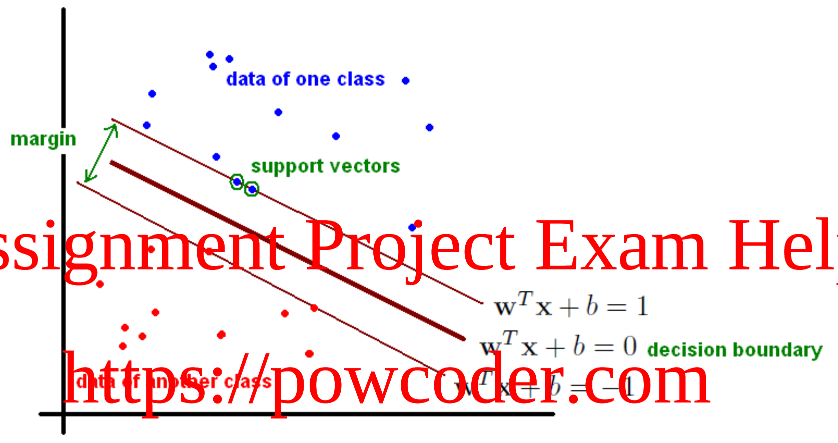
SVM Tutorial

Zoya Gavrilov

Just the basics with a little bit of spoon-feeding...


data of one class
margin
support vectors
$\mathbf{w}^T\mathbf{x} + b = 1$
$\mathbf{w}^T\mathbf{x} + b = 0$ decision boundary
data of another class
$\mathbf{w}^T\mathbf{x} + b = -1$

1 Simplest case: linearly-separable data, binary classification

Goal: we want to find the hyperplane (i.e. decision boundary) linearly separating our classes. Our boundary will have equation: $\mathbf{w}^T\mathbf{x} + b = 0$.

Anything above the decision boundary should have label 1.
i.e., $\mathbf{x_i}$ s.t. $\mathbf{w}^T\mathbf{x_i} + b > 0$ will have corresponding $y_i = 1$.
Similarly, anything below the decision boundary should have label -1.
i.e., $\mathbf{x_i}$ s.t. $\mathbf{w}^T\mathbf{x_i} + b < 0$ will have corresponding $y_i = -1$.

The reason for this labelling scheme is that it lets us condense the formulation for the decision function to $f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b)$ since $f(\mathbf{x}) = +1$ for all $\mathbf{x}$ above the boundary, and $f(\mathbf{x}) = -1$ for all $\mathbf{x}$ below the boundary.

Thus, we can figure out if an instance has been classified properly by checking that $y(\mathbf{w}^T\mathbf{x}+b) \geq 1$ (which will be the case as long as either both $y, \mathbf{w}^T\mathbf{x}+b > 0$ or else $y, \mathbf{w}^T\mathbf{x} + b < 0$).

You'll notice that we will now have some space between our decision boundary and the nearest data points of either class. Thus, let's rescale the data such that anything on or above the boundary $\mathbf{w}^T\mathbf{x} + b = 1$ is of one class (with label 1), and anything on or below the boundary $\mathbf{w}^T\mathbf{x} + b = -1$ is of the other class (with label -1).

What is the distance between these newly added boundaries?
First note that the two lines are parallel, and thus share their parameters $\mathbf{w}, b$.
Pick an arbirary point $\mathbf{x_1}$ to lie on line $\mathbf{w}^T\mathbf{x} + b = -1$. Then, the closest point on line $\mathbf{w}^T\mathbf{x} + b = 1$ to $\mathbf{x_1}$ is the point $\mathbf{x_2} = \mathbf{x_1} + \lambda\mathbf{w}$ (since the closest point will always lie on the perpendicular; recall that the vector $\mathbf{w}$ is perpendicular to both lines). Using this formulation, $\lambda\mathbf{w}$ will be the line segment connecting $\mathbf{x_1}$ and $\mathbf{x_2}$, and thus, $\lambda\|\mathbf{w}\|$, the distance between $\mathbf{x_1}$ and $\mathbf{x_2}$, is the shortest distance between the two lines/boundaries. Solving for λ:

$\Rightarrow \mathbf{w}^T\mathbf{x_2} + b = 1$ where $\mathbf{x_2} = \mathbf{x_1} + \lambda\mathbf{w}$
$\Rightarrow \mathbf{w}^T(\mathbf{x_1} + \lambda\mathbf{w}) + b = 1$
$\Rightarrow \mathbf{w}^T\mathbf{x_1} + b + \lambda\mathbf{w}^T\mathbf{w} = 1$ where $\mathbf{w}^T\mathbf{x_1} + b = -1$
$\Rightarrow -1 + \lambda\mathbf{w}^T\mathbf{w} = 1$
$\Rightarrow \lambda\mathbf{w}^T\mathbf{w} = 2$
$\Rightarrow \lambda = \frac{2}{\mathbf{w}^T\mathbf{w}} = \frac{2}{\|\mathbf{w}\|^2}$
And so, the distance $\lambda\|\mathbf{w}\|$ is $\frac{2}{\|\mathbf{w}\|^2}\|\mathbf{w}\| = \frac{2}{\|\mathbf{w}\|} = \frac{2}{\sqrt{\mathbf{w}^T\mathbf{w}}}$
It's intuitive that we would want to maximize the distance between the two

boundaries demarcating the classes (Why? We want to be as sure as possible that we are not making classification mistakes, and thus we want our data points from the two classes to lie as far away from each other as possible). This distance is called the margin, so what we want to do is to obtain the *maximal margin*.

Thus, we want to maximize $\frac{2}{\sqrt{\mathbf{w^T w}}}$, which is equivalent to minimizing $\frac{\sqrt{\mathbf{w^T w}}}{2}$, which is in turn equivalent to minimizing $\frac{\mathbf{w^T w}}{2}$ (since square root is a monotonic function).

This *quadratic programming* problem is expressed as:
$min_{\mathbf{w},b} \frac{\mathbf{w^T w}}{2}$
subject to: $y_i(\mathbf{w}^T \mathbf{x_i} + b) \geq 1$ ($\forall$ data points $\mathbf{x_i}$).

2 Soft-margin extention

Consider the case that your data isn't perfectly linearly separable. For instance, maybe you aren't guaranteed that all your data points are correctly labelled, so you want to allow some data points of one class to appear on the other side of the boundary.

We can introduce *slack variables* - an $\epsilon_i \geq 0$ for each $\mathbf{x_i}$. Our quadratic programming problem becomes:
$min_{\mathbf{w},b,\epsilon} \frac{\mathbf{w^T w}}{2} + C \sum_i \epsilon_i$
subject to: $y_i(\mathbf{w}^T \mathbf{x_i} + b) \geq 1 - \epsilon_i$ and $\epsilon_i \geq 0$ ($\forall$ data points $\mathbf{x_i}$).

3 Nonlinear decision boundary

Mapping your data vectors, $\mathbf{x_i}$, into a higher-dimension (even infinite) feature space may make them linearly separable in that space (whereas they may not be linearly separable in the original space). The formulation of the quadratic programming problem is as above, but with all $\mathbf{x_i}$ replaced with $\phi(\mathbf{x_i})$, where ϕ provides the higher-dimensional mapping. So we have the standard SVM formulation:

$min_{\mathbf{w},b,\epsilon} \frac{\mathbf{w^T w}}{2} + C \sum_i \epsilon_i$

subject to: $y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b) \geq 1 - \epsilon_i$ and $\epsilon_i \geq 0$ ($\forall$ data points $\mathbf{x_i}$).

4 Reformulating as a Lagrangian

We can introduce Lagrange multipliers to represent the condition:

$y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b)$ must be as close to 1 as possible.

This condition is captured by: $max_{\alpha_i \geq 0} \alpha_i [1 - y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b)]$

This ensures that when $y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b) \geq 1$, the expression above is maximal when $\alpha_i = 0$ (since $[1 - y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b)]$ ends up being negative). Otherwise, $y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b) < 1$, so $[1 - y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b)]$ is a positive value, and the expression is maximal when $\alpha_i \to \infty$. This has the effect of penalizing any misclassified data points, while assigning 0 penalty to properly classified instances.

We thus have the following formulation:

$min_{\mathbf{w},b}[\frac{\mathbf{w^T w}}{2} + \sum_i max_{\alpha_i \geq 0} \alpha_i [1 - y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b)]]$

To allow for slack (soft-margin), preventing the α variables from going to ∞, we can impose constraints on the Lagrange multipliers to lie within: $0 \leq \alpha_i \leq C$.

We can define the *dual problem* by interchanging the max and min as follows (i.e. we minimize after fixing alpha):

$max_{\alpha \geq 0}[min_{\mathbf{w},b} J(\mathbf{w}, b; \alpha)]$ where $J(\mathbf{w}, b; \alpha) = \frac{\mathbf{w^T w}}{2} + \sum_i \alpha_i [1 - y_i(\mathbf{w}^T \phi(\mathbf{x_i}) + b)]$

Since we're solving an optimization problem, we set $\frac{\partial J}{\partial \mathbf{w}} = 0$ and discover that the optimal setting of $\mathbf{w}$ is $\sum_i \alpha_i y_i \phi(x_i)$, while setting $\frac{\partial J}{\partial b} = 0$ yields the constraint $\sum_i \alpha_i y_i = 0$.

Thus, after substituting and simplifying, we get:

$min_{\mathbf{w},b} J(\mathbf{w}, b; \alpha) = \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})$

And thus our dual is:

$max_{\alpha \geq 0}[\sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(\mathbf{x_i})^T \phi(\mathbf{x_j})]$

Subject to: $\sum_i \alpha_i y_i = 0$ and $0 \leq \alpha_i \leq C$

5 Kernel Trick

Because we're working in a higher-dimension space (and potentially even an infinite-dimensional space), calculating $\phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$ may be intractable. However, it turns out that there are special *kernel* functions that operate on the lower dimension vectors $\mathbf{x_i}$ and $\mathbf{x_j}$ to produce a value equivalent to the dot-product of the higher-dimensional vectors.

For instance, consider the function $\phi : \mathbb{R}^3 \mapsto \mathbb{R}^{10}$, where:
$\phi(\mathbf{x}) = (1, \sqrt{2}\mathbf{x}^{(1)}, \sqrt{2}\mathbf{x}^{(2)}, \sqrt{2}\mathbf{x}^{(3)}, [\mathbf{x}^{(1)}]^2, [\mathbf{x}^{(2)}]^2, [\mathbf{x}^{(3)}]^2, \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(2)}, \sqrt{2}\mathbf{x}^{(1)}\mathbf{x}^{(3)}, \sqrt{2}\mathbf{x}^{(2)}\mathbf{x}^{(3)})$
Instead, we have the *kernel trick*, which tells us that $K(\mathbf{x_i}, \mathbf{x_j}) = (1 + \mathbf{x_i}^T\mathbf{x_j})^2 = \phi(\mathbf{x_i})^T\phi(\mathbf{x_j})$ for the given ϕ. Thus, we can simplify our calculations.

Re-writing the dual in terms of the kernel, yields:
$max_{\alpha \geq 0}[\sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j K(\mathbf{x_i}, \mathbf{x_j})]$

6 Decision function

To classify a novel instance $\mathbf{x}$ once you've learned the optimal α_i parameters, all you have to do is calculate $f(\mathbf{x}) = sign(\mathbf{w}^T\mathbf{x} + b) = \sum_i \alpha_i y_i K(\mathbf{x_i}, \mathbf{x}) + b$
(by setting $\mathbf{w} = \sum_i \alpha_i y_i \phi(\mathbf{x_i})$ and using the kernel trick).
Note that α_i is only non-zero for instances $\phi(\mathbf{x_i})$ on or near the boundary - those are called the *support vectors* since they alone specify the decision boundary. We can toss out the other data points once training is complete. Thus, we only sum over the $\mathbf{x_i}$ which constitute the support vectors.

7 Learn more

A great resource is the MLSS 2006 (Machine Learning Summer School) talk by Chih-Jen Lin, available at: videolectures.net/mlss06tw_lin_svm/. Make sure to download his presentation slides. Starting with the basics, the talk continues on to discussing practical issues, parameter/kernel selection, as well as more advanced topics.

The typical reference for such matters: Pattern Recognition and Machine Learning by Christopher M. Bishop.