Fondations of Machine Learning Assignment 2

Linear Regression

DA24D402, Shuvrajeet Das

November 3, 2024

Contents

List of Tables

List of Figures

Setup:

In this assignment, you will build a spam classifier from scratch. No training data will be provided. You are free to use whatever training data is publicly available/does not have any copyright restrictions (You can build your own training data as well if you think that is useful). You are free to extract features you think will be appropriate for this problem. The final code you submit should have a function/procedure which when invoked will be able to automatically read a set of emails from a folder titled test in the current directory. Each file in this folder will be a test email and will be named 'email.txt' ('email1.txt', 'email2.txt', etc). For each of these emails, the classifier should predict +1 (spam) or 0 (non-spam). You are free to use whichever algorithm learned in the course to build a classifier (or even use more than one). The algorithms (except SVM) need to be coded from scratch. Your report should detail information relating to the data set chosen, the features extracted and the exact algorithm/procedure used for training including hyperparameter tuning/kernel selection if any. The performance of the algorithm will be based on the accuracy on the test set.

1 Dataset:

Dataset Description

The dataset used in this study is divided into two main types based on the nature of data distribution:

- **Linearly Separable Dataset**: This type of dataset has data points that can be separated by a linear hyperplane. In other words, a straight line (or plane in higher dimensions) can distinguish between the different classes without any overlap or complex boundary.
- **Non-linearly Separable Dataset**: This dataset consists of data points that cannot be separated by a linear hyperplane. Instead, it requires a more complex, non-linear boundary to distinguish between classes effectively.

Data Composition

- **Total Number of Data Points**: The entire dataset comprises 1000 data points, which include both the training set and the test set.
- **Training Set**:
  - The training set consists of 800 data points in total.
  - These 800 points are split between linearly separable and non-linearly separable data types to ensure diversity in training. This setup helps in testing the ability of various algorithms to handle both simple and complex decision boundaries.
- **Test Set**:

  - The test set contains 200 data points.
  - Similar to the training set, the test set includes both linearly separable and non-linearly separable data points to evaluate the model's generalization ability on different types of data distributions.

- **Feature Space**:
  - Each data point in the dataset lies in a 10-dimensional feature space. This means that each data point x_i can be represented as a vector with 10 features:

    $$x_i = [x_{i1}, x_{i2}, \dots, x_{i10}]$$

    where x_{ij} denotes the jth feature of the ith data point.
  - A higher-dimensional feature space such as this allows for capturing complex relationships between features, making the problem more challenging and realistic for modern machine learning algorithms.

Purpose of the Dataset

- The **linearly separable dataset** is primarily used to assess the performance of algorithms that excel in scenarios where a simple decision boundary is sufficient. This type of dataset helps verify the fundamental capability of an algorithm to separate classes with minimal complexity.
- The **non-linearly separable dataset** is designed to evaluate how well algorithms can adapt to more complex scenarios. It tests the algorithm's ability to model and create decision boundaries that are not linear, reflecting real-world challenges where simple boundaries are often insufficient.

2 Algorithms used:

2.1 Naive Bayes Classifier:

Naive Bayes is a probabilistic classifier based on Bayes' theorem, assuming feature independence given the class.

- **Bayes' Theorem**:
  $$P(y|x) = \frac{P(x|y)P(y)}{P(x)}$$
  where:
  - $P(y|x)$ is the posterior probability of class y given feature vector x.
  - $P(x|y)$ is the likelihood of observing x given class y.
  - $P(y)$ is the prior probability of class y.
  - $P(x)$ is the probability of observing x.

- **Naive Bayes Assumption (Conditional Independence)**: Assuming features x_i are conditionally independent given y, we get:

$$P(x|y) = \prod_{i=1}^{n} P(x_i|y)$$

This simplifies the computation of $P(y|x)$.

- **Classification Rule**: To classify x, we compute the posterior for each class and choose the one with the highest probability:

$$\hat{y} = \arg\max_{y} P(y) \prod_{i=1}^{n} P(x_i|y)$$

2.2 Logistic Regression Classifier:

Logistic regression is used for binary classification, predicting probabilities of classes $y = 0$ or $y = 1$ by modeling a log-odds relationship with the predictors x.

- **Logistic Function**:

$$P(y = 1|x) = \sigma(z) = \frac{1}{1 + e^{-z}}$$

where $z = x^T\theta$.

- **Cost Function (Binary Cross-Entropy)**:

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

- **Gradient Descent Update Rule**:

$$\theta := \theta - \alpha \nabla J(\theta)$$

where $\nabla J(\theta)$ is the gradient of the cost function with respect to θ.

- **Hyperparameters**: - **Learning Rate (α)**: Controls the step size in gradient descent. - **Regularization Parameter (λ)**: Adds a penalty to the cost to prevent overfitting:

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^{m} \left[y^{(i)} \log(\sigma(z^{(i)})) + (1 - y^{(i)}) \log(1 - \sigma(z^{(i)})) \right] + \frac{\lambda}{2m} \sum_{j=1}^{n} \theta_j^2$$

2.3 K Nearest Neighbor Classifier:

K-Nearest Neighbors (KNN) is a non-parametric, instance-based algorithm used for classification and regression. It classifies a data point based on the classes of its k closest neighbors.

- **Distance Metric**:
  - Commonly used distance metrics include:
  - **Euclidean Distance**:
  $$d(x, x') = \sqrt{\sum_{i=1}^{n} (x_i - x'_i)^2}$$
  - **Manhattan Distance**:
  $$d(x, x') = \sum_{i=1}^{n} |x_i - x'_i|$$
- **Classification Rule**: - For a given point x, find the k closest points and assign x to the majority class among them.
- **Hyperparameters**:
  - **Number of Neighbors (k)**: Controls the smoothness of the decision boundary. A small k captures more local structures; a larger k smooths out noise.
  - **Distance Metric**: Determines the measure of "closeness" (e.g., Euclidean, Manhattan).
  - **Weighting**:
    - **Uniform Weighting**: All neighbors have equal weight.
    - **Distance-Weighted**: Closer neighbors have higher weights, often:
    $$w_i = \frac{1}{d(x, x_i)}$$

2.4 Decision Tree Classifier:

- **Decision Tree:**

  A decision tree is a model that splits data into subsets based on feature values, forming a tree-like structure to make predictions. The tree grows by selecting the best features to split at each node.
- **Split Criterion**: - Measures used to determine the best split include:
  - **Gini Impurity**:
  $$G = 1 - \sum_{i=1}^{C} p_i^2$$
  where p_i is the probability of a class i.
  - **Entropy**:
  $$H = -\sum_{i=1}^{C} p_i \log_2(p_i)$$

- **Information Gain**: - Used to measure the reduction in impurity from a split:

$$\text{Information Gain} = H_{\text{parent}} - \sum_{j=1}^{k} \frac{n_j}{n} H_j$$

where H_{parent} is the impurity of the parent node, H_j is the impurity of child node j, n_j is the number of samples in child node j, and n is the total number of samples.

- **Hyperparameters**:
  - **Max Depth**: Limits the depth of the tree to prevent overfitting.
  - **Min Samples Split**: Minimum number of samples required to split a node.
  - **Min Samples Leaf**: Minimum number of samples required to be in a leaf node.

3 Result Analysis:

3.1 Naive Bayes Classifier:

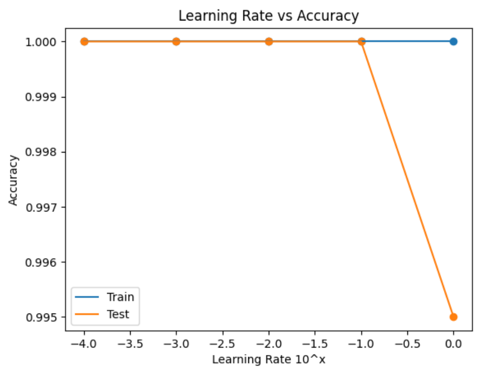
The Naive bayes Classifier was able to produce results of

Naive Bayes Classifier		
Dataset Type	Train Accuracy	Test Accuracy
Linear	1.0	1.0
Non-Linear	1.0	1.0

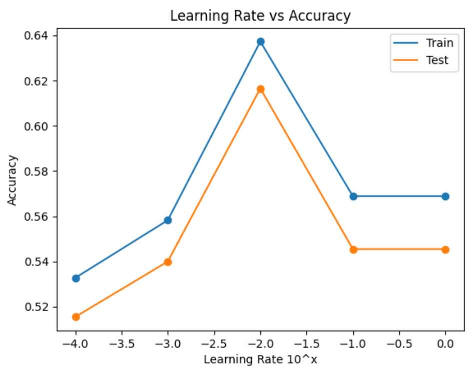
3.2 Logistic Regression Classifier:

The Logistic Regression Classifier was able to produce results of

Logistic Regression		
Dataset Type	Train Accuracy	Test Accuracy
Linear	1.0	1.0
Non-Linear	0.57	0.55



(a) Linear Dataset



(b) Non Linear Dataset

Also for various hyper-parameter tuning like learning rate:

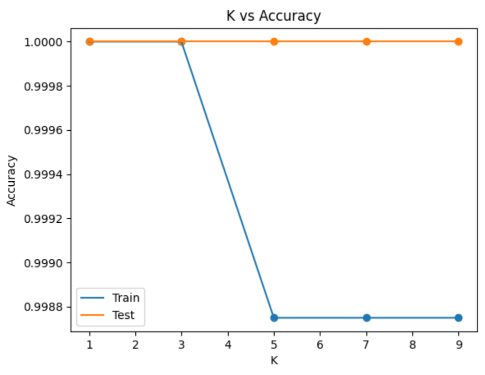
Logistic Regression			
Dataset Type	α	Train Accuracy	Test Accuracy
Linear	0.0001	1.0	1.0
	0.001	1.0	1.0
	0.01	1.0	1.0
	0.1	1.0	1.0
	1	1.0	1.0
Non-Linear	0.0001	0.54	0.52
	0.001	0.56	0.54
	0.01	0.64	0.62
	0.1	0.57	0.55
	1	0.57	0.55

The best hyperparameter is with α=0.01

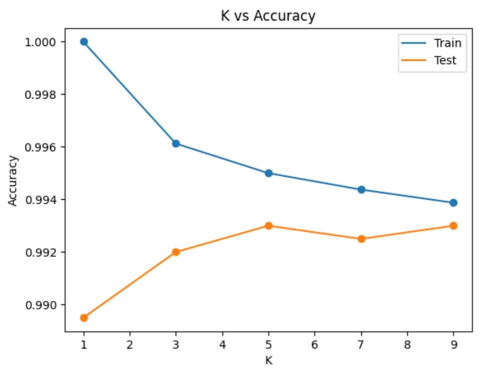
3.3 K Nearest Neighbor Classifier:

The K-Nearest Neighbor Classifier was able to produce results of

K Nearest Neighbor Classifier		
Dataset Type	Train Accuracy	Test Accuracy
Linear	1.0	1.0
Non-Linear	0.994	0.994



(a) Linear Dataset



(b) Non Linear Dataset

Also for various hyper-parameter tuning like learning rate:

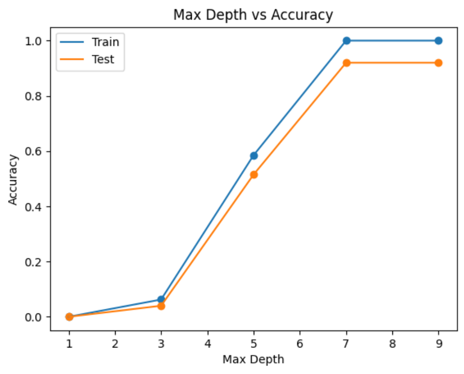
K Nearest Neighbor Classifier			
Dataset Type	k	Train Accuracy	Test Accuracy
Linear	1	1.0	1.0
	3	1.0	1.0
	5	0.99	1.0
	7	0.99	1.0
	9	0.99	1.0
Non-Linear	1	1	0.99
	3	0.996	0.992
	5	0.995	0.993
	7	0.994	0.9925
	9	0.994	0.994

The best hyperparameter is with k=9

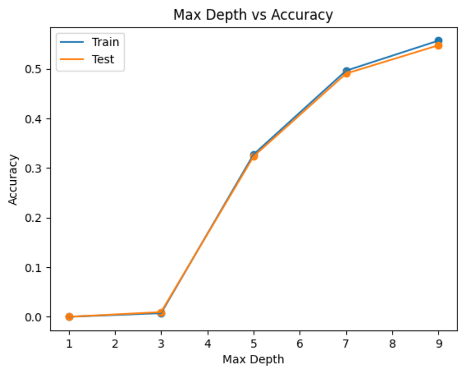
3.4 Decision Tree Classifier:

The Decision Tree Classifier was able to produce results of

Decision Tree Classifier		
Dataset Type	Train Accuracy	Test Accuracy
Linear	1.0	0.9
Non-Linear	0.55	0.53



(a) Linear Dataset



(b) Non Linear Dataset

Also for various hyper-parameter tuning like learning rate:

Decision Tree Classifier			
Dataset Type	Max Depth	Train Accuracy	Test Accuracy
Linear	1	0.0	0.0
	3	0.13	0.1
	5	0.6	0.55
	7	1.0	0.90
	9	1.0	0.90
Non-Linear	1	0	0
	3	0.05	0.05
	5	0.32	0.315
	7	0.492	0.487
	9	0.534	0.521

The best hyperparameter is with max depth=9

4 Conclusion

- **Naive Bayes Classifier:**
  - Selecting an appropriate prior distribution is crucial for achieving optimal performance with the Naive Bayes classifier. The choice of prior can significantly influence classification results.
  - To handle continuous features, it is common to use multiple bins to discretize the feature space. This process, known as feature binning, allows the algorithm to categorize continuous data effectively for probabilistic analysis.
- **Logistic Regression:**
  - The learning rate is a critical hyperparameter in training logistic regression models. An inappropriate learning rate can either slow down convergence or cause instability in training, leading to suboptimal accuracy. Proper tuning of the learning rate is essential for reliable performance.
  - Regularization techniques, such as Lasso (L1 regularization) and Ridge (L2 regularization), can be introduced to prevent overfitting and improve model generalization. These methods add penalties to the loss function, encouraging simpler and more robust models.
- **K-Nearest Neighbors (KNN):**
  - Choosing an appropriate number of neighbors, denoted by K, is essential for the K-Nearest Neighbors algorithm. A small K may lead to overfitting, while a large K may oversimplify the model. Therefore, careful tuning of K is necessary for optimal classification accuracy.

  - Various distance metrics, such as Euclidean, Manhattan, or Minkowski distance, can be used to compute the similarity between instances. The choice of metric depends on the nature of the data and can significantly impact the performance of the KNN algorithm.

- **Decision Tree:**
  - Increasing the depth of a decision tree generally improves classification accuracy, as it allows the model to capture more intricate patterns in the data. However, deeper trees can lead to higher complexity, increased training time, and a greater risk of overfitting.
  - Gini impurity and cross-entropy are common metrics used to measure the quality of splits within a decision tree. These criteria help determine the optimal splits at each node, balancing between purity and information gain for effective classification.

References