Lesson_11: Support Vector Machine (SVM) Classifier

Ali Aburas, PhD

Slide credit:Andrew Ng

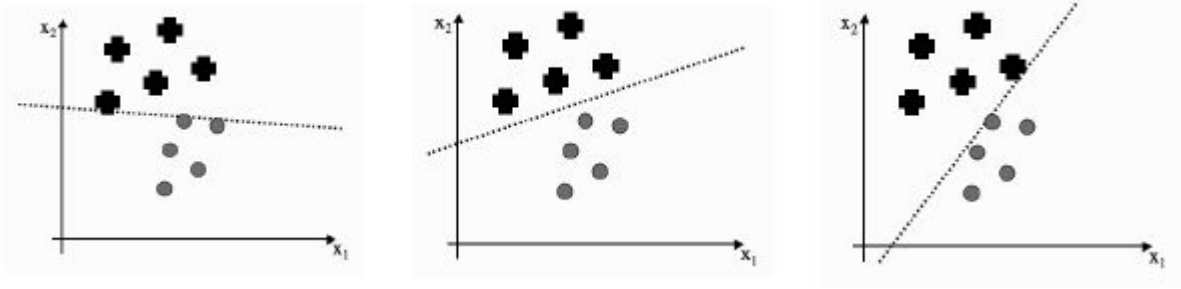
Outline

1. Intuition (Vladimir Vapnik)
2. SVM for linearly separable binary set

Support Vector Machine (SVM) Classifier

- **SVM** for linearly separable binary dataset.
- **SVM** is one of the most popular algorithms in modern machine learning.
- **SVM** is used both for ***classification and regression problems***. However, it mostly used in classification problems.
- **SVM** was introduced by Vapnik in 1992 and has taken off radically since then, principally because it often (but not always) provides very impressive classification performance on reasonably sized datasets.
- **SVMs** do not work well on extremely large datasets, since (as we shall see) the computations don't scale well with the number of training examples, and so become computationally very expensive.
- Main Goal to design a **hyperplane** that classify all training vectors into two classes
- The best model that leaves the maximum margin from both classes the two classes labels +1 (positive examples and -1 (negative examples)

OPTIMAL SEPARATION

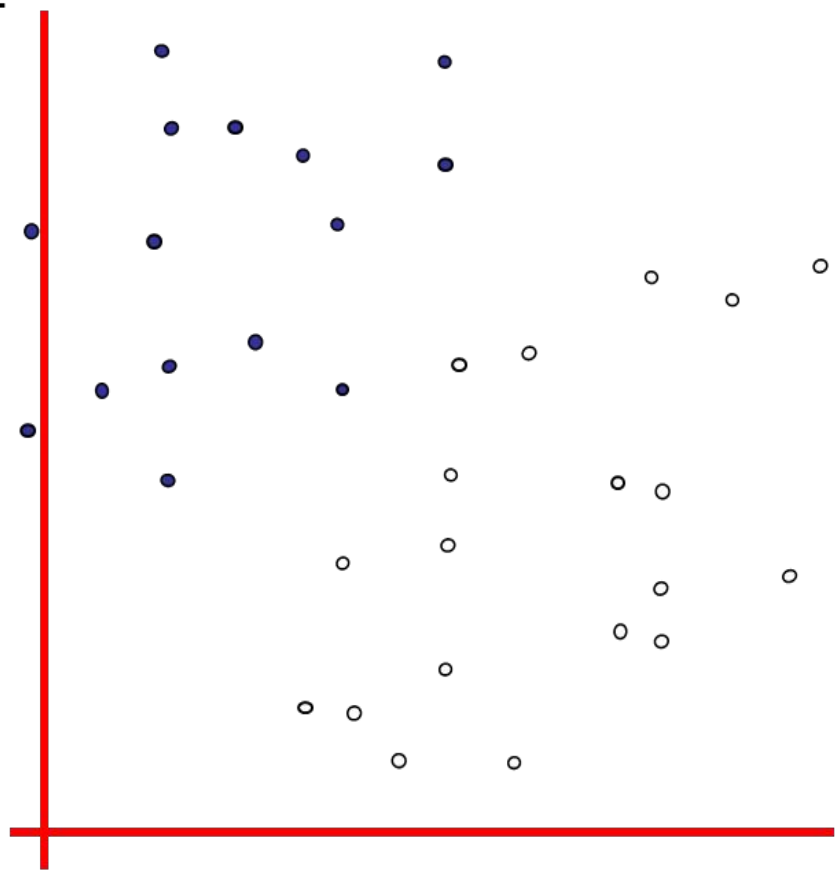


Three different classification lines. Is there any reason why one is better than the others?

- a simple classification problem with three different possible linear classification lines.
- All three of the lines that are drawn separate out the two classes, so in some sense they are ‘correct’,
- and the model (e.g., LR) would stop its training if it reached any one of them.
- However, if we had to pick one of the lines to act as the classifier for a set of test data, we would pick the line shown in the middle picture.
- we prefer a line that runs through the middle of the separation between the datapoints from the two classes, staying **approximately equidistant from the data in both classes**.

Linear SVM Classification

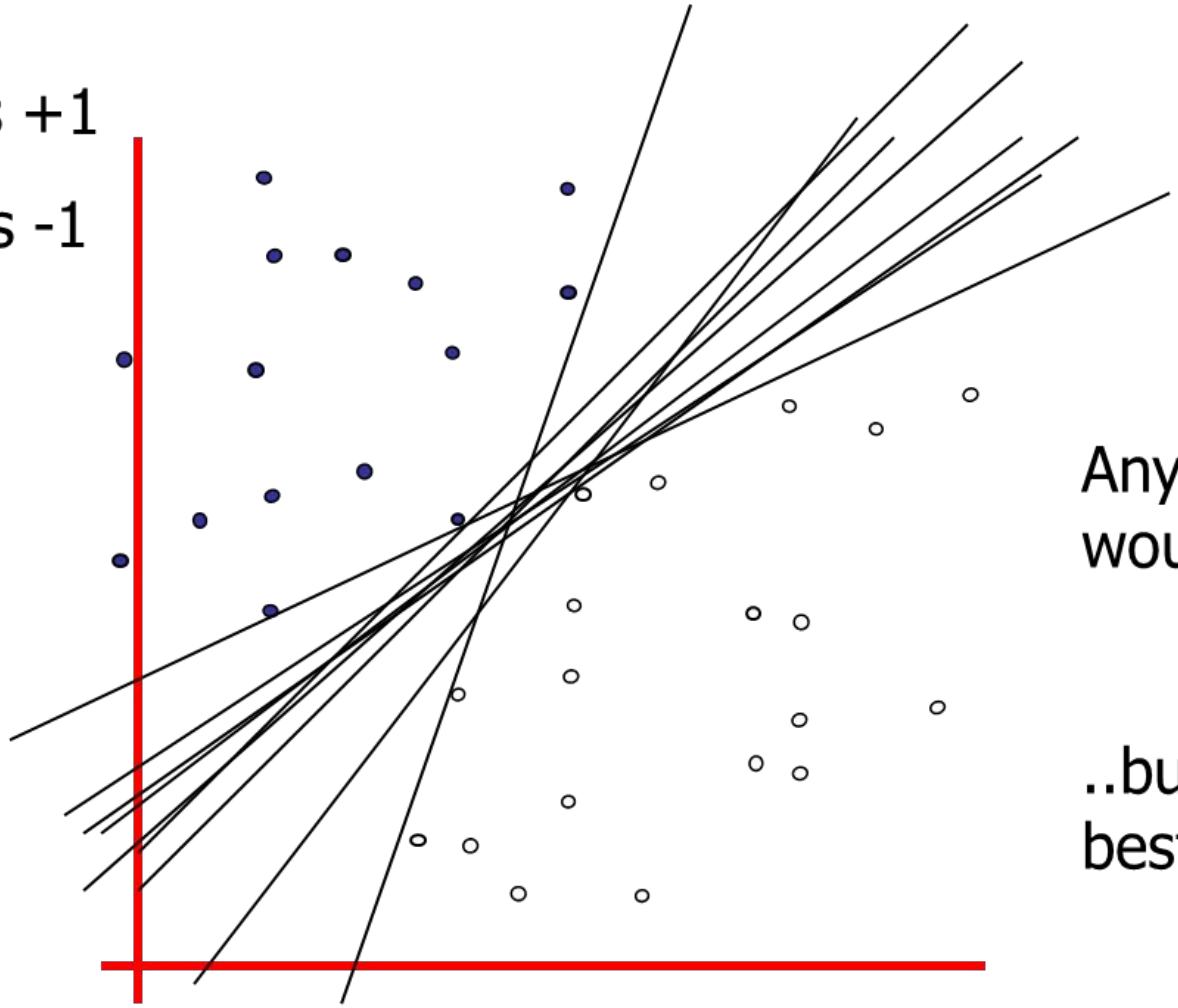
- denotes +1
- denotes -1



How would you classify this data?

Linear SVM Classification

- denotes +1
- denotes -1

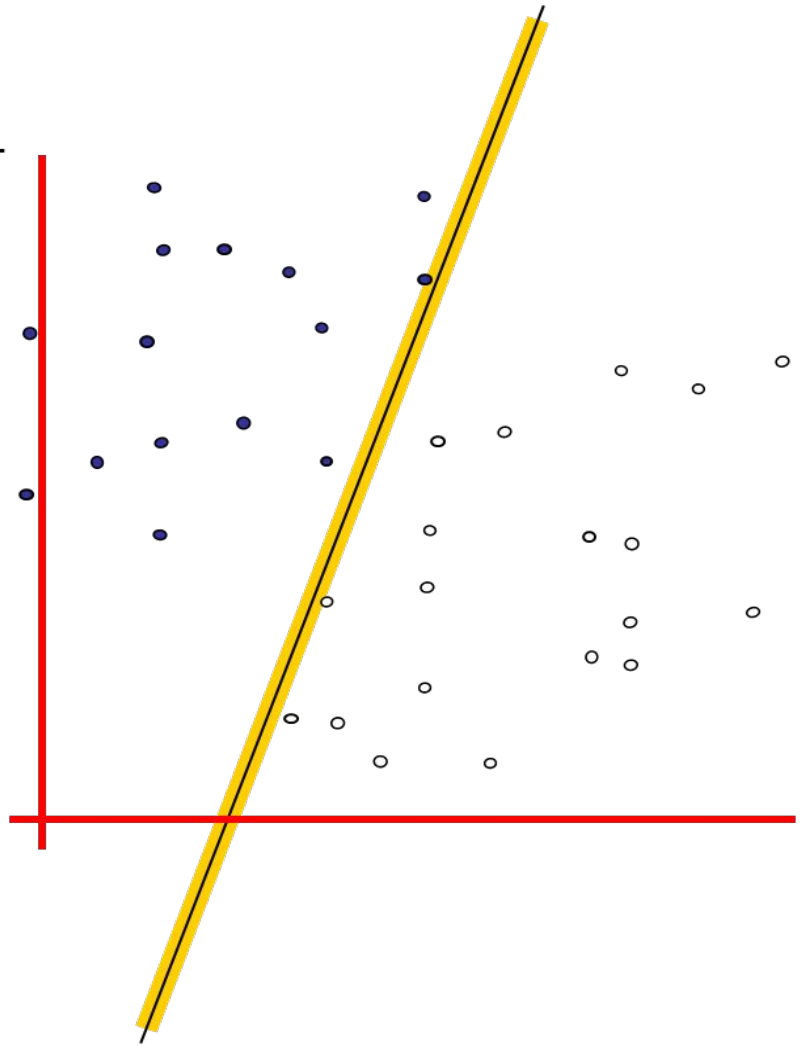


Any of these would be fine..

..but which is best?

Linear SVM Classification

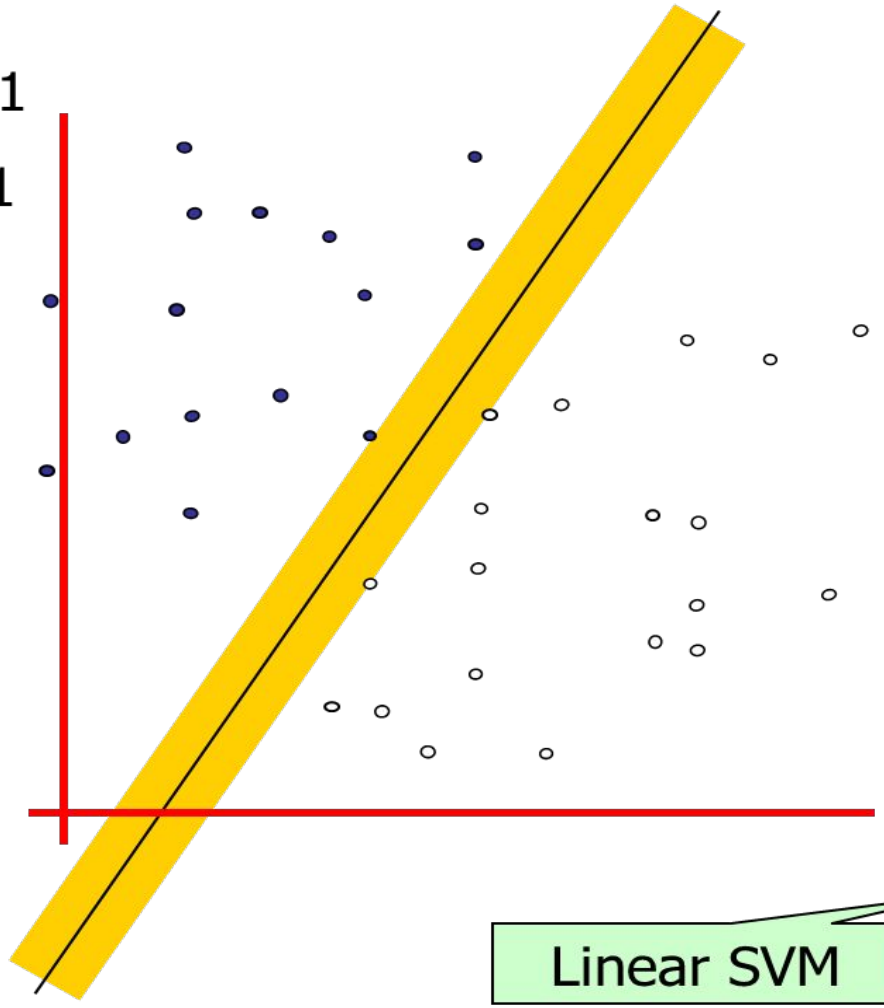
- denotes +1
- denotes -1



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

Linear SVM Classification

- denotes +1
- denotes -1



Linear SVM

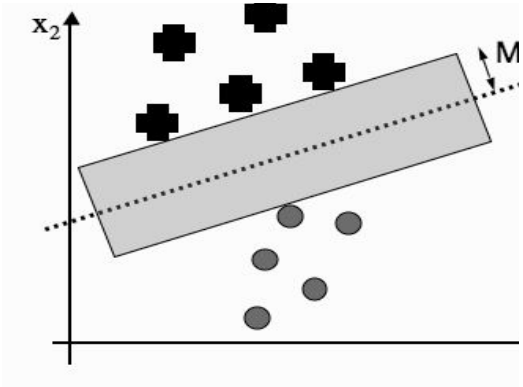
The maximum margin linear classifier is the linear classifier with the, um, maximum margin.

This is the simplest kind of SVM (Called an LSVM)

The Margin and Support Vectors

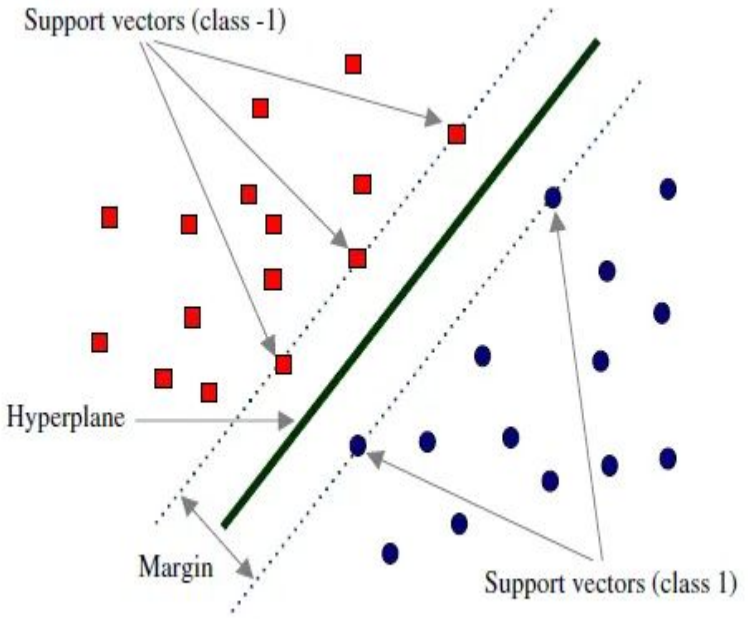
How can we quantify this?

- We can measure the distance that we have to travel away from the line (in a direction perpendicular to the line) before we hit a datapoint.
- This largest radius is known as the ***margin***, labelled M.
- The margin is the largest region we can put that separates the classes without there being any points inside, where the box is made from two lines that are parallel to the decision boundary.
- The datapoints in each class that lie closest to the classification line are called **support vectors**.



Support Vector Machine

- The goal of the SVM algorithm is to create the best line or **decision boundary** that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future.
- This best decision boundary is called a **hyperplane**.
- **Hyperplane** is a line that separates the data points into 2 classes.
- SVM chooses the extreme **points/vectors** that help in creating the hyperplane. These extreme cases are called as **support vectors**,
- **Support Vectors** are the data points which lie nearest to the hyperplane.
- and hence algorithm is termed as Support Vector Machine.



Support Vector Machine Classifier

Let's assume that we have only two independent variables.

A **hyperplane** is a plane in space that is used to separate the two data points classes. The main task of the SVM model is to find the best hyperplane to classify data points.

The hyperplane can be expressed by the following equation.

$$y = \omega^T x + b$$

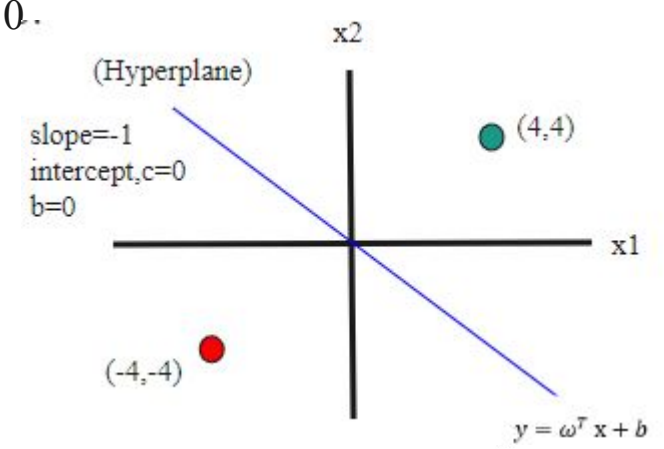
W is a vector of constants that represent the slopes of the plan.

In this example, the ***(w)*** **vector can be represented with two constants [-1 and 0].**

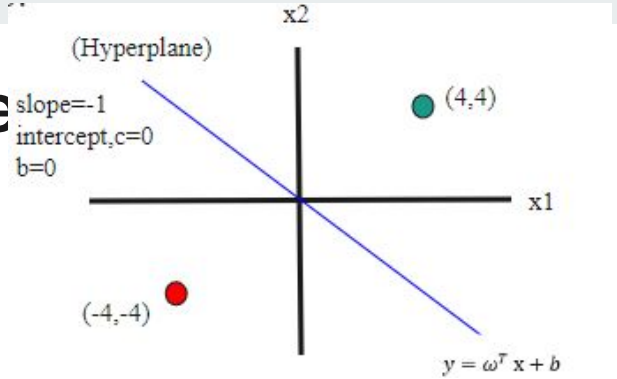
Assuming that the hyperplane is passing through origin so, b = 0 .

Let's take two data points, one from each class

(x1 = 4, x2 = 4) and (x1 = -4, x2 = -4)



Margin in Support Vector Machine



Hyperplane equation: $y = \omega^T x + b$

Let's take two data points, one from each class

For the first data point (x1 = 4, x2 = 4) after substituting it to the hyperplane equation

$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [4 \quad 4]$ y = -1 * 4 + 0* 4 = -4 → negative

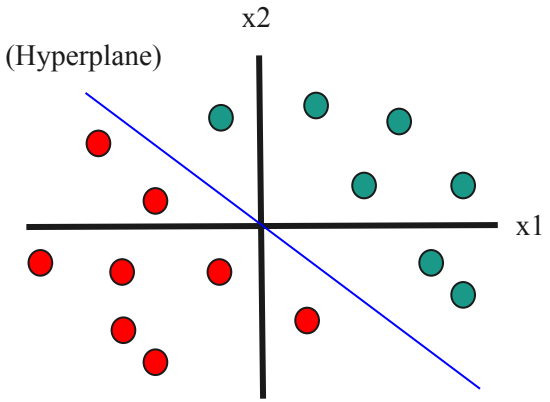
For the second data point (x1 = -4, x2 = -4), the value of y will be positive.

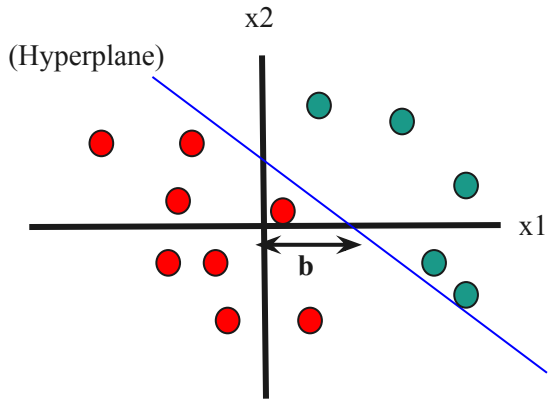
$y = \begin{bmatrix} -1 \\ 0 \end{bmatrix} [-4 \quad -4]$ y = -1 * -4 + 0* -4 = 4 → positive

Inference:

1. Any data point at one side of the hyperplane, $\omega^T x$ - is always positive
2. Any data point at one side of the hyperplane, $\omega^T x$ - is always negative

Support Vector Machine Classifier





- In the context of Support Vector Machines (SVM), the **bias term (b)** represents the **offset** of the **hyperplane from the origin** in the feature space. It is also known as the **intercept** term.
- The **value of the bias term** is determined during the **training process** of the SVM algorithm.
- The goal of the training process is to find the **optimal hyperplane that separates the classes with the largest margin,** and **the bias term is part of the parameters** that are adjusted to achieve this.
- Once the SVM model is trained, the value of the bias term is known and can be used to make predictions for new data points.

Optimization for Maximum Margin

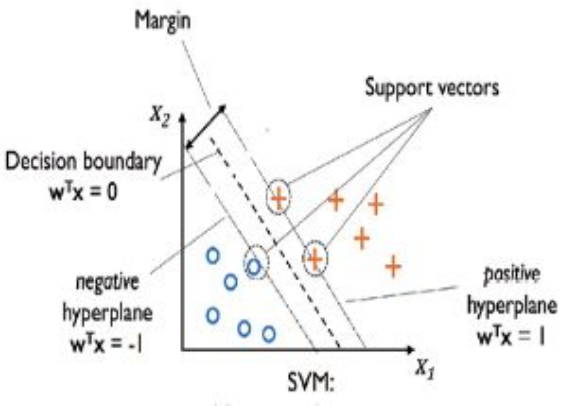
We need to find the equation of the hyperplane and the margin at the two sides of the hyper-plan in order to classify the data points.

Also, we need to find the optimization function used to find the best vector for hyperplane.

$\omega^T x + b = 0$ The equation of the hyper-plan

$\omega^T x + b = 1$ The equation of the margin line in the positive area.

$\omega^T x + b = -1$ The equation of the margin line in the negative area.



The distance between these two vectors x1 and x2 will be (x2-x1) vector.

$$\omega^T(x_2 - x_1) = 2$$

$$x_2 - x_1 = \frac{2}{\|\omega^T\|}$$

The model should find the values of w and b that maximize the following function.

$$(\omega^*, b^*) \max \frac{2}{\|w^T\|} \qquad \text{s.t.} \qquad y_i \begin{cases} +1 \text{ where } \omega^T x_i + b >= 1 \\ -1 \text{ where } \omega^T x_i + b <= -1 \end{cases}$$

Soft Margin SVM vs Hard Margin SVM

- In a **hard margin SVM**, the objective is to identify a hyperplane that completely separates data points belonging to different classes, ensuring a clear margin width possible.
- **Soft margin SVM** allows for some margin violations, meaning that it permits certain data points to fall within the margin or even on the wrong side of the decision boundary. Suitable for scenarios where the data may contain **noise or outliers**.
- The regularization parameter (C) controls the trade-off between these objectives.
  - A higher **C** value prioritizes a wider margin, even if it allows some misclassifications.
  - A lower **C** value allows for more margin violations to achieve a smoother decision boundary.

$$(\omega^*, b^*) \min \frac{\|w^T\|}{2} + \mathrm{C} \sum_{i=1}^{n} \varepsilon_i$$

C is the number of errors that will be misclassified

$\sum_{i=1}^{n} \varepsilon_i$ is the sum of the errors.

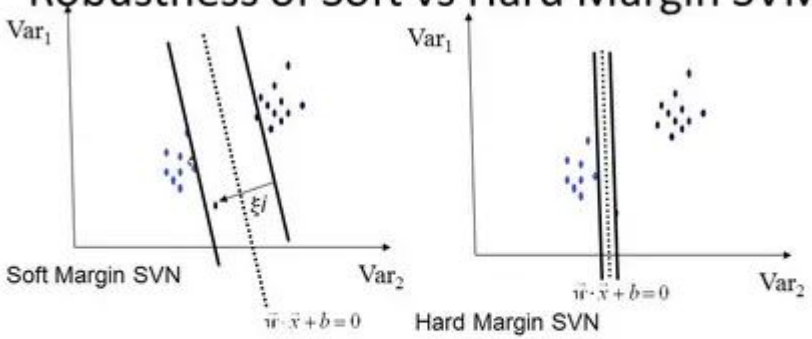
Soft Margin vs Hard Margin

$$(\omega^*, b^*) \min \frac{\|w^T\|}{2} + C \sum_{i=1}^{n} \varepsilon_i$$

C is the number of errors that will be misclassified

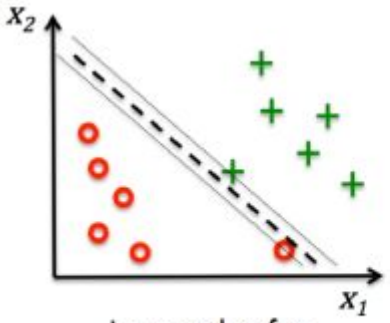
$\sum_{i=1}^{n} \varepsilon_i$ is the sum of the errors.

Robustness of Soft vs Hard Margin SVMs

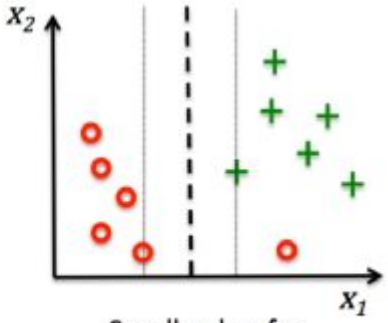


- Soft margin – underfitting
- Hard margin – overfitting

Trade-off: width of the margin vs. no. of training errors committed by the linear decision boundary



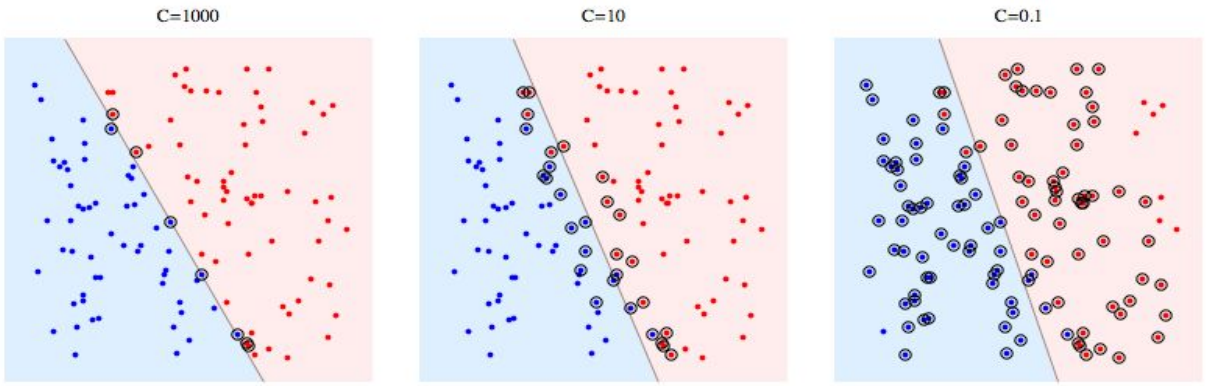
Large value for parameter *C*



Small value for parameter *C*

C (Regularization)

C is the penalty parameter, which represents misclassification or error term. The misclassification or error term tells the SVM optimisation how much error is bearable. when C is high it will classify all the data points correctly, also there is a chance to overfit.



Optimization for Maximum Margin

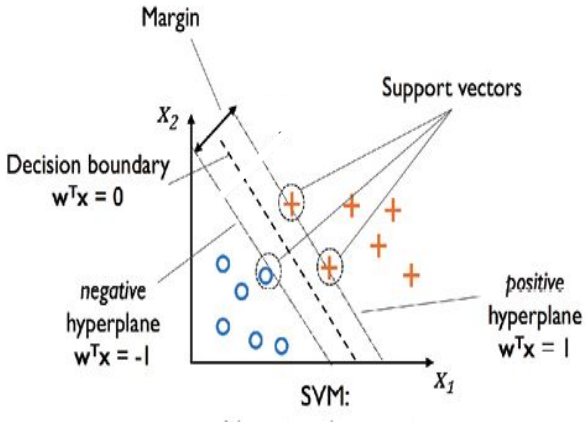
The model should find the values of w and b that maximize the following function.

$$(\omega^*, b^*) \max \frac{2}{\|w^T\|} y_i \begin{cases} +1 \text{ where } \omega^T x_i + b >= 1 \\ -1 \text{ where } \omega^T x_i + b <= -1 \end{cases}$$

It can be expressed in a simpler way as below.

$$(\omega^*, b^*) \max \frac{2}{\|w^T\|} y_i * (\omega^T x_i + b_i) >= 1$$

It can be expressed as minimize function as below.



$$(\omega^*, b^*) \min \frac{\|w^T\|}{2} + C \sum_{i=1}^{n} \varepsilon_i$$

C is the number of errors that will be misclassified

$\sum_{i=1}^{n} \varepsilon_i$ is the sum of the errors.

Note that, the term of error summation has been added to the optimization function to overcome the over-fitting problem.

Loss function (Hinge Loss)

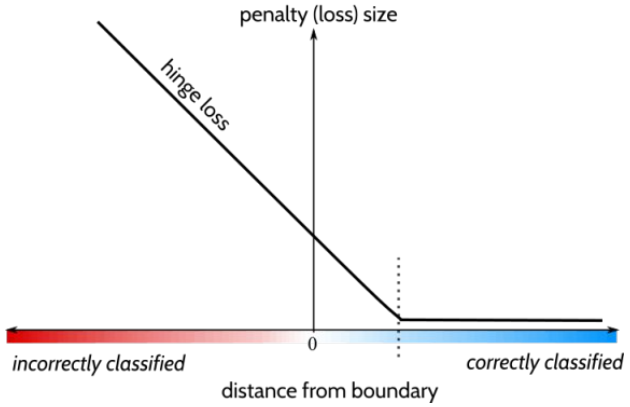
- **Hinge Loss** is one of the types of Loss function, mainly used for maximum margin classification models.
- Hinge Loss **incorporates a margin** or distance from the classification **boundary into the loss calculation**.
- Even if **new observations** (new data point) are **classified correctly**, they can incur a **penalty** if the margin from the decision boundary is not large enough.

The loss function is defined as:

L(y, f(x)) = max(0, 1 - y*f(x))

- where y is the true class label **(y = -1 or y = 1)** and
- **f(x)** is the predicted score for the positive class.

- If **y*f(x) >= 1**, then the loss is zero, which means that the prediction is correct.
- If **y*f(x) < 1**, then the loss is proportional to the distance from the correct prediction.



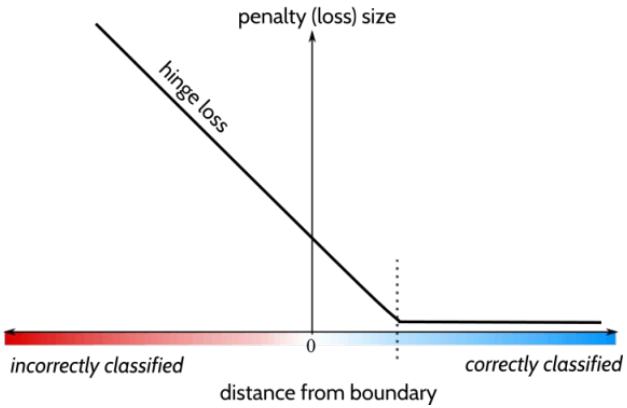
Loss function (Hinge Loss)

L(y, f(x)) = max(0, 1 - y*f(x))

- 0 - for correct classification
- 1- for misclassification

Example : **Misclassification**

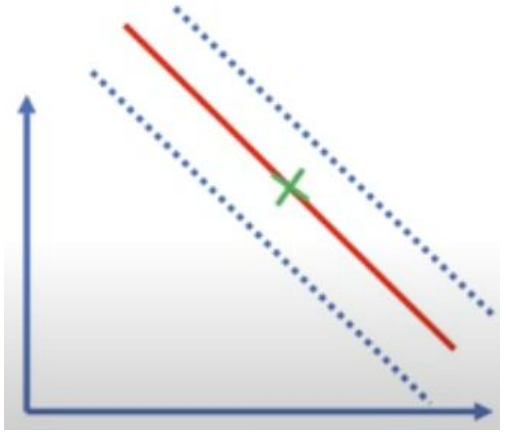
$y_i = 1 \quad \hat{y}_i = -1 \qquad y_i = -1 \; \hat{y}_i = 1$



- L=max(0, (1-(1)(-1)))
- L=2 (high loss value)

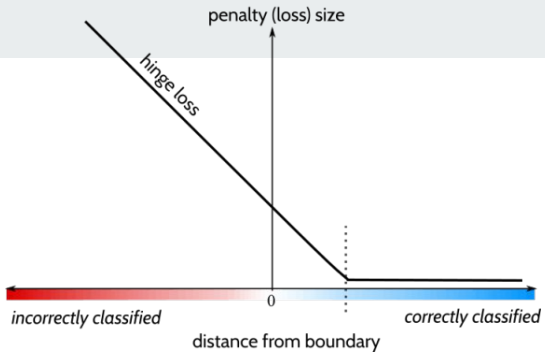
Correct classification

$y_i = 1 \quad \hat{y}_i = 1 \qquad y_i = -1 \quad \hat{y}_i = -1$



- L=(0, 1-(1)(1)))
- L=0

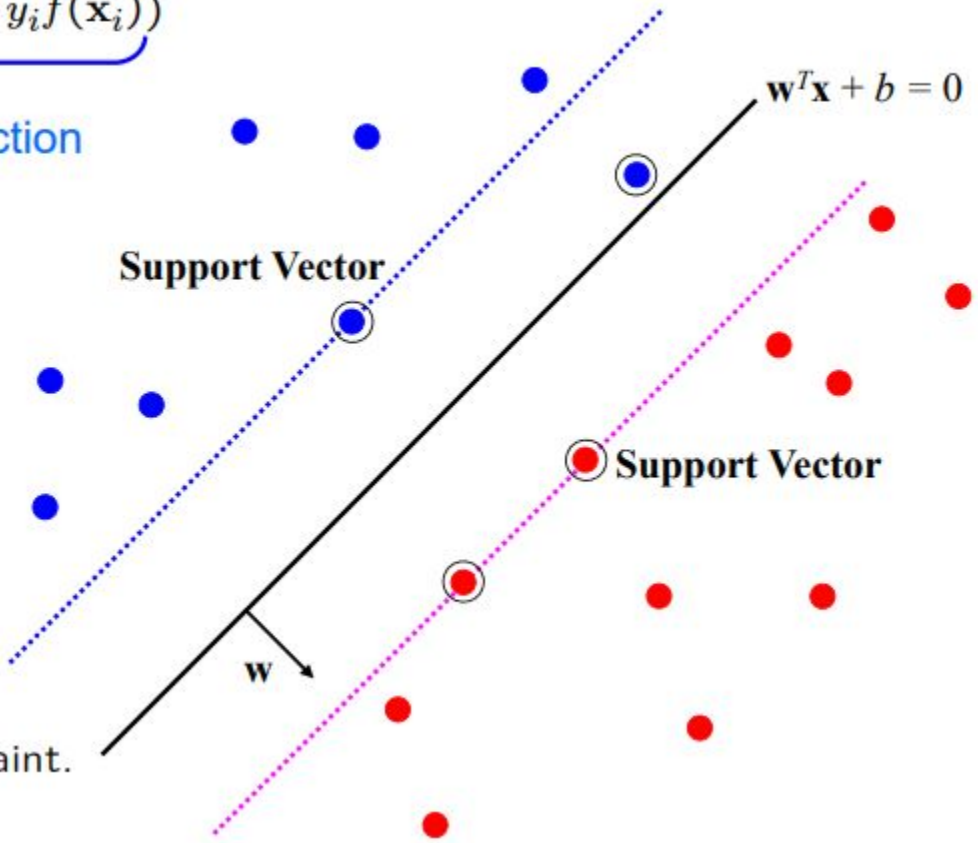
Loss function (Hinge Loss)



$$\frac{1}{2}\sum_{i=1}^{n}||w_i^2|| + C\sum_{i}^{N}\underbrace{\max(0, 1 - y_i f(\mathbf{x}_i))}_{\text{loss function}}$$

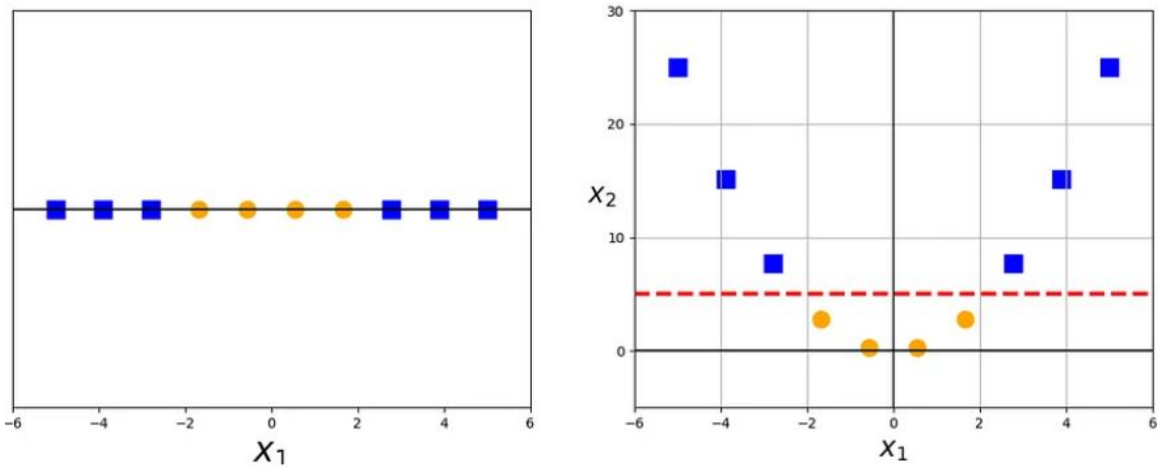
Points are in three categories:

1. $y_i f(x_i) > 1$
   Point is outside margin.
   No contribution to loss
2. $y_i f(x_i) = 1$
   Point is on margin.
   No contribution to loss.
   As in hard margin case.
3. $y_i f(x_i) < 1$
   Point violates margin constraint.
   Contributes to loss



Non-linear classification using SVM

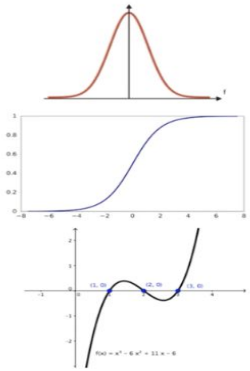
- Sometimes, in real-world problems, linear separation is not possible.
- In such non-linear classification, the input data is mapped into high-dimensional feature space using non-linear functions (**feature or kernel functions**), and linear classifier is then used for data classification.
- Some well-known kernel functions include Polynomial kernel, Gaussian kernel, sigmoid kernel, Radial basis function (RBF) kernel, etc. (sklearn.svm.SVC)
- **For example**:
  - In this example, the picture on the left shows our original data points.
  - In 1-dimension, this data is not linearly separable
  - but after applying the transformation $\phi(x) = x^2$ and adding this second dimension to our feature space, the classes become linearly separable



This data becomes linearly separable after a quadratic transformation to 2-dimensions.

Non-linear classification using SVM

Kernels: The main function of the kernel is to take low dimensional input space and transform it into a higher-dimensional space. It is mostly useful in non-linear separation problem.



Gaussian RBF Kernel

$$K(\vec{x}, \vec{l}^{i}) = e^{-\frac{\|\vec{x}-\vec{l}^{i}\|^2}{2\sigma^2}}$$

Sigmoid Kernel

$$K(X, Y) = \tanh(\gamma \cdot X^T Y + r)$$

Polynomial Kernel

$$K(X, Y) = (\gamma \cdot X^T Y + r)^d, \gamma > 0$$

Support Vector Machine (SVM)

- Advantages
  - Works well with smaller datasets
  - Accuracy

- Disadvantages
  - Isn’t suited to larger datasets as the training time with SVMs can be high
  - Less effective on noisier datasets with overlapping classes