Fundamentals of Algorithm Design and Machine Learning

Instructor: Dr. Aritra Hazra
Department of Computer Science Engineering
Indian Institute of Technology, Kharagpur

Prince Himadri Mayank : 24BM6JP43

1. Recap of Previous Concepts

In earlier sessions, we explored the primary objective of linear classification problems, which is to create a hyperplane that distinctly separates two classes.

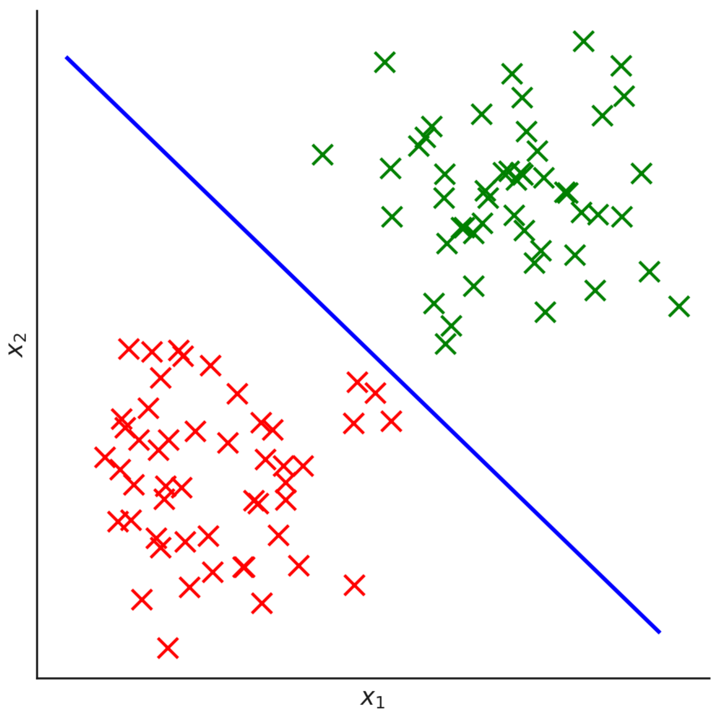

Figure 1: A linear classifier attempts to find a boundary that separates two classes.

The model classifies any input vector $\mathbf{x}$ with d features as follows:

$$\hat{y} = \begin{cases} 1 & \text{if } \mathbf{W}^T\mathbf{X} > 0 \\ -1 & \text{if } \mathbf{W}^T\mathbf{X} < 0 \end{cases}$$

Where $\mathbf{W}$ represents the weight vector that defines the hyperplane. A similar approach is applied in linear regression, where the predicted value is $\hat{y} = \mathbf{W}^T\mathbf{X}$.

The in-sample error, or the error during training, is calculated as:

$$E_{in} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

This can alternatively be written as:

$$E_{in} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{W}^T\mathbf{X}_i - y_i)^2$$

To minimize this error, we differentiate the expression with respect to **W** and solve for the optimal weights:

$$\mathbf{W} = \left(\mathbf{X}^T\mathbf{X}\right)^{-1}\mathbf{X}^T\mathbf{Y}$$

Thus, the optimal weights can be directly computed without the need for further iterations.

2. Non-linearly Separable Data

In some cases, the data points may not be linearly separable. For example, consider the following data distribution:

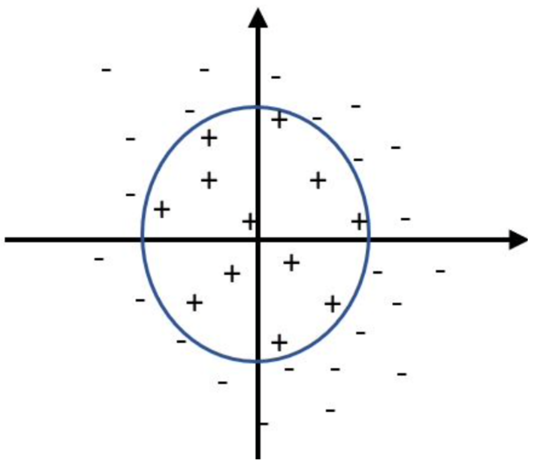

Figure 2: An example of data that is not linearly separable.

In such situations, we aim to transform the feature space to make the data linearly separable. For instance, we can map the data into a new feature space where $x = x_1^2$.

A more general approach involves transforming the features into a higher-dimensional space, such as:

$$\phi(1, x_1, x_2) = \{1, x_1, x_2, x_1x_2, x_1^2, x_2^2\}$$

However, this approach comes with some drawbacks:

- The expanded feature set might lead to overfitting the training data.
- It is computationally expensive to calculate and train the model on these expanded features.

To address these issues, we can reduce the feature set to:

$$\phi(1, x_1, x_2) = \{1, x_1^2 + x_2^2\}$$

This transformation could improve classification performance when compared to using raw features.

However, the transformation process must be performed carefully. This practice, known as *Data Snooping*, risks allowing the user, rather than the machine, to "learn" the data patterns.

3. Modified Linear Regression

Linear regression typically attempts to fit a single line to the entire dataset.

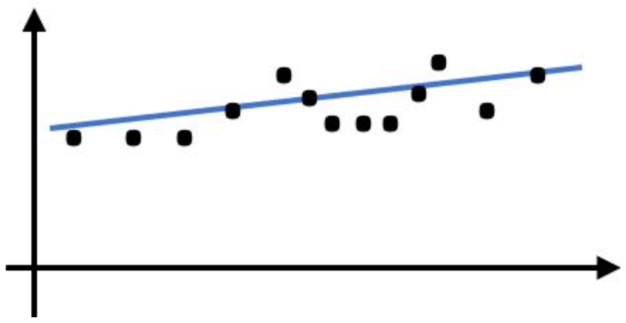

Figure 3: Linear Regression seeks to fit a single line with minimal error.

Alternatively, we can divide the data into smaller segments and fit separate linear regression models for each segment.

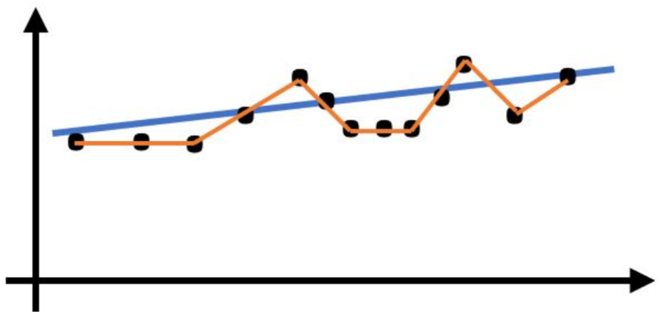

Figure 4: Modified Linear Regression approach fitting multiple lines for different intervals.

To minimize the error within each segment, we compute the error in each neighborhood δ:

$$\text{Fit}_W = \sum_{i=0}^{n} \alpha_i \left(\mathbf{W}^T \mathbf{X}_i - y_i\right)^2$$

Where:

$$\alpha = \exp\left(-\frac{(x_i - x)^2}{2\tau^2}\right)$$

This formulation results in a bell-shaped curve, illustrating how the influence of each data point decreases as we move further from the point of interest.

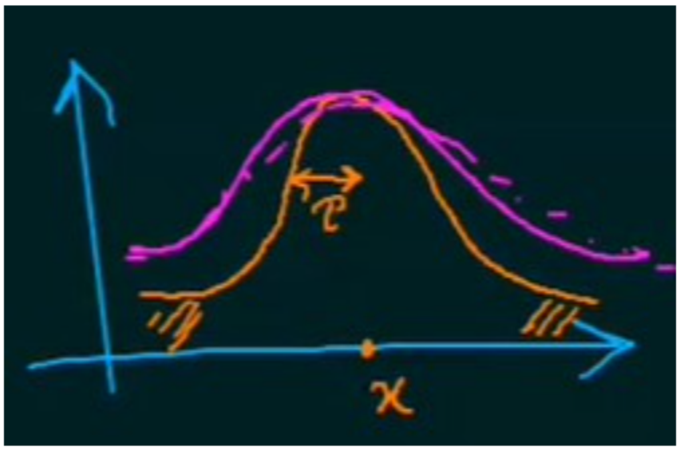

Figure 5: Change in α as we move away from the point of interest.

4. Logistic Regression

In classification tasks, we typically aim to predict a binary outcome, such as 0 or 1. The goal is to identify a smooth function that models the data distribution.

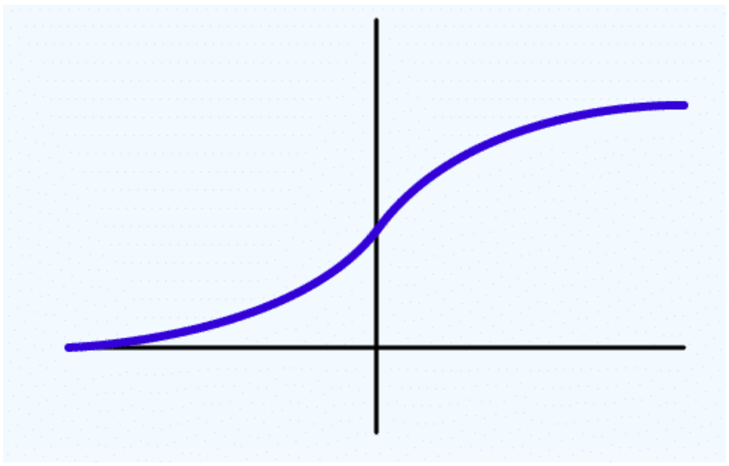

Figure 6: A function that yields 1 beyond a certain threshold.

One such function is the logistic function:

$$\theta(s) = \frac{e^s}{1 + e^{-s}}$$

Additionally, the following property holds:

$$\theta(-s) = 1 - \theta(s)$$

Thus, the hypothesis is:

$$H(s) = \theta(s)$$

Where $s = \mathbf{W}^T\mathbf{X}$.
The likelihood function is calculated as:

$$\text{Prob}(y|x) = \begin{cases} \theta(s) & \text{if } y = +1 \\ \theta(-s) & \text{if } y = -1 \end{cases}$$

We aim to maximize the likelihood:

$$\text{Maximize } \frac{1}{N} \prod_{i=1}^{N} \theta(y_i \mathbf{W}^T \mathbf{X}_i)$$

Alternatively, we minimize the following expression:

$$-\frac{1}{N} \sum_{i=1}^{N} \ln \left(1 + e^{y_i \mathbf{W}^T \mathbf{X}_i}\right)$$

The in-sample error is given by:

$$E_{in} = \frac{1}{N} \sum_{i=1}^{N} \ln \left(1 + e^{y_i \mathbf{W}^T \mathbf{X}_i}\right)$$

During the optimization process, the weights are updated according to:

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \hat{v}$$

Where $\hat{v} = \frac{\nabla E_{in}(\mathbf{W})}{\|E_{in}(\mathbf{W})\|}$ and η represents the learning rate. Initially, the learning step is larger, but as the model approaches the minimum, the step size gradually decreases.

5. Learning Rate and Its Effect on Optimization

The learning rate (η) is a hyperparameter that determines the size of the steps taken during optimization when updating the model's weights. It controls how much the weights are adjusted with respect to the gradient of the loss function. The choice of learning rate is crucial to the efficiency and effectiveness of the training process.

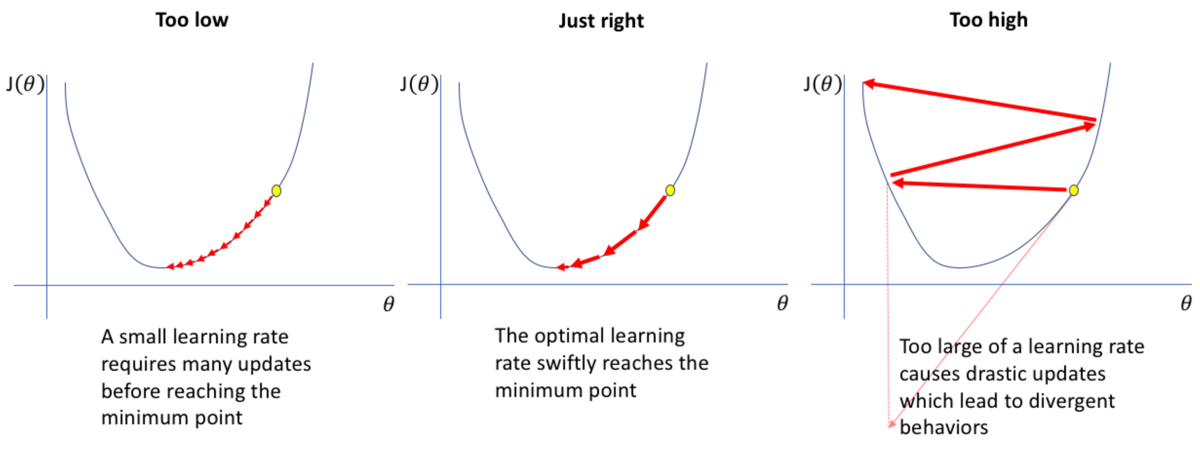
Too low
J(θ)
θ
A small learning rate requires many updates before reaching the minimum point
Just right
J(θ)
θ
The optimal learning rate swiftly reaches the minimum point
Too high
J(θ)
θ
Too large of a learning rate causes drastic updates which lead to divergent behaviors

Figure 7: Learning Rate Variations

Effect of a Low Learning Rate

When the learning rate is set too low, the model takes very small steps in the direction of the gradient. As a result, the optimization process becomes very slow. While the model is less likely to overshoot the optimal solution, it may get stuck in local minima and take an unreasonably long time to converge.

For example, if the learning rate is too low, it might take hundreds or even thousands of iterations to reach the optimal solution. This can lead to inefficiency, especially in cases where time and computational resources are limited.

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla E_{in}(\mathbf{W})$$

Where η is very small, making $\mathbf{W}_{t+1}$ only slightly different from $\mathbf{W}_t$.

Effect of a High Learning Rate

On the other hand, if the learning rate is set too high, the steps taken by the model during optimization can be too large, causing it to overshoot the optimal solution. Instead of converging to a minimum, the weights may oscillate around the optimal value or even diverge, leading to a failure in training.

For instance, when the learning rate is very large, the updates to the weights may cause the model to jump over the optimal solution, making it impossible for the model to converge to the minimum. This can result in erratic behavior and poor performance.

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla E_{in}(\mathbf{W})$$

When η is very large, the change in weights $\mathbf{W}_{t+1} - \mathbf{W}_t$ can be excessive, causing overshooting.

Effect of an Optimum Learning Rate

An optimum learning rate strikes a balance between being too slow and too fast. It ensures that the model converges quickly to the optimal solution without overshooting or getting stuck in suboptimal solutions. The learning rate is typically tuned using techniques like grid search or adaptive learning rate methods such as Adam.

With an optimal learning rate, the model makes efficient progress toward minimizing the error and converges at an acceptable pace. It reduces the total number of iterations needed to reach the optimal solution while ensuring stability.

$$\mathbf{W}_{t+1} = \mathbf{W}_t - \eta \nabla E_{in}(\mathbf{W})$$

In this case, η is neither too small nor too large, allowing for steady and fast convergence.