# Support Vector Machines

October 4, 2016

# 1 Linear SVM

We have a set of points $(x_i, y_i)$ where each $x_i$ is a vector and each $y_i$ is either -1 or 1.

## 1.1 Hard Margin

- First consider the line $L = \{y = mx\}$.
- What is the distance from a point $p_0 = (x_0, y_0)$ to this line?
- As we know from high school the answer is $p_0 \cdot n$ where $n$ is the unit normal of the line.
- The equation $n \cdot p_0 + \tilde{b} = 0$ defines the set of points lying on another line with gradient $m$, but with a different intercept.
- In fact, for the line (defined by) $y = mx + \tilde{b}$, then the points lying on the line is defined by $\left(p_0 - \tilde{b}\right) \cdot n = 0$, ie $p_0 \cdot n - \tilde{b} \cdot n = 0$, and we can denote $b = \tilde{b} \cdot n$.

### 1.1.1 The Optimization Problem

- Equivalently this can be written

  $$\frac{1}{\|w\|}\left(w \cdot p_0 + \frac{b}{\|w\|}\right) = 0$$

  We need to find $w$ and $b$ to maximize the "margin", in this case $1/\|w\|$.
- This is basically saying find $w$ and $b$ such that $w \cdot x_i + b \geq 1$ if $y_i = 1$, and $w \cdot x_i + b \leq -1$ if $y_i = -1$, and such that $\|w\|$ is minimized.
- Ie, minimize $\|w\|$ such that

  $$\operatorname{sgn}(w \cdot x_i + b) = y_i$$

  for all $i$.

- The point is that getting $n$ and $\tilde{b}$ is equivalent to specifying a hyperplane. The margin is then

$$\frac{1}{\|w\|} = \min_i \left(n \cdot x_i - \tilde{b}\right)$$

The good part about $w$ is that we can get rid of more unknowns in the constraint.

## 1.2 Soft Margin

- If the classification is wrong, we need a measure of how wrong it is.
- This can be given by the hinge loss

$$L(x,y) = \max(0, 1 - (w \cdot x + b)\, y)$$

Now minimize the average hinge loss.

- Remark: This is conceptually similar not entirely equivalent to maximizing

$$(w \cdot x + b)\, y$$

which is what was done at [1].

### 1.2.1 Regularization

- Have a parameter $\lambda$ that regulates the tradeoff between the margin and the hinge loss.
- Ie between getting more points right and getting wrong points less wrong.
- Now minimize

$$\frac{1}{n}\sum_i^n L(x_i, y_i) + \lambda \|w\|$$

# 2 General Kernels

What's actually happening here?

- We're basically picking a function $f$ so that the predictions are given by

$$\begin{aligned}
\mathrm{pred}(x) &= \mathrm{sgn}(f(x) + b) \\
f(x) &= w \cdot x \\
&= \sum_j w_j x_j
\end{aligned}$$

- Of course, we pick $f$ to minimize the hinge loss

$$\begin{aligned}
L(x_i, y_i) &= \max(1 - y_i (f(x_i) + b), 0) \\
\bar{L} &= \frac{1}{N}\sum_{i=1}^N L(x_i, y_i)
\end{aligned}$$

## 2.1 RBF Kernel

This is where it gets more interesting: we can pick something else for $f$

$$f(x) = \sum_i^n w_i y_i \exp\left(-4\|x_i - x\|^2\right)$$

- Think of this as just superimposing a Gaussian hump on top of every point in the training dataset, pointing up or down depending on $y_i$.
- Then to classify a point, just look at whether the humps sum to something positive at that location.

## 2.2 General

In fact we actually pick anything for $f$, as long as it's of the form

$$f(x) = \sum_i \alpha_i y_i K(x_i, x)$$

- We require that $K$ be positive-semidefinite. (otherwise there might be more than one solution).
- Without the positive semidefinite property all of these optimization problems would be able to "run to negative infinity" or use negative terms [1].
- Remark: the $\alpha_i$ are positive, otherwise having $y_i$ would be redundant (it would also imply we could flip the sign of $K$ with impunity, which is false).

## 2.3 Recasting the Linear Kernel

To compare the above to the linear kernel, write

$$\begin{aligned} K(x_i, x) &= \langle x_i, x\rangle = \sum_j^n x_{ij} x_j \\ f(x) &= \sum_i \alpha_i y_i \sum_j^n x_{ij} x_j \\ &= \sum_j^n \left(\sum_i \alpha_i y_i x_{ij}\right) x_j \end{aligned}$$

So that

$$\begin{aligned} w_j &= \sum_i \alpha_i y_i x_{ij} \\ w &= \sum_i \alpha_i y_i x_i \end{aligned}$$

- Ie we have a linear combination of dot prducts which will reduce to just a dot product in the end.

- Graphically, think of it like this: the 3d plot for $x \cdot y$ is just a tilted plane that passes through the origin. Superimposing such planes results in just another such plane. Eventually you just get a plane, and this is what you're testing is greater or less than zero.

### 2.3.1 Support Vectors

Are those for which $w_i \neq 0$. In the linear case there are two points $p_1$ and $p_2$ such that $p_1 - p_2$ is parallel to the dividing line. But this isn't necessarily the only way.

## 2.4 Higher Dimensional Projection

Abstractly, the positive-definiteness can be captured by setting $K(x_1, x_2) = \langle \phi(x_1), \phi(x_2) \rangle$ where $\phi : V \to W$ is some map between inner product spaces.

For example

$$\phi : (x,y) \mapsto \left(x, y, x^2 + y^2\right)$$

So that

$$\begin{aligned} K(x,y) &= \phi(x) \cdot \phi(y) \\ &= x \cdot y + \left(x_1^2 + x_2^2\right)\left(y_1^2 + y_2^2\right) \end{aligned}$$

- Note that this doesn't imply conjugate symmetry or sesquilinearity at all, since $\phi$ can be anything, perhaps even discontinuous.
- The whole projection to a higher dimensional space metaphor/interpretation really isn't all that useful. It's really trying to apply the intuition of the linear kernel to general kernels, when the other way around is much more natural.

## 2.5 Relationship to kNN

- The SVM is actually the same as weighted kNN with $k$ infinite, except that in this case $\alpha_i = 1$ for all $i$.
- So an SVM is like a weighted kNN on steroids.
- Note that the rbf kernel is not separable, ie cannot be written in the form

$$\exp\left(-4\left\|x_i - x\right\|^2\right) = f(x_i)\, g(x)$$

## 2.6 Relationship to Logistic Regression

- Remember that

$$\begin{aligned} \log\left(\frac{p}{1-p}\right) &= \sum_i w_i x_i \\ &= w \cdot x \end{aligned}$$

and

$$\text{pred}(x) = \text{sgn}(w \cdot x + \beta)$$

where $\beta$ is some function of the probability threshold.

- So you're optimizing $w$ again, which is the vector of coefficients.
- What we have is a linear SVM where you're maximizing the log likelihood instead of the margin (or minimizing the hinge loss).

# References


[1] http://www.win-vector.com/blog/2011/10/kernel-methods-and-support-vector-machines-de-mystified/

[2]