Support Vector Machine

Support vector machine is a supervised machine learning algorithm that classify data using a hyperplane that is defined by support vectors. Support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed. This dividing hyperplane is called the **Maximal-Margin hyperplane (MMH)**. It is the plane that can separate the two classes with the largest margin, or the largest distance between the hyperplane and the support vectors. The Maximal-Margin hyperplane is learned from the training data using an optimization procedure.

Constructing the Maximal Margin Classifier

Given n training observations/vectors $\boldsymbol{x}_1,....\boldsymbol{x}_n \in \boldsymbol{R}$ and n class labels $\boldsymbol{y}_{1,....}\boldsymbol{y}_n \in \{-1,1\}$, the MMH is the solution to the following optimization procedure:

Maximize $\boldsymbol{M} \in \boldsymbol{R}$, by varying $\boldsymbol{b}_1,...\boldsymbol{b}_p$ such that:

$$\sum_{j=1}^{p} \boldsymbol{b}_j^2 = 1$$

and

$$\boldsymbol{y}_i(\boldsymbol{b} \bullet \boldsymbol{x} + \boldsymbol{b}_0) \geq \boldsymbol{M}, \forall \boldsymbol{i} = 1,...\boldsymbol{n}$$

Soft Margin Classifier

Real data is usually messy and cannot be separated perfectly with a hyperplane. Thus, the constraint of maximizing the margin of the hyperplane that separates the classes must be relaxed. This is often called the **soft margin classifier**. This change allows some points in the training data to violate the separating line.

Mathematics behind the soft margin classifier are we are maximizing a new M, such that:

$$\sum_{j=1}^{p} \boldsymbol{b}_j^2 = 1$$

and

$$\boldsymbol{y}_i(\boldsymbol{b} \bullet \boldsymbol{x} + \boldsymbol{b}_0) \geq \boldsymbol{M}(1 - \varepsilon_i), \forall \boldsymbol{i} = 1,...\boldsymbol{n}$$

and

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

Here, the slack variable ε_i tell us where the *i*th observation is located relative to the margin and the hyperplane. $\varepsilon_i = 0$ states that the *i*th training observation is on the correct side of the margin; $\varepsilon_i > 0$ states that the *i*th training observation is on the wrong side of the margin; $\varepsilon_i > 1$ states that the *i*th training observation is on the wrong side of the hyperplane.

The **tuning parameter C** defines the amount of violation of the margin allowed. The larger the value of C, the more violations of the hyperplane are permitted and C=0 means no violation. As C affects the number of instances that are allowed to fall within the margin, C influences the number of support vectors used by the model. The smaller the value of C, the more sensitive the algorithm is to the training data (higher variance and lower bias). The larger the value of C, the less sensitive the algorithm is to the training data (lower variance and higher bias).

Kernel Trick

The main advantage of SVMs is that they allow a *non-linear* enlargening of the feature space, while still retaining a significant computational efficiency, using a process known as the 'kernel trick'.

When optimizing the *linear* SVC, the algorithm only need to make use of the inner products between the observations (see Andrew Ng's notes for geometric explanation) and not the observations themselves. The inner product is defined for two p-dimensional vectors u, v as:

$$\langle \boldsymbol{u}, \boldsymbol{v} \rangle = \sum_{j=1}^{p} \boldsymbol{u}_j \boldsymbol{v}_j$$

Hence for two observations an inner product is defined as:

$$\langle \boldsymbol{x}_{i,} \boldsymbol{x}_k \rangle = \sum_{j=1}^{p} \boldsymbol{x}_{ij} \boldsymbol{x}_{jk}$$

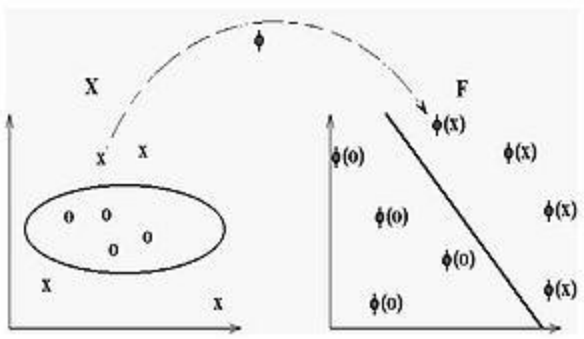
If we replace the inner product with a more general inner product Kernel function, we modify the SVC to process non-linear problems. Mathematically, Kernel is a way of computing the dot product of two vectors x and y in some feature space. In SVM, a kernel defines the similarity or a distance measure between new data and the support vectors. Suppose we have a mapping $\varphi : \boldsymbol{R}^n \to \boldsymbol{R}^m$ that brings our vectors in $\boldsymbol{R}^n$ to some feature space $\boldsymbol{R}^m$. Then the dot product of x and y in this space is $\varphi(\boldsymbol{x})^{\boldsymbol{T}} \varphi(\boldsymbol{y})$. A kernel is a function K that corresponds to this dot product: $\boldsymbol{K}(\boldsymbol{x}, \boldsymbol{y}) = \varphi(\boldsymbol{x})^{\boldsymbol{T}} \varphi(\boldsymbol{y})$.

Linear Kernel

$$\boldsymbol{K}(\boldsymbol{x},\boldsymbol{y}) = \boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{y}$$

Polynomial Kernel (of degree d)

$$\boldsymbol{K}(\boldsymbol{x},\boldsymbol{y}) = (1+\boldsymbol{x}^{\boldsymbol{T}}\boldsymbol{y})^{\boldsymbol{d}} \quad \text{where d > 1}$$



The polynomial Kernel map input value x into another feature space where different classes of x can be separated by a line. Ref: Wikipedia

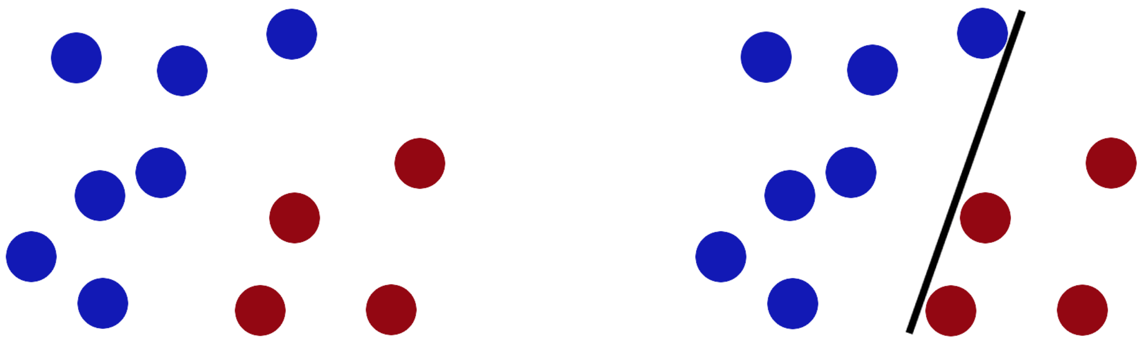
Gaussian Kernel

$$f_i = \exp(-\frac{\left\|\boldsymbol{x}-\boldsymbol{l}^{(i)}\right\|^2}{2\sigma^2}) \quad \text{where } \boldsymbol{l}^{(i)} = \boldsymbol{x}^{(i)}$$

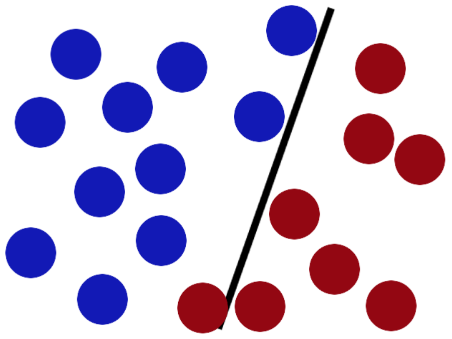
SVM to a five-year old

Reference:https://www.reddit.com/r/MachineLearning/comments/15zrpp/please_explain_support_vector_machines_svm_like_i/

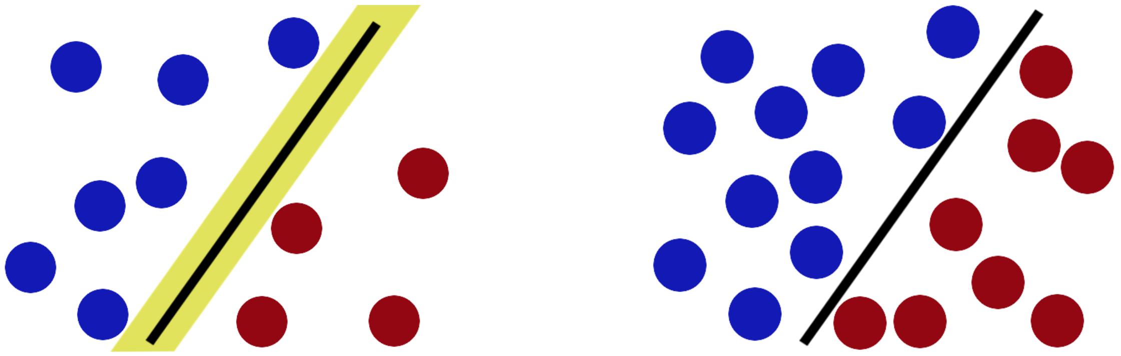
We have two colors of balls on the table (left figure) and we want to separate them. We simply place a stick in between the balls (right figure).



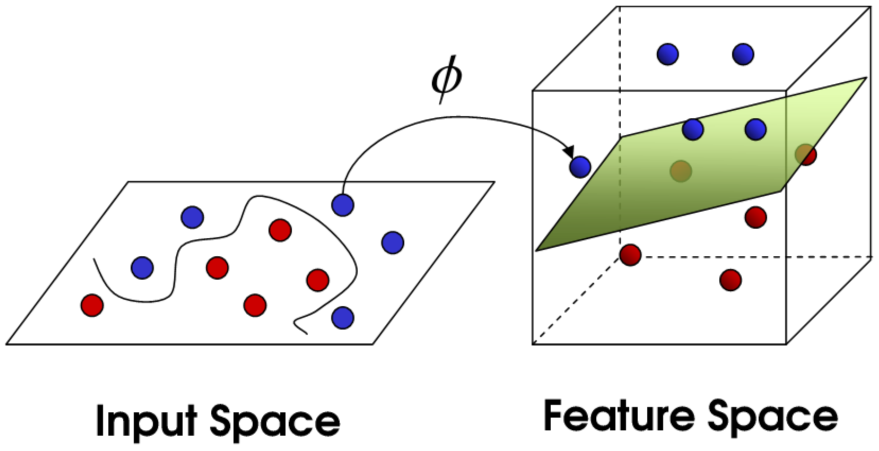
More balls are coming and it’s getting harder for the original line to separate the two types of balls. As shown below, one red ball is on the wrong side of the line.



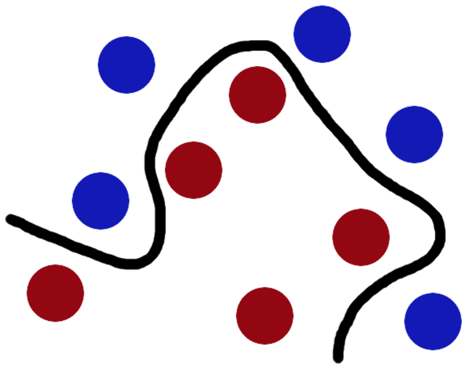
SVM try to put the stick in the best possible place by having as big a gap on either side of the stick as possible (left figure). With new coming balls, the position of the line are adjusted so that the stick can still separate the two types of balls (right figure).



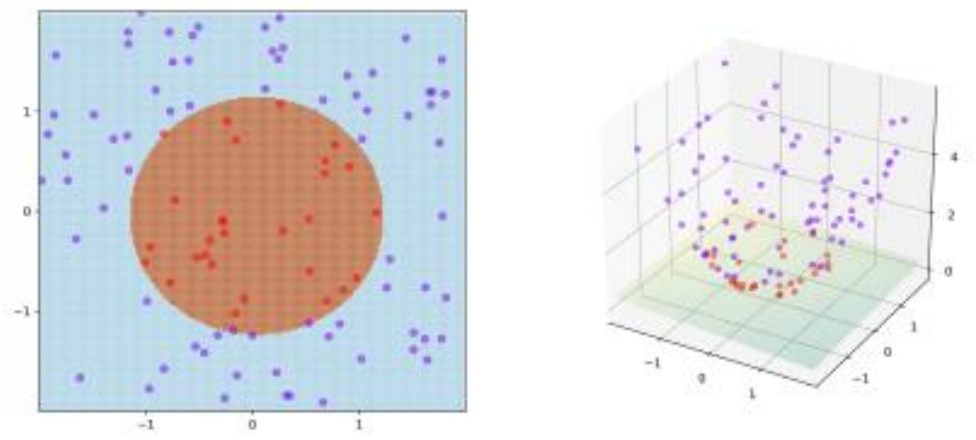
The positions of the balls can get more complicated and a stick is not enough to separate the balls. Instead, you throw the ball into the air and use a piece of paper to separate the balls.



If you look at the piece of paper from the table, it looks like a curved line which separate the two colors of balls, as shown below.



Mathematically, this mapping into feature space is done via using kernel tricks. For example, SVM with kernel given by $\varphi((a, b)) = (a, b, a^2 + b^2)$ and thus $K(\mathbf{x}, \mathbf{y}) = \mathbf{x}\bullet\mathbf{y} + \mathbf{x}^2\,\mathbf{y}^2$ (see figure below). The training points are mapped to a 3-dimensional space where a separating hyperplane can be easily found.



Ref: https://en.wikipedia.org/wiki/Kernel_method

References:

1, https://www.quantstart.com/articles/Support-Vector-Machines-A-Guide-for-Beginners

2, https://machinelearningmastery.com/support-vector-machines-for-machine-learning/