CS361: Machine Learning Notes

February 8, 2025

1 Linear Regression

1.1 Introduction

Linear regression is a supervised learning algorithm used to predict a continuous output variable y based on one or more input features x. The goal is to find the best-fit line that minimizes the error between the predicted and actual values.

The equation for simple linear regression is:

$$y = ax + b$$

where:

- y is the dependent variable (output).
- x is the independent variable (input).
- a is the slope (weight) of the line.
- b is the y-intercept.

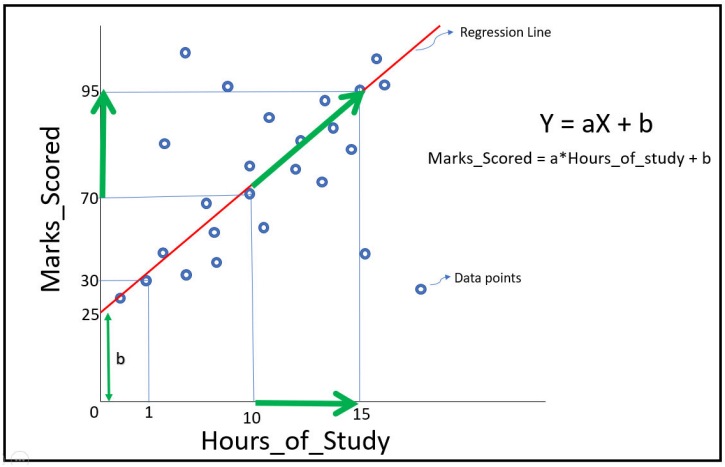

Figure 1: Linear Regression for a marks vs study hour example

1.2 Loss Function

The loss function measures how well the model is performing. For linear regression, we use the Mean Squared Error (MSE) as the loss function:

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

where:

- N is the number of data points.
- y_i is the actual value.
- $\hat{y}_i$ is the predicted value.

1.3 Gradient Descent

To minimize the loss function, we use gradient descent. The parameters a and b are updated as follows:

$$a = a - \eta \frac{\partial E}{\partial a}$$

$$b = b - \eta \frac{\partial E}{\partial b}$$

where η is the learning rate.

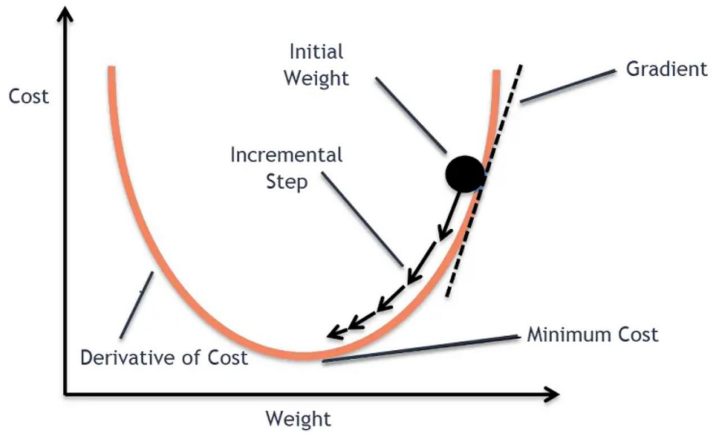

Figure 2: Gradient Descent Representation

1.4 Ridge Regression

Ridge regression adds a regularization term to the loss function to prevent overfitting:

$$E(\phi) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \|a\|^2$$

where λ is the regularization parameter.

1.5 Lasso Regression

Lasso regression also adds a regularization term, but it uses the absolute value of the coefficients:

$$E(\phi) = \frac{1}{2N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2 + \lambda \|a\|_1$$

Lasso can shrink some coefficients to zero, effectively performing feature selection.

2 Artificial Neural Networks (ANN)

2.1 Introduction

An Artificial Neural Network (ANN) is a computational model inspired by the human brain. It consists of layers of interconnected nodes (neurons) that process input data and produce an output.

2.2 Structure of ANN

- **Input Layer**: The first layer that receives the input features.
- **Hidden Layers**: Intermediate layers that process the data.
- **Output Layer**: The final layer that produces the output.

Each neuron in the network performs a weighted sum of its inputs and applies an activation function:

$$\text{Output} = f\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

where:

- w_i are the weights.
- x_i are the inputs.
- b is the bias.
- f is the activation function (e.g., sigmoid, ReLU).

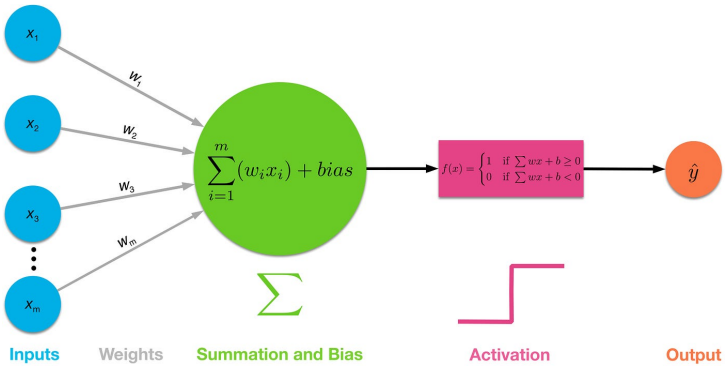

Figure 3: Basic Representation of a Neural Network

2.3 Activation Function

An activation function is a mathematical function applied to the output of a neuron. It decides whether the neuron should be "activated" or not based on the weighted sum of its inputs. Activation functions introduce non-linearity into the network, allowing it to learn complex patterns.

2.3.1 Sigmoid Function

The sigmoid function is a common activation function that maps any input to a value between 0 and 1. It is defined as:

$$f(x) = \frac{1}{1 + e^{-x}}$$

The sigmoid function is smooth and differentiable, making it useful for backpropagation. However, it can cause the "vanishing gradient" problem in deep networks.

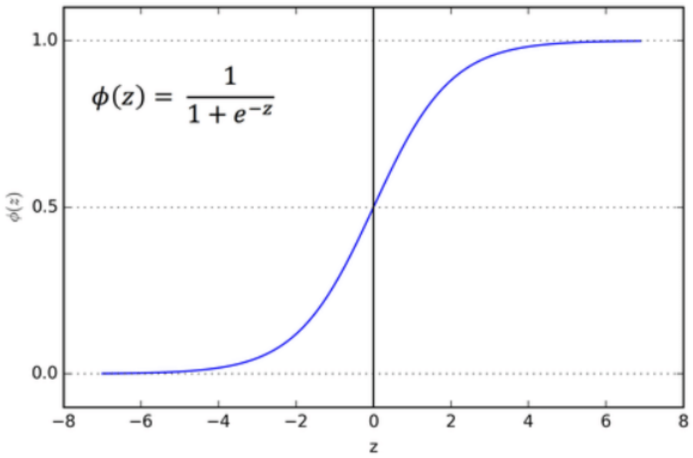

Figure 4: Sigmoid Activation Function

2.4 Feedforward Neural Network

A feedforward neural network is the simplest type of ANN where information flows in one direction: from the input layer, through the hidden layers, to the output layer. There are no cycles or loops in the network. Each neuron in one layer is connected to every neuron in the next layer.

2.5 Fully Connected Neural Network

In a fully connected neural network, every neuron in one layer is connected to every neuron in the next layer. This means that each neuron receives input from all the neurons in the previous layer and sends its output to all the neurons in the next layer. Fully connected layers are commonly used in the hidden layers of ANNs.

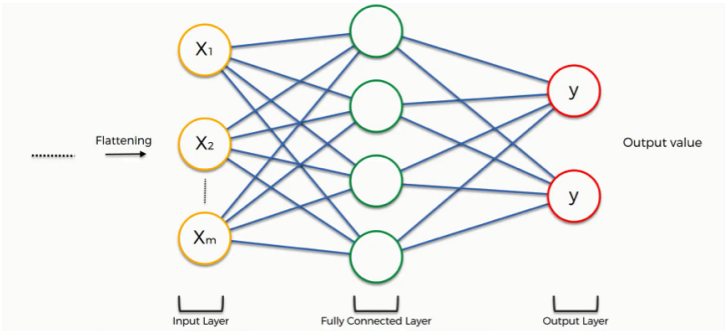

Figure 5: Fully Connected Neural Network

2.6 Forward Propagation

Forward propagation is the process of passing input data through the network to compute the output. The steps are:

1. Compute the weighted sum of inputs for each neuron.
2. Apply the activation function to get the neuron's output.
3. Pass the output to the next layer until the final output is obtained.

2.7 Backward Propagation

Backward propagation is used to update the weights of the network by minimizing the loss function. The steps are:

1. Compute the error at the output layer.
2. Propagate the error backward through the network.
3. Update the weights using gradient descent:

$$w_{ij} = w_{ij} - \eta \frac{\partial E}{\partial w_{ij}}$$

where η is the learning rate.

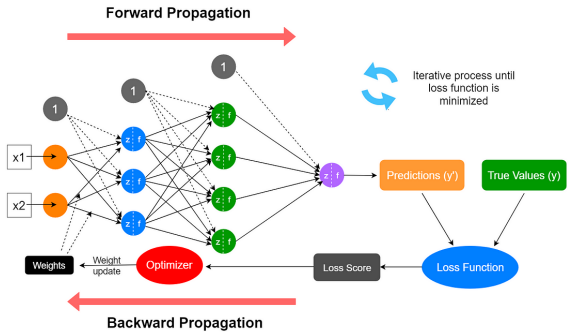

Figure 6: Forward and Backward Propagation

2.8 Loss Function for ANN

The loss function for ANN is typically the Mean Squared Error (MSE):

$$E(\omega) = \frac{1}{2} \sum_{k=1}^{K} (y_k - \hat{y}_k)^2$$

where y_k is the actual output and $\hat{y}_k$ is the predicted output.

2.9 Chain Rule in Backpropagation

The chain rule is used to compute the gradient of the loss function with respect to the weights:

$$\frac{\partial E}{\partial w_{ij}} = \frac{\partial E}{\partial o_j} \cdot \frac{\partial o_j}{\partial \text{net}_j} \cdot \frac{\partial \text{net}_j}{\partial w_{ij}}$$

where:

- o_j is the output of neuron j.
- net_j is the weighted sum of inputs to neuron j.