Journal of Computing and Electronic Information Management

ISSN: 2413-1660

The practical steps of the five algorithms

Yu Zhao [1], Tong Bai [2]

[1] Institute of Date Science, City University of Macao, Tapai, China

[2]Institute of Business Adminstration, Krirk University, Bangkok, Tailand

Abstract: This article puts forward some insights into the steps of the five algorithms, LDA, DT, XGBOOST, LIBSVM and BP. The purpose is to better help people understand the differences between the five algorithms and how to use them to bring more convenient operations to programming.

Keywords: algorithm, dispersion matrix.

1. LDA Algorithm

1.1 Define variables

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
#Calculate the mean value, the input data is required to be in numpy matrix format, the rows indicate the number of samples, and the columns indicate features
def meanX(data):
    return np.mean(data, axis=0) #axis=0 Means to find the average according to the column, if you enter list, then axis=1
```

1.2 Calculate the sub-item si of the within-class dispersion matrix (Si)

```
def compute_si(xi):
    n = xi.shape[0]
    ui = meanX(xi)
    si = 0
    for i in range(0, n):
        si = si + (xi[i, :] - ui).T * (xi[i, :] - ui)
    return si
```

1.3 Calculate the inter-class dispersion matrix (Sb)

```
def compute_Sb(x1, x2):
    dataX=np.vstack((x1,x2))# Merge sample
```

```
    print ("dataX:", dataX)
    #Calculate the mean
    u1=meanX(x1)
    u2=meanX(x2)
    u=meanX(dataX) #mean of all samples
    Sb = (u-u1).T * (u-u1) + (u-u2).T * (u-u2)
    return Sb
def LDA(x1, x2):
                    1.4 Calculate the within-class dispersion matrix (Sw)
    s1 = compute_si(x1)
    s2 = compute_si(x2)
    # Sw=(n1*s1+n2*s2)/(n1+n2)
    Sw = s1 + s2
                     1.5 Calculate the inter-class dispersion matrix (Sb)
    # Sb=(n1*(m-m1).T*(m-m1)+n2*(m-m2).T*(m-m2))/(n1+n2)
    Sb = compute_Sb(x1, x2)
             1.6 Find the eigenvector corresponding to the largest eigenvalue
    eig_value, vec = np.linalg.eig(np.mat(Sw).I * Sb)  # eigenvalue and eigenvector
    index_vec = np.argsort(-eig_value)  # sort eig_value from big to small, return the
index
    eig_index = index_vec[:1]  # Take out the index of the largest eigenvalue
    w = vec[:, eig_index]  # Take out the eigenvector corresponding to the largest
eigenvalue
    return w
                                    1.7 Construct the database
def createDataSet():
    X1 = np.mat(np.random.random((8, 2)) * 5 + 15)  # Category A
    X2 = np.mat(np.random.random((8, 2)) * 5 + 2)   # Category B
    return X1, X2
if __name__ == "__main__":
    x1, x2 = createDataSet()
    print (x1, x2)
    w = LDA(x1, x2)
    print ("w:", w)
```

2. DT algorithm

```
From sklearn. metrics import accuracy_score,precision_score,recall_score,f1_score
X_train,X_test,y_train,y_test=train_test_split(StS,y,test_size=0.4,random_state=0)
```

```
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X_train, y_train)
pre_labels = clf.predict(X_test)
print('accuracy score:',accuracy_score(y_test,pre_labels,normalize=True))
print('recall score:',recall_score(y_test,pre_labels))
print('precision score:',precision_score(y_test,pre_labels))
print('f1  score:',f1_score(y_test,pre_labels))
```

3. XGBoost algorithm

```
import xgboost as xgb
from sklearn.preprocessing import StandardScaler
# Record program running time
import time
start_time = time.time()
from xgboost.sklearn import XGBClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_ score,precision _score,recall_ score,f1_score,
classification_ report,roc_auc_score
bankChurn = pd.read_csv('D:/work/lost data and dictionary/test/bankChurn.csv')#
Original data
bankChurn_data = pd.read_csv('D:/work/lost data and dictionary/ test/bank
Churn_data.csv')# Preprocess data
Y_train=bankChurn['CHUR0_CUST_I0D']# tag
StS=StandardScaler().fit_transform(bankChurn_data)
X_train,X_test,y_train,y_test=train_test_split(StS,Y_train,test_size=0.4,random_state
=None)
print(X_train.shape, X_test.shape)
# Model parameter settings
xlf = xgb.XGBClassifier(max_depth=10,
                        learning_rate=0.1,
                        n_estimators=10,
                        silent=True,
                        objective='binary:logistic',
                        nthread=-1,
                        gamma=0,
                        min_child_weight=1,
                        max_delta_step=0,
                        subsample=0.85,
```

```
                colsample_bytree=0.7,
                colsample_bylevel=1,
                reg_alpha=0,
                reg_lambda=1,
                scale_pos_weight=1,# This value is because the categories are very
unbalanced.
                seed=1440)
xlf.fit(X_train, y_train, eval_metric='error', verbose = True, eval_set = [(X_test,
y_test)],early_stopping_rounds=100)
# Calculate auc score, forecast
preds = xlf.predict(X_test)
pre_pro = xlf.predict_proba(X_test)[:,1]
print('accuracy score:',accuracy_score(y_test,preds ,normalize=True))
print('classification report:',classification_report(y_test,preds ))
print('precision score:',precision_score(y_test,preds ))
print('roc_auc_score:%f' % roc_auc_score(y_test,pre_pro))
# Output running time
cost_time = time.time()-start_time
print("xgboost success!",'\n',"cost time:",cost_time,"(s)......")
```

4. Libsvm algorithm

```
import os
os.chdir('C:\libsvm-2.81\python')
from svmutil import *
from sklearn.metrics import accuracy_score,classification_report
y,x=svm_read_problem('bankchurnLibsvm.txt')# convert to libsvm format
# print(type(x))
x=np.array(x)
y=np.array(y)
stratified_folder=StratifiedKFold(n_splits=4,random_state=0,shuffle=True)
for train_index,test_index in stratified_folder.split(x,y):
    print('shuffled train index:',train_index)
    print('shuffled test index:', test_index)
    print('shuffled x_train:', x[train_index])
    print('shuffled x_test:', x[test_index])
    print('shuffled y_train:', y[train_index])
    print('shuffled y_test:', y[test_index])
    print('.......')
```

```
y_train=list(y[train_index])
y_test=list(y[test_index])
x_train=list(x[train_index])
x_test=list(x[test_index])
m=svm_train( y_train,x_train,'-c 4  -g 2')
p_label,p_acc,p_val=svm_predict(y_test,x_test,m)
print('accuracy score:',accuracy_score(y_test,p_label ,normalize=True))
print('classification report:',classification_report(y_test,p_label ))
```

5. BP neural network algorithm

```
import pandas as pd
import numpy as np
from sklearn.model_selection import cross_val_score
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score,roc_auc_score
from sklearn.metrics import accuracy_ score, precision_ score, recall_ score, f1_score,classification_report
bankChurn = pd.read_csv('D:/work/lost data and dictionary/test/bankChurn.csv')
X_data = pd.read_csv('D:/work/lost data and dictionary/test/bankChurn_data.csv')
X_data=X_data.values[:,:]
Y_label=bankChurn['CHUR0_CUST_I0D']
Y_label=Y_label.values[:]
data=np.hstack((X_data,Y_label.reshape(Y_label.size,1)))## Merge the sample set and the label
np.random.shuffle(data)## Shuffle data
X=data[:,:-1]
Y=data[:,-1]
train_x=X[:-8620]
test_x=X[-8620:]
train_y=Y[:-8620]
test_y=Y[-8620:]# Data5:5
######mlpclassifier_data():###Multilayer perceptron algorithm, BP algorithm
classifier=MLPClassifier(hidden_layer_sizes=(30,),activation='logistic',max_iter=1000)
clf=classifier.fit(train_x,train_y)
train_score=classifier.score(train_x,train_y)
test_score=classifier.score(test_x,test_y)
print('train_score:',train_score)
print('test_score:',test_score)
```

```
#### Get other classification effects####
pre_labels = clf.predict(test_x)
pre_pro = clf.predict_proba(test_x)[:,1]
print('accuracy score:',accuracy_score(test_y,pre_labels,normalize=True))
print('recall score:',recall_score(test_y,pre_labels))
print('classification report:',classification_report(test_y,pre_labels))
print('precision score:',precision_score(test_y,pre_labels))
print('f1  score:',f1_score(test_y,pre_labels))
print('roc_auc_score:%f' % roc_auc_score(test_y,pre_pro))
```

References

[1] DATTA P, MASAND B, MANI D R, et al. Automated cellular modeling and prediction on a large scale. Artificial Intelligence Review, 2000, 14(6) : 485- 502.

[2] YAN Lian, MILLER D J, MOZER M C, et al. Improving prediction of customer behavior in nonstationary environments/ /Proc of the1st International Joint Conference on Neural Networks. Washington DC: IEEE Press, 2001: 2258- 2263.

[3] AUW H, CHENKCC, YAO Xin. Anovel evolutionary data miningalgorithm with applications to churn prediction. Evolutionary Computation, 2003, 7(6): 532- 545.

[4] MOZERMC, WOLNIEWICZ R. Predicting subscriber dissatisfaction and improving retention in the wireless telecommunications industry. Neural Networks, 2000, 11(3) : 532-545.

[5] HWANG H, JUNG T, SUH E. An LTV model and customer segmentation based on customer value: acase study on the wireless telecommunication industry[J]. Expert Systems with Applications, 2004, 26(2) : 181- 188.