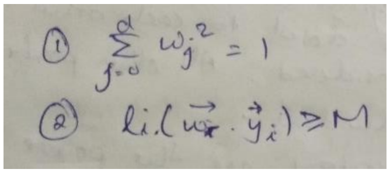
Question-1:

How is Soft Margin Classifier different from Maximal Margin Classifier?

Answer-1:

The *Maximal Margin Classifier* is a line or a hyperplane that maintains the largest possible equal distance from the nearest points of both the classifiers. These nearest points are called *support vectors* and are the only ones that are responsible for creating the hyperplane. The distance/band that exists between the closest datapoints and the hyperplane on both sides is referred to as the *margin*. The line with the maximum margin would be considered the best fit line for the given data. The Maximal Margin Classifier only exists for cases where the classes can be *perfectly separated*.

The constraints that need to be satisfied for the Maximal Margin Classifier include:

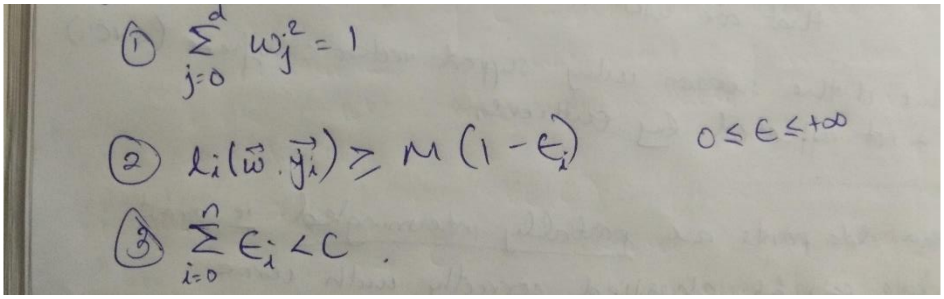


① $\sum_{j=0}^{d} w_j^2 = 1$

② $l_i.(\vec{w} \cdot \vec{y}_i) \geq M$

However, there may be cases when the classes cannot be perfectly classified i.e. it's not possible to come up with a line or a hyperplane that perfectly separates the different classes in the data. Here, we use the *Soft Margin Classifier*. In such cases where our datapoints are *partially intermingled* i.e. most of the data can be classified correctly with some *misclassifications*, the hyperplane that allows certain points to be deliberately misclassified is called the *Support Vector Classifier*. A slack variable, Є, is used to control the misclassifications.

The constraints that need to be satisfied for the Soft Margin Classifier include:



① $\sum_{j=0}^{d} w_j^2 = 1$

② $l_i(\vec{w} \cdot \vec{y}_i) \geq M(1-\epsilon_i)$ $0 \leq \epsilon \leq +\infty$

③ $\sum_{i=0}^{n} \epsilon_i < C$.

Question-2:

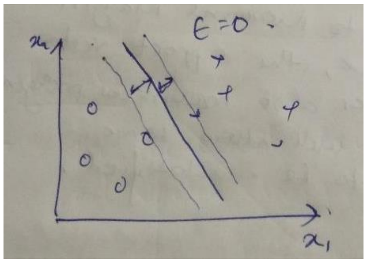
What does the slack variable Epsilon (Є) represent?

Answer-2:

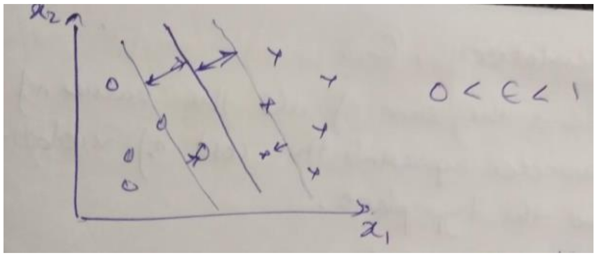
The slack variable is used to control the *misclassifications* allowed in the soft margin classification. The slack variable tells us where an observation is located relative to a margin and hyperplane. As discussed in the previous question, the second constraint that the soft margin classifier needs to satisfy is:

$$\textcircled{2} \; l_i(\vec{w}.\vec{y}_i) \geq M(1-\epsilon_i) \qquad 0 \leq \epsilon \leq +\infty$$

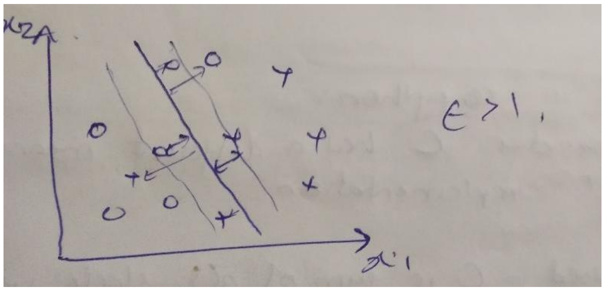
When the sum of Є's equals 0, the model behaves as a maximal margin classifier i.e. all the data points are at a distance at least more than the margin, M from the hyperplane.



When the value of Є's lies between 0 and 1 i.e. if the datapoints are correctly identified but falls inside the margin, M



Finally, when the value of Є's is greater than 1 i.e. if the datapoint is incorrectly classified



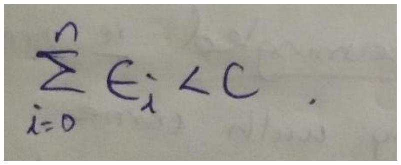
Question-3:

How do you measure the cost function in SVM? What does the value of C signify?

Answer-3:

The cost function in SVM is also known as the *cost of classification*, C. This is the sum of all the values of slack variables. This parameter represents the *cost of violations or misclassifications* to the margin and the hyperplane. The parameter is also called the tuning parameter of SVM and is one of the hyperparameters.

To choose the best support vector classifier, the total cost (sum of all Є's) should be as low as possible.

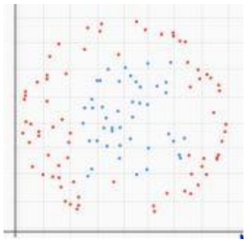


$$\sum_{i=0}^{n} \epsilon_i < C \, .$$

When C is large, the slack variables can be large i.e. many datapoints can be misclassified. Such a model has high bias – it is flexible, more generalizable and less likely to overfit.

When C is small, the slack variables can be small i.e. we do not allow many datapoints to be misclassified. Such a model has high variance – it is less flexible, less generalizable and more likely to overfit.

Question-4:

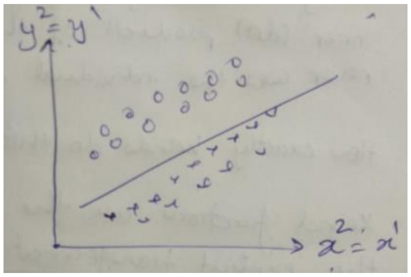


Given the above dataset where red and blue points represent the two classes, how will you use SVM to classify the data?

Answer-4:

Looking at the datapoints, we can see that these cannot be linearly separated. As a result, we use *kernels* that enable linear SVM models to separate the non-linearly separated datapoints, like the one shown in the figure.

We can perform non-linear feature transformation to create new features that can be linearly separated. Looking at the above distribution, we can see that the separator needs to be elliptical or circular form. Therefore, we can perform the transformation $\frac{x^2}{a}+\frac{y^2}{b}=c$. This will convert the above attribute space into a linear feature space as shown below:



This process is known as feature transformation. However, as the number of attributes increase, performing SVM would become computationally expensive.

Therefore, we use *kernels* that find the best possible fit model using SVM by using only the inner dot products of the observations. Kernels use this trick to bypass the explicit transformation process that we discussed above from the attribute space to create a linear feature space that can be separated using SVM.

There are different kernel functions that we can use, the 3 most common ones are: The linear kernel, the polynomial kernel, the radial basis function (RBF) kernel

Once we've selected the kernel, we can use grid search cross validation to choose the best combination of the hyperparameters (In non-linear SVM, we have 2 tuning parameters Gamma and C)

Question-5:

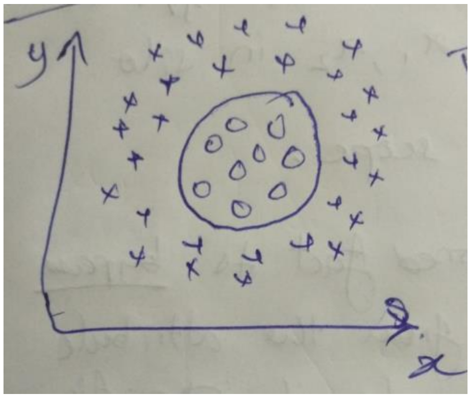
What do you mean by feature transformation?

Answer-5:

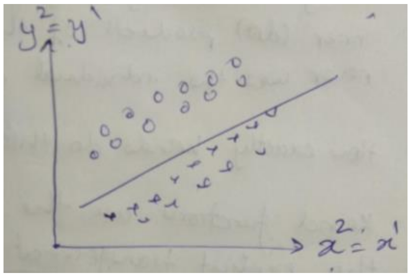
Feature transformation is essential in SVM when we're looking to perform classification for a no-linearly distributed data. Since SVM can only produce linear model, we are required to perform feature transformation on the non-linear datapoints to convert them into a linear feature space so that an SVM classifier can be implemented on it.

Example:

Consider the below distribution of points in the attribute space shown:



Looking at the above distribution, we can see that the separator needs to be elliptical or circular form. Therefore, we can perform the transformation $\frac{x^2}{a}+\frac{y^2}{b}=c$. This will convert the above attribute space into a linear feature space as shown below:



This process is known as feature transformation.

However, in the process of feature transformation, as the number of attributes increase, so do the number/combinations of the features that can be produced increases. This would make the SVM process computationally non-feasible and expensive.