Classification: Support Vector Machine (SVM)

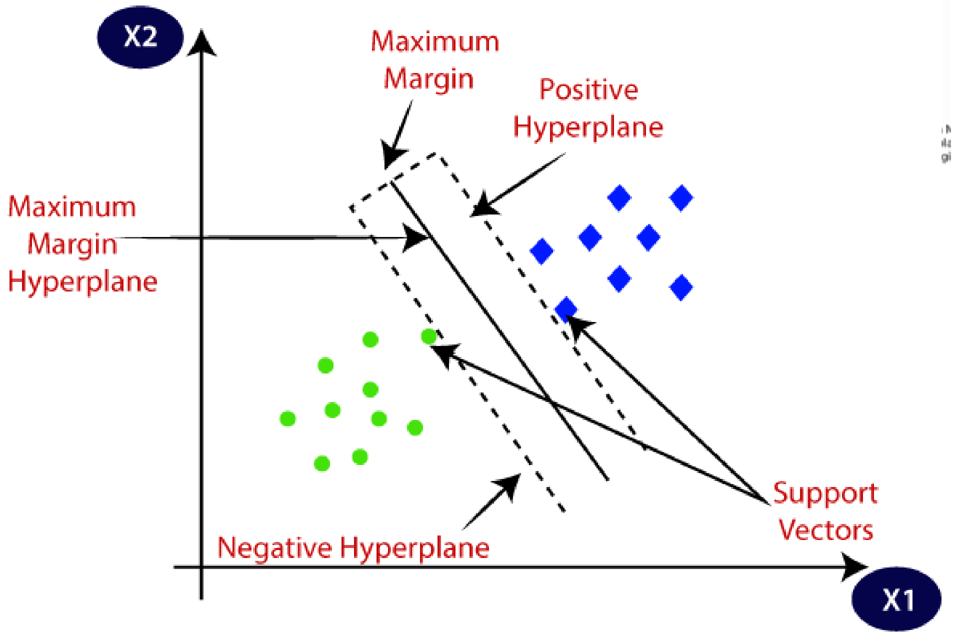
Penyusun Modul: Chairul Aulia

Editor: Citra Chairunnisa

Support Vector Machine

SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems (mostly for Classification).

SVM algorithm can perform really well with both linearly separable and non-linearly separable datasets. Even with a limited amount of data, the support vector machine algorithm does not fail to show its magic.



https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

Hyperplane

There can be multiple lines/decision boundaries to segregate the classes in n-dimensional space, but we need to find out the **best decision boundary** that helps to classify the data points. This best boundary is known as the hyperplane of SVM.

The dimension of the hyperplane depends upon the number of features. If the number of input features is two, then the hyperplane is just a line. If the number of input features is three, then the hyperplane becomes a 2-D plane.

We always create a hyperplane that has a maximum margin, which means the maximum distance between the data points.

Support Vectors

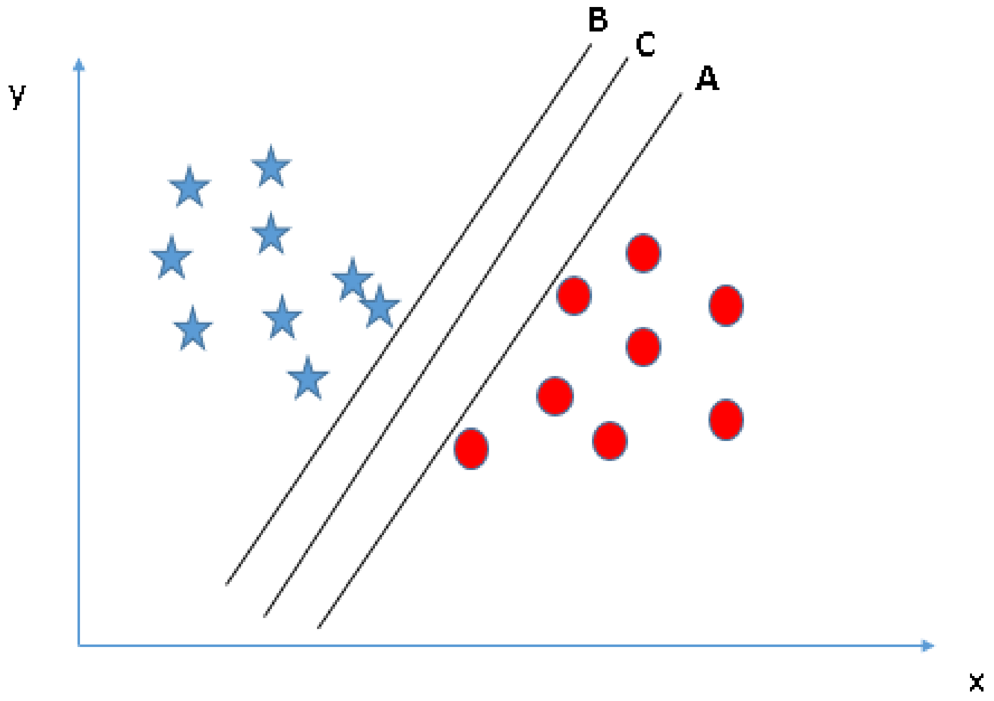
The data points or vectors that are the closest to the hyperplane and which affect the position of the hyperplane are termed as Support Vector. Since these vectors support the hyperplane, hence called a Support vector.

Types of SVM

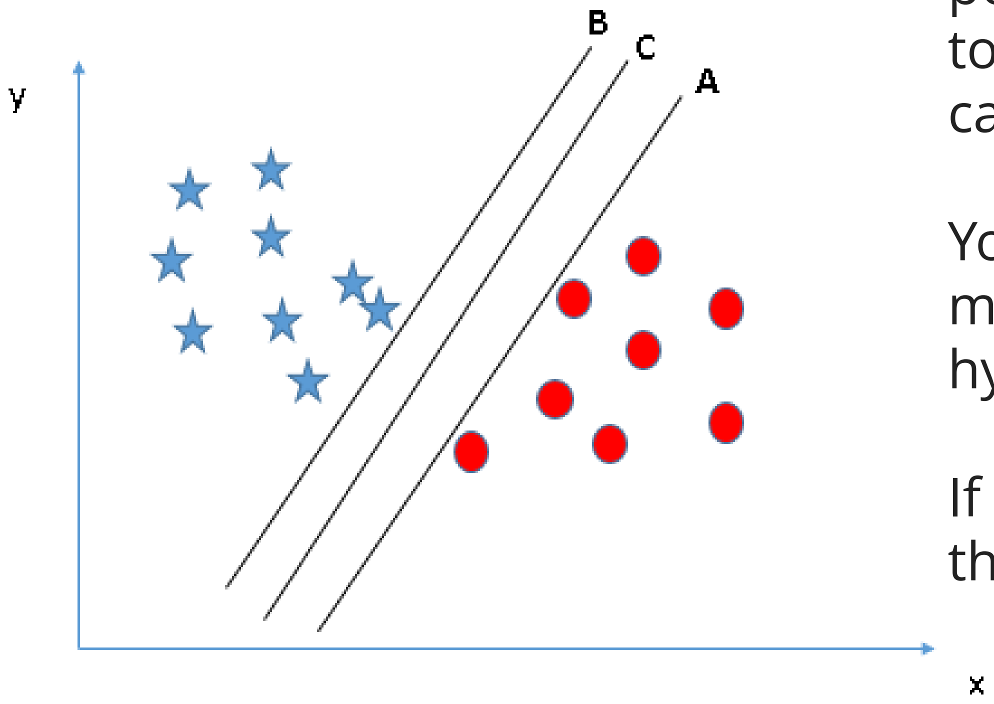
1. **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
2. **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

Choosing Hyperplane (Linear SVM)

Can you identify the right hyperplane in diagram below?



Choosing Hyperplane (Linear SVM)

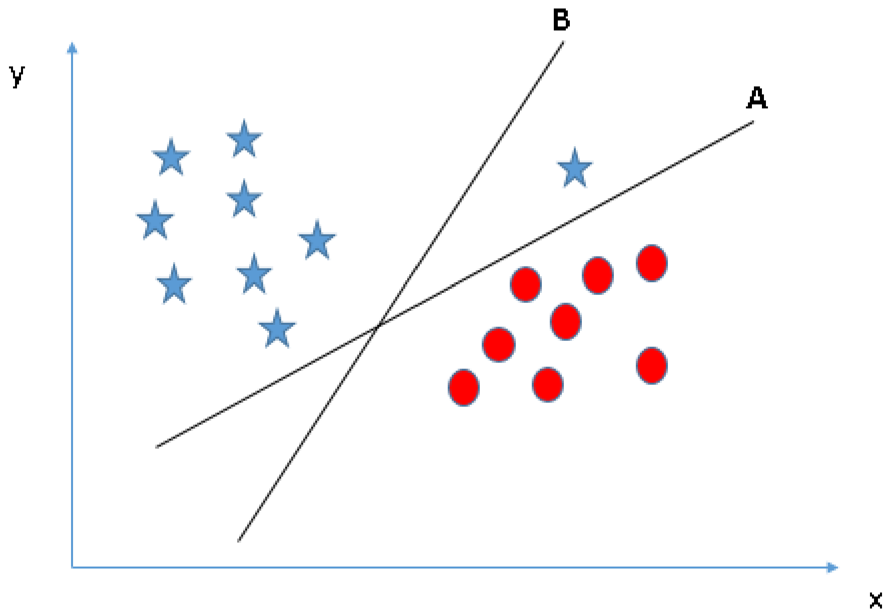


Maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called as **Margin**

You see that the **hyperplane C** has the highest margin to both classes, so it is the right hyperplane!

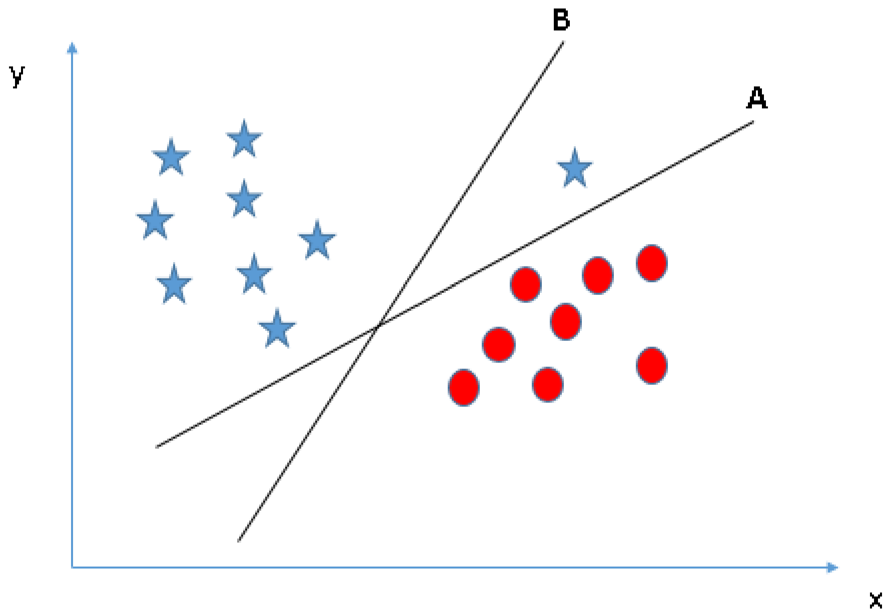
If we select a hyperplane having low margin, then there is high chance of miss-classification.

Choosing Hyperplane (Linear SVM)



How about the hyperplane for this scenario? Can you identify the right hyperplane?

Choosing Hyperplane (Linear SVM)

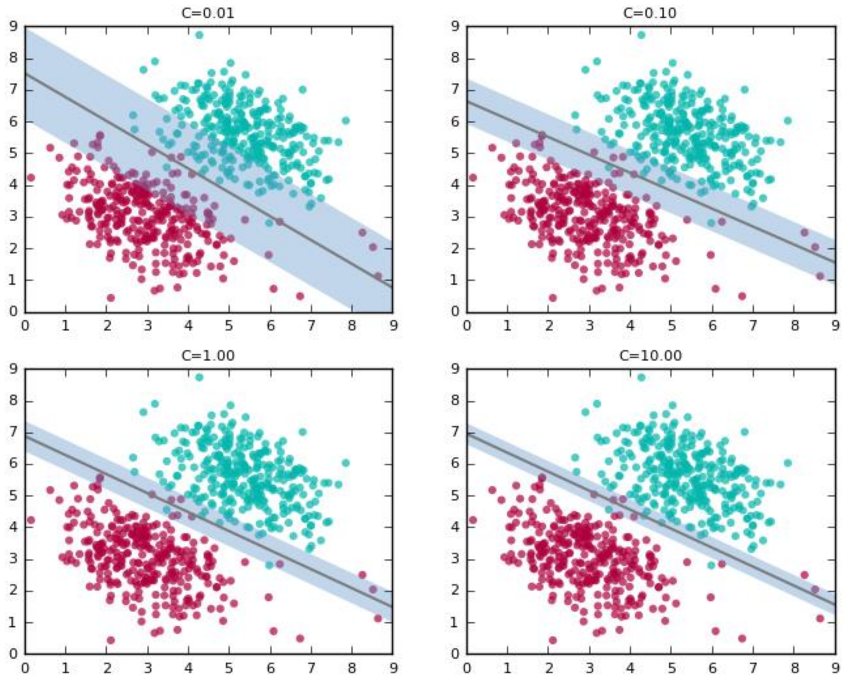


Some of you may have selected the hyper-plane **B** as it has higher margin compared to **A.**

But, here is the catch, SVM selects the hyperplane which classifies the classes accurately prior to maximizing margin.

Here, hyperplane B has a classification error and A has classified all correctly. Therefore, the right hyperplane is **A.**

Regularization Parameter



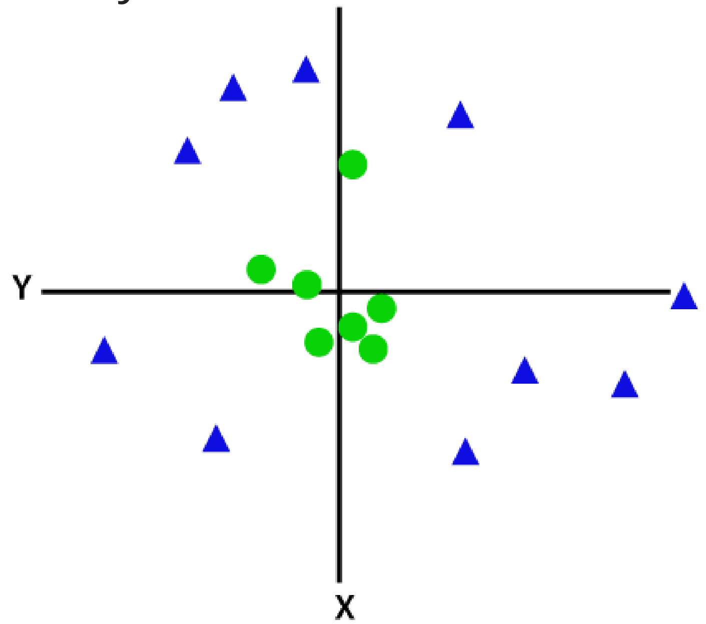
Real-world data is typically messy. You will almost always have a few instances that a linear classifier can't get right.

How do SVMs deal with this? They allow you to specify how many errors you are willing to accept. You can provide a regularization parameter called "C" to your SVM

An important practical problem is to decide on a good value of C. Since real-world data is almost never cleanly separable, this need comes up often. We typically use a technique like ***cross-validation*** to pick a good value for C (we'll cover it later).

Choosing Hyperplane (Non-Linear SVM)

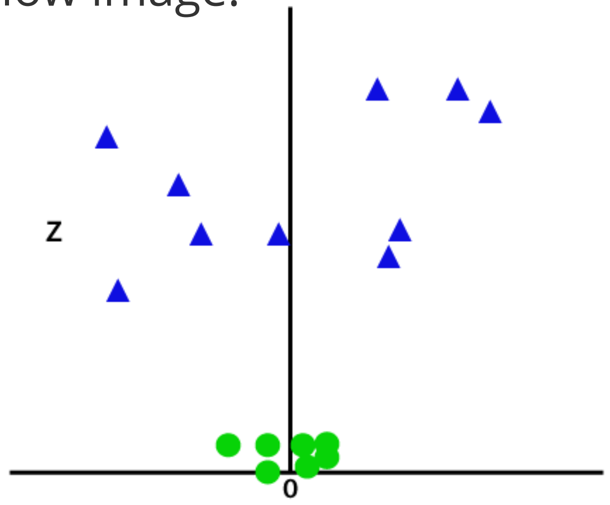
We can't have linear hyperplane between the two classes, so how does SVM classify these two classes?



So to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third dimension z. It can be calculated as:

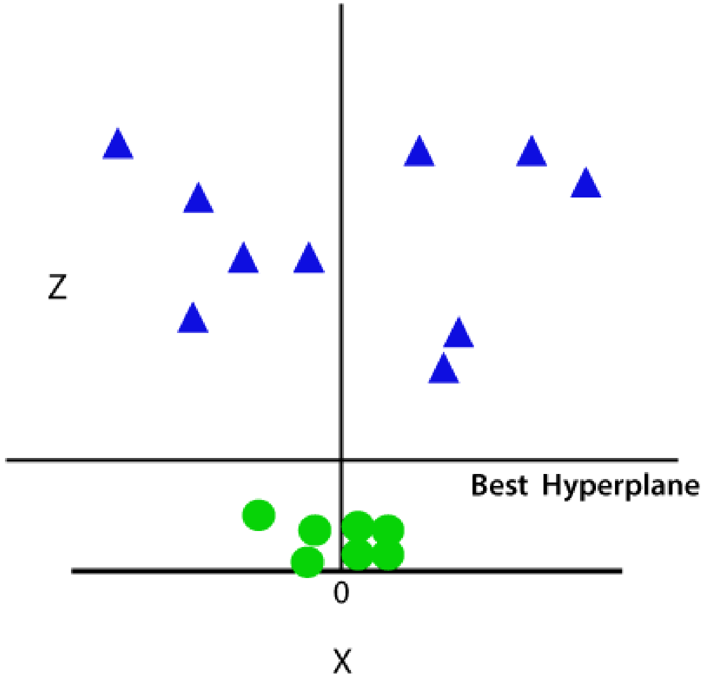
$z=x^2+y^2$.

By adding the third dimension, the sample space will become as below image:

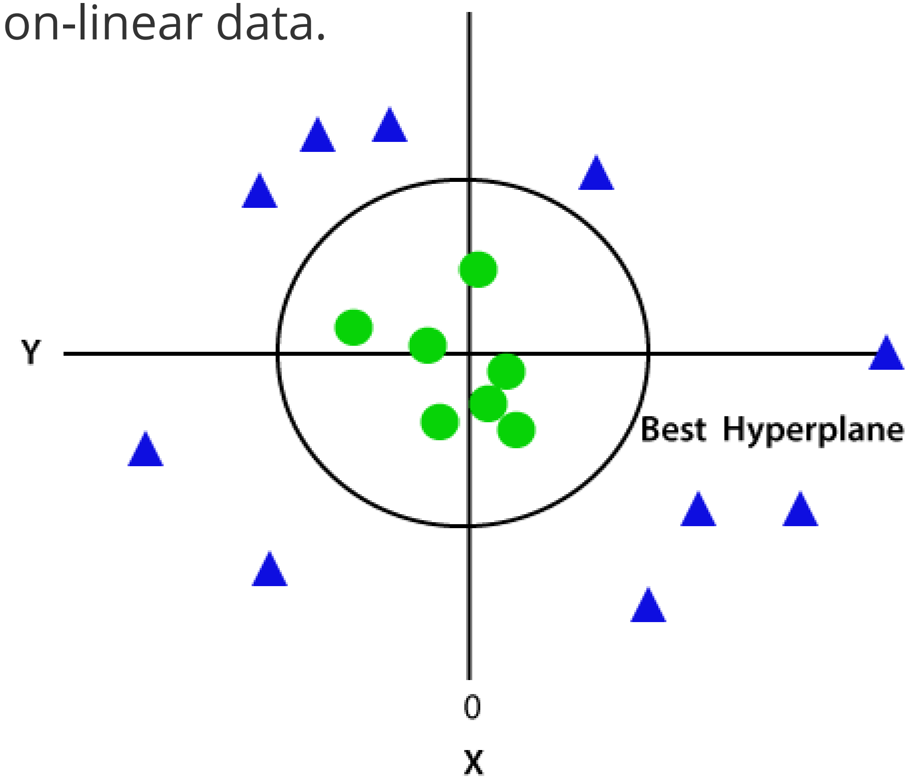


Choosing Hyperplane (Non-Linear SVM)

Since we are in 3-d Space, hence it is looking like a plane parallel to the x-axis. If we convert it in 2d space with z=r, then it will become as:



Hence we get a circumference of radius r in case of non-linear data.



What we have seen so far

1. For linearly separable data SVMs work amazingly well.
2. For data that's almost linearly separable, SVMs can still be made to work pretty well by using the right value of C.
3. For data that's not linearly separable, we can project data to a space where it is perfectly/almost linearly separable, which reduces the problem to 1 or 2 and we are back in business.

It looks like a big part of what makes SVMs universally applicable is projecting it to higher dimensions. And this is where kernels come in.

Kernel Trick

the SVM algorithm has a technique called the kernel trick.
The **SVM kernel** is a function that takes **low dimensional** input space and transforms it to a **higher dimensional** space i.e. it converts **not separable problem** to **separable problem**.

Simply put, it does some extremely complex data transformations, then finds out the process to separate the data.

But, surprisingly mathematical formula to do the transformation is never show up in SVM.
Calculating the transformation can get pretty expensive computationally. One may deal with a lot of new dimensions, each possibly involving a complicated calculation. Hence, doing this for every vector in the dataset will be a lot of work.

Here's the solution: SVM does not need actual vectors to work its magic. It can get by with dot products between all pairs of points in the projected space.
And a function takes two points input in the original space, and directly gives us the dot product in the projected space

Use of Kernels

1. We typically don’t define a specific projection for our data. Instead, we pick from available kernels, tweaking them in some cases, to find one best suited to the data.
2. Of course, nothing stops us from defining our own kernels, or performing the projection ourselves, but in many cases we don’t need to. Or we at least start by trying out what’s already available.
3. If there is a kernel available for the projection we want, we prefer to use the kernel, because that’s often faster.

Examples of Kernels

1. **Linear Kernel**. It is **faster** than other functions. The linear kernel is mostly preferred when classification problems can be linearly separated.
2. **Polynomial Kernel**. It is a more generalized representation of the linear kernel. It **is not** as preferred as other kernel functions as it is **less efficient** and accurate.
3. **Radial Basis Function (RBF)** Kernel. It is one of the most preferred and used kernel functions in SVM. It is usually chosen for non-linear data. It helps to make proper separation when there is no prior knowledge of data.
4. **Sigmoid Kernel.** It is mostly preferred for neural networks. This kernel function is similar to a two-layer perceptron model of the neural network, which works as an activation function for neurons.

Advantages of SVM

1. SVM works relatively well when there is a clear margin of separation between classes.
2. SVM is more effective in high dimensional spaces.
3. SVM is effective in cases where the number of dimensions is greater than the number of samples.
4. SVM is relatively memory efficient as it uses a subset of training points in the decision function called support vectors

Disadvantages of SVM

1. Does not work well with larger datasets
2. Sometimes, training time with SVMs can be high
3. If the number of features is significantly greater than the number of data points, it is crucial to avoid overfitting when choosing kernel functions and regularization terms
4. Probability estimates are not directly provided by SVMs; rather, they are calculated by using an expensive fivefold cross-validation
5. It works best on small sample sets due to its high training time

Example

Let’s see implementation of SVM classification with Scikit Learn on your JupyterLab and try it out. Using historical data about patients diagnosed with cancer, we will predict whether cancer is benign or malignant

Let’s use breast-cancer-wisconsin.data dataset

Your Turn

Let’s classify whether a fruit is an orange or an apple!

Use **apples_and_oranges.csv** dataset for this