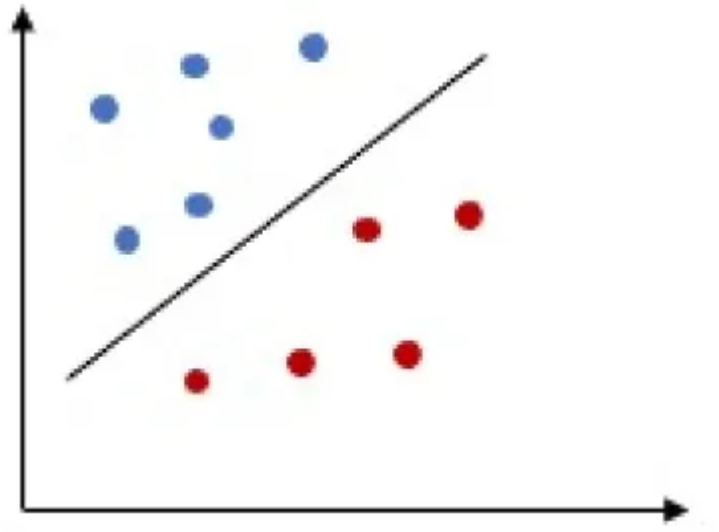
Non – Linear SVM

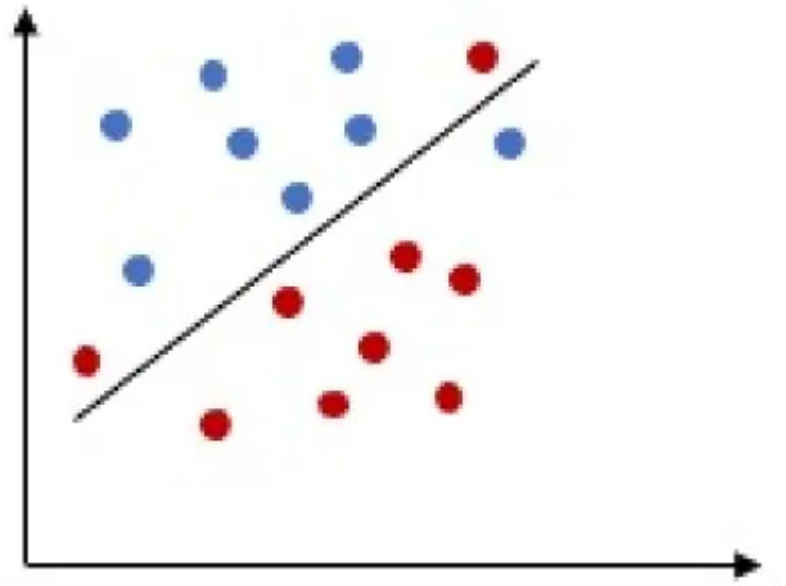
1. Linear SVM – Hard Margin Classifier

Here we will build our initial concept of SVM by classifying perfectly separated dataset (linear classification). This is also called “Linear SVM – Hard Margin Classifier”. We will define the objective function. This tutorial is dedicated for Hard Margin Classifier.



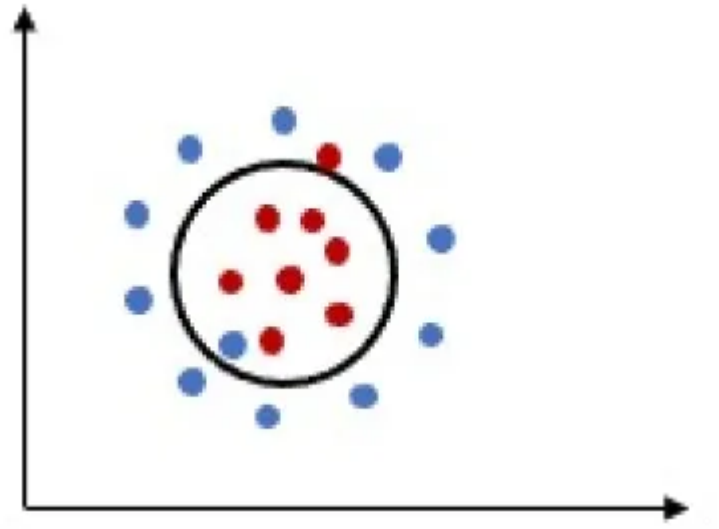
2. Linear SVM – Soft Margin Classifier

We will extend our concept of Hard Margin Classifier to solve for dataset where there are some outliers. In this case all of the data points cant be separated using a straight line, there will be some miss-classified points. This is similar of adding regularization to a regression model.



3. Non – Linear SVM

Finally we will learn how to derive Non-linear SVM using kernel. I will probably have a separate tutorial on kernel before this.



Additional Technical Concepts

Hyperplane

Margin
Functional Margin

Geometric Margin

Quadratic Programming

Lagrange Multiplier

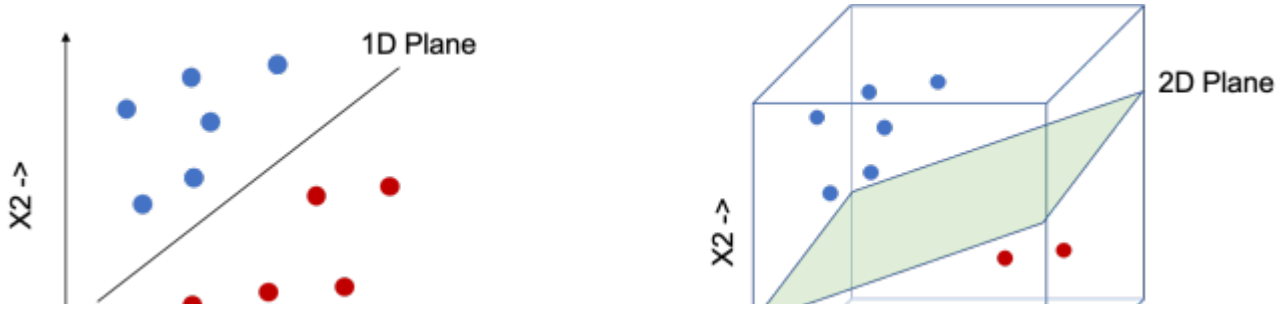
Kernel Methods

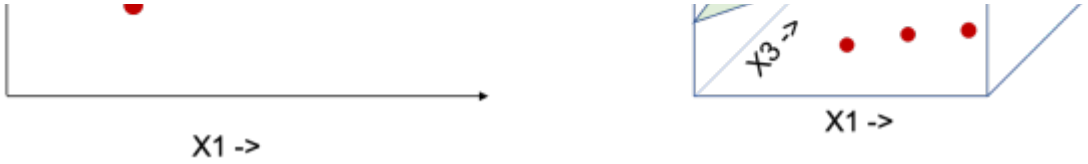
Maximal Margin Classifier

In order to explain Linear SVM many books or articles uses **Maximal Margin Classifier**. Hence it's good to understand the relationship between them. Linear SVM is a `generalization` of Maximal Margin Classifier. Remember that Maximal Margin Classifier does not have any practical use and its a theoretical concept.

Hyperplane

We can use a line to separate data which is in two dimension (Have 2 features `x1 and x2`). Similarly need a 2D plane to separate data in 3 dimension. In order to generalize the concepts, we will call them hyperplane, instead of line, plane or cube for n dimension of data, where `n > 0.`





Few important points to be noted for hyperplane,

A Hyperplane splits the original D – dimensional space into two half-spaces.

A dataset is `linearly separable` if each half space has points only from a single class.

Mathematical Definition of Hyperplane

Above we saw more of the intuition behind Hyperplane, here is the Mathematical Definition:

In a D-dimensional space, a hyperplane is a flat `affine subspace` of dimension (D-1)

If you are not familiar with **affine subspace**, please watch the video below for more clarification:

Equation of Hyperplane:

In `two dimension` we can represent the Hyperplane using the following equation. This similar to the equation of `affine combination`, however we have added the bias `b` here.

$$\beta_1 x_1 + \beta_2 x_2 + b$$

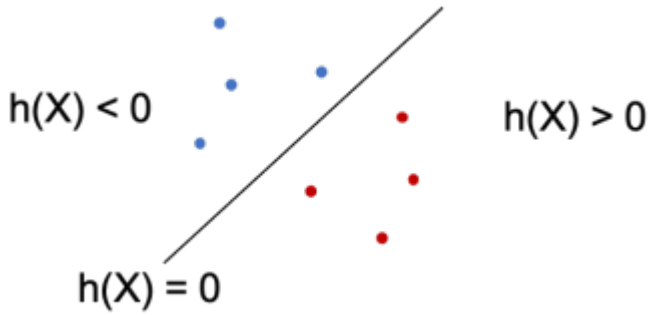
We can generalize this for d-dimensions and represent in vectorized form.

$$\begin{aligned} h(x) &= \beta_1 x_1 + \ldots + \beta_d x_d + b \\ &= \left(\sum_{i=1}^{d} \beta_i x_i \right) + b \\ &= \beta^T x + b \end{aligned}$$

For any point $X = (x_1, \ldots, x_d)$,

`h(X) = 0` then `X` lies on the hyperplane

`h(X) < 0 or h(X) > 0` then `X` falls to one side of the hyperplane.



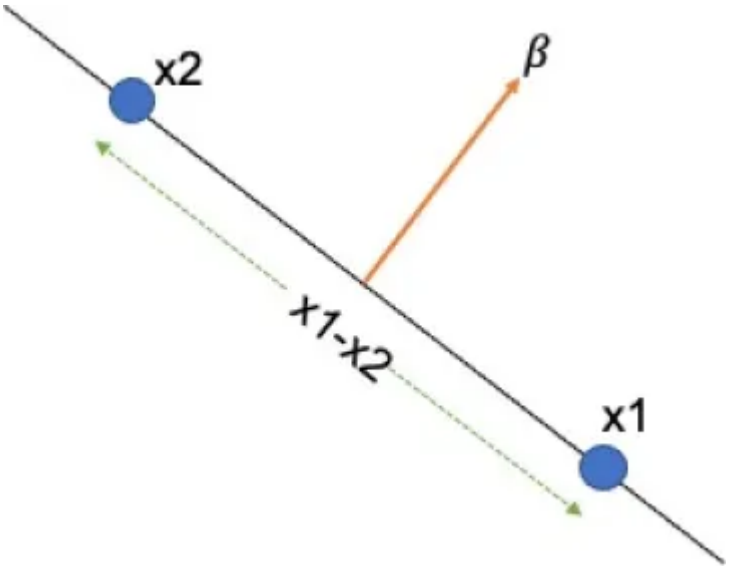
Separating Hyperplane

We will now make a very important assumption about the coefficient \ weight vector β. If x_1 and x_2 are two arbitrary points that lie on the hyperplane, we can write,

$$h(x_1) = \beta^T x_1 + b = 0$$
$$h(x_2) = \beta^T x_2 + b = 0$$
$$\text{Hence, } \beta^T x_1 + b = \beta^T x_2 + b$$
$$\beta^T (x_1 - x_2) = 0$$

We know if the dot product of two vectors is zero then the vectors are orthogonal to each other. Here the weight vector *beta* is orthogonal to $(x_1 - x_2)$. As $(x_1 - x_2)$ lie on the hyperplane, the *weight vector β is orthogonal to the Hyperplane too.*

We know if the dot product of two vectors is zero then the vectors are orthogonal to each other. Here the weight vector β is orthogonal to $(x_1 - x_2)$. As $(x_1 - x_2)$ lie on the hyperplane, the *weight vector β is orthogonal to the Hyperplane too.*



In other words, the weight vector β indicates the direction that is normal to the hyperplane. The bias **b** is the offset of the hyperplane in the d-dimensional space.

Maximal Margin Classifier

We can perform classification using a separating hyperplane. The sign of the $h(x_i)$ indicates whether the output label is `+1 or -1` and the magnitude defines how far the x_i lies from the Hyperplane.

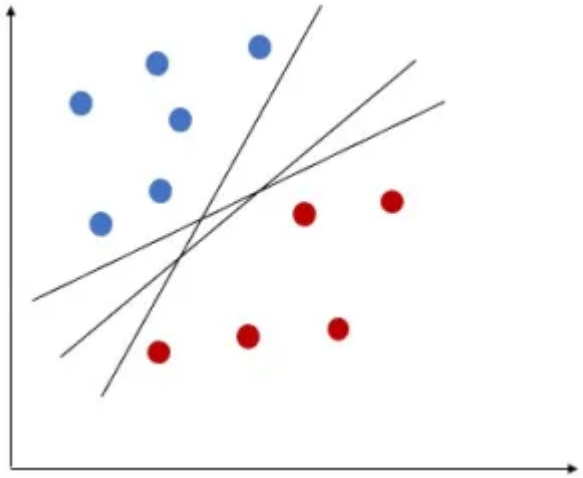
We know that,

$$h(X_i) = \beta^T X_i + b = 0$$

Now we can define two cases, one for +1 and another for -1. Then we can combine the two equations to one.

$$\left.\begin{array}{l} h(X_i) < 0 \text{ if } y_i = -1 \\ h(X_i) > 0 \text{ if } y_i = +1 \end{array}\right\} y_i(\beta^T X_i + b) > 0$$

How to choose a Hyperplane?



The above equation will help to find the classifier performance, however we need to find the best Hyperplane. If the data can be perfectly separated by a Hyperplane, then there can be infinite number of possible Hyperplanes, as a given separating hyperplane can be shifted/rotated. Hence we need to have a way to decide which Hyperplane to use.

What is Margin?

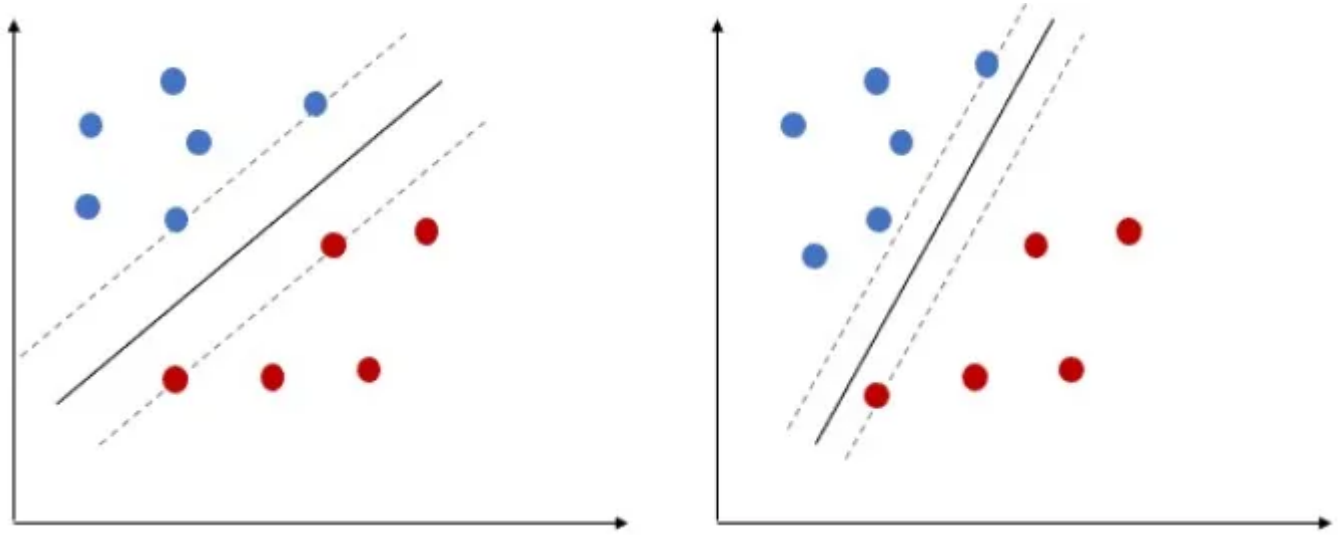
Margin can be defined using the `minimum distance` (normal distance) from each observations to a given separating hyperplane. Let's see how we can use Margin to find optimal Hyperplane.

Margin

Classifier Confidence

The size of the Margin defines the `confidence` of the classifier, hence the most wide margin is preferable.

Let's select two Hyperplanes, based on the margin. The first one has much wider margin than the 2nd one, hence the first Hyperplane is more optimal than 2nd one.



Finally, we can say, in `Maximal Margin Classifier`, in order to classify the data, we will use a

separating hyperplane, which has the `farthest` (max) `minimum` (min) distance from the observations. (Remember will still use the Margin to select the optimal separating Hyperplane)

The above statement is very important, make sure that you understand it. Watch the video or post comments if you need help.

Next, we will understand how we can define margin mathematically.

Margin

Use the observable dataset to find the best margin, so that the appropriate Hyperplane can be identified. We will discuss about two types of margin.

We can conceptually define the equation of margin (known as `Functional Margin`).

However later need to use geometry to re-define the equation (known as `Geometric Margin`).

Functional Margin

Functional Margin is used to define the theoretical aspect of margin. Given a training example (X_i, y_i), the Functional Margin of (β, b) w.r.t the training example will be,

$$y_i(\beta^T X_i + b) = \hat{\gamma_i}$$

Instead of the value being just greater than 0, we have defined a value for the margin using γ.

Hence we can define the following conditions:

if $y_i = 1$ then $\hat{\gamma_i} >> 0$

if $y_i = 0$ then $\hat{\gamma_i} << 0$

However there is an issue with Functional Margin, it's dependent on the values of β and `b`. So if we scale β and `b` (multiply with some scaler `s`), the equation of the hyperplane **does not change** but the `margin widens`.

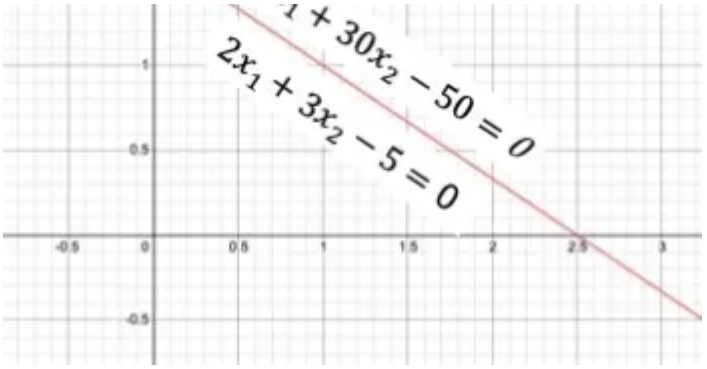
If you plot following two equations, they will represent the same hyperplane, however in our case the margin width will be different.

$2x_1 + 3x_2 - 5 = 0$

$20x_1 + 30x_2 - 50 = 0$

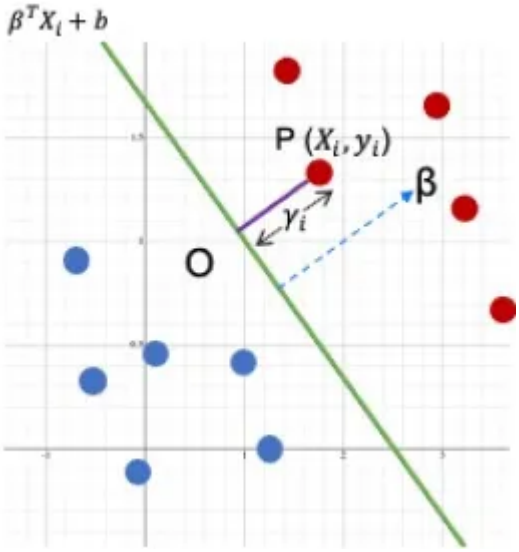
Here is the plot for both the equations.

We will use **Geometry** to redefine the equation of the margin in order to overcome this issue.



Geometric Margin

Now let's take help from Geometry to find the equation of the margin. Please refer the below diagram.



We will calculate the distance from point P to the hyperplane.

So assume a point $P(x_i, y_i)$ where $y_i = 1$

The minimum distance from the point P to the hyperplane is the normal distance PO

The distance of the line segment PO is γ_i

The equation of the hyperplane is $\beta^T X_i + b$

We already know that β is orthogonal (at 90 degree) to the separating hyperplane.

Hence can get the direction of β by using the unit vector $\frac{\beta}{||\beta||}$

Since PO will also have the same direction of β, we can calculate define point O as,

$$X_i - \gamma_i \cdot \frac{\beta}{||\beta||}$$

As point O lies on the separating hyperplane we can write the following equation as,

$$\beta^T\left(X_i - \gamma_i \cdot \frac{\beta}{||\beta||}\right) + b = 0$$

$$\beta^T X_i - \gamma_i \frac{\beta^T . \beta}{||\beta||} + b = 0$$

$$\gamma_i \frac{\beta^T . \beta}{||\beta||} = \beta^T X_i + b$$

We can express β using following equation,

$$||\beta|| = \sqrt{\sum_{i=1}^{D} \beta_i^2}$$

$$= \sqrt{\beta^T \beta}$$

$$\beta^T \beta = ||\beta||^2$$

Hence we can write,

$$\gamma_i \frac{||\beta||^2}{||\beta||} = \beta^T X_i + b$$

$$\gamma_i ||\beta|| = \beta^T X_i + \beta^T b$$

$$\gamma_i = \left(X_i \frac{\beta^T}{||\beta||} + \frac{b}{||\beta||}\right)$$

Finally, we can combine both positive and negative training examples in one equation,

$$\gamma_i = y_i\left(X_i \frac{\beta^T}{||\beta||} + \frac{b}{||\beta||}\right)$$

Geometric Margin

This is called the `Geometric Margin` and the normalizer $||\beta||$ does not let the margin widen even if we multiply using a scaler.

Going forward we will use this equation in all of our derivations.

Support Vectors of a Hyperplane

Given a training dataset `( X , y )` and a separating `Hyperplane`, we can find the distance between each point and the Hyperplane as,

$$\gamma_i = \frac{y_i h(x_i)}{||\beta||} = \frac{y_i(\beta^T x_i + b)}{||\beta||}$$

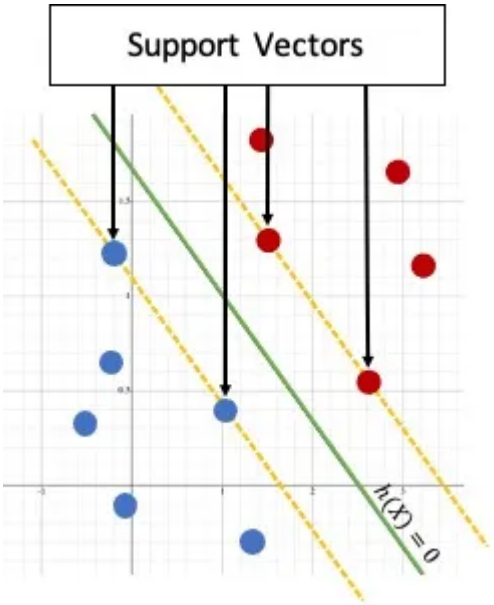
Over all the n points, we define the margin of the linear classifier as the minimum distance of a point from the separating hyperplane.

$$\gamma_i^* = \min_{x_i} \left\{ \frac{y_i(\beta^T x_i + b)}{||\beta||} \right\}$$

Why this algorithm is called

Support Vector Machine

All the points that achieve this minimum distance are called **Support Vectors** (x^*, y^*) for the Hyperplane. Thats the reason this algorithm is named as `Support Vector Machines`.



The Support Vectors lies precisely on the margin of the classifier.

$$\gamma_i^* = \frac{y_i^*(\beta^T x^* + b)}{||\beta||}$$

Canonical Hyperplane

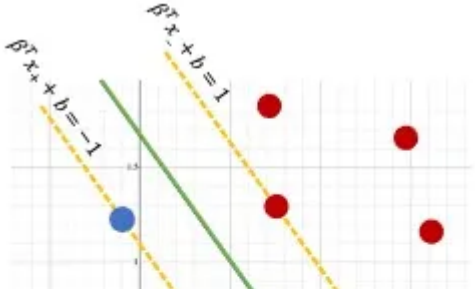
A Hyperplane is said to be Canonical w.r.t the support vectors if,

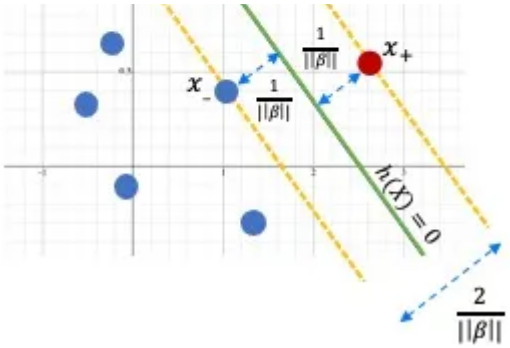
$$|\beta^T x^* + b| = 1$$

So the Geometric Margin will be,

$$\gamma_i^* = \frac{y^* h(x*))}{||\beta||} = \frac{y_i^*(\beta^T x^* + b)}{||\beta||} = \frac{1}{||\beta||}$$

We can represent this using a diagram.





This implies that the min distance between two classes will be at least $\frac{2}{||\beta||}$

For each support vector x_i^* we have $y_i^* h(x_i^*) = 1$ and for any other points which is not a support vector we can define a single combined equation (for both support vectors and other points),

$$y_i(\beta^T x_i + b) \geq 1$$

Linear SVM : Hard Margin Classifier

We will use the **Canonical Hyperplane** for Linear SVM (Hard Margin Classifier), which yields the maximum margins among all separating hyperplanes. So the goal of the objective function is to find the optimal hyperplane h^* :

$$h^* = \max_h \{\gamma_h^*\} = \max_{\beta,b} \left\{\frac{1}{||\beta||}\right\}$$

$$\text{Subject to the constraint : } y_i(\beta^T x_i + b) \geq 1$$

Instead of `maximizing` the margin $\frac{1}{||\beta||}$ we can also `minimize` $||\beta||$.

However, $||\beta|| = 1$ is a `non-convex optimization problem`, which means, there can be **many local optima**.

Hence, we can use following minimization formulation (which is convex in nature)

$$\text{Objective Function : } \min_{\beta,b} \left\{\frac{||\beta^2||}{2}\right\}$$

$$\text{s.t Linear Constraint : } y_i(\beta^T x_i + b) \geq 1$$

Optimization Problem

Primal Problem

This objective function is a `Primal Convex Optimization Problem` and it can be solved with **Linear Constraints using standard algorithms**.

However, this way we wont be able use the objective function to solve for `non-linear` cases. Hence we will find an equivalent problem named `Dual Problem` and solve that using `Lagrange Multipliers`.

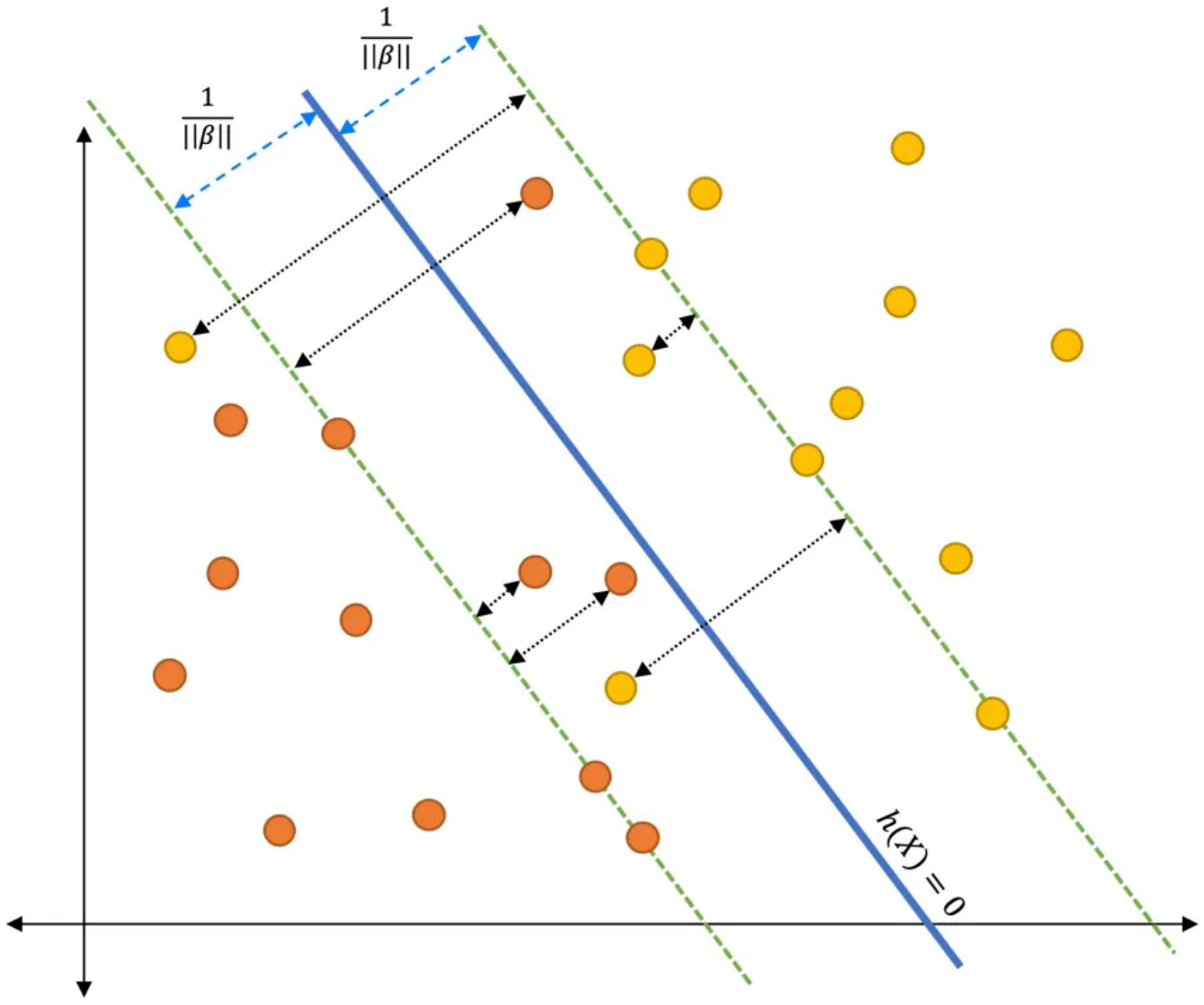
Linear SVM : Soft Margin Classifier

So far we have assumed that the dataset is **perfectly** linearly separable, which doesn't really happen in real scenario. Hence lets look at a little bit more complicated case. We are still working on Linear SVM, however, this time some of the classes overlap such way that a perfect separation is impossible, but remember the data is still linearly separable.

Here is an example of that. Consider a 2-dimension dataset. Thera are mainly two possibilities,

Single outlier can push the decision boundary greatly, so that the margin becomes very narrow.

Even though a linear decision boundary can classify the target classes properly, the data may not be separable using a straight line (no clear boundary)



In other words, previously derived Hard Margin Classifier wont work due to the `inequality constraint` $y_i(\beta^T x_i + 1) \geq 1$

Slack Variable

We can solve this problem by introducing a new variable named Slack Variable and then redefine our `inequality constraint` as,

$$y_i(\beta^T x_i + b) \geq 1 - \xi_i$$

where $\xi_i \geq 0$ is the `Slack Variable` for the point x_i.

Slack Variable

The Slack Variable indicates how much the point can violate the margin.

In case of `Canonical Hyperplane`, the point may not be at least $\frac{1}{||\beta||}$ away from the hyperplane.

The Slack Variable helps to define 3 types of data points:

if $\xi = 0$ then the corresponding point ξ is on the margin or further away.

if $0 < \xi < 1$ then the point ξ is within the margin and classified correctly (Correct side of the hyperplane).

If $\xi \geq 1$ then the point is misclassified and present at the wrong side of the hyperplane.

The ξ is the misclassification penalty. Hence we want to minimize it during optimization.

Objective Function

Out Hard Margin Classifier's Objective Function needs to also minimize the slack variable (misclassification penalty). We can write the objective function as,

$$\text{Objective Function}: \min_{\beta, b, \xi_i} \left\{ \frac{||\beta^2||}{2} + C \sum_{i=1}^{n} (\xi_i)^k \right\}$$

$$\text{s.t Linear Constraint}: y_i(\beta^T x_i + b) \geq 1 - \xi_i, \text{where } \xi_i \geq 0$$

`C` and `k` are constants which balances the cost of misclassification. The $\sum_{i=1}^{n} (\xi_i)^k$ is the loss term and C is a HyperParameter which controls the tread-off between maximizing the margin and minimizing the loss.

The HyperParameter `C` is also called as *`Regularization Constant`*.

If $C \to 0$, then the loss is zero and we are trying to maximize the margin.

If $C \to \infty$ then the margin does not have any effect and the objective function tries to just minimize the loss.

In other words, the Hyper Parameter `C` controls the relative weighting between the twin goals of making margin large and ensures that most examples have functional margin at least 1.

The value of C is chosen by Cross Validation during training time.

k is typically set to 1 or 2. If k = 1 , then the loss is named as `Hinge Loss` and if k =2 then its called `Quadratic Loss`. We will use both of these in later tutorials.

Gradient Descent

Understand that Gradient Descent can be used to optimize the objective function when `k = 1 (Hinge Loss)` by changing the inequality constraint to equality constraint.

Notes

I will explain how Gradient Descent can be used to optimize the objective function here, but this method does not help us to solve Non-Linear boundaries, hence this is not a preferred method. However it will be a good practice to see how this classifier might work (At my University the prof followed this approach).

As you might already know that we can't use Gradient Descent to optimize the Soft Margin Objective Function due to the inequality constraint. Hence need to first find a way to change the constraint to equality constraint.

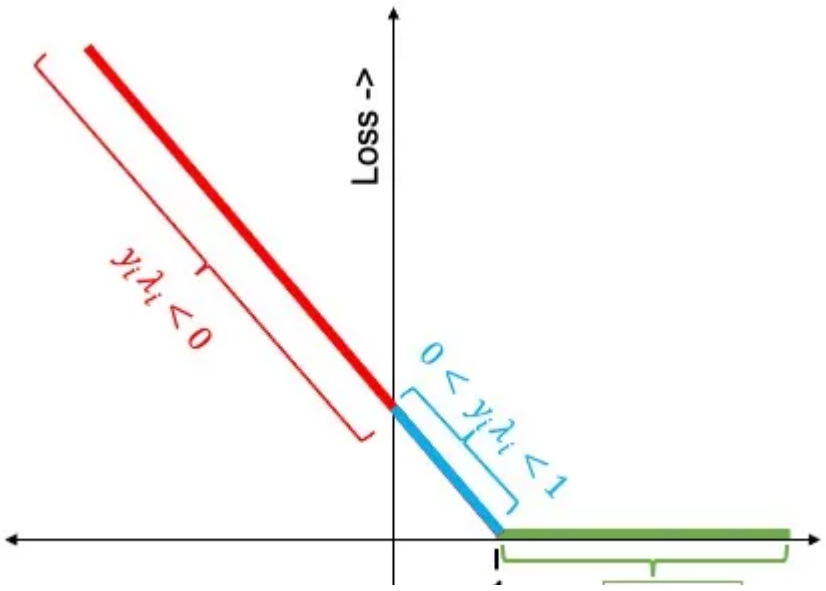
Lets define $\lambda_i = (\beta^T x_i + b)$. Here are some of the important points to consider,

If $(y_i \lambda_i) > 1$ then the classifier predicts the sign correctly (both of them have same sign) and x_i is far from the margin, hence there is no penalty/loss.

If y_i and λ_i have the same sign, but $\lambda_i < 1$ then x_i is in between the margin and Hyperplane. Event though the classifier predicts the sign correctly, there will be some penalty/loss. Moreover the penalty get larger as x_i gets closer to the Hyperplane.

If y_i and λ_i have different sign, then the penalty will be large and as x_i moves further away from the boundary on the wrong side the penalty/loss will increase linearly.

Now let's plot the penalty/loss using the above 3 statements. I have highlighted each section of the plot with the related three conditions.



Hinge Loss

`Hinge Loss` is referred in many books and you should remember the above plot. We will again talk about Hinge Loss in the next tutorial.

So as we don't need any loss when $y_i\lambda_i > 1$ we can change the inequality constraint to equality constraint by rewriting the equation $y_i(\beta^T x_i + b) \geq 1 - \xi_i$ in following way:

$$\xi_i = max\Big(0, 1 - y_i(\beta^T x_i + b)\Big)$$

We can incorporate this directly to the Objective Function itself and calculate the Loss Function as,

$$L = \frac{||\beta^2||}{2} + C\sum_{i=1}^{n} max\Big(0, 1 - y_i(\beta^T x_i + b)\Big)$$

In order to find the minima, we need to take derivative w.r.t β and `b` and then use them in Gradient Descent formula (Same as in Linear/Logistic Regression).

$$\frac{\delta L}{\delta \beta} = \beta - C\sum_{i=1,\xi_i \geq 0}^{n} y_i x_i$$

$$\frac{\delta L}{\delta b} = -C\sum_{i=1.\xi_i > 0}^{n} y_i$$