MO444 – Aprendizado de Máquina

Edgar Rodolfo Quispe Condori - RA 192617

June 17, 2017

1 Dataset

The original dataset is ex6-train.csv but in order to test the algorithms we create a new dataset small_7000.csv that contains 7000 random elements from the original dataset. This let us test the algorithms in smallest time and have and intuition of which algorithm would be better in ex6-train.csv.

The algorithms where tested in ex-train.csv in ascending order of its MAE obtained in small_7000.csv and the final classifier is the one with best MAE in the complete dataset. Note that some methods did not end to train in the complete dataset because of the high number data.

2 Preprocessing

For the preprocessing we consider normalization. Results with and without normalization are compared.

3 Regressors

The list of regressors and the list of hyperparameters tested is shown below.

Regressors	**hyperparameter**
Support Vector Machine (SVM)	kernel: linear, rbf C : $2^{-5}, 2^{0}, 2^{5}, 2^{10}$ gamma : $2^{-15}, 2^{-10}, 2^{-5}, 2^{0}, 2^{5}$
Random Forest (RF)	n_estimators : 100, 200, 400, 800 max_features : 8, 10, 30, 32
Multi Layer Perceptron (MLP)	hidden_layer_sizes : 3, 7, 10, 20 solver : lbfgs
Gradient Boosting Machine (GBM)	n_estimators : 30, 70, 100, 400, 800 learning_rate : 0.1, 0.05 max_depth: 5, 10
Bagging Regressor Machine (BRM)	base_estimator: $KNeighborsRegressor(n\_neighbors = 1)$, $MLPRegressor(solver =' lbfgs', hidden\_layer\_sizes = (20,))$ n_estimators : 100, 400, 800 max_features : 5,10, 30,40

Table 1: Algorithm and hyperparameters tested.

4 Evaluation

Because of the quantities of data, the evaluation was made with a 3-fold, a grid search is run for each fold and the score of an algorithm is the best score in the folds.

5 Results

The results are show in the tables below. Note that some algorithm results are not included for the complete dataset because of computer power and time limitations.

Regressor	**small_7000.csv**		**ex6-train.csv**	
	hyperparameters	**MAE**	**hyperparameters**	**MAE**
SVM	kernel: linear C : 0.03125 gamma : $3.0517578125e-05$	0.08919	- - -	- - -
RF	n_estimators : 200 max_features : 10	0.08165	n_estimators : 800 max_features : 8	0.08820
MLP	hidden_layer_sizes : 3 solver : lbfgs	0.08575	hidden_layer_sizes : 7 solver : lbfgs	0.11058
GBM	n_estimators : 70 learning_rate : 0.1 max_depth: 5	0.08616	n_estimators : 800 learning_rate : 0.05, max_depth: 5	0.09554
BRM	base_estimator: *MLP* n_estimators : 800 max_features : 40	0.07394	base_estimator: *MLP* n_estimators : 800 max_features : 40	0.08382

Table 2: Results for data without normalization.

Regressor	**small_7000.csv**		**ex6-train.csv**	
	hyperparameters	**MAE**	**hyperparameters**	**MAE**
SVM	kernel: linear C : 0.03125 gamma : $3.0517578125e-05$	0.08949	- - -	- - -
RF	n_estimators : 400 max_features : 10	0.08182	n_estimators : 800 max_features : 10	0.08866
MLP	hidden_layer_sizes : 3 solver : lbfgs	0.08755	hidden_layer_sizes : 7 solver : lbfgs	0.11058
GBM	n_estimators : 70 learning_rate : 0.1 max_depth: 5	0.08646	n_estimators : 800 learning_rate : 0.05, max_depth: 5	0.09554
BRM	base_estimator: *MLP* n_estimators : 800 max_features : 40	0.07418	base_estimator: *MLP* n_estimators : 800 max_features : 40	0.08322

Table 3: Results for data with normalization.

From the results we can see that the hyperparameter does not change to much from small_7000.csv and ex6-train.csv, the same happens for the results with and without normalization. The best regressor is the Bagging Regressor Machine with a Multilayer Perceptron as base estimator, it gets a MAE of 0.08382 with normalization and 0.08322 without normalization.

6 Code

```
import numpy as np
import csv
import pandas as pd

from sklearn import preprocessing
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import KFold
from sklearn.metrics import mean_absolute_error

from sklearn.neighbors import KNeighborsRegressor
from sklearn.svm import SVR
from sklearn.neural_network import MLPRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.ensemble import BaggingRegressor
from sklearn.decomposition import PCA
from sklearn.externals import joblib


from sklearn import discriminant_analysis
def read_data():
  file_name = "ex6-train.csv"
  data = pd.read_csv(file_name)
  data = np.array(data).astype(np.float)

  labels = data[:, -1]
  data = data[:, 0:-1]
  return data, labels

def preprocess_data(x, standardize = True, pca = True):
  #standardize data
  if standardize:
      x = preprocessing.StandardScaler().fit_transform(x)
  if pca:
      x = PCA(n_components =  0.9999).fit_transform(x)
  return x

def three_validation(x, y, clf_name, params, name_metada = "", cnt = 1):
    clfs = { 'knn': KNeighborsRegressor, 'svm': SVR, 'mlp': MLPRegressor,
    'rf': RandomForestRegressor, 'gbm': GradientBoostingRegressor, 'brm': BaggingRegressor}
    gridSearch = GridSearchCV(clfs[clf_name](), params, cv = 3,
    scoring = 'neg_mean_absolute_error', n_jobs = 5)
    gridSearch.fit(x, y)
    name = clf_name + str(gridSearch.best_params_) + str(gridSearch.best_score_) + name_metada
    if clf_name != 'brm':
        joblib.dump(gridSearch, name + '_GS.pkl')
    else:
        print name
        joblib.dump(gridSearch, str(cnt) + name_metada + '_GS.pkl')
    clf_eval = clfs[clf_name](**gridSearch.best_params_).fit(x, y)
    if clf_name != 'brm':
        joblib.dump(clf_eval, name + '_REG.pkl')
    else:
        joblib.dump(clf_eval, str(cnt) + name_metada + '_REG.pkl')

def svm_evaluation(x, y, model = "five_three", name_metada = ""):
    print "Support Vector Machine"
    params = {'kernel': ['linear', 'rbf'], 'C' : [2**(-5), 2**(0), 2**(5), 2**(10)],
    'gamma': [2**(-15), 2**(-10), 2**(-5), 2**(0), 2**(5)]}
    if model == "five_three":
        five_three_validation(x, y, 'svm', params)
    if model == "three":
        three_validation(x, y, 'svm', params, name_metada)

def mlp_evaluation(x, y, model = "five_three", name_metada = ""):
    print "Multi Layer Perceptron"
    params = {'hidden_layer_sizes' : [(3,), (7, ), (10, ), (20, )], 'solver' : ['lbfgs']}
    if model == "five_three":
        five_three_validation(x, y, 'mlp', params)
    if model == "three":
        three_validation(x, y, 'mlp', params, name_metada)
def rf_evaluation(x, y, model = "five_three", name_metada = ""):
```

```
    print "Random Forest"
    params = {'n_estimators' : [100, 200, 400, 800], 'max_features' : [8, 10, 30, 32]}
    if model == "five_three":
        five_three_validation(x, y, 'rf' , params)
    if model == "three":
        three_validation(x, y, 'rf' , params, name_metada)

def gb_evaluation(x, y, model = "five_three", name_metada = ""):
    print "Gradient Boosting Machine"
    params = {'n_estimators' : [30, 70, 100, 400, 800], 'learning_rate' : [0.1, 0.05],
    'max_depth': [5, 10]}
    if model == "five_three":
        five_three_validation(x, y, 'gbm', params)
    if model == "three":
        three_validation(x, y, 'gbm', params, name_metada)

def brm_evaluation(x, y, model_evaluation = "five_three", name_metada = ""):
    models = [PRegressor(solver = 'lbfgs', hidden_layer_sizes = (20,)),
              KNeighborsRegressor(n_neighbors = 1)]
    cnt = 666
    for model in models:
        print "Bagging Regressor Machine"
        params = {'base_estimator': [model], 'n_estimators' : [100, 400, 800],
        'max_features' : [5,10, 30,40]}
        if model_evaluation == "five_three":
            five_three_validation(x, y, 'brm', params)
        if model_evaluation == "three":
            three_validation(x, y, 'brm', params, name_metada, cnt)
        cnt += 1

def evaluate_finalRegressor():
    reg = joblib.load('models/big/666_1_0_REG.pkl')
    data = pd.read_csv('ex6-test.csv')
    data = np.array(data).astype(np.float)
    print  data.shape
    pred = reg.predict(data)
    df = pd.DataFrame(pred)
    df.to_csv("solutions.csv", header = False, index = False)

if __name__ == "__main__":
    x, y = read_data()

    print "standardize = False, pca = False ======================== "
    mlp_evaluation(x, y, "three", name_metada = "_0_0")
    svm_evaluation(x, y, "three", name_metada = "_0_0")
    rf_evaluation(x, y, "three", name_metada = "_0_0")
    brm_evaluation(x, y,"three", name_metada = "_0_0")
    gb_evaluation(x, y, "three", name_metada = "_0_0")

    xx = preprocess_data(x.copy(), standardize = True, pca = False)
    print "standardize = True, pca = False ======================== "
    svm_evaluation(xx, y, "three", name_metada = "_1_0")
    mlp_evaluation(xx, y, "three", name_metada = "_1_0")
    rf_evaluation(xx, y, "three", name_metada = "_1_0")
    brm_evaluation(xx, y,"three", name_metada = "_1_0")
    gb_evaluation(xx, y, "three", name_metada = "_1_0")

    evaluate_finalRegressor()
```