FBA | QUANTITATIVE FINANCE RESEARCH GROUP

Financial Machine Learning

Homework 4

Due at 07:00 pm (Korea Standard Time) on Sunday, September 4.

Submit one file: written solutions with executable Python code

Problem 1. Text book: Hands-on Machine Learning. Submit .ipynb file.

(a) Practice all the codes in the Text book Chapter 6, 7. And show that they work well.

(b) Load the MNIST data (introduced in Chapter 3), and split it into a training set, a validation set, and a test set (e.g., use 50,000 instances for training, 10,000 for validation, and 10,000 for testing). Then train various classifiers, such as a Random Forest classifier, and Extra-Trees classifier, and an SVM classifier. Next, try to combine them into an ensemble that outperforms each individual classifier on the validation set, using soft or hard voting. Once you have found one, try it on the test set.

(c) Run the individual classifiers from the previous exercise to make predictions on the validation set, and create a new training set with the resulting predictions: each training instance is a vector containing the set of predictions from all your classifiers for an image, and the target is the image's class. Train a classifier on this new training set. Now evaluate the ensemble on the test set. For each image in the test set, make predictions with all your classifiers, then feed the predictions to the blender to get the ensemble's predictions

(d) Train and fine-tune a Decision Tree for the moons dataset by following these steps:

a. Use *make_moons(n_samples=10000, noise=0.4)* to generate a moos dataset.
b. Use *train_test_splist()* to split the dataset into a training set and test set.
c. Use grid search with cross-validation (with the help of the GridSearchCV class) to find good hyperparameter values for a DecisionTreeClassifier. (Try various values for max_leaf_nodes)
d. Train it on the full training set using these hyperparameters, and measure your model's performance on the test set.

Problem 2. Following the pseudo code, implement the algorithm

(a) Implement TreeGenerate method. Use gini index or entropy for attribute selection method. (A\{a} means complementary set)

Input:
Train dataset: $S=\{(\boldsymbol{x}_1,y_1), (\boldsymbol{x}_2,y_1), (\boldsymbol{x}_3,y_1),\dots,(\boldsymbol{x}_m,y_m)\}$.
Attribute set: $A=\{a_1,a_2,\dots,a_d\}$.
Attribute selection method : a procedure to determine the splitting criterion that best partitions the data samples into individual classes. The splitting criterion includes a splitting attribute, and splitting subset. C5.0 utilizes *Gain Ratio* in Eq.(3) to choose the splitting attribute.

Method:

```
1. create node $N$
2. if samples in S are all of the same class, $C$, then
3.     return $N$ as a leaf node labeled with class $C$
4. end if
5. if $A=\emptyset$, or the value of attribute in $S$ are same, then
6.     return $N$ as a leaf node labeled with the majority class in $S$
7. end if
8. find the best splitting attribute $a_*$ in $A$ using attribute selection method
9. for every value $a_*^v$ of $a_*$
10.    label node $N$ with splitting criterion, let $S_v$ be the set of data in $S$ which equal to $a_*^v$ in $a_*$.
11.    if $S_v=\emptyset$, then
12.        attach a leaf labeled with majority class in $S$ to node $N$
13.    else
14.        attach the node returned by TreeGenerate ($S_v$, $A\setminus\{a_*\}$) to node $N$
15.    end if
16. end for
```

Output: a decision tree

(b) Implement bagging method. (Optional. Implement Adaboost algorithm)

```
Input: Data set $TR = \{(x_1,y_1),(x_2,y_2),\dots,(x_1,y_n)\}$;
       Base learning algorithm $L$;
       Number of learning rounds $K$.

Process:
  for $k = 1,\dots,K$:
       $TR_k = Bootsrap(TR)$;     % Generate a bootstrap sample from $TR$
       $h_k = L(TR)$               % Train a base learner $h_k$ from the bootstrap sample
  end.

Output:
$H(X) = argmax_{y\in Y} \sum_{k=1}^{K} l(y = h_k(x))$     % the value of $l(a)$ is 1 if $a$ is $true$ and 0 otherwise.
```

Algorithm 4: AdaBoost algorithm

```
Input: Data set $\mathcal{D} = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \cdots, (\boldsymbol{x}_m, y_m)\}$;
       Base learning algorithm $\mathcal{L}$;
       Number of learning rounds $T$.
Process:
  $D_1(i) = 1/m$.    % Initialize the weight distribution
  for $t = 1, \cdots, T$:
      $h_t = \mathcal{L}(\mathcal{D}, D_t)$;    % Train a weak learner $h_t$ from $\mathcal{D}$ using distribution $D_t$
      $\epsilon_t = \Pr_{i \sim D_i}[h_t(\boldsymbol{x}_i \neq y_i)]$;    % Measure the error of $h_t$
      $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;    % Determine the weight of $h_t$
      $D_{t+1}(i) = \frac{D_t(i)}{Z_t} \times \begin{cases} \exp(-\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) = y_i \\ \exp(\alpha_t) & \text{if } h_t(\boldsymbol{x}_i) \neq y_i \end{cases}$
                  $= \frac{D_t(i)\exp(-\alpha_t y_i h_t(\boldsymbol{x}_i))}{Z_t}$    % Update the distribution, where $Z_t$ is
                      % a normalization factor which enables $D_{t+1}$ be a distribution
  end.
Output: $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$
```