Support Vector Machine (SVM) Derivations

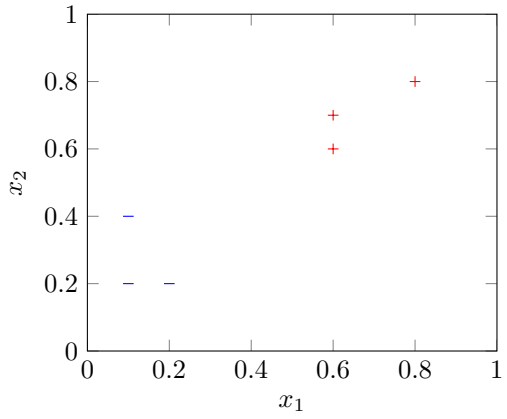
Mickey Chao

January 2016

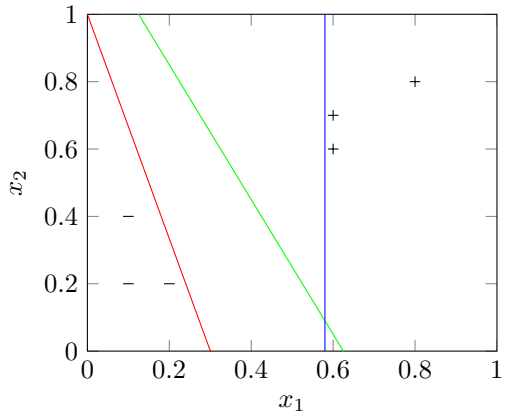
1 Intuition Behind SVMs

Machine Learning often involves drawing boundaries (classifications). There are many methods of producing boundaries: the Perceptron method, applying linear or logistic regression, neural networks, and many more.

SVMs follow the idea of separating data by producing a linear classifier and drawing a line between them. SVMs address the difficulties of selecting a classifier when many lines can be drawn. Suppose you have some datapoints that are linearly separable and look like this:



There are many different boundaries that could be drawn, as shown in the following picture:



The question is which boundary is the best? The blue and red boundaries are not good because they are too close to either the negative or positive examples. We will eventually use the test set to measure the performance of our boundary. If the test set is noisy, the red and blue classifiers will likely misclassify something near the linear boundary.

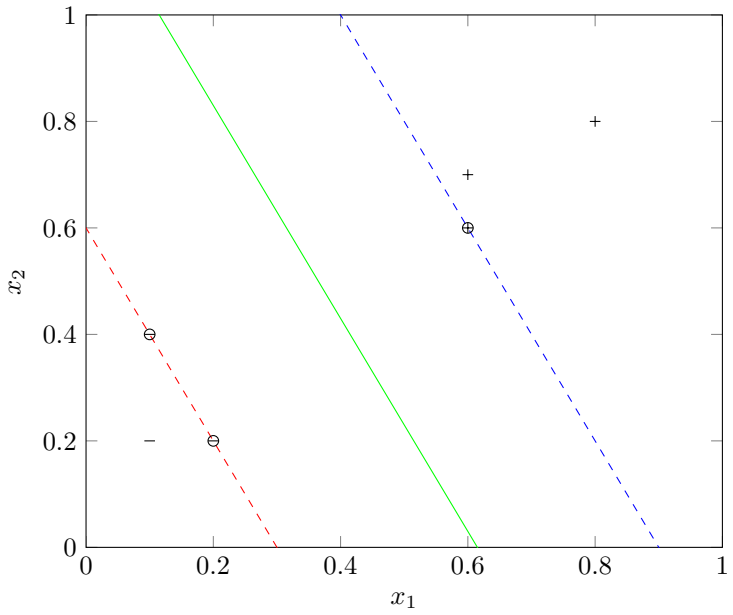
Now consider the green boundary. It is in the middle of a wide margin that separates the positive and negative examples. When there is noise, the green boundary can accommodate more noise than the red and blue boundaries. Thus, we might say the green boundary is the most robust because it has the widest margin and allows the most room for error.

The idea behind SVMs is to maximize this margin because a linear classifier with a wide margin generalizes better than a linear classifier with a narrow margin.

2 Maximization Objective

Now, we will attempt to formalize the objective of "maximizing the margin" in mathematical terms.

Suppose we have a boundary defined as $\bar{\theta} \cdot \bar{x} + b = 0$ and we have found the points closest to it. Here is an example:



The decision boundary is the green line. The boundaries of the margin are the dotted red and blue line. The circled points are the points closest to the decision boundary. We call these circled points "support vectors."

Now, suppose $\bar{x}^{(i)}$ is a point on the margin boundary. Then we have

$$\text{proj}_\theta\left(\bar{x}^{(i)}\right) = \frac{\bar{x}^{(i)} \cdot \bar{\theta}}{\|\bar{x}^{(i)}\|\|\bar{\theta}\|}\bar{x}^{(i)} \quad \text{and} \quad \|\text{proj}_\theta\|\left(\bar{x}^{(i)}\right) = \frac{\bar{x}^{(i)} \cdot \bar{\theta}}{\|\bar{\theta}\|}$$

$\|\text{proj}_\theta(\bar{x}^{(i)})\|$ is the signed distance of $\bar{x}^{(i)}$ from the boundary $\bar{\theta} \cdot \bar{x} = 0$. If $\bar{x}^{(i)}$ is a negative datapoint, this value will be negative and if $\bar{x}^{(i)}$ is a positive datapoint, this value will be positive.

By similar calculations using projections, the distance of $\bar{\theta} \cdot \bar{x} + b = 0$ to $\bar{\theta} \cdot \bar{x} = 0$ is $\frac{b}{\|\bar{\theta}\|}$.

Thus, the unsigned distance of $\bar{x}^{(i)}$ from the decision boundary is

$$\text{margin size} = \frac{(\bar{\theta} \cdot \bar{x}^{(i)} + b)y^{(i)}}{\|\bar{\theta}\|}$$

For mathematical convenience, let us define $\bar{\theta} \cdot \bar{x} + b = 1$ and $\bar{\theta} \cdot \bar{x} + b = -1$ to be the positive and negative boundaries of the margin.This is okay because we can always rescale $\bar{\theta}$ to anything we want, so the size of the margin still remains generalized. Now, the width of the margin becomes

$$\text{margin size} = \frac{(\bar{\theta} \cdot \bar{x}^{(1)} + b)y^{(i)}}{\|\bar{\theta}\|} = \frac{1}{\|\bar{\theta}\|}$$

so we want to maximize $\frac{1}{\bar{\theta}}$. This is also equivalent to minimizing $\frac{1}{2}\|\bar{\theta}\|^2$, which is just the scaled denominator.

Recall that the data are linearly separable. We want to make sure that every datapoint is on the correct side of the classifier, so we must also guarantee that $y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1$ while optimizing our minimization objective.

Thus, our minimization objective is as follows:

$$\min_{\bar{\theta}} \frac{1}{2}\|\bar{\theta}\|^2 \quad \text{subject to} \quad y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1$$

3 Minimizing the Objective Function with Lagrange Multipliers

In the SVM problem, we wish to find $\bar{\theta}$ satisfying

$$\min_{\theta} \frac{1}{2}\|\bar{\theta}\|^2 \quad \text{subject to} \quad y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1$$

We can apply method of Lagrange Multipliers to solve this optimization objective. Lagrange Multipliers solve optimization problems of the following form:

"Minimize $f(x_1, x_2, \ldots, x_n)$ subject to $g_k(x_1, x_2, \ldots, x_n) = 0$" where the g_k are constraints.

The idea behind Lagrange Multipliers is that our optimal solution occurs when the the gradient of our minimization objective function and the constraint functions are all 0. That is

$$\nabla f = 0 = \alpha_k g_k \quad \text{for all } k$$

where α_k is a constant called a Lagrange Multiplier.

This can be more concisely represented in an auxilary function

$$\mathcal{L}(x_1, x_2, \ldots, x_n, \alpha_1, \alpha_2, \ldots, \alpha_n) = f(x_1, x_2, \ldots, x_n) + \sum_k \alpha_k g_k(x_1, x_2, \ldots, x_n)$$

where we wish to solve for $\nabla \mathcal{L} = 0$

In our minimization problem for SVMs, we can rewrite all our constraints to be of the form

$$1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \leq 0$$

This basically says the distance of all points from the decision boundary is at least 1. In other words, these constraints enforce the margin size of 1. Note that

$$1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) = 0$$

only when $\bar{x}^{(i)}$ is a support vector. Thus, only when $\bar{x}^{(i)}$ is a support vector, we get

$$1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) = 0$$

If $\bar{x}^{(i)}$ is not a support vector, we don't care. If we satisfy the condition $1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \leq 0$ for all the support vectors, we satisfy the same condition for all the datapoints because all the datapoints are at least as far as the support vectors from the decision boundary. Therefore, we can drop the constraints for any datapoints $\bar{x}^{(i)}$ that are not support vectors by setting $\alpha_i = 0$.

Our auxilary function is

$$\mathcal{L}(\bar{\theta}, \bar{\alpha}, b) = \frac{1}{2}\|\bar{\theta}\|^2 + \sum_{i=1}^{n} \alpha_i \left(1 - y^{(i)}(\bar{\theta} \cdot \bar{x} + b)\right) \tag{1}$$

where $\alpha_i = 0$ if $x^{(i)}$ is not a support vector.

To find the gradient of $\mathcal{L}$ and set it to 0, we take the partial derivatives of $\mathcal{L}$ with respect to $\bar{\theta}$ and b:

$$\frac{\partial \mathcal{L}}{\partial \bar{\theta}} = 0 = \bar{\theta} - \sum_{i=1}^{n} \alpha_i y^{(i)} \cdot \bar{x}^{(i)} \tag{2}$$

$$\frac{\partial \mathcal{L}}{\partial b} = 0 = \sum_{i=1}^{n} \alpha_i y^{(i)} \tag{3}$$

Rearranging (2) gives us

$$\bar{\theta}^* = \sum_{i=1}^{n} \alpha_i y^{(i)} \cdot \bar{x} \tag{4}$$

Next, we can substitute (3) back into (1) and rewrite out auxiliary function:

$$\mathcal{L}(\bar{\theta}, \bar{\alpha}, b) = \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y^{(i)} \bar{x}^{(i)} \right)^T \left(\sum_{j=1}^{n} \alpha_j y^{(j)} \bar{x}^{(j)} \right) + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \left(y^{(i)} \left(\sum_{j=1}^{n} \alpha_j y^{(j)} \bar{x}^{(j)} \right) \cdot \bar{x}^{(i)} \right) + \sum_{i=1}^{n} \alpha_i y^{(i)} b$$

We know that $\sum_{i=1}^{n} \alpha_i y^{(i)} = 0$, so the last term disappears and we get

$$\mathcal{L}(\bar{\theta}, \bar{\alpha}, b) = \frac{1}{2} \left(\sum_{i=1}^{n} \alpha_i y^{(i)} \bar{x}^{(i)} \right)^T \left(\sum_{j=1}^{n} \alpha_j y^{(j)} \bar{x}^{(j)} \right) + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \left(y^{(i)} \left(\sum_{j=1}^{n} \alpha_j y^{(j)} \bar{x}^{(j)} \right) \cdot \bar{x}^{(i)} \right)$$

Expanding the the products of the summations yields

$$\mathcal{L}(\bar{\theta}, \bar{\alpha}, b) = \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} \right) + \sum_{i=1}^{n} \alpha_i - \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} \right)$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} \right)$$

where $\mathcal{L}$ remains subject to

$$\alpha_i \geq 0 \quad , \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

Finally, we ask a quadratic programming algorithm to find

$$\max_{\bar{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} \right) \quad \text{subject to} \quad \alpha_i \geq 0 \quad , \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

Recall that all the support vectors will have nonzero α and all the non-support vectors must have $\alpha = 0$. Therefore, given the optimal $\hat{\alpha}$ from the quadratic programming algorithm, we can determine the support vectors and non-support vectors.

Once we identify all the support vectors, $\bar{x}^{sv}$, we can calculate $\bar{\theta}^*$ by applying (4):

$$\bar{\theta}^* = \sum_{i=0}^{n} \hat{\alpha}_i y^{(i)} \bar{x}^i$$

Note also that since $\alpha_i = 0$ for all non-support vectors, we're essentially just taking a linear combination of the support vectors.

Finally, to calculate our value of b, we pick any support vector $\bar{x}^{sv}$ and solve for b in the equation

$$y^{(i)} \left(\bar{\theta}^* \bar{x}^{sv} + b \right) = 1$$

If we were worried about floating point imprecisions, we could solve this for multiple support vectors $\bar{x}^{sv}$ and take the median.

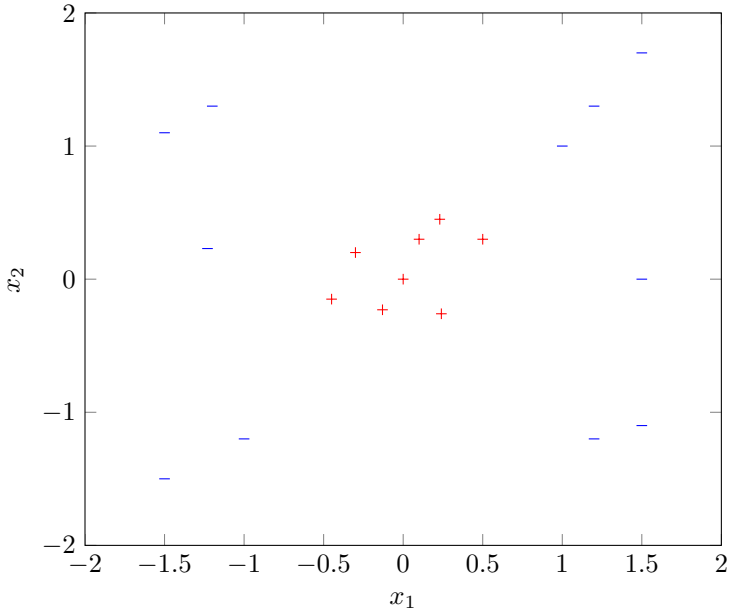
We have now gone through all the steps to calculate the widest margin classifier for some linearly separable data.

4 Kernels

In this section, we discuss how to apply SVMs to producing nonlinear decision boundaries.

4.1 Feature Space Transformations

Consider training a classifier on the following training data:



There is no line that can separate the positive from the negative, but it should be easy to see how they can be separated with a nonlinear classifier - most likely by a circle of radius 1 about the origin.

As was done with logistic regression or linear regression, we could just add more complex features such as x_1^2, x_1x_2 and x_2^2 and run our SVM algorithm on that. It would certainly produce a valid boundary. In particular, we could define a function

$$\Phi : \mathbb{R}^d \to \mathbb{R}^n$$

where $n > d$ that maps features in the original d-dimensional space to a more complex space with n-dimensions. For example, in the above example, we could define Φ to map training data in the space spanned by $\left[1, x_1, x_2\right]^T$ to the space spanned by $\left[1, x_1, x_2, x_1^2, x_2^2, x_1x_2\right]^T$

The problem is that when we end up working with hundreds of thousands of features, it becomes computationally infeasible to produce a feature mapping that would map the original 100,000 dimensions to a new 2^{100000} dimensional space that considers all possible combinations of features. In addition, if we could even have 2^{100000} features, we'd probably overfit the data terribly. We will address the problem with efficiency now, and in the following section, we will address the problem with overfitting.

4.2 Making Use of the Dot Product

Previously, we derived a maximization problem to be passed into a quadratic programming algorithm:

$$\max_{\bar{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\left(\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i\alpha_j y^{(i)}y^{(j)}\bar{x}^{(i)} \cdot \bar{x}^{(j)}\right) \quad \text{subject to} \quad \alpha_i \geq 0 \quad , \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

Notice that the this maximization objective depends on a dot product between two pieces of training data, but not the actual training data $\bar{x}^{(i)}$ and $\bar{x}^{(j)}$. In the computationally infeasible scenario where we apply a

feature mapping Φ, our maximization problem would be

$$\max_{\bar{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \Phi(\bar{x}^{(i)}) \cdot \Phi(\bar{x}^{(j)}) \right) \quad \text{subject to} \quad \alpha_i \geq 0 \quad , \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

We can now replace the dot product $\Phi(\bar{x}^{(i)}) \cdot \Phi(\bar{x}^{(j)})$ with a new function, $K(\bar{x}^{(i)}, \bar{x}^{(j)})$, which we call a kernel function. If we can find an easy-to-compute kernel function $K(\bar{x}^{(i)}, \bar{x}^{(j)})$, we can avoid computing a the feature mapping Φ and thus make our calculations computationally feasible again. However, one important caveat to using applying kernel functions is that the kernel functions must be valid. That is, there must exist a feature mapping $\Phi(\bar{x}^{(i)}, \bar{x}^{(j)})$ such that

$$K(\bar{x}^{(i)}, \bar{x}^{(j)}) = \Phi(\bar{x}^{(i)}) \cdot \Phi(\bar{x}^{(j)})$$

Given that we use valid kernels, our new maximization problem becomes

$$\max_{\bar{\alpha}} \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \left(\sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} K\left(\bar{x}^{(i)}, \bar{x}^{(j)}\right) \right) \quad \text{subject to} \quad \alpha_i \geq 0 \quad , \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

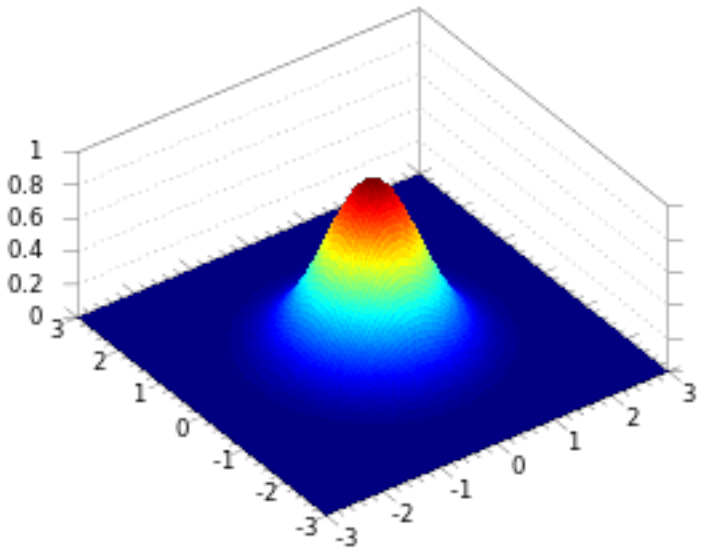
and the solution to the optimization problem depends solely on our choice of the function to use for K.

4.3 The Radial Basis Function (Gaussian) Kernel

Now, we will demonstrate how we can use a powerful kernel known as the Radial Basis Function kernel (RBF kernel). The Radial Basis Function kernel is defined as follows:

$$K_{RBF}(\bar{x}^{(i)}, \bar{x}^{(j)}) = e^{-\frac{(\|\bar{x}^{(i)} - \bar{x}^{(j)}\|)^2}{2\sigma^2}}$$

This is what the graph of K_{RBF} looks like for 2 dimensional vectors $\bar{x}^{(i)}$ and $\bar{x}^{(j)} = \vec{0}$ and $2\sigma^2 = 1$:



The x and y axes (spanning the blue plane on the bottom) represent the values $\bar{x}_1^{(i)} - \bar{x}_1^{(j)}$ and $\bar{x}_2^{(i)} - \bar{x}_2^{(j)}$. The z axis represents the output of the function K_{RBF}.

From the plot, we notice that the further away $\bar{x}^{(i)}$ goes from $\bar{x}^{(j)}$, the lower the value of K_{RBF}. The closer $\bar{x}^{(i)}$ gets to $\bar{x}^{(j)}$, the higher the value of K_{RBF}. This seems like a promising kernel function as it

mirrors a property of dot products: two similar vectors yield a high output and two dissimilar vectors yield a low output.

We will now derive the feature mapping for K_{RBF} which will show that it is a valid kernel. Let $\gamma = \frac{1}{2\sigma^2}$. If we expand the exponent of e in K_{RBF} we get

$$\begin{aligned} K_{RBF} &= e^{-\gamma(\|\bar{x}^{(i)} - \bar{x}^{(j)}\|)^2} \\ &= e^{-\gamma(\bar{x}^{(i)} - \bar{x}^{(j)})^T(x^{(i)} - \bar{x}^{(j)})} \\ &= e^{-\gamma\|\bar{x}^{(i)}\|^2 + 2\gamma\left(\bar{x}^{(i)}\right)^T\bar{x}^{(j)} - \gamma\|\bar{x}^{(j)}\|^2} \\ &= e^{-\gamma\|\bar{x}^{(i)}\|^2} \cdot e^{-\gamma\|\bar{x}^{j}\|^2} \cdot e^{2\gamma(\bar{x}^{(i)})^T\bar{x}^{(j)}} \end{aligned}$$

If the apply the Taylor series expansion to $e^{2\gamma(\bar{x}^{(i)})^T\bar{x}^{(j)}}$ then we get

$$K_{RBF} = e^{-\gamma\|\bar{x}^{(i)}\|^2} \cdot e^{-\gamma\|\bar{x}^{(j)}\|^2} \cdot \sum_{n=0}^{\infty}\left(\frac{(2\gamma(\bar{x}^{(i)})^T\bar{x}^{(j)})^n}{n!}\right)$$

To express K_{RBF} in the form $\Phi(\bar{x}^{(i)}) \cdot \Phi(\bar{x}^{(j)})$, we will want to group all the $\bar{x}^{(i)}$ terms together and all the $\bar{x}^{(j)}$ terms together. Therefore, we will multiply $e^{-\gamma\|\bar{x}^{(i)}\|^2} \cdot e^{-\gamma\|\bar{x}^{(j)}\|^2}$ into the summation:

$$K_{RBF} = \sum_{n=0}^{\infty}\left(e^{-\gamma\|\bar{x}^{(i)}\|^2}\left(\bar{x}^{(i)}\right)^n \cdot \frac{(2\gamma)^n}{n!} \cdot e^{-\gamma\|\bar{x}^{(j)}\|^2}\left(\bar{x}^{(j)}\right)^n\right)$$

We can split split $\frac{(2\gamma)^n}{n!}$ into two pieces by taking the square root and our summation becomes:

$$K_{RBF} = \sum_{n=0}^{\infty}\left(\left(\sqrt{\frac{(2\gamma)^n}{n!}}e^{-\gamma\|\bar{x}^{(i)}\|^2}\left(\bar{x}^{(i)}\right)^n\right) \cdot \left(\sqrt{\frac{(2\gamma)^n}{n!}}e^{-\gamma\|\bar{x}^{(j)}\|^2}\left(\bar{x}^{(j)}\right)^n\right)\right)$$

Now it should be clear how the feature mapping is defined. We have

$$\Phi(\bar{x}^{(i)}) \cdot \Phi(\bar{x}^{(j)}) = K_{RBF}(\bar{x}^{(i)}, \bar{x}^{(j)}) = \begin{bmatrix} \left(\sqrt{\frac{(2\gamma)^0}{0!}}e^{-\gamma\|\bar{x}^{(i)}\|^2}\left(\bar{x}^{(i)}\right)^0\right) \\ \left(\sqrt{\frac{(2\gamma)^1}{1!}}e^{-\gamma\|\bar{x}^{(i)}\|^2}\left(\bar{x}^{(i)}\right)^1\right) \\ \vdots \end{bmatrix} \cdot \begin{bmatrix} \left(\sqrt{\frac{(2\gamma)^0}{0!}}e^{-\gamma\|\bar{x}^{(j)}\|^2}\left(\bar{x}^{(i)}\right)^0\right) \\ \left(\sqrt{\frac{(2\gamma)^1}{1!}}e^{-\gamma\|\bar{x}^{(j)}\|^2}\left(\bar{x}^{(i)}\right)^1\right) \\ \vdots \end{bmatrix}$$

It is also important to notice that the exponent of the $\bar{x}^{i)}$ and $\bar{x}^{(j)}$ goes from $n = 0$ to ∞. This means that the feature mapping Φ creates an infinite dimensional vector. In fact, $e^{-\gamma\|x\|^2}\{1, \sqrt{\frac{2\gamma}{1!}}x, \sqrt{\frac{(2\gamma)^2}{2!}}x^2, \ldots\}$ is a basis for an infinite-dimensional feature space to which Φ maps. This means that the radial basis function kernel implicitly specifies an infinite dimensional mapping and is capable of producing any boundary and separate any data.

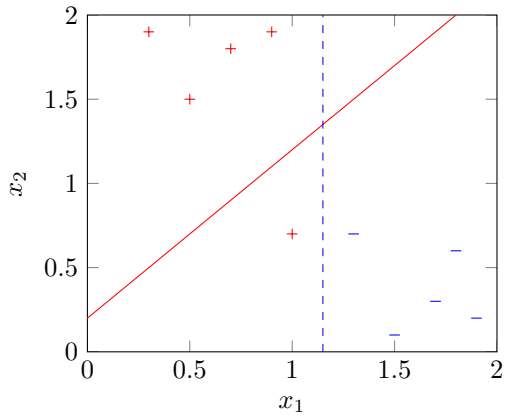
5 SVMs with Slack Variables and Regularization

In the Kernel section, we concluded that SVMs applying the RBF kernel can theoretically separate any data. However, this does not make the RBF kernel the ideal one to use in every situation. The RBF kernel will severely overfit to any noise in the training set and most likely won't generalize well.

An alternative is to use a lower-complexity SVM. However, we may not always be able to separate the data with a lower-complexity SVM, so the optimization objective presented in section 3 is no longer valid. In this section, we will discuss slack variables that allow noisy training data to be inside the SVM margin or even on the wrong side of the separator and discuss how to apply regularization to SVMs.

5.1 Slack Variables

Sometimes, our training data is noisy. Consider the following data where we have proposed two linear separators (solid red and dashed blue):



The positive datapoint at (1, 0.7) is very likely to be noise. Although we can still separate the data taking (1, 0.7) into consideration, our classifier in dashed blue has a much narrower margin and doesn't seem to work well with the rest of the non-noisy datapoints. If we ignore the noisy datapoint, we get the classifier in solid red which has a much wider margin.

To deal with noisy data, we can introduce slack variables, which we call ξ_i for $i = 1 \dots n$. Instead of having the hard constraint $y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1$, we can have a soft constraint $y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1 - \xi$.

Our new optimization problem becomes

$$\min_{\bar{\theta},b,\bar{\xi}} \frac{1}{2}\|\bar{\theta}\|^2 + C\sum_{i=1}^{n} \xi_i \quad \text{subject to} \quad y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) \geq 1 - \xi_i \quad \text{where} \quad \xi_i \geq 0$$

where $C = \frac{1}{\lambda}$ is some constant set by the user (we will discuss the significance of C in the next section).

If a variable has some slack, we impose the penalty ξ_i for it so that points we deem "noisy" but actually aren't noise will still be taken into account. However, we no longer have the hard constraint that all noisy points must be correctly classified.

It can also be noted that if the slack variable for a datapoint satisfies $0 = \xi_i$, then this equivalent to not using a slack variable and the SVM must have the ith datapoint categorized correctly and outside the SVM margin. If $0 < \xi_i \leq 1$, then we essentially encode that the ith datapoint must lie on the correct side of the boundary but is allowed to lie inside the SVM margin. If $1 \leq \xi_i$, then we are encoding that the ith datapoint lies on the wrong side of the margin.

5.2 Regularization

Before we begin to apply Lagrange Multiplers and solve this optimization problem, we should take a second look at the optimization function involving slack variables.

Notice that the penalty due for violating the margin by $1 - \xi_i$ is ξ_i. If we rearrange the constraint function, we obtain

$$\xi_i = 1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b)$$

Now, we see that ξ_i is essentially encoding the hinge loss, which was defined as

$$\text{Loss}_n(z) = \max\{1 - z, 0\}$$

where $z = y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b)$. In addition, the $\frac{1}{2}\|\bar{\theta}\|^2$ term serves essentially as our regularization term. If we increase C, we add more significance to the loss function and lower the significance of the regularization. If we decrease C, we remove significance from the loss function and thus increase the significance of the regularization.

5.3 Optimizing $\bar{\theta}, b$ and $\bar{\xi}$

(This section uses Lagrange Multipliers. For a brief summary of Lagrange Multipliers, please review the beginning of section 3.)

From the previous section, we found that our optimization objective is

$$\min_{\bar{\theta},b,\bar{\xi}} \frac{1}{2}\|\bar{\theta}\|^2 + C\sum_{i=1}^{n}\xi_i \quad \text{subject to} \quad y^{(i)}(\bar{\theta}\cdot\bar{x}^{(i)}+b) \geq 1-\xi_i \quad \text{where} \quad \xi_i \geq 0$$

We can rewrite the constraints as

$$0 \geq 1-\xi_i - y^{(i)}(\bar{\theta}\cdot\bar{x}^{(i)}+b)$$

$$0 \geq -\xi_i$$

and so the constraint functions are

$$g_i(x) = 1-\xi_i - y^{(i)}(\bar{\theta}\cdot\bar{x}^{(i)}+b)$$

with Lagrange Multiplier α_i and

$$h_i(x) = -\xi_i$$

with Lagrange Multiplier ζ_i.

Our auxiliary function is

$$\mathcal{L}(\bar{\theta},b,\bar{\xi}) = \frac{1}{2}\|\bar{\theta}\|^2 + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i(1-\xi_i-y^{(i)}(\bar{\theta}\cdot\bar{x}^{(i)}+b)) - \sum_{i=1}^{n}\zeta_i\xi_i \tag{1}$$

and, we want to set $\nabla_{\bar{\theta}}\mathcal{L}=0$, $\frac{\partial\mathcal{L}}{\partial b}=0$ and $\frac{\partial\mathcal{L}}{\partial\xi_i}=0$. Thus, we get

$$\nabla_{\bar{\theta}}\mathcal{L} = 0 = \bar{\theta} - \sum_{i=1}^{n}\alpha_i y^{(i)}\bar{x}^{(i)} \quad \Rightarrow \quad \bar{\theta}^* = \sum_{i=1}^{n}\alpha_i y^{(i)}\bar{x}^{(i)} \tag{2}$$

$$\frac{\partial\mathcal{L}}{\partial b} = 0 = \sum_{i=1}^{n}\alpha_i(-y^{(i)}) \tag{3}$$

$$\frac{\partial\mathcal{L}}{\partial\xi_i} = 0 = C-\alpha_i-\zeta_i \quad \Rightarrow \quad 0 \leq \alpha_i \leq C \tag{4}$$

The implication in equation (4) is correct because all the ζ_i and α_i implicitly must be non-negative (if ζ_i or α_i could be negative, then the optimal solution would be set $\zeta_i=-\infty$ or $\alpha_i=-\infty$ giving us a useless solution). Thus, when $\zeta_i=0$, we get the upper bound on α_i is C and we already know that α_i needs to be nonnegative for the optimization objective to be useful.

Now, we substitute equations (2), (3), and (4) into equation (1) and we find that our new auxiliary function is

$$\mathcal{L}(\bar{\theta},b,\bar{\xi}) = \frac{1}{2}\left(\sum_{i=1}^{n}\left(\alpha_i y^{(i)}\bar{x}^{(i)}\right)\cdot\sum_{i=1}^{n}\left(\alpha_i y^{(i)}\bar{x}^{(i)}\right)\right) + C\sum_{i=1}^{n}\xi_i + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\alpha_i\xi_i -$$

$$\sum_{i=1}^{n}\alpha_i y^{(i)}\left(\sum_{i=j}^{n}\left(\alpha_j y^{(j)}\bar{x}^{(j)}\right)\right) - \sum_{i=1}^{n}\alpha_i y^{(i)}b - \sum_{i=1}^{n}\zeta_i\xi_i$$

We know that $\sum_{i=1}^{n}\alpha_i y^{(i)}=0$ so the sixth term drops out. We can also combine the second, fourth, and seventh terms to get

$$C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\xi_i - -\sum_{i=1}^{n}\zeta_i\xi_i = \sum_{i=1}^{n}(C-\alpha_i-\zeta_i)\,\xi_i$$

and by equation (4), this sum is also 0. Thus, after removing all terms and sums that zero out, we have

$$\mathcal{L}(\bar{\theta}, b, \bar{\xi}) = \sum_{i=1}^{n} \alpha_i + \frac{1}{2} \left(\sum_{i=1}^{n} \left(\alpha_i y^{(i)} \bar{x}^{(i)} \right) \cdot \sum_{i=1}^{n} \left(\alpha_i y^{(i)} \bar{x}^{(i)} \right) \right) - \sum_{i=1}^{n} \alpha_i y^{(i)} \left(\sum_{i=j}^{n} \left(\alpha_j y^{(j)} \bar{x}^{(j)} \right) \right)$$

and rewriting the dot product as a double summation gives us

$$\mathcal{L}(\bar{\theta}, b, \bar{\xi}) = \sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)}$$

$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)}$$

The optimization objective we pass in to a quadratic programming algorithm is

$$\min_{\bar{\alpha}} \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \sum_{j=1}^{n} \alpha_i \alpha_j y^{(i)} y^{(j)} \bar{x}^{(i)} \cdot \bar{x}^{(j)} \quad \text{subject to} \quad 0 \leq \alpha_i \leq C \quad , \quad \sum_{i=1}^{n} \alpha_i y^{(i)} = 0$$

This is like the optimization objective in section 3, but we bound α_i by C.

Once we find our α_i, we can plug them back in to equation (2) to find the optimal $\bar{\theta}$.

The datapoints with $\alpha_i = 0$ will be non-support vectors that lie outside the margin. For any non-support vector,

$$y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) > 1 \quad \Rightarrow \quad 1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) < 0$$

Since $\xi_i \geq 0$, the constraint $1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) - \xi_i$ is never 0, so the Lagrange Multiplier $\alpha_i = 0$ to satisfy $\alpha_i g_i(x) = 0$. This essentially suggests that non-support vectors do not influence the decision boundary, which makes sense, as the margin is completely determined by the support vectors.

The datapoints with $\alpha_i = C$ will be the margin violators, also known as support vectors with nonzero slack. For any margin violator,

$$y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b) < 1 \quad \Rightarrow \quad 0 < 1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b)$$

Therefore, to satisfy the constraint that $0 \geq 1 - y^{(i)}(\bar{\theta} \cdot \bar{x}^{(i)} + b)$, we must have $\xi_i > 0$. Thus, to satisfy $\zeta_i h_i(x) = 0$, we must have the Lagrange Multiplier $\zeta_i = 0$. If $\zeta_i = 0$, then by equation (4), we have $\alpha_i = C$.

By process of elimination, the datapoints with $0 < \alpha_i < C$ will be the support vectors that do not violate the margin.

Once we know the support vectors in the margin, we can solve the equation

$$y^{(sv)}(\bar{\theta} \cdot \bar{x}^{(sv)} + b) = 1$$

to find values of b. To account for imprecisions, we take the median value of b and use that in our final decision boundary. We have now obtained b and $\bar{\theta}^*$ to be used in our SVM classifier with slack.