Indian Institute of Technology, Kharagpur

PGDBA Programme

Department of Computer Science Engineering

Fundamentals of Algorithm Design and Machine Learning

Sachin Goyal: 24BM6JP46

Instructor: Dr. Aritra Hazra

Week 9 (20th Mar 2025) Scribe

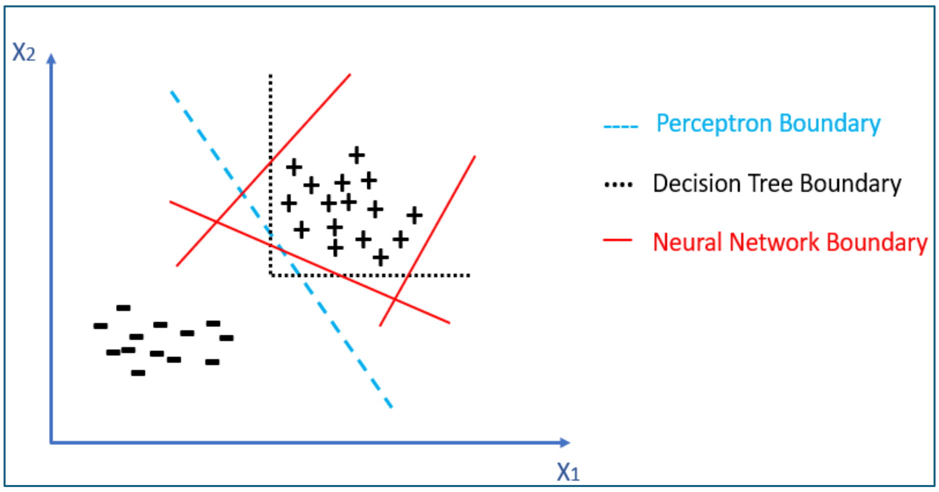
Until now, we have learned the following topics:

- What is Learning
- Can we learn it? (Is Learning Feasible?)
- How to do it (using DT, Perceptron, ANN, etc.)

Now, our focus will be on **"How to do it well".**

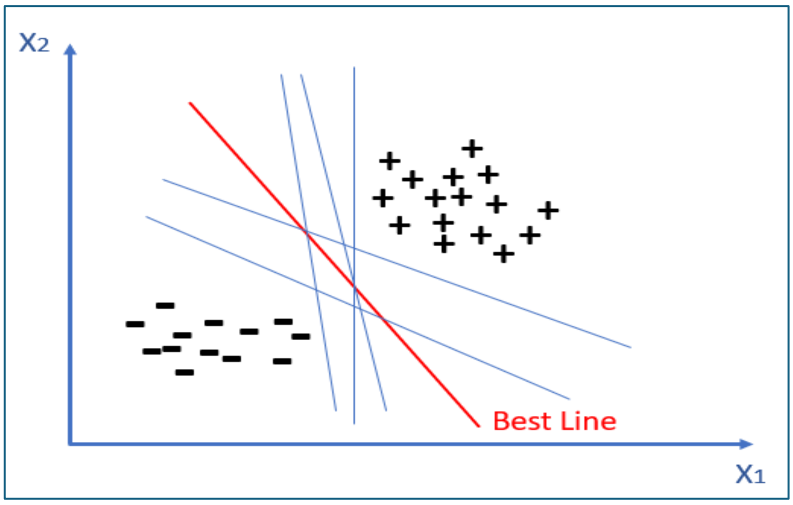
1. Linear Discriminant Analysis (LDA)

Given a set of points (in binary classification problem as in following image), we can either use Decision tree, or Perceptron Linear classifier or Neural Network to classify the points.



Now, which model to pick out of the 3 models, this question will be answered later on, (when we study evaluation metrics and validation set).

But for now, under LDA, we will focus on that amongst all the possible lines (where each line depicts a linear discriminant model in the following image), which line is the best for dividing/generalizing the dataset.



Any Linear discriminator can be represented as follows:

$$W^T x + b_1 = 0$$

For a 2D plane, $w_1 x_1 + w_2 x_2 + b_1 = 0$

For any new point $X_n = \langle a_1, a_2 \rangle$,

$$w_1 a_1 + w_2 a_2 + b_1 \geq 0 \quad \text{(Class +1)}$$

$$w_1 a_1 + w_2 a_2 + b_1 < 0 \quad \text{(Class -1)}$$

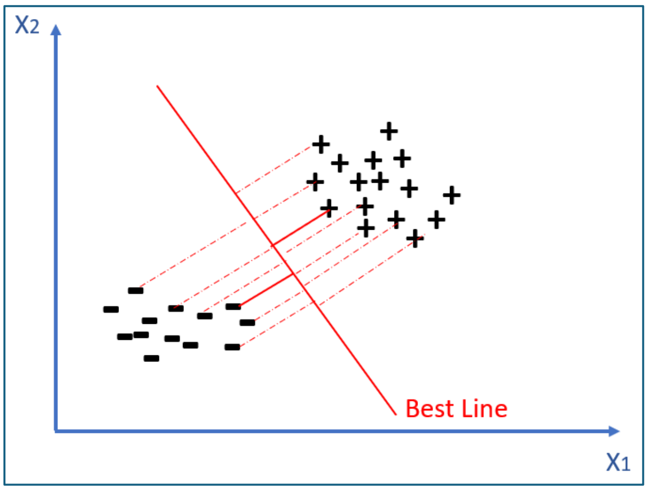
So, we can say for any Linear Discriminant,

$$\boxed{y_i(w_1 x_{i1} + w_2 x_{i2} + b_1) \geq 0} \cdots\cdots \boxed{\text{Equation 1}}$$

2. Defining the Best Line:

Intuitively, we can say that the Ideal linear discriminator (best line) will be the one that will be almost in the **middle** of both classes, but just saying middle seems very vague, so we will define formally what this **middle** actually means.

By placing the line in the **middle**, we mean that we have to select that line (out of all the possible lines) for which the minimum distance among all the points is maximized.



We know the perpendicular distance of a point from a plane is given by:

$$d_i = \frac{|w_1 x_{i1} + w_2 x_{i2} + b_1|}{\sqrt{w_1{}^2 + w_2{}^2}}$$

So mathematically we can write the above condition as:

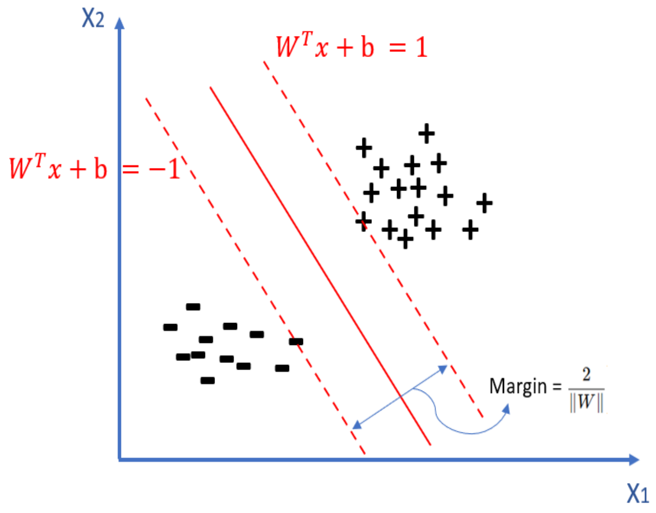
$$\text{Max}\,\{\min\,(d_i)\} = \max\left[\min_i \frac{w_1 x_{i1} + w_2 x_{i2} + b}{\sqrt{w_1^2 + w_2^2}}\right]$$

Here, our optimizer function includes both max and min, but we can simplify it further since the optimization depends only on the numerator.

The values of *w*₁, *w*₂, *and b* can be chosen in such a way that for the closest point, the minimum distance (**numerator**) is set to 1. To achieve this, we will modify equation 1.

$$\boxed{y_i(w_1x_{i1} + w_2x_{i2} + b_1) \geq 1}$$

This will ensure that the numerator for the closest point from the discriminator is 1, and for all the other points, it is greater than 1.



Hence, the optimization problem simplifies to:

$$\max\left(\frac{1}{\|W\|}\right)$$

where $\|W\| = W^TW$.

3. Primal Optimization Problem

Our Optimization function is

$$\max\left(\frac{1}{\|W\|}\right)$$

We can also write it as $\min_{W,b} \frac{1}{2}W^TW$

subject to constraint: $y_i(W^TX_i + b) \geq 1, \quad \forall i.$

- At any point in time, there exist **two support points** that define the optimal decision boundary.
- This leads to the concept of the **Support Vector Machine (SVM)**.

4. Dual Problem Formulation

- Since the above-mentioned Primal Objective function is an optimization problem subjected to **inequality** constraints, we can't directly solve it in the usual way.
- So, we will use Karush-Kuhn-Tucker (**KKT**) conditions and **Lagrangian multipliers** to convert it to form the dual problem.
- The steps are as follows:

Step 1: Construct the Lagrangian function

- To handle the constraints, we will introduce Lagrange multipliers $(\alpha_i) \geq 0$ for each constraint and construct the Lagrangian function.

$$\mathcal{L}(W, b, \alpha) = \frac{1}{2}W^TW + \sum_{i=1}^{n} \alpha_i \left(1 - y_i(W^Tx_i + b)\right)$$

Step 2: Compute Partial Derivatives

- In this step, we will find the gradient of Lagrangian function with respect to W and b and equate them to zero.

$$\frac{\partial \mathcal{L}}{\partial W} = W - \sum_{i=1}^{n} \alpha_i y_i x_i = 0$$

$$\Rightarrow W = \sum_{i=1}^{n} \alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n} \alpha_i y_i = 0$$

$$\Rightarrow \sum_{i=1}^{n} \alpha_i y_i = 0$$

Step 3: Formulate the Dual Problem

- In this step, we will substitute W back into the Lagrangian:

$$\mathcal{L}(\alpha) = \frac{1}{2}\left(\sum_{i=1}^{n} \alpha_i y_i x_i\right)^T \left(\sum_{j=1}^{n} \alpha_j y_j x_j\right) + \sum_{i=1}^{n} \alpha_i (1 - y_i((\sum_{j=1}^{n} \alpha_j y_j x_j)^T x_i + b))$$

After expanding and simplifying,

$$\mathcal{L}(\alpha) = \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j + \sum_{i=1}^{n} \alpha_i - \sum_{i=1}^{n} \alpha_i y_i (\sum_{j=1}^{n} \alpha_j y_j x_j^T x_i + b)$$

Using the constraint, $\sum_{i=1}^{n} \alpha_i y_i = 0$ the term involving b vanishes. Thus, the dual problem simplifies to:

$$\mathcal{L}(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

Dual Form (in terms of α)

$$\max_{\alpha} \quad \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subjected to constraints,

$$\sum_{i=1}^{n} \alpha_i y_i = 0, \quad \alpha_i \geq 0 \quad \forall i$$

Remember We started with **minimization** of optimization function over W and b but end up with **Maximization** of dual form over α.

- This dual problem form will allow us to solve the optimisation problem in terms of the Lagrange multipliers α_i instead of the primal variables W and b.
- This dual formulation can be solved by using a quadratic programming (QP) solver.
- We can also represent the above-mentioned dual problem in matrix form.

Dual Problem (in matrix form)

$$\max_{\alpha} \quad 1^T\alpha - \frac{1}{2}\alpha^T H\alpha$$

subjected to constraints, $$y^T\alpha = 0,$$ $$\alpha \geq 0.$$

where ,

- α as the $n \times 1$ vector of Lagrange multipliers.
- H as the $n \times n$ Hessian matrix with entries
$$H_{ij} = y_i y_j \, x_i^T x_j$$
- 1 as an $n \times 1$ vector of ones.
- y as the $n \times 1$ vector of class labels.

- Solving this dual problem will get us the values of α_i for each data point.

Step 4: Finding Weight (W) and bias (b)

The Karush-Kuhn-Tucker (KKT) conditions will play a crucial role in explaining the behaviour of the Lagrange multipliers:

KKT Condition - Complementary Slackness:

The KKT condition states that for each training point,

$$\alpha_i \left(1 - y_i(W^T x_i + b)\right) = 0$$

This entails:

i. If $\alpha_i > 0$, then the corresponding constraint is active, i.e., $y_i(W^T x_i + b) = 1$ This means that these points lie exactly on the margin and are known as **support vectors**.

ii. If a training point is **not a support vector**, then the constraint is not active. i.e.,

$$y_i(W^T x_i + b) > 1$$

which forces $\alpha_i = 0$.

Thus, during the optimization, the algorithm adjusts the Lagrange multipliers in such a way that only the points that are "critical" to defining the margin (**the support vectors**) will end up with a positive α_i and all the other points will have $\alpha_i = 0$.

We will use this information to find out W and b.

- From the 2nd step, we have,

$$W = \sum_{i=1}^{n} \alpha_i y_i x_i$$

- But we know now that only support vectors will have positive α_i and all the rest will have α_i = 0. So, we can restrict the sum to the support vector set S.

$$\boxed{W = \sum_{i \in S} \alpha_i \, y_i \, x_i}$$

- **This means that only the Support Vectors contribute to the final decision boundary.**
- And bias can be found simply by taking any of the **Support Vector (x_i)** for which,

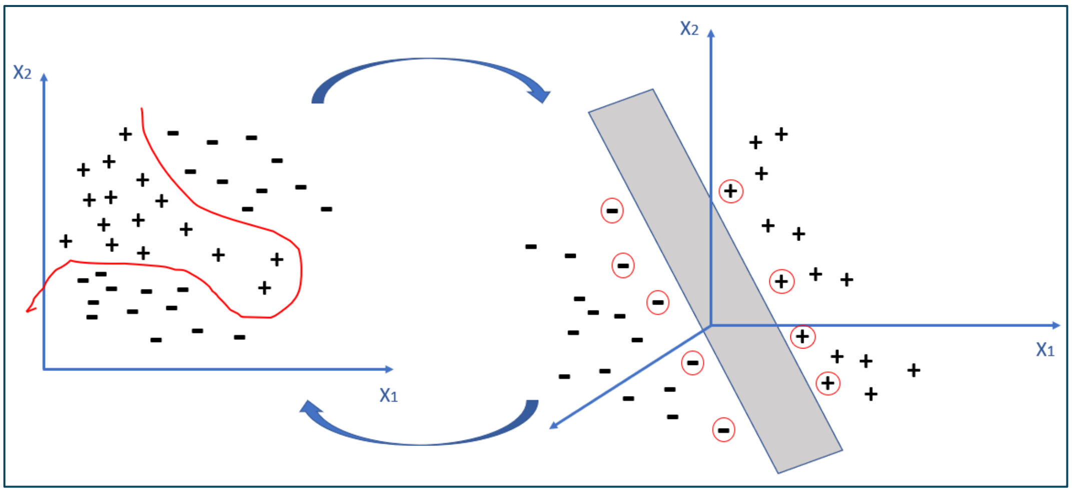
$$y_i \left(W^T x_i + b\right) = 1$$

$$\Rightarrow \boxed{b = y_i - W^T x_i}$$

This will be the Weight and Bias for the best Linear Discriminator Model.

5. Better Generalization with SVM in Higher dimensions

- The Support Vector machine is also used when the data is not linearly separable in lower dimensions but separable (linearly) in higher dimensions.
- Unlike other models (like the Perceptron, Decision-Tree, etc.) that rely on all training points, SVMs depend only on **support vectors** (points closest to the margin, for which α_i = 0). This makes SVMs resistant to the "curse of dimensionality" since the



number of support vectors is often much smaller than the total number of data points.

- And since we are considering only a few no. of points in higher dimensions to decide on the decision boundary, it significantly reduces the risk of overfitting. Hence, it can generalize the data better than most of the models.

6. SVM Dual form with soft margin:

- The above-mentioned derived equations for SVM are valid only when data is perfectly linearly separable. They are called Hard-Margin SVM.
- But most of the time, data will not be perfectly linear separable; in that case, we use **Soft-Margin SVM**. In the soft-margin SVM, we allow some misclassification by introducing **slack variables ξ_i**.

1. **Primal Form** of Soft-Margin SVM:

$$\boxed{\begin{array}{l} \min_{W,b,\xi} \quad \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\xi_i \\ \\ y_i(W^T x_i + b) \geq 1 - \xi_i, \quad \forall i = 1,\ldots,n \\ \qquad \xi_i \geq 0, \quad \forall i \end{array}}$$

subject to constraint:

2. Forming the **Lagrangian function**:

$$L(W,b,\xi,\alpha,\mu) = \frac{1}{2}\|W\|^2 + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i\left(y_i(W^T x_i + b) - 1 + \xi_i\right) - \sum_{i=1}^{n}\mu_i\xi_i$$

3. Computing the **Partial Derivatives** with respect to W, b and ξ_i:

$$\frac{\partial \mathcal{L}}{\partial W} = W - \sum_{i=1}^{n}\alpha_i y_i x_i = 0$$

$$\Rightarrow W = \sum_{i=1}^{n}\alpha_i y_i x_i$$

$$\frac{\partial \mathcal{L}}{\partial b} = -\sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\Rightarrow \sum_{i=1}^{n}\alpha_i y_i = 0$$

$$\frac{\partial L}{\partial \xi_i} = C - \alpha_i - \mu_i = 0$$

$$\Rightarrow \alpha_i \leq C$$

- The partial derivation with respect to ξ_i gives the **upper bound** constraint on α_i, which is different from the hard-margin case.

4. Forming the **Dual Problem**:

- Substituting W back into the Lagrangian

$$L = \frac{1}{2}\left(\sum_{i=1}^{n}\alpha_i y_i x_i\right)^T\left(\sum_{j=1}^{n}\alpha_j y_j x_j\right) + C\sum_{i=1}^{n}\xi_i - \sum_{i=1}^{n}\alpha_i y_i\left(\sum_{j=1}^{n}\alpha_j y_j x_j^T x_i + b\right) + \sum_{i=1}^{n}\alpha_i - \sum_{i=1}^{n}\mu_i\xi_i$$

After Expanding and simplifying we get dual form,

Soft Margin Dual Form

$$\max_{\alpha} \sum_{i=1}^{n}\alpha_i - \frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n}\alpha_i\alpha_j y_i y_j x_i^T x_j$$

subjected to constraints,

$$\sum_{i=1}^{n}\alpha_i y_i = 0, \quad 0 \le \alpha_i \le C$$

- This Soft Margin Dual form is the same as the Hard Margin SVM Dual form. The only difference from the hard-margin case is the **upper bound on α_i** ($0 \le \alpha_i \le C$), which will allow for some misclassifications.

5. Calculating **Weight (W) and Bias (b):**

$$\boxed{W = \sum_{i \in S}\alpha_i^* y_i x_i}$$

- Non-support vectors have $\alpha_i^* = 0$, so they do not affect the final decision boundary.
- Only the **Support vectors** contribute to W (i.e., points with $0 < \alpha_i^* \le C$)
  - If a point is on the margin, $0 < \alpha_i^* < C$ and $\xi_i = 0$
  - If a point is within the margin, $0 < \alpha_i^* < C$ and $0 < \xi_i < 1$
  - If a point is misclassified, $\alpha_i^* = C$ and $\xi_i > 1$

And bias can be found simply by taking any of the **Support Vector (x_i)** for which,

$$y_i\left(W^T x_i + b\right) = 1$$

$$\Rightarrow \boxed{b = y_i - W^T x_i}$$