BBM409 : Introduction to Machine Learning Lab.
Fall 2024
Instructor: Assoc. Prof. Dr. Hacer Yalım Keleş
TA: Sevginur İnce

Assignment 2: Understanding Logistic Regression, Support Vector Machine and Decision Tree

Due date: Tuesday, 26-11-2024, 11:59 PM.

In this assignment, in **binary classification task** first you will explore the Logistic Regression algorithm and SVM algorithm by implementing it for a binary classification task using the **Bank Marketing Dataset**. The dataset contains 10,578 samples with 17 features ('age', 'job', 'marital', 'education', 'default', 'balance', 'housing', 'loan', 'contact', 'day of week', 'month', 'duration', 'campaign', 'pdays', 'previous', 'poutcome', 'y'), and your task will be to classify the client whether it will subscribe (class 2) a term deposit or not (class 1) using class label feature 'y'.

The steps you will follow include:

1. **Implementation:** You will implement the Logistic Regression Algorithm from scratch to train it on the Bank Marketing Dataset. A Logistic Regression model must be created without using Sklearn or any built-in libraries. You will also implement the SVM model to train it on the Bank Marketing Dataset. You can use the sklearn library for implementation of the SVM model.

2. **Evaluation and Metrics:** You will evaluate both two model's performance using metrics such as accuracy, F1 score, recall, and precision. A set of questions will guide you to understand the importance of using these metrics and validation data in assessing the model's performance. Also you will evaluate confusion matrix of every classification operation.

3. **Visualization:** You will visualize Logistic Regression model's decision boundary on 2D space. In this section, you will explore how the Logistic Regression Algorithm forms the decision boundary (or hyperplane) that separates the two classes. To be able to do that you will reduce the feature space to two dimensions, train the Logistic Regression on the selected features, and visualize the hyperplane. First, the correlation of all features with the target 'y' feature will be calculated. Then, modeling will be performed using the 2 features that have the highest correlation with the target feature. Then, the class separation boundary will be visualized and compared in 2d space and as in PA1. The classification report and confusion matrix will be rewrite along with the decision boundaries. Then, the classification will be performed using all the features. The results of this classification will be reported together with the resultant confusion matrices.

   The goal is to understand how the Logistic Regression changes the decision boundary as the data changes, and how it adapts based on the features used.
   You will also visualize the SVM model's decision boundary. First, perform the

classification with the SVM model using the 2 features with the highest correlation with the target feature. After the classification, visualize the decision boundaries. In addition, the classification results, confusion matrix and classification report, should also be rewrite as output. After this process, perform the classification using the 2 features with the lowest correlation with the target feature. After the classification, visualize the decision boundaries. In addition, the classification results, confusion matrix and classification report, should also be printed as output. After these two classifications made with the SVM model, try to get the best classification score by including all the features in the GridSearchCV model. For this classification, the classification report and confusion matrix outputs should also be included in your homework.

The best results of the experiments with SVM and Logistic regression models will be compared in a table.

This part of assignment will provide a comprehensive understanding of the Logistic Regression and SVM algorithms, with a focus on how its performance is evaluated and how its decision boundary is influenced by the underlying data.

1 Logistic Regression Algorithm

The **Logistic Regression Algorithm** is a widely-used linear model for binary classification. Unlike the Perceptron, which classifies directly, Logistic Regression outputs probabilities for each class and uses these probabilities to classify data into one of two categories. Logistic Regression applies a logistic (sigmoid) function to the linear combination of input features to obtain a probability score, which can be thresholded for classification.

How the Algorithm Works

1. **Initialization:** Logistic Regression starts by initializing a weight vector $\mathbf{w}$ and a bias b to zeros (or small random values). Each input feature vector $\mathbf{x}$ is associated with a label $y \in \{0, 1\}$, representing two possible classes.

2. **Prediction Function:** For any input $\mathbf{x}$, Logistic Regression computes a weighted sum and passes it through the *sigmoid activation function* to obtain a probability score:
$$\hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b) = \frac{1}{1 + e^{-(\mathbf{w}^T\mathbf{x}+b)}}$$
This probability score $\hat{y}$ represents the likelihood of the input belonging to the positive class. A threshold (e.g., 0.5) is applied to convert the probability into a class label:
$$\text{Predicted class} = \begin{cases} 1, & \text{if } \hat{y} \geq 0.5 \\ 0, & \text{if } \hat{y} < 0.5 \end{cases}$$

3. **Loss Function and Gradient Descent:** Logistic Regression uses the *cross-entropy loss* function to measure the prediction error. The weights and bias are

updated iteratively using gradient descent to minimize this loss. For each sample $(\mathbf{x}_i, y_i)$, the gradients for the weights and bias are calculated as follows:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial L}{\partial \mathbf{w}}$$

$$b \leftarrow b - \eta \frac{\partial L}{\partial b}$$

where L is the cross-entropy loss, and η is the learning rate controlling the step size of updates.

4. **Convergence:** Gradient descent is run for a fixed number of iterations or until the loss function converges to a stable minimum. Logistic Regression is suitable for both linearly and non-linearly separable data, though performance may degrade on complex non-linear data structures.

Key Concepts

- **Logistic (Sigmoid) Function:** A function that maps real-valued inputs into a range between 0 and 1, suitable for probability estimation.
- **Cross-Entropy Loss:** A loss function used in Logistic Regression to measure the dissimilarity between predicted probabilities and actual labels.

In this assignment, you will implement this algorithm and evaluate its performance on the *Bank Marketing Dataset*.

2 Support Vector Machine (SVM) Algorithm

The **Support Vector Machine (SVM)** is a supervised learning algorithm for binary classification that finds the optimal hyperplane separating two classes with maximum margin. Introduced by Vapnik, SVM aims to create a robust classifier with a large margin between classes, ensuring better generalization on unseen data.

How the Algorithm Works

1. **Feature Transformation (optional):** If the data is not linearly separable, a kernel function can be applied to transform the data into a higher-dimensional space where a linear separation is possible.
2. **Decision Function:** For any input $\mathbf{x}$, the SVM computes a weighted sum and bias as follows:

$$f(\mathbf{x}) = \mathbf{w}^T \mathbf{x} + b$$

The SVM classifies $\mathbf{x}$ based on the sign of $f(\mathbf{x})$:

$$\text{Predicted class} = \begin{cases} 1, & \text{if } f(\mathbf{x}) \geq 0 \\ -1, & \text{if } f(\mathbf{x}) < 0 \end{cases}$$

3. **Optimization with Hinge Loss:** SVM optimizes the weights and bias by minimizing the hinge loss and maximizing the margin:

$$L = \frac{1}{2}\|\mathbf{w}\|^2 + C\sum \max(0, 1 - y_i f(\mathbf{x}_i))$$

where C is a regularization parameter balancing margin size and classification error.

4. **Support Vectors:** Only data points that lie on the margin boundaries (support vectors) influence the final model. Other points do not affect the optimization.

Key Concepts

- **Maximum Margin Hyperplane:** The hyperplane that maximizes the distance to the nearest data points of both classes.
- **Kernel Trick:** A method to apply SVM on non-linearly separable data by transforming it into a higher dimension.

3 Implementation

3.1 Data Preparation

1. **Download and Load the Data:** Download the Bank Marketing Dataset from the HERE.
2. Load the dataset with Pandas, ensuring that features and target labels are correctly formatted.
3. Calculate the correlation between target feature 'y' and other features
4. **Preprocess the Data:** Analyze and preprocess the data (e.g., normalization).
5. **Training and Validation Split:** Split the dataset into an 80% training and 20% validation set. Explain why splitting is essential for generalization.

3.2 Initialize the Model

1. **Initialization:** For Logistic Regression, initialize weights and bias. And you should create the necessary functions such as sigmoid, fit, predict. For SVM, initialize parameters including the regularization parameter C and any necessary kernel parameters.You will try to reach the best performance of the SVM model by using GridSearchCV.

3.3 Train the Model

1. **Logistic Regression:** Implement Logistic Regression model to training data.
2. **SVM:** Implement SVM model to training data.

4 Evaluation

Test your model's performation using validation set. Show accuracy, precision, recall, and F1 score as metrics for both Logistic Regression and SVM.

4.1 Step 1: Evaluate with Accuracy

$$\text{Accuracy} = \frac{\text{Correct Predictions}}{\text{Total Predictions}}$$

4.2 Step 2: Introduce Precision, Recall, and F1 Score

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

5 Visualization of the Decision Boundary

Visualize the decision boundaries of both Logistic Regression and SVM models. Also you should visualize the confusion matrix, classification report of both model.

6 Multi-class classification

In this assignment, in **multi-class classification task** first you will explore the **Decision Tree algorithm** by implementing it for a multi class classification task using the Weights Dataset. The dataset consists of data collected from 3,360 individuals aged 20 and over, comprising 9 ('BMI_CLASS', 'UNIT_NUM', 'STUB_NAME_NUM', 'STUB_LABEL_NUM', 'YEAR_NUM', 'AGE_NUM', 'ESTIMATE', 'SE', 'FLAG') features and 6 discrete classes ('BMI_CLASS' feature is the label feature). Each of the BMI classes is distributed equally. You will classify which BMI class a person belongs to among the six classes.

7 Implementation

7.1 Step 1: Data Preparation

1. **Download and Load the Data:** In this assignment, you will implement the Decision Tree algorithm and evaluate its performance on the **Weights Dataset**. You can download dataset from HERE. Make sure to load the dataset into your environment correctly.

2. Load the dataset with Pandas, ensuring that features and target labels are correctly formatted.

3. **Training and Validation Split:** Split the dataset into an 80% training and 20% validation set. This split is essential for evaluating the model's ability to generalize to new data.

7.2 Step 2: Initialize the Model

1. Create Decision Tree model using sklearn module.

7.3 Step 3: Train the Model

1. **Decision Tree:** Train the Decision Tree model on the training data. The tree will grow by recursively splitting the data based on the feature that maximizes the information gain or minimizes the impurity (e.g., Gini index or entropy).

2. The tree will continue growing until it reaches the maximum depth or when the stopping criteria (e.g., minimum samples required for a leaf) are met.

7.4 Step 4: Evaluate the Model

1. **Model Evaluation:** Once the tree is trained, evaluate the model on the validation set by making predictions and comparing them to the true labels. Compute performance metrics such as accuracy, precision, recall, F1-score, and confusion matrix.

7.5 Step 5: Visualize the Tree Structure

1. **Visualize the Tree:** After evaluation the Decision Tree, use a tree visualization library such as `plot_tree` from `sklearn.tree` to generate a graphical representation of the decision tree.

2. This visualization will allow you to observe the structure of the tree, including the decision rules at each node, the features used for splitting, and the values at the leaves.

3. The decision tree visualization is an excellent tool to understand how the model makes decisions based on the feature values.

This part of assignment will provide a comprehensive understanding of the Decision Tree algorithm, with a focus on how its performance is evaluated and how its decision boundary is influenced by the underlying data.

8 Decision Tree Algorithm

The **Decision Tree Algorithm** is a non-linear model used for both classification and regression tasks. It splits the data into distinct subsets based on feature values, creating a tree-like structure. Each internal node represents a feature test, each branch represents an outcome of the test, and each leaf node represents a class label (in classification) or a continuous value (in regression). The algorithm recursively divides the dataset into subsets that maximize the homogeneity of the resulting classes or values.

How the Algorithm Works

1. **Initialization:** The decision tree starts with the entire dataset at the root node. The algorithm chooses the best feature to split the data based on a splitting criterion (e.g., Gini Impurity, Information Gain, or Mean Squared Error). The goal is to choose the feature that results in the most homogeneous subsets.

2. **Splitting:** At each node, the decision tree algorithm evaluates all features and selects the one that best separates the data into different classes or values. This is done by calculating a measure of impurity (such as Gini Impurity or Entropy for classification, or Variance for regression) for each possible split and choosing the one with the lowest impurity.

3. **Recursive Partitioning:** After a feature is chosen for the current node, the data is split into subsets based on the possible values of that feature. This process is repeated recursively for each subset, building a tree structure. This continues until one of the stopping criteria is met, such as:

   - The node reaches a predefined depth.
   - The data in the node cannot be further split (e.g., all data points belong to the same class).
   - The node contains fewer than a predefined minimum number of samples.

4. **Prediction:** Once the tree is built, predictions are made by traversing the tree from the root to a leaf node. For classification, the predicted class is the majority class in the leaf node; for regression, it is the average value of the data points in the leaf.

5. **Pruning (Optional):** In some cases, decision trees may overfit the training data, leading to poor generalization. To prevent overfitting, pruning can be applied. This involves removing branches that have little to no impact on the prediction accuracy, simplifying the model and improving its generalization to new data.

Key Concepts

- **Gini Impurity:** A measure of the "impurity" or "purity" of a node. Lower values indicate more homogeneous nodes, where all the data points belong to the same class.

- **Information Gain:** A criterion used to measure the effectiveness of a feature in classifying the data. It is the reduction in entropy (or disorder) after a dataset is split based on a feature.

- **Overfitting:** When the model is too complex, capturing noise or outliers in the training data, which leads to poor performance on unseen data.

- **Pruning:** A technique used to reduce the complexity of a decision tree by removing branches that contribute little to the model's accuracy, thus preventing overfitting.

9 Grading

- Logistic Regression Task (40 points)
- Support Vector Machine Task (30 points)
- Decision Tree Task (30 points)

10 Submit

- The filled-in Jupyter Notebook as both your source code and report.
- You should prepare a ZIP file named <student id>.zip containing the Jupyter notebook in ipynb format as well as its .py (Python file) version containing all your codes. Submit it to submit.cs.hacettepe.edu.tr, where you will be assigned a submission. The file hierarchy is up to you as long as your Jupyter notebook and Python file works fine.

11 Academic Integrity

All work on assignments must be done individually unless stated otherwise. You are encouraged to discuss with your classmates about the given assignments, but these discussions should be carried out in an abstract way. That is, discussions related to a particular solution to a specific problem (either in actual code or in the pseudocode) will not be tolerated. In short, turning in someone else's work, in whole or in part, as your own will be considered a violation of academic integrity. Please note that the former condition also holds for the material found on the web as everything on the web has been written by someone else.