Note 3

03 Support Vector Machines

by Alvin Wan

In this note, we maximize the **margin** for a classifier, namely the distance between a decision boundary H and the point closest to it. A larger margin is preferred, as it indicates a higher confidence that our classifier is correct. Let us start by deriving the objective function. We stick with the same setup - a binary classification problem where $y_i \in \{1, -1\}$, $\hat{y}_i = \frac{1}{\|w\|_2}(w^T x_i + \beta)$ and $x_i, w \in \mathbb{R}^d$. We define the margin to be the following:

$$\min_i \frac{1}{\|w\|_2}(w^T x_i + \beta)$$

1 Hard Margin Classifier

First, we desire all classifications to be correct. Per the previous note, we thus want $y_i = \hat{y}_i$ or equivalently, $y_i \hat{y}_i \geq 0$ and

$$y_i(w^T x_i + \beta) \geq 0$$

However, this introduces a trivial solution with $w = \vec{0}$. We can thus equivalently write the following, where $m > 0$ is a constant that will become proportional to the margin.

$$y_i(w^T x_i + \beta) \geq m$$

Now, divide both sides by $\|w\|_2$.

$$\frac{y_i}{\|w\|_2}(w^T x_i + \beta) \geq \frac{m}{\|w\|_2}$$

Recall the distance between a point x_i and the decision boundary H is $\frac{1}{\|w\|_2}(w^T x_i + \beta)$. Since $y_i \in \{1, -1\}$, multiplying by y_i does not change the absolute value, so the left hand side is the distance from the decision boundary. Thus, the above tells us that the margin is at least $\frac{m}{\|w\|_2}$ and that the slab formed by the decision boundary is at least $\frac{2m}{\|w\|_2}$ in width. To maximize the width of this slab $\frac{2}{\|w\|_2}$, we need to minimize $\|w\|_2$. To make our objective function smooth everywhere, we will take our objective function to be $w^T w = \|w\|_2^2$. Thus, our optimization problem is the following:

$$\min_{w,\beta} \|w\|_2^2 \text{ subject to } y_i(w^T x_i + \beta) \geq m, \forall i \in [1, n]$$

This is a quadratic program in $d + 1$ dimensions with n constraints. $d + 1$ comes from the simplification we made in the last note, converting $w^T x_i + \beta$ into $w'^T x_i'$, where $x_i' \in \mathbb{R}^{d+1}$. What happens if our data is not linearly separable, or if our data was corrupted by a misplaced sample point? We next discuss a soft-margin classifier, which tolerates errors but incurs penalties for each error committed. This generalizers better and is not as susceptible to outliers in data.

2 Soft Margin Classifier

For a soft margin classifier, we introduce a slack variable ξ_i. This variable should be 0 for points at least the minimum margin $\frac{m}{\|w\|_2}$ away from the decision boundary. To effect this, we enforce a non-negativity constraint $\xi_i \geq 0$ and include it in the following manner:

$$\frac{y_i}{\|w\|_2}(w^T x_i + \beta) \geq \frac{m - \xi_i}{\|w\|_2}$$

Note that this inequality is equivalent to the following.

$$y_i(w^T x_i + \beta) \geq m - \xi_i$$

In other words, if x_i is classified correctly *and* is more than $\frac{m}{\|w\|_2}$ away from the decision boundary, ξ_i is 0. For x_i that lie on the wrong side of the margin (so any point in the slab, even if it is classified correctly, is included), then ξ_i is greater than 0 in order for the constraint to be satisfied.

We could simply modify the constraint from the quadratic program derived above. However, means the algorithm could simply choose $w = 0$ and let all ξ_i tend to arbitrarily large values. Thus, we penalize all non-zero ξ_i by including it in our objective function.

$$\min_{w,\beta,\xi_i} \|w\|_2^2 + C\sum_{i=1}^{n} \xi_i$$

subject to

$$\forall i \in [1,n], y_i(w^T x_i + \beta) \geq m - \xi_i$$

$$\forall i \in [1,n], \xi_i \geq 0$$

The quadratic program above is in $d+n+1$ dimensions and $2n$ constraints. C above is a **regularization hyperparameter**. Think of C as the penalty you pay for an error. If C approaches infinity, the above optimization problem becomes equivalent to a hard-margin classifier, since all errors are intolerable. The following is a table comparing different values of C, taken verbatim from Professor Jonathan Shewchuk's notes.

	small C	big C
desire	maximize margin	keep most slack variables zero or small
danger	underfitting	overfitting
outliers	less sensitive	very sensitive
boundary	more "flat"	more sinuous (for other, nonlinear decision boundaries)

Notice that the above optimization problem can actually be reformulated, to give us additional intuition. Recall that C is a hyperparameter, $m > 0$ is a constant proportional to the size of our margin.

$$\min_{w} \|w\|_2^2 + C\sum_{i=1}^{n} \max(m - y_i(w^T x_i + \beta), 0)$$

Intuitively, we take the distance from the decision boundary and multiply by y_i so that our new term $y_i(w^T x_i + \beta)$ is negative if x_i is misclassified and positive otherwise. $m - y_i(w^T x_i + \beta)$ is then always positive if x_i is misclassified and is negative if x_i is not more than $\frac{m}{\|w\|_2}$ away from the decision boundary. In short, we are minimizing the sum of all distances between x_i and the wrong side of the margin.

3 Nonlinear Features

Now, we present several conventional approaches to finding nonlinear decision boundaries. In short, we add nonlinear features that **lift** points from $\mathbb{R}^d$ to a higher dimensional space. Let us introduce a more formal notation, namely the **featurization**

$\phi(\cdot)$. Each featurization will take x to some new ϕ space, with the goal of find a linear decision boundary in a higher dimensional space or to find a larger margin. In either case, an increasing number of dimensions corresponds to a higher chance of overfitting.

3.1 Parabolic Lifting Map

Take the original vector $x \in \mathbb{R}^d$, and add an additional entry $|x|^2$ to $x \in \mathbb{R}^{d+1}$. $\phi(x) : \mathbb{R}^d \to \mathbb{R}^{d+1}$.

$$\phi(x) = \begin{bmatrix} x \\ |x|^2 \end{bmatrix}$$

The featurization ϕ encapsulates the notion of lifting x to a paraboloid.

3.2 Axis-Aligned Ellipsoid/Hyperboloid

We extend the previous featurization. Take the original set of features $x \in \mathbb{R}^d$ and square every feature $\phi(x) : \mathbb{R}^d \to \mathbb{R}^{2d}$.

$$\phi(x) = \begin{bmatrix} x_1^2 & \cdots & x_d^2 & x_1 & \cdots & x_d \end{bmatrix}^T$$

Axis-aligned ellipsoids in $\mathbb{R}^n$ follow the general structure $\sum_{i=1}^n a_i x_i^2 + \sum_{i=1}^n b_i x_i + \beta = 0$ for some constant β. (For example, $a_1 x_1^2 + a_2 x_2^2 + b_1 x_1 + b_2 x_2 + \beta = 0$.) Thus, the hyperplane is $w^T \phi(x) + \beta = 0$, where

$$w^T = \begin{bmatrix} a_1 & \cdots & a_n b_1 & \cdots & b_n \end{bmatrix}$$

3.3 Non-Axis-Aligned Ellipsoid/Hyperboloid

We again extend the previous featurization by considered non-axis-aligned quadrics. Now, we consider all possible cross terms, $\phi(x) : \mathbb{R}^d \to \mathbb{R}^{(d^2+3d)/2}$. To be more explicit, we consider $d = 3$.

$$\phi(x) = \begin{bmatrix} x_1^2 & x_2^2 & x_3^2 & x_1 x_2 & x_2 x_3 & x_1 x_3 & x_1 & x_2 & x_3 \end{bmatrix}^T$$

Non-axis-aligned ellipsoids follow the general structure $\sum_{i=1}^n a_i x_i^2 + \sum_{i \neq j} b_{ij} x_i x_j + \sum_{i=1}^n c_i x_i + \beta = 0$ for some constant β. Thus, as before, the hyperplane is defined by $w^T \phi(x) + \beta = 0$, where w contains the coefficient of each term.

3.4 Degree-p Polynomial

Again a generalization of the previous feature, this technique takes an arbitrary degree p polynomial, $\phi(x) = \mathbb{R}^d \to \mathbb{R}^{O(d^p)}$. For example, take $d = 3$ again.

$$\phi(x) = \begin{bmatrix} x_1^3 & x_1^2 x_2 & x_1 x_2^2 & x_2^3 & x_1^2 & x_1 x_2 & x_2^2 & x_1 & x_2 \end{bmatrix}$$

Using this form of featurization, we now have two hyperparameters to tune: p, the degree of the polynomial, and C, the degree of slack.

3.5 Edge Detection

Use an edge detector. The basic premise is to detect discontinuities in brightness, as these may indicate changes in depth, material, or orientation. However, since these discontinuities exist, we need to approximate gradients. Common techniques include the Sobel filter, tap filter, and the oriented Gaussian derivative filter.