CSE 4554
Machine Learning Lab

Experiment No: 3

Name of the experiment: Logistic Regression with Regularization, Decision Trees, KNN and Naive Bayes

Dr. Hasan Mahmud
Professor, Department of CSE
Md. Tanvir Hossain Saikat
Junior Lecturer, Department of CSE

October 4, 2024

Contents

1 Objectives

- To know the basics of Logistic Regression
- To know about Regularization
- To know the use of Decision Trees, KNN and Naive Bayes classifier
- To understand how to do result analysis

2 Problem Discussion

2.1 Logistic Regression

Logistic regression is a supervised learning algorithm used for binary classification. It models the probability that an instance belongs to a specific class by fitting a logistic function to the data.

The hypothesis function for logistic regression is:

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

Where x is the input feature vector and θ are the model parameters.

2.1.1 Cost Function

The cost function for logistic regression is given by:

$$J(\theta) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log(h_\theta(x^{(i)})) + (1-y^{(i)})\log(1-h_\theta(x^{(i)}))\right]$$

Minimizing this cost function using gradient descent updates the model parameters.

2.1.2 Model Training

Model training is essentially minimization of the loss function. We achieve that with the Gradient Descent technique, which can be broken into a few steps:

1. First, we find derivatives of the loss function with respect to each weight. Derivatives can tell which direction and by how much we should change weight to make the model loss a bit smaller.
2. Updating each weight according to derivative until the local minimum is found, i.e. model doesn't improve anymore so we can stop.

2.2 Regularization

Firstly why do we need Regularization? The biggest reasons for regularization are

1. to avoid overfitting by not generating high coefficients for predictors that are sparse.
2. to stabilize the estimates especially when there's collinearity in the data.

For Point No.1 - is inherent in the regularization framework. Since there are two forces pulling each other in the objective function, if there's no meaningful loss reduction, the increased penalty from the regularization term wouldn't improve the overall objective function. This is a great property since a lot of noise would be automatically filtered out from the model.

To give you an example for 2), if you have two predictors that have the same values, if you just run a regression algorithm on it since the data matrix is singular, your beta coefficients will be Inf if you try to do a straight matrix inversion. But if you add a very small regularization lambda to it, you will get stable beta coefficients with the coefficient values evenly divided between the equivalent two variables.

In L2 regularization, the penalty term is proportional to the square of the coefficients:

$$J(\theta) = -\frac{1}{m}\sum [y\log(h_\theta(x)) + (1-y)\log(1-h_\theta(x))] + \frac{\lambda}{2m}\sum \theta_j^2$$

Where λ is the regularization parameter.

In L1 regularization, the penalty term is proportional to the absolute value of the coefficients:

$$J(\theta) = -\frac{1}{m}\sum [y\log(h_\theta(x)) + (1-y)\log(1-h_\theta(x))] + \frac{\lambda}{m}\sum |\theta_j|$$

Secondly, When Do we need L1 and L2 regularization?

The main intuitive difference between the L1 and L2 regularization is that L1 regularization tries to estimate the median of the data while the L2 regularization tries to estimate the mean of the data to avoid overfitting. L1 regularization tends to push some coefficients to zero, effectively selecting a simpler model by eliminating less important features. This is particularly useful when you have many features and want to perform feature selection.

3 Decision Trees

Decision trees are a non-parametric supervised learning method used for both classification and regression. They are simple yet powerful algorithms that make decisions by recursively splitting the dataset based on feature values.

3.1 Tree Structure

A decision tree consists of:

- **Nodes**: Each node represents a feature and a condition to split the data.
- **Branches**: The branches represent the outcomes of the condition.
- **Leaves**: The leaves represent the final classification or decision.

3.2 Splitting Criteria

Common criteria for splitting nodes include:

- **Gini Impurity**: Measures how often a randomly chosen element would be incorrectly classified.
- **Entropy**: Measures the amount of uncertainty in the data.

The tree grows by recursively splitting the data until it reaches a stopping condition, such as maximum depth or minimum number of samples per node.

4 K-Nearest Neighbors (KNN)

KNN is a simple, instance-based learning algorithm used for classification and regression. It operates by identifying the k-nearest points (neighbors) to a given input and then making a prediction based on the majority label of those neighbors.

4.1 Distance Metric

KNN relies on a distance metric to determine the closeness of data points. The most commonly used distance metric is the Euclidean distance:

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

Where x and y are feature vectors.

4.2 Classification

For classification tasks, KNN assigns the majority class label among the k-nearest neighbors. The value of k is a hyperparameter that must be chosen carefully to balance bias and variance.

5 Naive Bayes Classifier

The Naive Bayes classifier is a probabilistic model based on Bayes' theorem. It assumes that the features are conditionally independent, meaning the value of one feature does not affect the value of another feature.

5.1 Bayes' Theorem

Bayes' theorem is stated as:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

Where:

- $P(C|X)$ is the posterior probability of class C given the features X.
- $P(X|C)$ is the likelihood of the features given the class.
- $P(C)$ is the prior probability of the class.
- $P(X)$ is the prior probability of the features.

5.2 Assumption of Independence

Despite its simplicity, the Naive Bayes classifier performs surprisingly well in many applications, especially when the assumption of independence holds approximately true.

6 Result Analysis

Evaluating the performance of machine learning models requires a variety of metrics. Some commonly used metrics for classification problems include:

6.1 Confusion Matrix

The confusion matrix is a table that describes the performance of a classification model by displaying the counts of true positives, false positives, true negatives, and false negatives.

6.2 Accuracy, Precision, Recall, F1-Score

- **Accuracy**: The ratio of correctly predicted instances to total instances.
- **Precision**: The ratio of correctly predicted positive instances to all predicted positives.
- **Recall**: The ratio of correctly predicted positives to all actual positives.
- **F1-Score**: The harmonic mean of precision and recall.

These metrics give a clear understanding of a model's performance, especially in the case of imbalanced datasets.

7 Data

The first step for any machine learning problem is getting the data. There is no machine "learning" if there is nothing to "learn" from. So for this lab we will be using a dataset named **"Biomechanical features of orthopedic patients"** which you can download from Kaggle. You will also find all the details about the dataset in the website. For today's lab use the two class portion of the dataset.