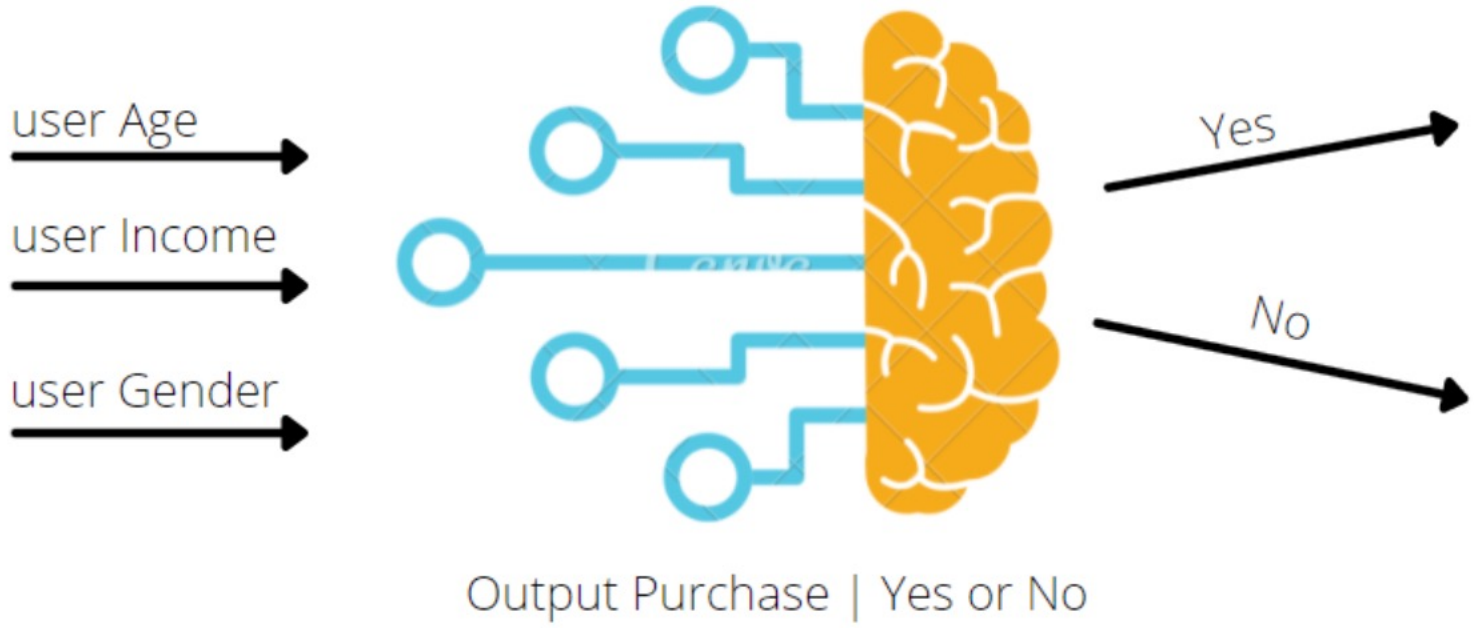
Logistic Regression



Introduction and Background

- Logistic Regression, a popular machine learning classification algorithm is used to estimate the probability that an instance belongs to a particular task.
- It extends the concepts of linear regression to handle classification problems.
- Logistic regression model computes a weighted sum of the input features along with the bias added with the result. The sum is then passed through the sigmoid function to achieve the probability.
- **Binary Outcome**: The response variable in logistic regression is categorical with two possible outcomes, often coded as 0 and 1.
- **Linear Combination**: Similar to linear regression, logistic regression starts by creating a linear combination of the input features. For an input vector x and a coefficient vector β, the linear combination is given by:

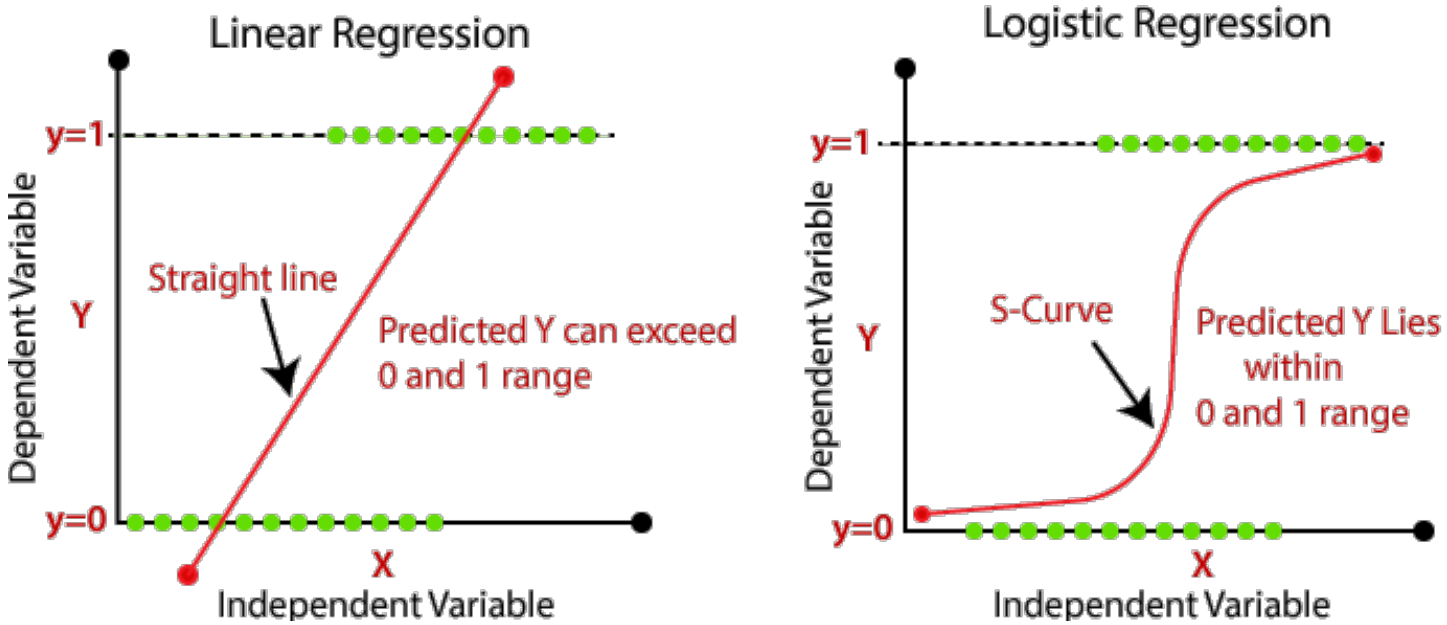
$$z = \beta^T \mathbf{x} = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots + \beta_n x_n$$

- **Sigmoid Function**: The linear combination z is then passed through a sigmoid function (also known as the logistic function) to map it to a probability value between 0 and 1. The sigmoid function is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

Introduction and Background

- This function ensures that the output is always between 0 and 1, which can be interpreted as the probability of the outcome being 1.
- **Prediction**: The model predicts the outcome based on the probability. If the probability $p=\sigma(z)$ is greater than a threshold (typically 0.5), the predicted class is 1. Otherwise, it is 0.
- Imagine you want to classify whether a student will pass or fail an exam based on their study hours. The study hours represent the input feature (x), and the outcome (pass or fail) is the binary target variable (y).



Introduction and Background

- Logistic regression has two phases:
  - training: We train the system (specifically the weights w and b) using stochastic gradient descent and the cross-entropy loss.
  - test: Given a test example x we compute p(ylx) and return the higher probability label y = 1 or y = 0.
- **Model Training:**
  - The coefficients β are typically estimated using iterative optimization algorithms such as gradient descent or least squares (LS).
  - Logistic regression solves the classification task by learning, from a training set, a vector of weights and a bias term.
  - Each weight β_i is a real number and is associated with one of the input features x_i .
  - The weight β_i represents how important that input feature is to the classification decision and can be positive (providing evidence that the instance being classified belongs in the positive class) or negative (providing evidence that the instance being classified belongs in the negative class).

Introduction and Background

- The cost function used in logistic regression is the log-loss (or binary cross-entropy loss), defined as:

$$\text{Log-loss} = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{p}_i) + (1-y_i)\log(1-\hat{p}_i)]$$

- where m is the number of samples, y_i is the actual label and $\hat{p}_i$ is the predicted probability.
- **Model Testing:**
  - The model testing phase in logistic regression involves evaluating the performance of the trained model on a separate dataset that was not used during training.
  - This phase is crucial to understand how well the model generalizes to new, unseen data.

Introduction to Loss Functions

- A loss function, also known as a cost function or error function, is a mathematical function that quantifies the difference between the predicted values generated by a model and the actual values from the data.
- It serves as a guide for training machine learning models, indicating how well or poorly the model is performing. By minimizing the loss function, we improve the model's accuracy.
- **Role in Machine Learning**
  - **Model Evaluation**:
    - The loss function provides a single scalar value that measures the quality of the model's predictions. Lower values indicate better performance.
  - **Optimization**:
    - During training, optimization algorithms (such as gradient descent) adjust the model parameters to minimize the loss function. The gradient of the loss function with respect to the model parameters indicates the direction to update the parameters to reduce the loss.
- The choice of the loss function depends on the type of problem being solved (e.g., classification, regression, or others) and the specific characteristics of the data.

Introduction to Loss Functions

- **Regression Loss Functions**:
  - **Mean Squared Error (MSE)**: Measures the average squared difference between predicted and actual values.

$$\text{MSE} = \frac{1}{m}\sum_{i=1}^{m}(y^{(i)} - \hat{y}^{(i)})^2$$

  - **Mean Absolute Error (MAE)**: Measures the average absolute difference between predicted and actual values.

$$\text{MAE} = \frac{1}{m}\sum_{i=1}^{m}|y^{(i)} - \hat{y}^{(i)}|$$

- **Classification Loss Functions**:
  - **Binary Cross-Entropy Loss**: Used for binary classification tasks to measure the performance of a classification model.

$$\text{Binary Cross-Entropy} = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(\hat{y}^{(i)}) + (1-y^{(i)})\log(1-\hat{y}^{(i)})\right]$$

  - **Categorical Cross-Entropy Loss**: Used for multi-class classification tasks.

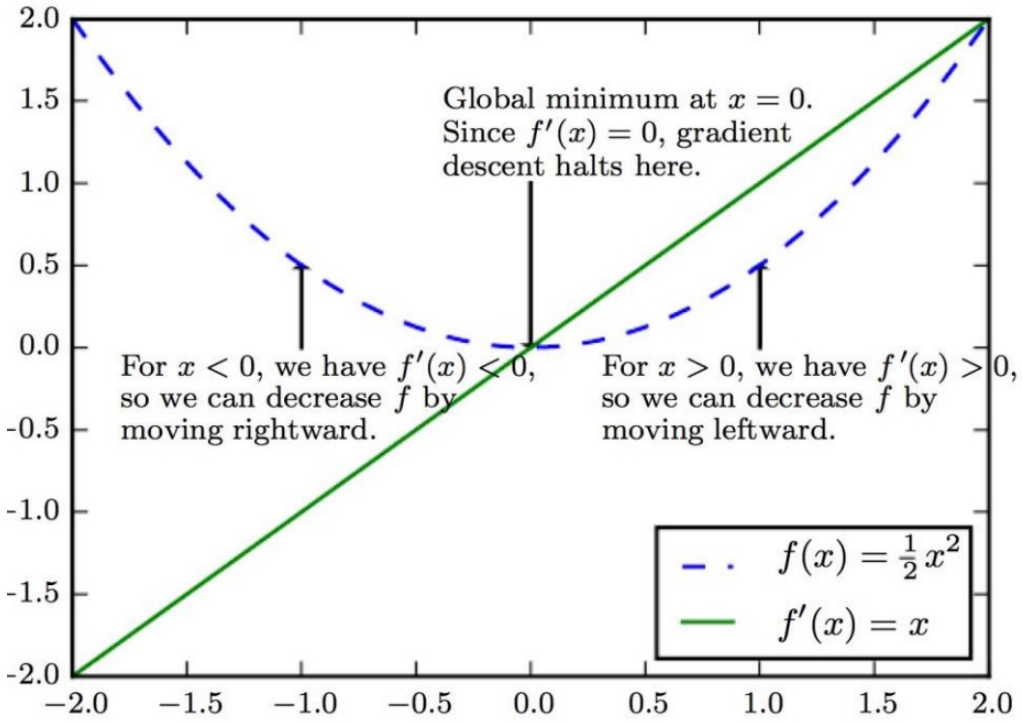
$$\text{Categorical Cross-Entropy} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{c=1}^{C} y_c^{(i)}\log(\hat{y}_c^{(i)})$$

Introduction to Gradient Descent

- Gradient Descent is an optimization algorithm commonly used in machine learning and deep learning to minimize a given loss function.
- By iteratively adjusting the model's parameters, gradient descent seeks to find the values that result in the lowest possible loss, thereby improving the model's performance.
- Gradient descent is an optimization algorithm used to find the values of parameters (coefficients) of a function (f) that minimizes a cost function (cost).
- Gradient descent is best used when the parameters cannot be calculated analytically (e.g. using linear algebra) and must be searched for by an optimization algorithm.
- Suppose a large bowl like what you would eat cereal out of or store fruit in. This bowl is a plot of the cost function (f).
- A random position on the surface of the bowl is the cost of the current values of the coefficients (cost).
- The bottom of the bowl is the cost of the best set of coefficients, the minimum of the function.

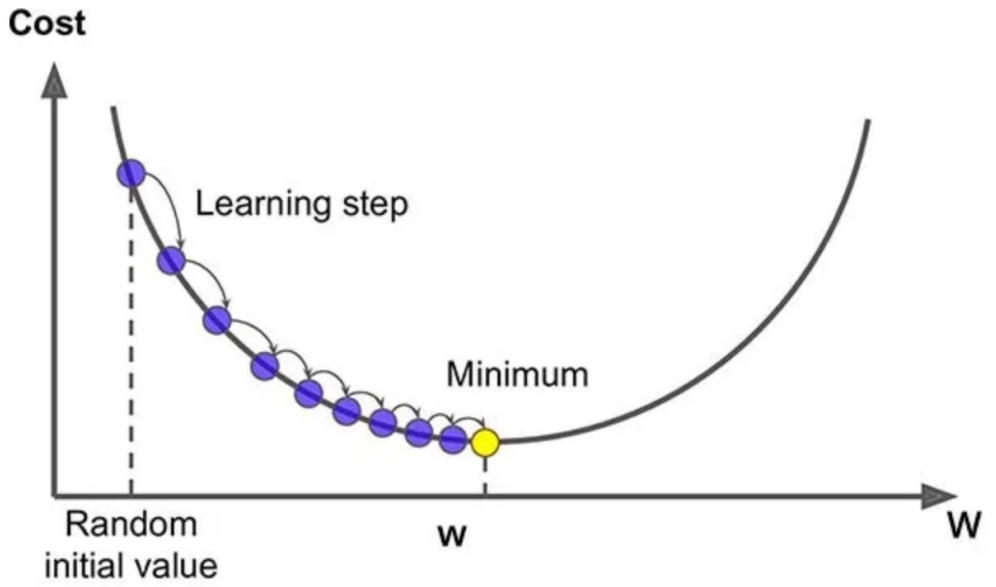
Introduction to Gradient Descent

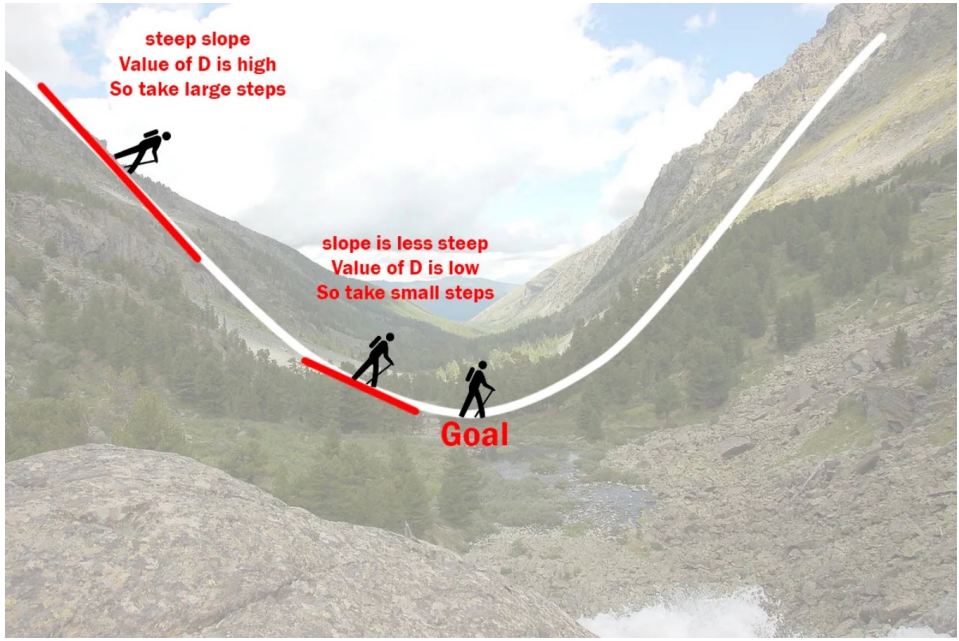
- The goal is to continue to try different values for the coefficients, evaluate their cost and select new coefficients that have a slightly better (lower) cost.
- Repeating this process enough times will lead to the bottom of the bowl and you will know the values of the coefficients that result in the minimum cost.
- Given function is f (x)=½ x 2 which has a bowl shape with global minimum at x=0.
- Since f ’(x)=x
  - for x>0, f(x) increases with x
  - and f’(x)>0 for x<0
- Use f’(x) to follow function downhill.
- Reduce f (x) by going in direction opposite sign of derivative f’(x).



Introduction to Gradient Descent

- It determines a weight vector w that minimizes E(w) by
  - Starting with an arbitrary initial weight vector.
  - Repeatedly modifying it in small steps.
  - At each step, weight vector is modified in the direction that produces the steepest descent along the error surface.
- The gradient points directly uphill, and the negative gradient points directly downhill.
- Thus, we can decrease function f by moving in the direction of the negative gradient.
  - This is known as the method of steepest descent or gradient descent.
- Steepest descent proposes a new point: $x' = x - \eta \nabla x\ f(x)$; where η is the learning rate.





Gradient Descent – Steps

Steps Involved in Linear Regression with Gradient Descent Implementation

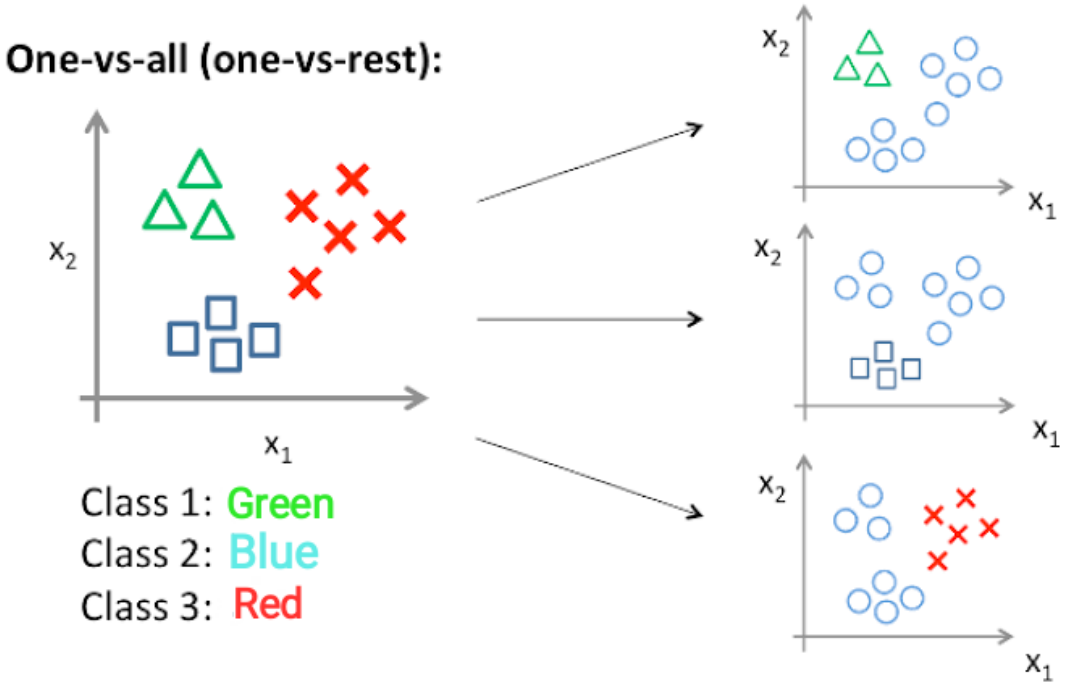
1. Initialize the weight and bias randomly or with 0(both will work).
2. Make predictions with this initial weight and bias.
3. Compare these predicted values with the actual values and define the loss function using both these predicted and actual values.
4. With the help of differentiation, calculate how loss function changes with respect to weight and bias term.
5. Update the weight and bias term so as to minimize the loss function.

Logistic Regression for Multiclass Classification

- **One-Vs-Rest Approach**
  - One-vs-rest (OvR for short, also referred to as One-vs-All or OvA) is a heuristic method for using binary classification algorithms for multi-class classification.
  - It involves splitting the multi-class dataset into multiple binary classification problems. A binary classifier is then trained on each binary classification problem and predictions are made using the model that is the most confident.
  - For example, given a multi-class classification problem with examples for each class '*red*,' '*blue*,' and '*green*'. This could be divided into three binary classification datasets as follows:
    - **Binary Classification Problem 1**: red vs [blue, green]
    - **Binary Classification Problem 2**: blue vs [red, green]
    - **Binary Classification Problem 3**: green vs [red, blue]
  - For each classifier, the class is treated as the positive class, and all other classes are treated as the negative class.

Logistic Regression for Multiclass Classification - OVR

- **Procedure**:
  1. **Training**: For each class k (where k∈{1,2,...,K} and K is the total number of classes), train a binary classifier that distinguishes between class kkk and the rest of the classes.
  2. **Prediction**: To classify a new instance, run it through all K classifiers. Each classifier outputs a confidence score (or probability) for the instance belonging to its respective class. The class with the highest score is selected as the predicted class.
- A possible downside of this approach is that it requires one model to be created for each class. For example, three classes requires three models. This could be an issue for large datasets (e.g. millions of rows), slow models (e.g. neural networks), or very large numbers of classes (e.g. hundreds of classes).



Logistic Regression for Multiclass Classification

- **One-Vs-One Approach**
  - One-vs-One (OvO for short) is another heuristic method for using binary classification algorithms for multi-class classification.
  - Like one-vs-rest, one-vs-one splits a multi-class classification dataset into binary classification problems. Unlike one-vs-rest that splits it into one binary dataset for each class, the one-vs-one approach splits the dataset into one dataset for each class versus every other class.
  - For example, consider a multi-class classification problem with four classes: '*red*,' '*blue*,' and '*green*,' '*yellow*.' This could be divided into six binary classification datasets as follows:
    - **Binary Classification Problem 1**: red vs. blue
    - **Binary Classification Problem 2**: red vs. green
    - **Binary Classification Problem 3**: red vs. yellow
    - **Binary Classification Problem 4**: blue vs. green
    - **Binary Classification Problem 5**: blue vs. yellow
    - **Binary Classification Problem 6**: green vs. yellow

Logistic Regression for Multiclass Classification - OVO

- In the One-vs-One approach, a separate binary classifier is trained for every possible pair of classes. For K classes, this results in $K\times(K-1)/2$ binary classifiers.
- **Procedure**:
  - **Training**: For each pair of classes (i, j) (where $i,j\in\{1,2,...,K\}$ and $i\neq j$), train a binary classifier that distinguishes between class i and class j.
  - **Prediction**: To classify a new instance, run it through all $K\times(K-1)/2$ classifiers. Each classifier votes for one of its two classes. The class that receives the most votes across all classifiers is selected as the predicted class.
- Can suffer from the imbalance problem where the number of negative samples far exceeds the number of positive samples for each classifier.
- The training time can be lengthy if the number of classes is large since it requires training K separate models.