ARPITA ROY
6/13/24

Logistic Regression

What is Logistic Regression?

Logistic regression is a statistical method used for binary classification, which means it helps in predicting whether an instance belongs to one of two possible classes (like yes/no, spam/not spam, pass/fail).

Basic Concept

1. Input Features (X): These are the factors or attributes that influence the outcome. For example, in predicting whether an email is spam, features could be the presence of certain keywords, length of the email, etc.

2. Output (Y): This is the result or the class we want to predict. In the case of spam detection, the output could be 1 for spam and 0 for not spam.

How Does It Work?

1. Linear Combination of Inputs: Just like in linear regression, logistic regression starts by combining input features linearly. If we have input features (X_1, X_2, ..., X_n), we form a linear combination like this:

$$Z = b_0 + b_1X_1 + b_2X_2 + ... + b_nX_n$$

Here, b_0 is the intercept and b_1, b_2, ..., b_n are the coefficients for each feature.

2. Sigmoid Function: The key difference from linear regression is that logistic regression applies a sigmoid function to this linear combination. The sigmoid function, also known as the logistic function, squashes the output to a range between 0 and 1. The sigmoid function is given by:

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

This means no matter how large or small Z is, the output sigma(Z) will always be between 0 and 1.

3. Interpretation as Probability: The output of the sigmoid function, sigma(Z), can be interpreted as the probability of the instance belonging to the positive class (e.g., spam). If sigma(Z) is close to 1, it means high probability of being spam; if close to 0, it means low probability.

4. Threshold for Decision Making: To make a final decision, we choose a threshold (usually 0.5). If sigma(Z) >= 0.5, we classify the instance as 1 (e.g., spam). If sigma(Z) < 0.5, we classify it as 0 (e.g., not spam).

Training Logistic Regression

1. Cost Function: To train the model, we need to find the best values for the coefficients b_0, b_1, ..., b_n. This is done by minimizing a cost function, specifically the log loss or binary cross-entropy loss:

$$\text{Cost}(b_0, b_1, ..., b_n) = -\frac{1}{m}\sum_{i=1}^{m}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

Here, m is the number of training instances, y_i is the actual label, and $\hat{y}_i$ is the predicted probability.

2. Optimization: Techniques like gradient descent are used to adjust the coefficients b_0, b_1, ..., b_n to minimize the cost function.

Summary

- **Logistic Regression** is used for binary classification.

- It **combines features linearly** and applies a **sigmoid function** to output a probability between 0 and 1.

- The result is interpreted as a probability, and a **threshold** is used to make a final classification.

- The model is trained by **minimizing a cost function** using optimization techniques.

This is a high-level overview of logistic regression. The mathematical details involve more advanced concepts, but the essence is combining inputs linearly, squashing them with a sigmoid function, and interpreting the output as a probability for making binary decisions.

Decision Tree

A decision tree is a popular machine learning algorithm used for classification and regression tasks. It models decisions and their possible consequences in the form of a tree structure.

Basic Concept

Think of a decision tree as a flowchart. Each node in the tree represents a decision point (based on a feature of the data), and each branch represents the outcome of that decision. The leaves of the tree represent the final outcomes (or predictions).

Building a Decision Tree

1. **Start with the entire dataset**.
2. **Choose the best feature to split the data**. This is done based on a criterion like Gini impurity or Information Gain (entropy).
3. **Split the data into subsets** based on the chosen feature.
4. **Repeat the process** for each subset, using the remaining features, until you either:
    - Achieve a pure subset (all elements are of the same class).
    - Reach a stopping criterion (like maximum depth of the tree).

Example

Imagine you have a dataset of animals, and you want to classify them as mammals or birds. Your features are:

- Does it have feathers? (Yes/No)
- Can it fly? (Yes/No)
- Does it lay eggs? (Yes/No)

Step-by-Step Construction

1. **Root Node (First Split)**
    - **Feature**: "Does it have feathers?"
        - **Yes**: Likely a bird.
        - **No**: Could be a mammal or a non-flying bird like a penguin.
2. **Second Level**
    - For the "No" branch (no feathers):
        - **Feature**: "Does it lay eggs?"
            - **Yes**: Could be a mammal like a platypus or a reptile.
            - **No**: Likely a mammal.
3. **Third Level**

- For the "Yes" branch (lays eggs):
  - **Feature**: "Can it fly?"
    - **Yes**: Likely a bird.
    - **No**: Could be a flightless bird or a platypus.

Here's a simple decision tree based on the above example:

Does it have feathers?

```
├── Yes: Bird
└── No
    ├── Does it lay eggs?
    │   ├── Yes
    │   │   ├── Can it fly?
    │   │   │   ├── Yes: Bird
    │   │   │   └── No: Platypus
    │   └── No: Mammal
```

Mathematical Explanation

1. **Splitting Criteria**:

   - **Gini Impurity**: Measures how often a randomly chosen element would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the set.

$$\text{Gini Impurity} = 1 - \sum_{i=1}^{n} (P_i)^2$$

     where P_i is the probability of each class in the node.

   - **Entropy**: Measures the impurity or disorder of a set.

$$\text{Entropy} = -\sum_{i=1}^{n} P_i \log_2(P_i)$$

where P_i is the probability of each class in the node.

2. **Information Gain**: The reduction in entropy or Gini impurity from a split.

$$\text{Information Gain} = \text{Entropy before split} - \sum_{k} \left(\frac{|D_k|}{|D|} \times \text{Entropy}(D_k) \right)$$

where D is the dataset and D_k are the subsets created from the split.

Summary

- **Decision Trees** are flowchart-like structures used for classification/regression.
- **Nodes** represent decisions based on features.
- **Branches** represent the outcomes of decisions.
- **Leaves** represent the final prediction.

Gini impurity

Gini impurity is a metric used to measure how often a randomly chosen element from a set would be incorrectly classified if it was randomly labeled according to the distribution of labels in the set. It is commonly used in decision trees to decide how to split the data at each node.

Binary Classification Example

Imagine you have a box containing apples and oranges. You want to measure how mixed the box is in terms of these two fruits.

Gini Impurity Calculation

1. **Calculate the probabilities**: Determine the probability of picking an apple and the probability of picking an orange.
   - If there are p apples and q oranges, the probability of picking an apple, $P(\text{apple})$, is $\frac{p}{p+q}$, and the probability of picking an orange, $P(\text{orange})$, is $\frac{q}{p+q}$.
2. **Formula for Gini impurity**: The Gini impurity is calculated using the probabilities:

$$\text{Gini impurity} = 1 - \sum_{i=1}^{n} (P_i)^2$$

   where P_i is the probability of each class.
3. **Binary example**: For our apple and orange example, the Gini impurity would be:

$$\text{Gini impurity} = 1 - \left(\left(\frac{p}{p+q} \right)^2 + \left(\frac{q}{p+q} \right)^2 \right)$$

Interpretation

- **Pure Node**: If the box contains only apples or only oranges (one class), the Gini impurity is 0, meaning no impurity or perfect purity.

- **Mixed Node**: If the box contains a mix of apples and oranges, the Gini impurity is greater than 0, increasing with the degree of mix. The maximum impurity (0.5 for a binary classification) occurs when the box contains an equal number of apples and oranges.

Why Gini Impurity is Useful

In decision trees, Gini impurity is used to decide how to split the data:

- **Lower Gini impurity**: Indicates a purer split. Decision trees aim to create splits that reduce Gini impurity, making each resulting subset of data as pure as possible (i.e., containing predominantly one class).

Example

Suppose you have a dataset with the following distribution:

- 4 apples and 6 oranges.

First, calculate the probabilities:

$$P(\text{apple}) = \frac{4}{10} = 0.4$$
$$P(\text{orange}) = \frac{6}{10} = 0.6$$

Next, compute the Gini impurity:

$$\text{Gini impurity} = 1 - (0.4^2 + 0.6^2)$$
$$= 1 - (0.16 + 0.36)$$
$$= 1 - 0.52$$
$$= 0.48$$

This means the box is somewhat mixed, with a Gini impurity of 0.48.

Summary

- **Gini impurity** measures how mixed a set is.

- **Range**: 0 (perfectly pure) to a maximum of 0.5 (perfectly mixed for binary classification).

- **Usage**: Helps decision trees determine the best splits to create purer subsets of data.

Gini impurity provides a simple, intuitive way to evaluate the effectiveness of splits in decision trees, guiding them towards creating more homogeneous nodes.

Entropy

Entropy is a concept from information theory that measures the amount of uncertainty or disorder in a system. In the context of decision trees and machine learning, entropy is used to measure the impurity or randomness in a dataset.

Simple Analogy

Imagine you have a jar filled with colored balls. If all the balls are of the same color, the jar is very ordered and there's no uncertainty about which color you'll pick. However, if the jar contains balls of many different colors, there's more uncertainty and disorder about which color you'll pick next.

Definition

In simple terms, entropy tells us how mixed or impure the data is. High entropy means the data is very mixed up, and low entropy means the data is more pure.

Calculation

Entropy is calculated using the formula:

$$\text{Entropy} = -\sum_{i=1}^{n} P_i \log_2(P_i)$$

where:

- P_i is the probability of each class in the dataset.
- $\log_2$ is the logarithm base 2.

Steps to Calculate Entropy

1. **Calculate the probability** of each class in the dataset.
2. **Multiply** each probability by the logarithm (base 2) of that probability.
3. **Sum up** these values for all classes.
4. **Multiply the sum by -1** to get the entropy.

Example

Let's say we have a dataset of 10 animals with the following distribution:

- 4 cats
- 6 dogs

Step-by-Step Calculation

1. **Calculate probabilities**:
   - Probability of cat (P_{cat}): $\frac{4}{10} = 0.4$
   - Probability of dog (P_{dog}): $\frac{6}{10} = 0.6$
2. **Calculate individual terms**:
   - For cats: $0.4 \times \log_2(0.4) \approx 0.4 \times (-1.322) \approx -0.5288$
   - For dogs: $0.6 \times \log_2(0.6) \approx 0.6 \times (-0.737) \approx -0.4422$
3. **Sum the values**:
   - Sum: $-0.5288 + (-0.4422) = -0.971$
4. **Multiply by -1**:
   - Entropy: $-(-0.971) = 0.971$

This value (0.971) represents the entropy of the dataset, indicating a moderate level of uncertainty or disorder since both classes are somewhat balanced.

Interpretation

- **Low Entropy (close to 0)**: The data is pure, meaning most items belong to a single class.
- **High Entropy (closer to 1 for binary classification)**: The data is very mixed, with items distributed evenly among classes.

Why It Matters

In decision trees, we use entropy to decide how to split the data. We want to split the data in a way that reduces entropy, leading to purer subsets. This makes the decision-making process clearer and more efficient.

Summary

- **Entropy** measures uncertainty or disorder.
- **High entropy** means more disorder (more mixed data).
- **Low entropy** means less disorder (more pure data).
- Used in decision trees to help create splits that make the data more homogenous, improving the model's predictions.

Entropy provides a quantitative measure to evaluate how well a split organizes the data, helping us create more effective and accurate decision trees.