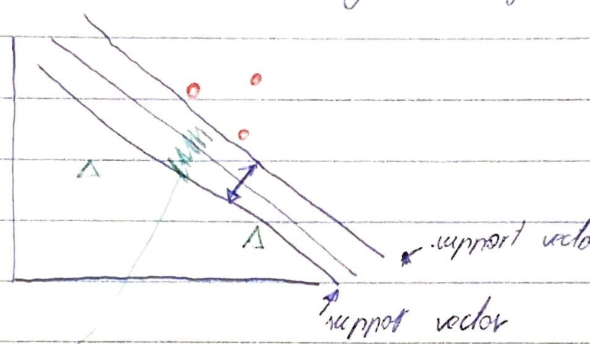
Tutorial session 05. Kernel Machines.

Question 1)

Part I) Maximum margin classifier



This space

Classifier that tries to figure out the maximum space between 2 classes

To do so, the classifier will look for a hiper plane (line) and the support vectors that maximise this space ....
We will see how this happens in a while.

Tutorial session 05. Kernel Machines.

Question 1)

Part II) What is a kernel?

First, Just as a premise, We can use functions To move our data to a higher dimensional space. Then, once we are in this space, when can better separate the ~~classes~~ different classes.

Example:

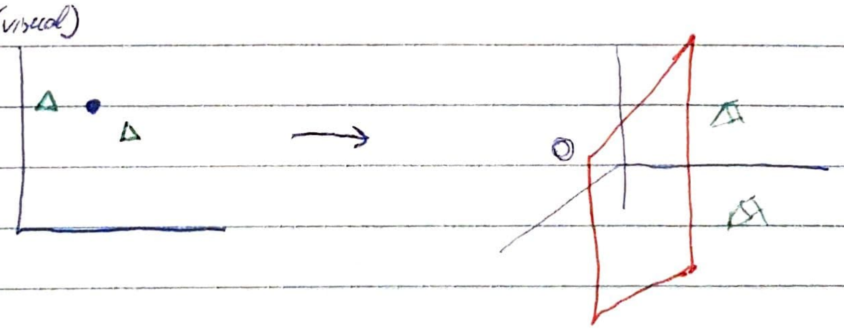
function: can be any function

$$x=\underbrace{(x_1, x_2)}_{\text{attributes}} \longrightarrow F(x) \longrightarrow x=(x_1^2, x_2^2, \sqrt{2x_1x_2})$$

moving 2D data to a 3D space.

It migh be the case that the dataset is lineary separable in ~~a~~ the 3D realm.

Example (visual)



→ Later on, you will see that we will need to do calculations in the new "Realm" or "space". Therefore we need to move data to the new "space". However moving all the data is computationally expensive. So we can move just the ~~parts~~ result of the calculation we use

Tutorial session 05. Kernel Machines.

In SVM we need to multiply our feature vectors. So why don't we move the result of that multiplication?

We can do this with the kernel trick. So rather than

$$\overset{\text{data point}}{F(x)} \cdot \underset{\text{data point}}{F(z)}$$

we can use a kernel to do all the process in one

$$K(x, z)$$

examples:

$$K(x_i, x_j) = \exp\left(- \frac{\|x_i - x_j\|_2^2}{2\sigma^2}\right)$$

$$= \tanh\left(\kappa (x_i \cdot x_j) + \Theta\right)$$

etc...

Tutorial session 05. Kernel Machines

Question 2) Advantages of the Kernel trick.

At some point, in the optimisation process To get the weights of the SVM, we will need to solve this equation:

$$\arg\max_{\alpha} \sum_{i=1}^{m} a_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \underbrace{(x_i \cdot x_j)}$$

we need To move this To a higher dimension

$$\arg\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j \underbrace{F(x_i) \cdot F(x_j)}$$

Too Expensive

we need the kernel trick

$$\arg\max_{\alpha} \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} \alpha_i \alpha_j y_i y_j K(x_i, x_j)$$

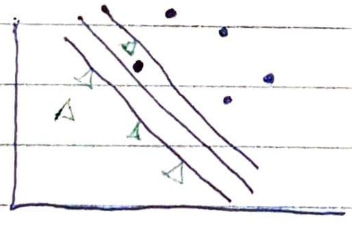
Tutorial session 05. Kernel Machines

Question 3) Soft margin classifier.

a) Sometimes, even using a kernel, it might be impossible to find a set of weights that separate the classes in a dataset. So, our optimizers will never find the solution.

As a solution, we can frame the optimisation process in a way that we allow some points in the space between the support vectors. This is the main concept of soft SVMs.

Example



→ here we allow 2 data instances to stay within the margings.

b) We can translate this into the optimisation problem as follows

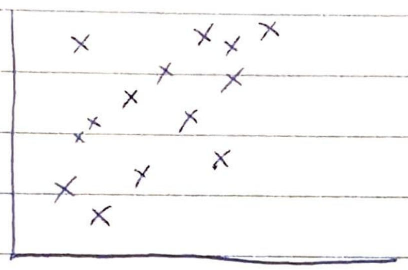
C controls the number of points that we allow in the margin

$$\min \frac{1}{2}\|w\|^2 + C\sum \varepsilon_i$$

Tutorial session 05. Kernel machines

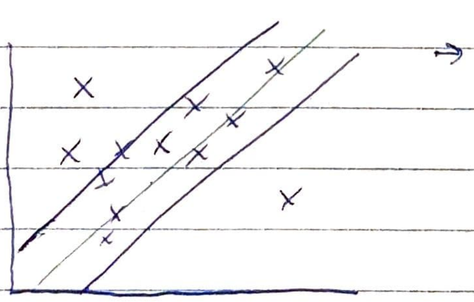
Question 4):

Suppor vector regression use the concept of SVMs but in a defferent way



→ We have this regression dotaset. Rather than separating the points, we need to find a line to predict the points in the plane

We can create a line supported by the support vector that enclose, in a small space, the highest amount of instances.



→ So, now, we have to find the minimum space that encloses the maximum number of instances

Obviously, we need to modify the optimisation process.

slack that allows points outside the region

$$\min \underbrace{\frac{1}{2}\|w\|^2}_{\text{minimizes the space between the support vector}} + C\sum_{i=1}^{m} \underbrace{|y_i - (w \cdot x + b)|_\varepsilon}_{\text{error of the regression}}$$