Introduction

There are multiple ways to classify data with machine learning. You could run a logistic regression, use decision trees, or build a neural network to accomplish the task. In 1963, Vladimir Vapnik and Alexey Chervonenkis developed another classification tool, the support vector machine. Vapnik refined this classification method in the 1990's and extended uses for SVMs.

Machine learning involves predicting and classifying data and to do so we employ various machine learning algorithms according to the dataset.

SVM or Support Vector Machine is a linear model for classification and regression problems. It can solve linear and non-linear problems and work well for many practical problems. The idea of SVM is simple: The algorithm creates a line or a hyperplane which separates the data into classes.

How it works?

At first approximation what SVMs do is to find a separating line(or hyperplane) between data of two classes. SVM is an algorithm that takes the data as an input and outputs a line that separates those classes if possible.

Let's begin with a problem. Suppose you have a dataset as shown below and you need to classify the red rectangles from the blue ellipses(let's say positives from the negatives). So your task is to find an ideal line that separates this dataset in two classes (say red and blue).

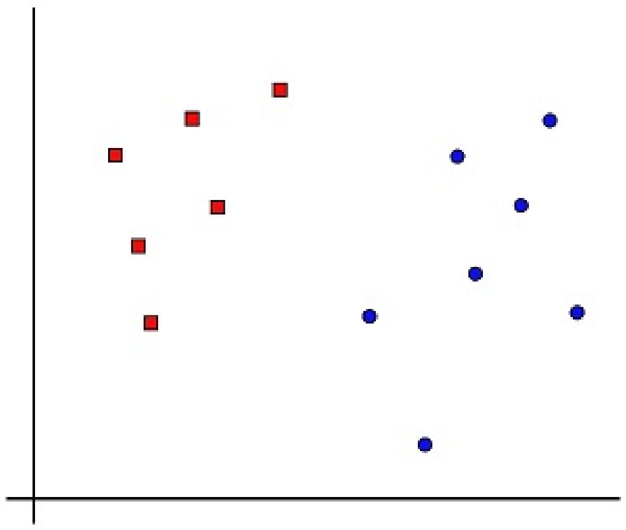

Figure 1

But, as you notice there isn't a unique line that does the job. In fact, we have an infinite lines that can separate these two classes. So how does SVM find the ideal one?

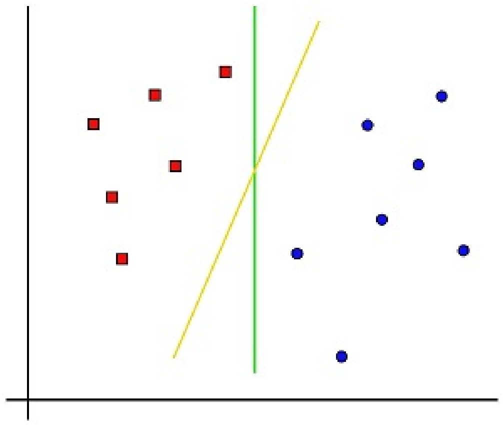

Figure 2

We have two candidates here, the green colored line and the yellow colored line. Which line according to you best separates the data?

If you select the yellow line, then thats the line we are looking for. It's visually quite intuitive in this case that the yellow line classifies better. But, we need something concrete to fix our line.

The green line in the image above is quite close to the red class. Though it classifies the current datasets it is not a generalized line and in machine learning our goal is to get a more generalized separator.

SVM's way to find the best line

According to the SVM algorithm, we find the points closest to the line from both the classes. These points are called support vectors. Now, we compute the distance between the line and the support vectors. This distance is called the margin. Our goal is to maximize the margin. The hyperplane for which the margin is maximum is the optimal hyperplane.

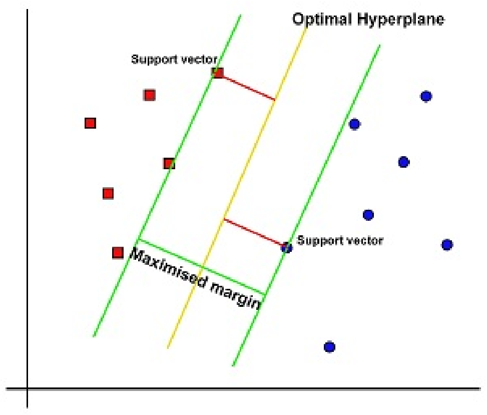

Figure 3

Thus SVM tries to make a decision boundary in such a way that the separation between the two classes (that street) is as wide as possible.

Let's consider a bit complex dataset, which is not linearly seperable.

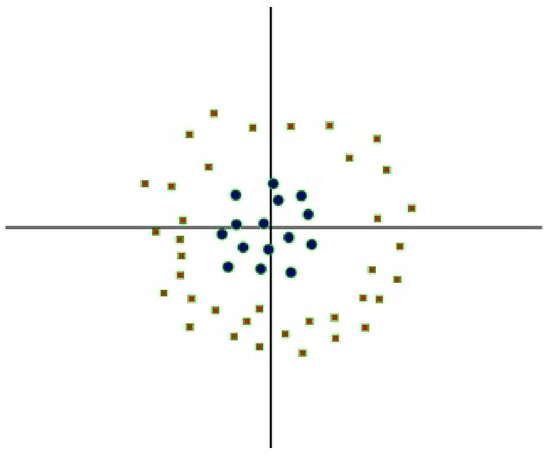

Figure 4

This data is clearly not linearly separable. We cannot draw a straight line that can classify this data. But, this data can be converted to linearly separable data in higher dimension. Lets add one more dimension and call it z-axis. Let the co-ordinates on z-axis be given by the constraint,

$$z = x^2 + y^2$$

So, basically z co-ordinate is the square of distance of the point from origin. Let's plot the data on z-axis.

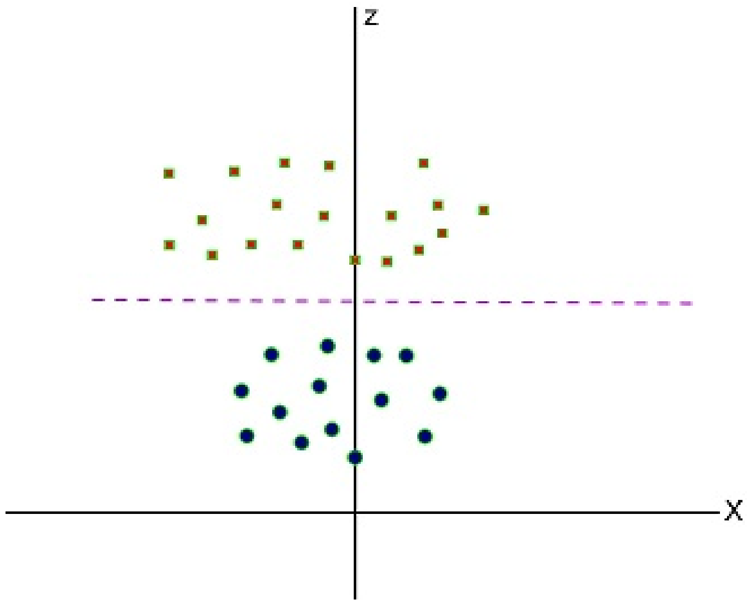

Figure 5

Now the data is clearly linearly separable. Let the purple line separating the data in higher dimension be $z = k$, where k is a constant.

Since, $z = x^2 + y^2$ we get $x^2 + y^2 = k$; which is an equation of a circle. So, we can project this linear separator in higher dimension back in original dimensions using this transformation.

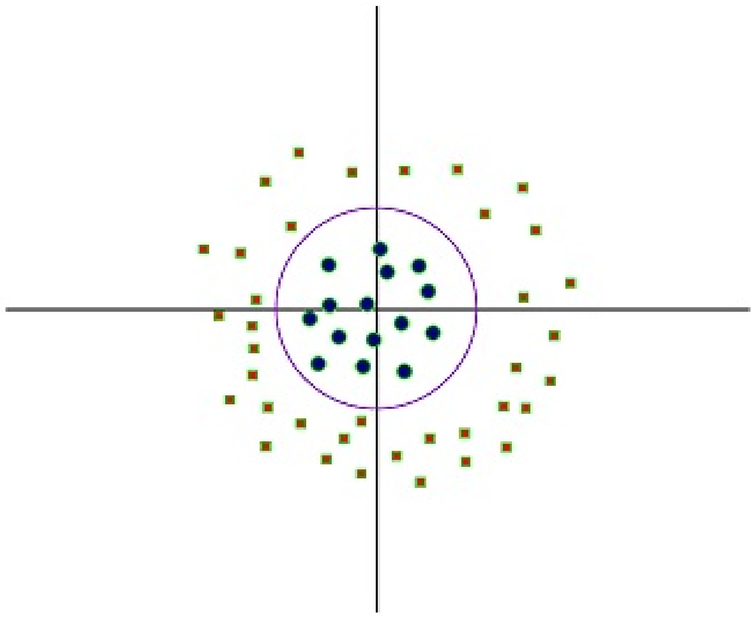

Figure 6

Thus we can classify data by adding an extra dimension to it so that it becomes linearly seperable and then projecting the decision boundary back to original dimensions using mathematical transformation. But finding the correct transformation for any given dataset isn't that easy.

The goal of SVM is to identify a optimal seperating hyperplane which maximizes the margin between different classes of the training data.

Hyperplane: It is basically a generalization of plane

- in one dimension, an hyperplane is called a point
- in two dimensions, it is a line
- in three dimensions, it is a plane
- in more dimensions you can call it an hyperplane

SVM : Formulation

- Let $g(x) = w^T x + b$
- We want to minimize margin such that:
  - $w^T x_i + b \geq 1$ for $y_i = 1$.
  - $w^T x_i + b \neq 1$ for $y_i = 1$.
  - $y_i(w^T x_i + b) \geq 1$ for all i.

For example, The distance from $(0,0)$ to $ax + by + c = 0$ is $\frac{c}{\sqrt{a^2+b^2}}$. Similarly distance from origin to $w^T x + b = 1$ is $\frac{b-1}{||w||}$ and to $w^T x + b = -1$ is $\frac{b+1}{||w||}$.

The distance between the two lines/planes (margin) is

$$\frac{b+1}{||w||} - \frac{b-1}{||w||} = \frac{2}{||w||}$$

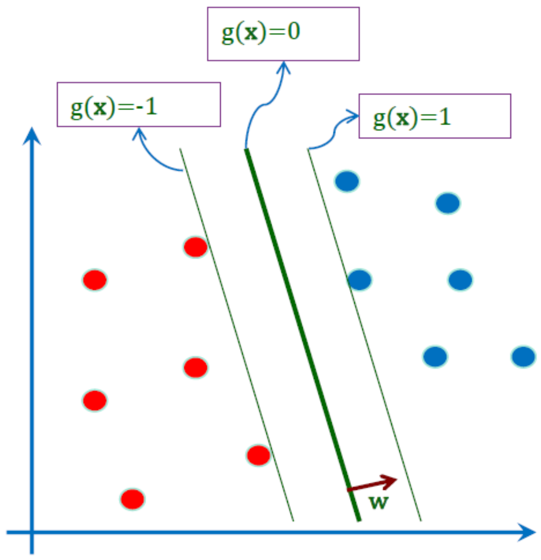

Figure 7

The objective of maximize the margin is same as minimize $\frac{1}{2}||w||$, such that all positive samples and negative samples are beyond the respective half planes.
This is a convex optimization problem.

SVM: Primal and Dual

Primal problem

$$min \frac{1}{2} w^T w$$
$$\text{subject to } y_i(w^T x_i + b) - 1 \geq 0 \forall i$$
$$y_i \in \{-1, 1\}$$

This results in maximization of

Dual problem

$$J_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j x_i^T x_j$$
$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$
$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

If we can solve for α_is (dual problem), then we have a solution for w (primal problem).

So why solve the dual SVM?

There are some quadratic programming algorithms that can solve the dual faster than the primal, specially in high dimensions.

What if Data is not Linearly separable?

In the case of non-linearly separable data points SVM uses the kernel trick. The idea stems from the fact that if the data cannot be partitioned by a linear boundary in its current dimension, then projecting the data into a higher dimensional space may make it linearly separable.

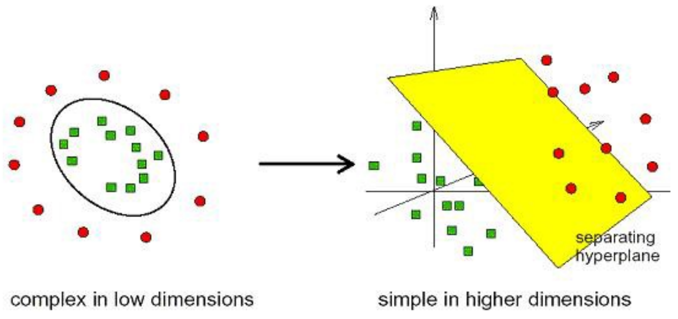

Figure 8

Kernel Trick

The Kernel Trick is the maximization of

$$J_d(\alpha) = \sum_{i=1}^{N} \alpha_i - \frac{1}{2} \sum_{i=1}^{N} \sum_{i=1}^{N} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$
$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$K(x_i, x_j) = \phi(x_i).\phi(x_j)$$

where $K(x_i, x_j)$ is the Kernel function and ϕ is a mapping used to project data in higher dimension.

Kernel Strategy

We have $w = \sum_{i=1}^{N} \alpha_i y_i x_i$.
What we need only is $w^T x = \sum_{i=1}^{N} \alpha_i y_i x_i^T x$.
We can do the same in a new feature space:

$$w^T x = \sum_{i=1}^{N} \alpha_i y_i \phi(x_i)^T \phi(x) \; w^T x = \sum_{i=1}^{N} \alpha_i y_i K(x_i, x)$$

Using Kernels, it is possible to train and test SVMs this without explicitly doing the non-linear mapping to high dimensions.
We need only a Kernel function

$$K(x_i, x_j) = \phi(x_i)\phi(x_j)$$

Types of Kernels

- linear: $K(x, y) = x^T y$.
- polynomial: $K(x, y) = (\gamma x^T y + r)^d, \gamma > 0$.
- radial basis function (RBF): $K(x, y) = \exp(-\gamma ||x - y||^2)\gamma > 0$.
- sigmoid: $K(x, y) = \tanh(\gamma x^T y + r)$.

References:

For more details on SVM,
`http://cs229.stanford.edu/notes/cs229-notes3.pdf`
`https://www.youtube.com/watch?v=9_DJ4KvyYoo`
`https://www.youtube.com/watch?v=ffF8UnbheLk`