbf{v}}(\mathbf{x}, y) \geq \mu^*$$

are classified correctly with a margin at least μ^*, and incur zero hinge loss. On the other hand, when

$$\mu_{\mathbf{v}}(\mathbf{x}, y) < \mu^* \; ,$$

the training sample $(\mathbf{x}, y)$ falls short of the reference margin μ^*, and the hinge loss is strictly positive. Specifically, if $0 < \ell_{\mathbf{v}} < 1$, the training sample is correctly classified but by a smaller margin. When $\ell_{\mathbf{v}} = 1$, the sample is on the separating hyperplane, and when $\ell_{\mathbf{v}} > 1$, the sample is misclassified. Figure 2 illustrates.

The Empirical Risk A good SVM classifier has both a small average hinge loss (*i.e.*, a small risk) *and* a large reference margin μ^*. To define a risk function that captures this criterion, the separating hyperplane in equation 1 is rescaled as follows:

$$\mathbf{w}^T\mathbf{x} + b = 0 \quad \text{with} \quad \mathbf{w} = \frac{\mathbf{n}}{\mu^*} \; , \;\; b = \frac{c}{\mu^*}$$

where $\mathbf{n}$ is a unit vector, so that the norm of the hyperplane coefficient vector $\mathbf{w}$ is $1/\mu^*$: The smaller the norm of $\mathbf{w}$, the larger the margin.

Then, the margin of sample $(\mathbf{x}, y)$ is

$$\mu_{(\mathbf{w},b)}(\mathbf{x}, y) \; = \; y\,(\mathbf{n}^T\mathbf{x} + c) \; = \; \mu^* y\,(\mathbf{w}^T\mathbf{x} + b) \; . \tag{4}$$

With these definitions, the hinge loss can be rewritten as

$$\ell_{(\mathbf{w},b)}(\mathbf{x}, y) \; = \; \max\{0, 1 - y(\mathbf{w}^T\mathbf{x} + b)\} \; . \tag{5}$$

The empirical risk function is defined as the sum of half of $\|\mathbf{w}\|^2 = \frac{1}{(\mu^*)^2}$ and a multiple of the average hinge loss:

$$L_T(\mathbf{w}, b) \; \stackrel{\text{def}}{=} \; \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C_0}{N}\sum_{n=1}^{N} \ell_{(\mathbf{w},b)}(\mathbf{x}_n, y_n) \tag{6}$$

[3]The margin $\mu_{\mathbf{v}}(\mathbf{x}, y)$ is a property of sample $(\mathbf{x}, y)$ and the classifier with parameters $\mathbf{v}$. The reference margin μ^* is a parameter of the classifier alone.

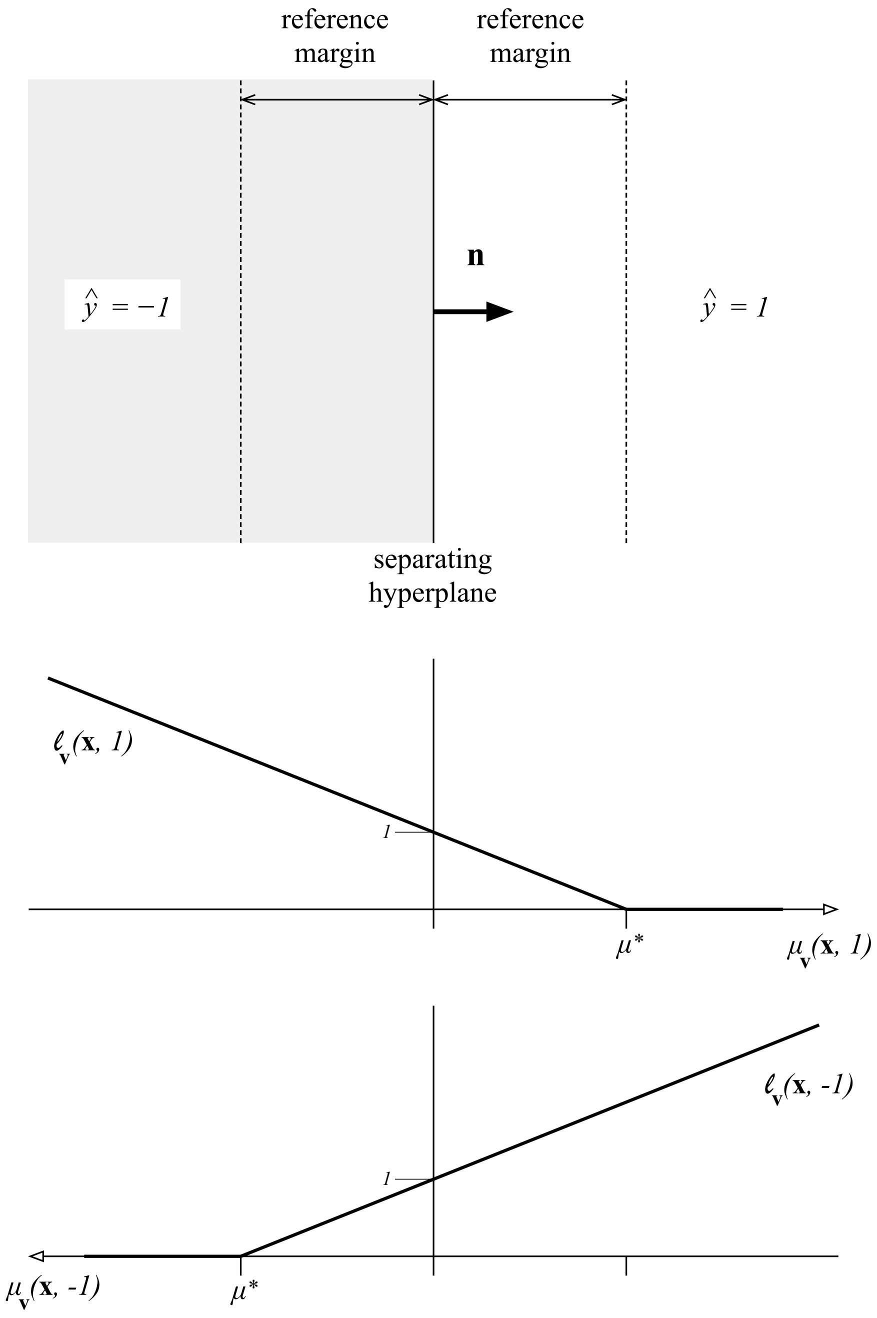
reference margin
reference margin
n
$\hat{y} = -1$
$\hat{y} = 1$
separating hyperplane
$\ell_{\mathbf{v}}(\mathbf{x}, 1)$
1
μ^*
$\mu_{\mathbf{v}}(\mathbf{x}, 1)$
$\ell_{\mathbf{v}}(\mathbf{x}, -1)$
1
$\mu_{\mathbf{v}}(\mathbf{x}, -1)$
μ^*

Figure 2: *Top:* The white area is the decision region for label 1, and the gray area is that for label -1. The unit vector $\mathbf{n}$ is perpendicular to the decision boundary and points towards the region for label 1. *Middle:* The hinge loss $\ell_{\mathbf{v}}(\mathbf{x}, 1)$ for data whose true label is 1 is plotted as a function of the margin $\mu_{\mathbf{v}}(\mathbf{x}, 1)$. The loss is equal to zero at the reference margin μ^* and beyond, and is equal to one on the decision boundary. *Bottom:* The function that relates the hinge loss $\ell_{\mathbf{v}}(\mathbf{x}, -1)$ to the margin $\mu_{\mathbf{v}}(\mathbf{x}, -1)$ for data whose true label is -1 is the same as that for true label 1. However, it is plotted here with the abscissa pointing to the left so that it relates more directly to the diagram at the top. Either way, even correctly classified samples can incur some loss, if their margin is too small.

where C_0 is some strictly positive constant.

The ERM classifier $(\mathbf{w}^*, b^*) = \text{ERM}_T(\mathbf{w}, b)$ implicitly finds a reference margin μ^* that strikes a balance between the two terms of the risk $L_T(\mathbf{w}, b)$: Reducing the norm of $\mathbf{w}$ increases the reference margin μ^* and reducing the average hinge loss decreases either the number of samples that violate the margin or the amount by which they violate it, or both. However, these two terms are in conflict with each other, because a wider reference margin causes more training samples to fall short of it. By changing the value of C_0, one can tune the trade-off implied by the two terms of $L_T(\mathbf{w}, b)$. Specifically, a large value of C_0 emphasizes the hinge loss and therefore leads to a narrower reference margin (bigger $\|\mathbf{w}\|$, because the influence of the first term of $L_T(\mathbf{w}, b)$ is reduced relative to that of the second) and fewer and/or smaller violations of the margin. Decreasing C_0 has the opposite effect. A good value for C_0 for a specific problem can be found by cross-validation.

3 Soft Linear Support Vector Machines

The ERM classifier

$$(\mathbf{w}^*, b^*) = \text{ERM}_T(\mathbf{w}, b) = \arg\min_{(\mathbf{w},b)} L_T(\mathbf{w}, b) \tag{7}$$

where the empirical risk L_T is defined in equation 6,

$$L_T(\mathbf{w}, b) \stackrel{\text{def}}{=} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C_0}{N}\sum_{n=1}^{N} \max\{0, 1 - y_n(\mathbf{w}^T\mathbf{x}_n + b)\} ,$$

is called a *Soft Linear Support Vector Machine* (Linear Soft SVM). The qualifier "linear" reminds us that these SVMs are linear classifiers, to distinguish them from the nonlinear versions that we will see when we introduce kernels. The term "soft" refers to the fact that the training set is not required to be linearly separable: Even the best separating hyperplane may still misclassify some samples. The formulation of SVMs would be simpler for linearly separable training sets, because it would then be possible to make the second term of the risk vanish. However, these "hard" SVMs are not too useful in practice, since it is difficult to guarantee linear separability. We may drop both qualifiers when there is no ambiguity.

Since the norm squared (the first term int he expression of the risk) is a strictly convex function of $\mathbf{w}$ and the hinge loss is weakly convex in both $\mathbf{w}$ and b, the risk $L_T(\mathbf{w}, b)$ is strictly convex, and *the solution to this minimization problem is therefore unique.* Uniqueness of the solution is one big advantage of SVMs, which however these classifiers share with the regularized version of the logistic regression classifier. We will see other advantages of SVMs soon.

Computing a Soft Linear SVM Suppose that we are given a training set

$$T = \{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$$

with $\mathbf{x}_n \in X = \mathbb{R}^d$ and $y_n \in Y = \{-1, 1\}$. We are also given a value for C_0, a positive real number that perhaps is to be refined through cross-validation. How can we compute the parameters $\mathbf{w}^* \in \mathbb{R}^d$ and $b^* \in \mathbb{R}$ that specify the separating hyperplane for the corresponding soft SVM? That is, how do we find $\text{ERM}_T(\mathbf{w}, b)$ for this machine learning problem?

Conceptually, the simplest, although by no means the most efficient, method for solving this convex optimization problem is to use gradient descent, or even Stochastic Gradient Descent (SGD). While the derivative of the hinge function

$$\rho(z) = \max\{0, z\}$$

is undefined at the origin, we know that gradient descent and SGD can be used with sub-gradients, and that we can use any sub-gradient where the gradient is undefined. The modification then is very simple. If we choose zero as the sub-derivative of $\rho(z)$ for $z = 0$, we can write

$$\rho'(z) = \begin{cases} 1 & \text{for } z > 0 \\ 0 & \text{elsewhere.} \end{cases} \tag{8}$$

The risk $L_B(\mathbf{w}, b)$ can be written as follows for a batch B of M samples with $1 \leq M \leq N$:

$$L_B(\mathbf{w}, b) \;=\; \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C_0}{M}\sum_{n=1}^{M} \rho(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$$

and the sub-gradient is then

$$\begin{aligned} \frac{\partial L_B}{\partial \mathbf{w}} &= \mathbf{w} - \frac{C_0}{M}\sum_{n=1}^{M} \rho'(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))\, y_n\mathbf{x}_n \\ \frac{\partial L_B}{\partial b} &= -\frac{C_0}{M}\sum_{n=1}^{M} \rho'(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))\, y_n \;. \end{aligned}$$

Note that the quantity $\rho'(1 - y_n(\mathbf{w}^T\mathbf{x}_n + b))$, in spite of its complex formula, is equal to either 0 or 1 (equation 8), so the sub-gradient can be computed easily.

Appendix

A The Dual Formulation of Soft Linear SVMs

It is mathematically possible to write necessary and sufficient conditions for a minimum of $L_T(\mathbf{w}, b)$ by reformulating the unconstrained optimization problem in equation 7 as a constrained problem via the introduction of so-called *slack variables* ξ_n for $n = 1, \ldots, N$. It then turns out that minimizing the risk L_T over $\mathbf{w}$ and b is the same as minimizing the function

$$f(\mathbf{w}, b, \boldsymbol{\xi}) \overset{\mathsf{def}}{=} \frac{1}{2}\|\mathbf{w}\|^2 + \frac{C_0}{N}\sum_{n=0}^{N} \xi_n \tag{9}$$

over $\mathbf{w}$, b, and $\boldsymbol{\xi} = (\xi_1, \ldots, \xi_N)$ with the additional constraint

$$\xi_n \geq \ell_{(\mathbf{w},b)}(\mathbf{x}_n, y_n) \quad \text{for} \quad n = 1, \ldots, N \; . \tag{10}$$

However, proving this equivalence and then developing necessary and sufficient conditions for this problem, which are called the Karush-Kuhn-Tucker (KKT) conditions, would take us far into the field of convex programming. This is in fact the traditional approach to SVMs, and may well be worthwhile studying if you plan to specialize in this type of predictor.