Statistical Methods for Data Science

DATA7202

Semester 1, 2024

Lab 2

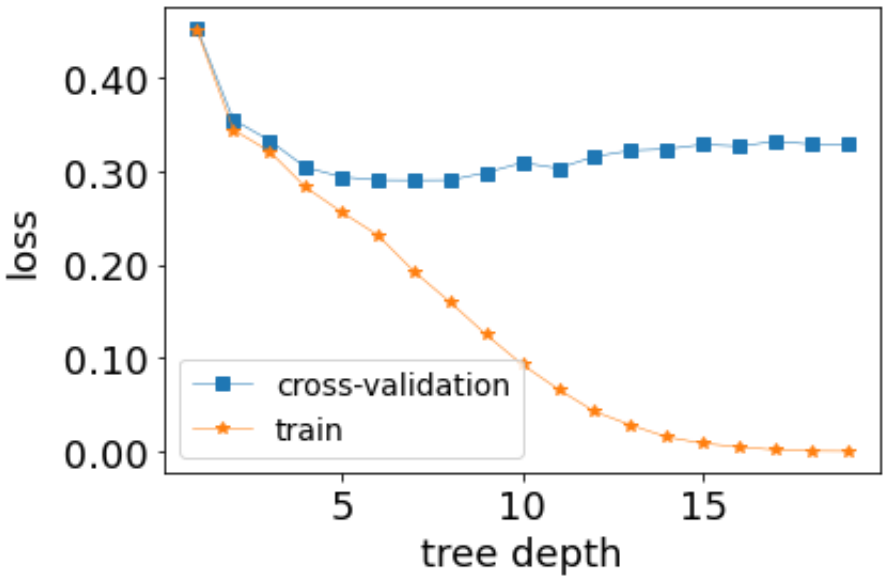
Objectives

On completion of this laboratory session you should be able to understand and implement decision trees.

1. Consider the following data generation process.

```
X, y =  make_blobs(n_samples=5000, n_features=10, centers=3,
                                random_state=10, cluster_std=10)
```

We are going to find the best decision tree depth using cross-validation procedure. Write a code to reproduce the following Figure.



Solution

```
from sklearn.datasets import make_blobs
from sklearn.metrics import zero_one_loss
from sklearn.model_selection import cross_val_score
from sklearn.tree import DecisionTreeClassifier
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.ticker import FormatStrFormatter

def custom_zer_one_score(model, X, y):
    y_pred = model.predict(X)
    return zero_one_loss(y, y_pred)

if __name__ == "__main__":
    X, y =  make_blobs(n_samples=5000, n_features=10, centers=3,
                               random_state=10, cluster_std=10)

    model = DecisionTreeClassifier(random_state=0)

    tree_depth = range(1,20)

    xlist = []
    trlist = []
    cvlist = []

    for d in tree_depth:
        xlist.append(d)
        model.max_depth=d
        cv = np.mean(cross_val_score(model, X, y, cv=10,
          scoring=custom_zer_one_score))
        cvlist.append(cv)
        model.fit(X, y)
        trlist.append(custom_zer_one_score(model, X, y))


    font = {'family' : 'normal',
            'weight' : 'normal',
            'size'   : 18}
    plt.rc('font', **font)


    f = plt.figure()
    ax=plt.gca()
    ax.yaxis.set_major_formatter(FormatStrFormatter('%5.2f'))

    cv = plt.plot(xlist, cvlist,'-s' , linewidth=0.5,
      label='cross-validation')
    tr = plt.plot(xlist, trlist,'-*' , linewidth=0.5, label='train')
    plt.xlabel('tree depth', fontsize=18, color='black')
    plt.ylabel('loss', fontsize=18, color='black')
    plt.xticks(fontsize=18)
    plt.yticks(fontsize=18)
    plt.legend(fontsize=14,loc=3)
    plt.show()
```

2. Explain why bagging decision trees is a special case of random forest.

   Solution

   It is easy to see that bagging is a special case of random forest. Specifically, a random forest h_{rf} with $m = p$, where m is the number of subsets that we consider at every split and p is the number of explanatory variables, satisfies $h_{rf} = h_{bag}$.

3. Consider the `mnist` dataset.

   (a) Plot several images from the dataset.

   (b) Split the dataset to train and test sets (75% train and 25% test).

(c) Fit logistic regression model and evaluate the miss-classification rate.

(d) Fit a random forest classifier, evaluate the miss-classification rate, and compare to the results obtained in (c).

Solution

```
# -*- coding: utf-8 -*-


import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import zero_one_loss
from PIL import Image
import warnings
warnings.filterwarnings("ignore")

# read the data
data = pd.read_csv("train.csv")
data.head()


X = data.iloc[:, 1:]
y = data['label']

# print image
tmp = np.array(X.iloc[10].values.reshape((28,28))).astype(np.uint8)
img = Image.fromarray(tmp)
img

X_train, X_test, y_train, y_test = \
    train_test_split(X, y, test_size=0.25, random_state=1179)


################################################################
# Logistic regression
################################################################

reg = LogisticRegression(solver = "lbfgs")
reg.fit(X_train,y_train)
y_pred = reg.predict(X_test)
#
print("Logistic Regression 0/1 loss = ", zero_one_loss(y_pred, y_test))



################################################################
# Random forest
################################################################
rfc = RandomForestClassifier(n_jobs=-1, n_estimators=100)
rfc.fit(X_train,y_train )
#
y_pred = rfc.predict(X_test)
#
print("Random Forest 0/1 loss = ", zero_one_loss(y_pred, y_test))
```