

National Taiwan University

Machine Learning: HW2

EE3 b03901016

陳昊

October 27, 2016

1 Logistic Regression

The concept of implementation of classifying in this program is based on **discriminative logistic regression**, using **gradient descent** to directly find the bias and weight of each feature of the model. Similar to linear regression, the only different between these two is the design of loss function, logistic regression calculate the **cross entropy**.

Algorithm 1 Compute Loss and Update parameters

```
1: Define model  $y = \text{sigmoid}(b + \sum w_i x_i)$ 
2: Initialize all parameters  $b, w_i, L, \text{Grad\_of\_}w_i = 0$ 
3: for  $i < \text{iteration}$  do
4:    $L \leftarrow 0$ 
5:   for models in all data do
6:      $y \leftarrow b$ 
7:     for each parameter do
8:        $y \leftarrow y + w_i * x_i$ 
9:        $L \leftarrow L + \text{cross\_entropy}(\hat{y}^n - y)$ 
10:    for each parameter do
11:      Compute  $G_i = \text{gradient of } w_i$ 
12:       $\text{Grad\_of\_}w_i \leftarrow \text{Grad\_of\_}w_i + G_i$ 
13:    Update each parameter by Adagrad
14: return Final model
```

(This method of gradient descent include **stochastic gradient descent** and **Adagrad**)

2 Random Forest

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks. It correct for decision trees' habit of overfitting to their training set.

2.1 Tree bagging

To construct the training model of each tree in random forest, we randomly sample part of the training data and use it to train a tree. In addition to the randomization of the sample data, training features of each tree are also randomized. In another word, different trees contain different training data and training features.

To have better performance, the data-size and features number of a **Tree** should be modified, in my design, the size of a **Tree** is randomly picked in range [2000, 2500], and the sample features number is in range [11, 14]. Another important design is the set of split-points of each feature, I compute the mean and variance of each feature base on training data, and sample several points using Gaussian distribution.

2.2 Build Tree

Now use the data and features that are already bagged to build a tree model. First, I construct a class **Node** and a class **Tree**, each **Node** contains some members, such as **feature**, **split-point**, **label**...etc; while a **Tree** contains several **Nodes**.

When input a training data to a node, it will randomly pick some features, and each feature has a set of split points, to calculate the **Gini index** of each split-point. Assume there are m features, and for each feature there are n split-points, then it will calculate the **gini-index** and get $m * n$ different values, then pick the feature with the split-point having the highest score(i.e. Decrease gini-index most) and set the **Node** with that feature and the split-point. Recursively do this process until it cannot split any more. So when it iterate to the leaf of the tree, the label of the node will be set to 0 or 1.

2.3 Ensemble

Since we've build a lot of **Trees**, and each of them have different structure, they combine a random forest. Then how do we decide the final classification of an input data? Just simply vote!

Input each testing data through all the **Trees**, then each **Tree** will return a corresponding value (0 or 1) due to the model structure itself. If more trees say the testing is in class 0, we assume it is 0; while if more trees say it is 1, we assume it to be 1.

3 Experiment

The best score I got in **Logistic Regression** is 0.91667, which is not an satisfying score. While using **Random Forest**, I got better score, which has an best score of 0.96333. The number of **Trees** in random forest effect the prediction result significantly, but the effect decrease when the number is larger and larger.

Tree-size	1	4	16	64	128	512	1024	4096
Training Acc	0.96626	0.98825	0.98925	0.99425	0.99650	0.99547	0.99500	0.99525
Public Acc	0.90000	0.92000	0.96000	0.96333	0.96333	0.96333	0.96333	0.96333

From the table, we can find that there are no significant change when the number of tree

is larger than 128, perhaps just error cause by randomization. Consider the runtime and efficiency, 128 **Trees** is enough for the model.