CSCI 3320: Fundamental of Machine Learning **2020-2021 Term 2**

Programming Assignment 2

Instructor: Prof. John C.S. Lui Due: 23:59 on Friday, Apr. 16, 2021

Introduction

This programming assignment consists of two parts.

- The first part will guide you to write a logistic linear discriminator for binary classification and solve it by gradient descent.
- The second part will guide you to implement decision tree in python. It contains gini index calculation, binary decision tree building and a decision tree depth experiment.

1 Binary Logistic Classification

In this section, we will use logistic discriminate to do a binary classification task.

In `ex1.py`, we generate two clusters of data points and split the data to training and test data with the following script:

```
n_samples = 1000
centers = [(-1, -1), (5, 10)]
X, y = make_blobs(n_samples=n_samples, n_features=2, cluster_std=1.8,
                  centers=centers, shuffle=False, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)
```

Note: please do **not** change the parameter `random_state` in the file `ex1.py`.

Notations: we use x, y to stand for scalers, boldface $\boldsymbol{x}, \boldsymbol{y}$ as column vectors, and capital boldface $\boldsymbol{X}, \boldsymbol{Y}$ as matrixes. Denote $\mathcal{D}$ as the data set and $d \in \mathcal{D}$ as a sample $d = \left(\boldsymbol{x}^{(d)}, y^{(d)}\right)$ in $\mathcal{D}$.

1.1 Logistic Function

The logistic function is

$$L(x) = \frac{1}{1 + e^{-x}}.$$

Complete `logistic_func()` in `ex1.py`.

1.2 Gradient Descent Update Rule

Set $g(\boldsymbol{x}|\boldsymbol{w}, w_0) = w_0 + \boldsymbol{w}^T \cdot \boldsymbol{x}$ as the linear function, where $\boldsymbol{x} = [x_1, x_2, \ldots] = [x_i]_i$ and each entry x_i corresponds to a feature. The update rule of logistic regression is as follows:

$$w_0 \leftarrow w_0 + \eta \cdot \sum_{d \in \mathcal{D}} \left(y^{(d)} - L\left(g\left(\boldsymbol{x}^{(d)}\right)\right)\right)$$
$$w_i \leftarrow w_i + \eta \cdot \sum_{d \in \mathcal{D}} x_i^{(d)} \left(y^{(d)} - L\left(g\left(\boldsymbol{x}^{(d)}\right)\right)\right)$$

Complete `train()` in `ex1.py`.

Hint: the convergence of gradient descent can be measured by weight's change, like $\sum_i |w_i^{t+1} - w_i^t| < 10^{-4}$, *where* t *denote the update round.*

1.3 Gradient Descent Update Rule in Matrix Form

To get the matrix form update rule, let's first review some notation tricks. Notice that $g(\boldsymbol{x}|\boldsymbol{w}, w_0) = w_0 + \boldsymbol{w}^T \cdot \boldsymbol{x} = \begin{bmatrix} w_0, \boldsymbol{w}^T \end{bmatrix} \cdot \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}$. Let's denote $\bar{\boldsymbol{w}} = \begin{bmatrix} w_0 \\ \boldsymbol{w} \end{bmatrix}$ and $\bar{\boldsymbol{x}} = \begin{bmatrix} 1 \\ \boldsymbol{x} \end{bmatrix}$. Now, the linear function is $g(\boldsymbol{x}|\boldsymbol{w}, w_0) = \bar{\boldsymbol{w}}^T \cdot \bar{\boldsymbol{x}} = \bar{\boldsymbol{x}}^T \cdot \bar{\boldsymbol{w}}$.

We then write the data set $\mathcal{D}$ as a matrix and its corresponding vector. Notice that each sample $d \in \mathcal{D}$ is $\left(\boldsymbol{x}^{(d)}, y^{(d)}\right)$. We denote $\boldsymbol{X} = \left[\bar{\boldsymbol{x}}^{(1)}, \bar{\boldsymbol{x}}^{(2)}, \ldots\right] = \left[\bar{\boldsymbol{x}}^{(d)}\right]_{d \in \mathcal{D}}$ and $\boldsymbol{y} = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \end{bmatrix} = \left[y^{(d)}\right]_{d \in \mathcal{D}}$. That is, $\mathcal{D} = (\boldsymbol{X}, \boldsymbol{y})$.

With above notations, there is a matrix form of gradient descent update rule:

$$\bar{\boldsymbol{w}} \leftarrow \bar{\boldsymbol{w}} + \eta \cdot \boldsymbol{X}\left(\boldsymbol{y} - L\left(\boldsymbol{X}^T \cdot \bar{\boldsymbol{w}}\right)\right),$$

where the logistic function L is applied to each entries of its input vector.

Complete `train_matrix()` in `ex1.py`.

1.4 Prediction Rule

Use the prediction rule of logistic classification, for input $\boldsymbol{x}$:

$$C(\boldsymbol{x}) = \begin{cases} 1, & p(\boldsymbol{x}) \geq 0.5 \\ 0, & otherwise \end{cases},$$

where $p(\boldsymbol{x}) = L(g(\boldsymbol{x}|\boldsymbol{w}, w_0))$.

Complete `predict()` in `ex1.py`.

1.5 Experiments

Use *ex1.py* to test both `train()` and `train_matrix()` function. Copy down both figures and number of wrong predictions to `Assignment2.pdf`.

2 Decision Tree Classification

In the first part, we implement linear logistic regression for binary classification. However, when there are more then two classes in our data set, applying linear logistic regression is cumbersome and requires further refinements. Instead of extending linear regression, we now turn to *decision tree* which can easily handle the multiple classification task. Specifically, we consider a *triple classification* task. Data is generated as follows in `ex2.py`.

```
n_samples = 1000
centers = [(-1, -1), (5, 10), (10,5)]
X, y = make_blobs(n_samples=n_samples, n_features=2, cluster_std=1.8,
                  centers=centers, shuffle=False, random_state=42)
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=42)
```

Note: please do **not** change the parameter `random_state` in the file `ex2.py`.

2.1 Calculate Gini Index of A Split

Gini index is used in Classification And Regression Trees (CART) algorithm. The Gini index of a set measures the set's *impurity*:

$$\text{GiniIndex}(S) = 1 - \sum_{i=1}^{C} p_i^2,$$

where C is the number of classes, p_i is the prior probability of class i in the set S. For example, if all points in set S are class 1, that is $p_1 = 1$ and $p_i = 0$ for $i > 1$, the Gini index of set S would be 0, which implies that set S is *pure*.

When we split a set S into S_1 and S_2, we use the Gini index to measure the split's impurity via a weighted summation of both subsets' Gini indexes:

$$\text{GiniIndex}(S \to S_1, S_2) = \text{GiniIndex}(S_1)\frac{|S_1|}{|S|} + \text{GiniIndex}(S_2)\frac{|S_2|}{|S|},$$

where $|\cdot|$ is the size of a set. For example, if S only contains points in class 1 and 2 and the split correctly separates S's class 1 points and class 2 points into set S_1 and S_2, the split's GiniIndex would be 0, as both S_1 and S_2 are pure.

Complete the function `gini_index()` of `ex2.py` to calculate the Gini Index of a split.

2.2 Find the Optimal Split

The `split()` function of `ex2.py` finds the optimal vertical or horizontal split plane (i.e., we only split data via one feature's value) of a set S and split it to left set S_l and right set S_r. As we aim to construct a tree, the left set S_l and the right set S_r are two children of the set S.

Complete the function `split()` of `ex2.py`.

Hints: One can use data points' feature value to construct split planes; the optimal split is the one with the smallest Gini index.

2.3 Recursively Build up Decision Tree

We start from the whole set and use `split()` to optimally split it to its left child set and right child set. Then, for each child set, we recursively split it unless the set meets any of the stopping criterion:

- The depth (height) of the decision tree is greater than `max_depth`;
- The number of point in the set is no greater than `min_size`.

Once the set meets any of the conditions, we terminate the split and denote the set as a `leaf`. When all children are `leaves`, we finish the decision tree's construction.

Complete function `build_tree()` of `ex2.py`.

2.4 The Depth of Decision Tree

Use `ex2.py` to test the influence of decision tree's depth in classification. Try `max_depth=3,5,7`. Copy down these three figures and number of wrong predictions to `Assignment2.pdf`. Compare three figures and their wrong prediction times. Write down possible reasons of the result.

3 Submission

Instructions for the submission are as follows. **Please follow them carefully.**

1. Make sure you have answered all questions in your report.
2. Test all your Python scripts before submission. Any script that has syntax error will not be marked.
3. Zip all Python script files, i.e., the `*.py` files in `asgn2.zip` (Please do not change the filenames of the scripts.) and your report (`Assignment2.pdf`) into a single zipped file named `<student-id>_asgn2.zip`, where `<student-id>` should be replaced with your own student ID. e.g., `1155012345_asgn2.zip`.
4. Submit the zipped file `<student-id>_asgn2.zip` via Blackboard System no later than 23:59 on Friday, Apr. 16, 2021.