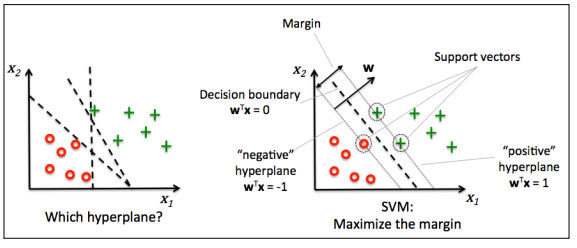
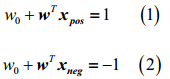
**SVM (support vector machine):**

It can be considered as an extension of the perceptron. Using the perceptron algorithm, we minimized misclassification errors. However, in SVMs, our optimization objective is to maximize the margin. The margin is defined as the distance between the separating hyperplane (decision boundary) and the training samples that are closest to this hyperplane, which are the so-called support vectors.



**Maximum margin intuition:**

The rationale behind having decision boundaries with large margins is that they tend to have a lower generalization error whereas models with small margins are more prone to overftting. To get an intuition for the margin maximization, let's take a closer look at those positive and negative hyperplanes that are parallel to the decision boundary, which can be expressed as follows:



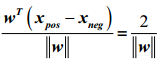
If we subtract those two linear equations (1) and (2) from each other, we get：



We can normalize this by the length of the vector w, which is defined as follows:

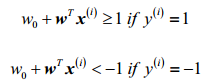


So we arrive at the following equation:



The left side of the preceding equation can then be interpreted as the distance between the positive and negative hyperplane, which is the so-called margin that we want to maximize.

Now the objective function of the SVM becomes the maximization of this margin by maximizing under the constraint that the samples are classified correctly, which can be written as follows:



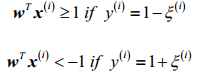
These two equations basically say that all negative samples should fall on one side of the negative hyperplane, whereas all the positive samples should fall behind the positive hyperplane. This can be written more compactly as follows:



**Dealing with the nonlinearly separable case using slack variables**

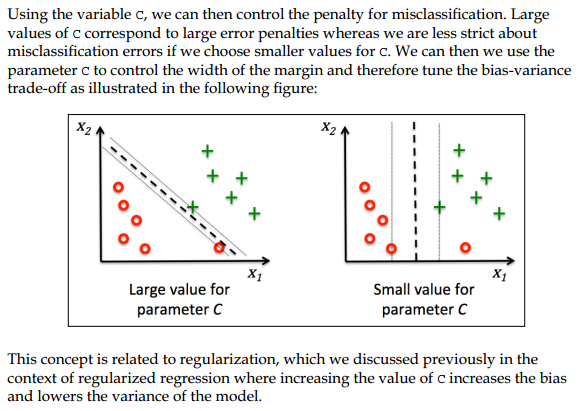
The motivation for introducing the slack variable was that the linear constraints need to be relaxed for nonlinearly separable data to allow convergence of the optimization in the presence of misclassifications under the appropriate cost penalization.

The positive-values slack variable is simply added to the linear constraints:



So the new objective to be minimized becomes:





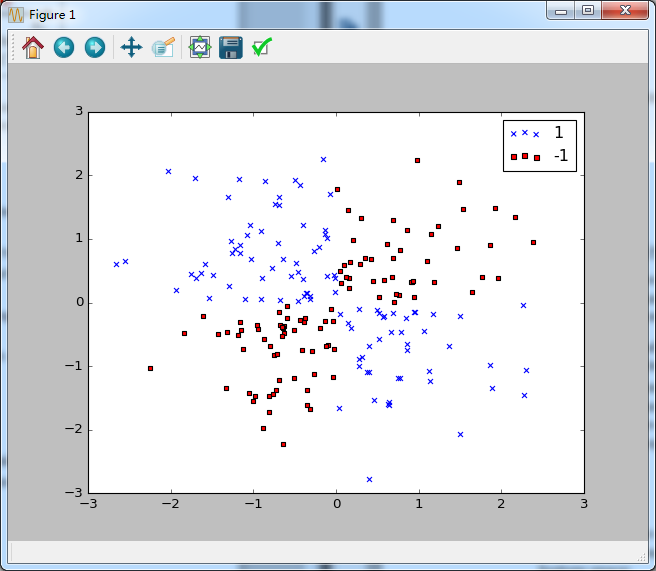
**Logistic regression Vs SVM**

In practical classification tasks, linear logistic regression and linear SVMs often yield very similar results.

1. Logistic regression try to maximize the conditional likelihoods of the training data, which makes it more prone to outliners than SVMs. The SVMs mostly care about the points that are closet to the decision boundary(support vectors).
2. On the other hand, logistic regression has the advantage that it is a simpler model that can be implemented more easily.
3. Furthermore, logistic regression models can be easily updated, which is attractive when working with streaming data.

**Solving nonlinear problem using a kernel SVM**

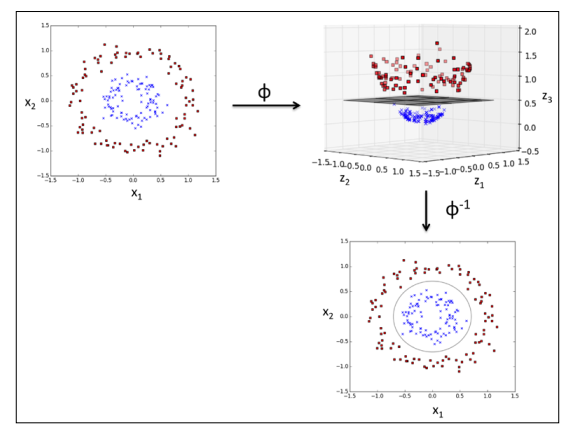
Another reason why SVMs enjoy high popularity among machine learning practitioners is that they can be easily *kernelized* to solve nonlinear classification problems.



The basic idea behind kernel methods to deal with such linearly inseparable data is to create nonlinear combinations of the original features to project them onto a higher dimensional space via a mapping function  where it becomes linearly separable. As shown in the next figure, we can transform a two –dimensional dataset onto a new three-dimensional feature space where the classes become separable via the following projection:



This allows us to separate the two classes shown in the plot via a linear hyperplane that becomes a nonlinear decision boundary if we project it back onto the original future space:

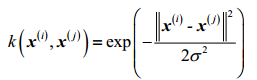


**Using the kernel trick to find separating hyperplanes in higher dimensional space**

To solve a nonlinear problem using an SVM, we transform the training data onto a higher dimensional feature space via a mapping function  and train a linear SVM model to classify the data in this new feature space. Then we can use the same mapping function  to transform new, unseen data to classify it using the linear SVM model.

However, one problem with this mapping approach is that the construction of the new features is computationally very expensive, especially if we are dealing with high-dimensional data. This is where the so-called kernel trick comes into play. Although we didn't go into much detail about how to solve the quadratic programming task to train an SVM, in practice all we need is to replace the dot product by. In order to save the expensive step of calculating this dot product between two points explicitly, we defne a so-called kernel function: .

One of the most widely used kernels is the **Radial Basis Function kernel (RBF kernel) or Gaussian kernel:**



**This is of simplified to:**



The minus sign inverts the distance measure into a similarity score and, due to the exponential term, the resulting similarity score will fall into a range between 1 (for exactly similar samples) and 0 (for very dissimilar samples).

Reference:

1. Python machine learning