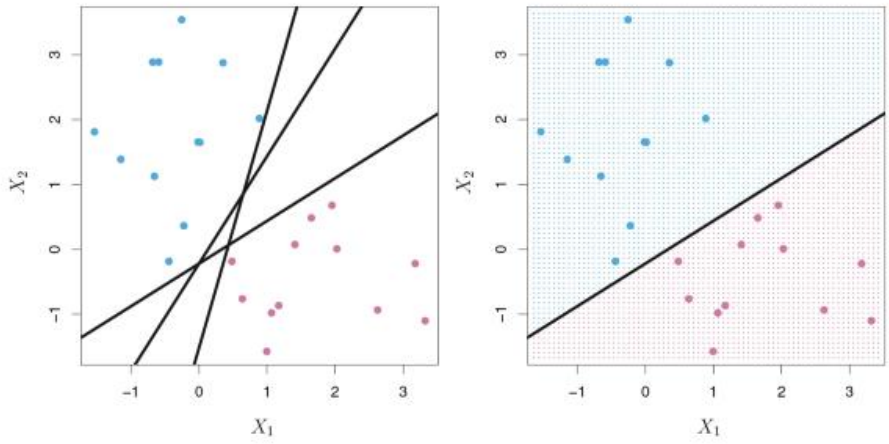
Lecture 15

Support Vector Machines

Support Vector Machine SVMs were introduced by Boser, Guyon, Vapnik in 1992.
The SVM model is a supervised machine learning model that is mainly used for classifications (but it could also be used for regression!). It learns how to separate different groups by forming decision boundaries.
SVM performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories.
Family of machine-learning algorithms that are used for mathematical and engineering problems including for example handwriting digit recognition, object recognition, speaker identification, face detections in images and target detection.
• SVMs have become popular because of their success in handwritten digit recognition.

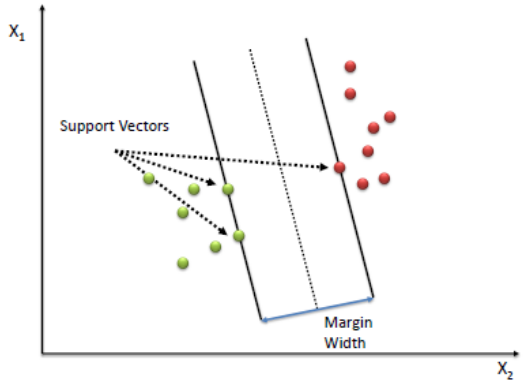
.



In the graph above, we notice that there are two classes of observations: the blue points and the purple points. There are tons of ways to separate these two classes as shown in the graph on the left. However, we want to find the "best" hyperplane that could maximize the margin between these two classes, which means that the distance between the hyperplane and the nearest data points on each side is the largest. Depending on which side of the hyperplane a new data point locates, we could assign a class to the new observation.

Support Vector Machine - Classification (SVM)

A Support Vector Machine (SVM) performs classification by finding the hyperplane that maximizes the margin between the two classes. The vectors (cases) that define the hyperplane are the support vectors.

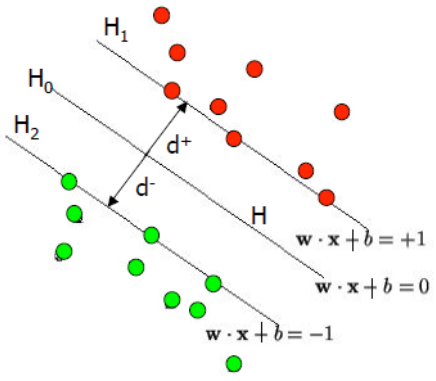


How It Works

Classifies data by finding the linear decision boundary (hyperplane) that separates all data points of one class from those of the other class. The best hyperplane for an SVM is the one with the largest margin between the two classes, when the data is linearly separable. If the data is not linearly separable, a loss function is used to penalize points on the wrong side of the hyperplane. SVMs sometimes use a kernel transform to transform nonlinearly separable data into higher dimensions where a linear decision boundary can be found.

Define the hyperplanes H such that:
$w \bullet x_i + b \geq +1$ when $y_i = +1$
$w \bullet x_i + b \leq -1$ when $y_i = -1$

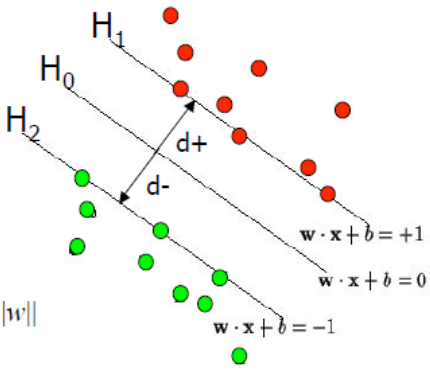
H_1 and H_2 are the planes:
H_1: $w \bullet x_i + b = +1$
H_2: $w \bullet x_i + b = -1$
The points on the planes H_1 and H_2 are the tips of the Support Vectors
The plane H_0 is the median in between, where $w \bullet x_i + b = 0$



d+ = the shortest distance to the closest positive point
d- = the shortest distance to the closest negative point
The margin (gutter) of a separating hyperplane is d+ + d–.

Maximizing the margin (aka street width)

We want a classifier (linear separator) with as big a margin as possible.



Recall the distance from a point(x_0,y_0) to a line: $Ax+By+c = 0$ is: $|Ax_0 +By_0 +c|$/sqrt(A^2+B^2), so, The distance between H_0 and H_1 is then: $|w•x+b|/||w||=1/||w||$, so

The total distance between H_1 and H_2 is thus: $2/||w||$

In order to maximize the margin, we thus need to minimize $||w||$. With the condition that there are no datapoints between H_1 and H_2:

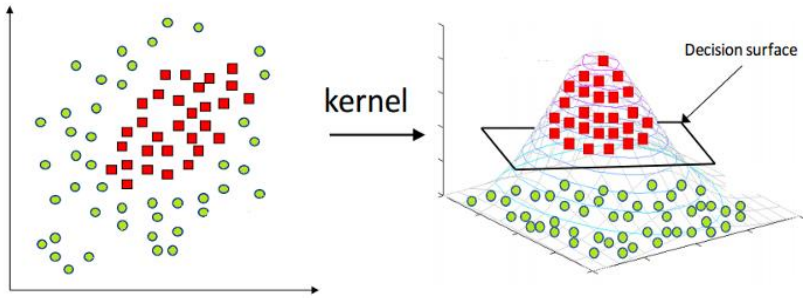
$\mathbf{x_i•w+b \geq +1}$ when $y_i = +1$
$\mathbf{x_i•w+b \leq -1}$ when $y_i = -1$

Can be combined into: $y_i(x_i•w) \geq 1$

Kernel methods

However, not all data are linearly separable. In fact, in the real world, almost all the data are randomly distributed, which makes it hard to separate different classes linearly.

SVMs are now important and active field of all Machine Learning research and are regarded as an main example of “kernel methods”.



Why is it important to use the kernel trick?

As you can see in the above picture, if we find a way to map the data from 2-dimensional space to 3-dimensional space, we will be able to find a decision surface that clearly divides between different classes. This data transformation process is to map all the data point to a higher dimension (in this case, 3 dimension), find the boundary, and make the classification.

However, when there are more and more dimensions, computations within that space become more and more expensive. This is when the kernel trick comes in. It allows us to operate in the original feature space without computing the coordinates of the data in a higher dimensional space.