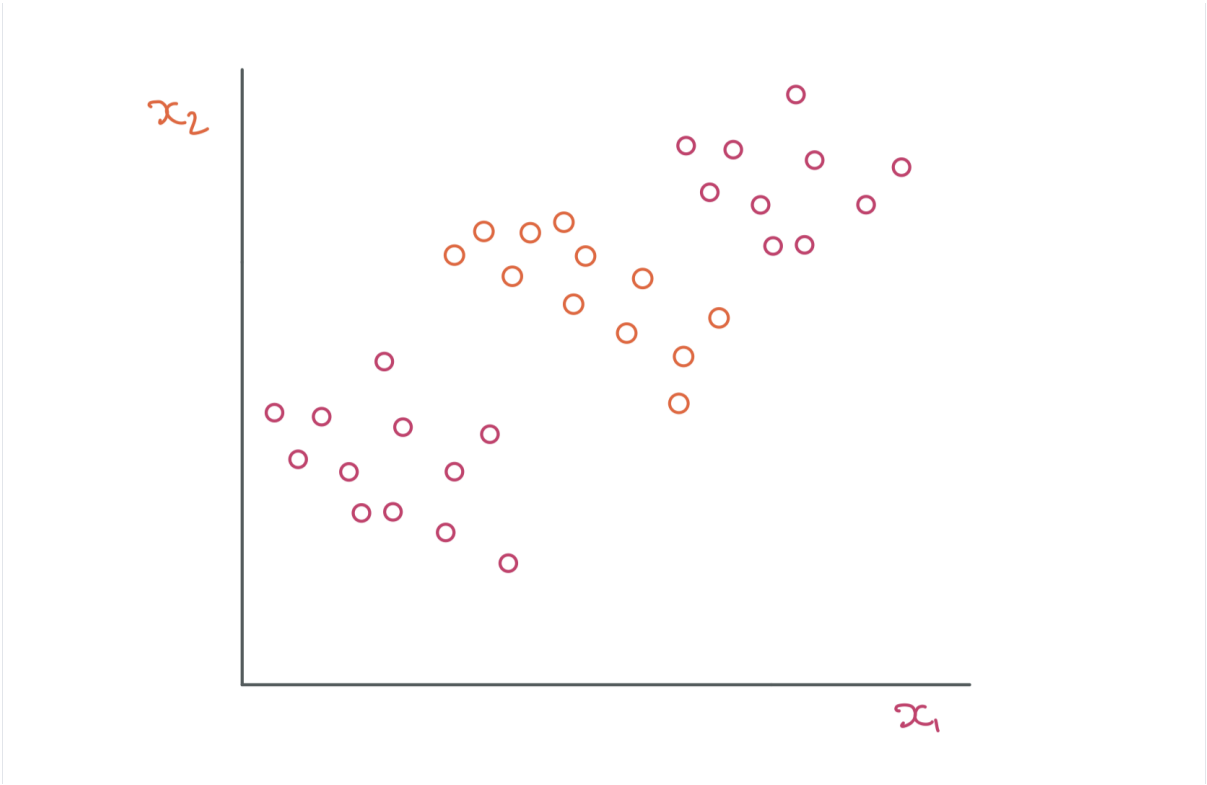
# Support Vector Machines Part 3

## Motivation

In the case where the two class labels can be separated by a linear boundary, support vector classifier can be a good choice. But in situations where the data  cannot be separated by a linear boundary, we may need to explore more options. Consider the following figure where a linear boundary will not be able to separate the orange and red bubbles.



What this means is that we need to consider non linear classifiers which may include quadratic or cubic polynomials.  For a quadratic polynomial the constraint equations will be

> maximize $M$

**Subject to**

$$y_i(\beta_0 + \sum_{j=1}^{p} \beta_{j1} x_{ij} + \sum_{j=1}^{p} \beta_{j2} x_{ij}^2) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \sum_{i=1}^{n} \epsilon_i \leq C$$

$$\sum_{j=1}^{p} \sum_{k=1}^{2} \beta_{jk}^2 = 1$$

In the above case, we have considered a quadratic polynomial, but higher order polynomials can also be used to build a non linear classifier. Consequently, the higher order terms would also lead to heavy computation. Support vector machines finds a way to work in enlarged space without the computations becoming inefficient.

Essentially in support vector machines, we are enlarging the feature space from the support vector classifier. So SVM is an extension of the SVC.  First let's look at the solution of the SVC optimization problem and then make some changes to it in a way that it generalizes to higher order classifiers.

The  solution to the SVC optimization problem mentioned in the second article can be written as:

> $f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i < x, x_i >$

Here $< x, x_i >$ is known as the inner product between $x$ and $x_i$ and is equal to $\sum_{j=1}^{p} x_{ij} x_{i'j}$

$x$ here is a **new data point** and we are calculating its inner product with each **training observation** $x_i$ for $i = 1$ to $n$.

The parameters $\alpha_1$, $\alpha_2$.... $\alpha_n$ and $\beta_0$ can be obtained by calculating the inner products $< x_i, x_j >$ between all the training observations. Now the good part is that these $\alpha_i$ is non zero only for those training data point that is a **Support Vector**. Remember, support vectors are the data points that lie either on the margin or on the wrong side of it. So say from the total set of training observation data points a subset of these $S$ are support vectors, then we only sum over these in the last equation, which we can rewrite as

$$f(x) = \beta_0 + \sum_{i \epsilon S} \alpha_i < x, x_i >$$

The choice of inner product used here uses the Pearson correlation to calculate the similarity of two observations. Now to generalize this, we can write the above equation as

$$f(x) = \beta_0 + \sum_{i \epsilon S} \alpha_i K(x, x_i)$$

$K$ here is called Kernel and is a function of $x$ and $x_i$

In case of the Support vector classifier, it is just equal to the the inner product considered above, but it can take many other forms.

The following Kernel called **Radial Kernel** offers a more flexible decision boundary.

$$K(x, x_i) = exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2)$$

$\gamma$ is a tuning parameter and decides how flexible the decision boundary will be. A high value of $\gamma$ will make the boundary too flexible, making it follow the noise instead of the signal and resulting in high variance and overfitting.

If a training observation $x_i$ is far from a new data point $x$, then the euclidean distance $(x_{ij} - x_{i'j})^2$ will be very large and thus the value of $K(x, x_i)$ will be very small. Essentially this particular training observation or any other training observations that are far from a new  data point, will not play a role in its classification since $K(x, x_i)$ will be tiny and hence $f(x)$ will not be impacted by such an $x_i$

Only those training observations that are close to the test data point will have an effect on its classification. Radial kernel has a local behavior.

## Reference

1. An introduction to Statistical learning by Gareth James et al