PROIECT INTELIGENTA ARTIFICIALA – 2019

BlackBox Classification

https://www.kaggle.com/c/ml-unibuc-2019-24

Henning Erik Raimar

Grupa 243

Facultatea de Matematica si Informatica,

Universitatea Bucuresti

Descrierea problemei: "A black-box learning challenge in which competitors train a classifier on a dataset that is not human readable, without knowledge of what the data consists of. They are scored based on classification accuracy on a given test set."

Scor obtinut pe datele de test (20%) : 0.835

Solutia propusa:

Pentru clasificarea datelor am aplicat o retea de perceptron (MLPClassifier) astfel:

Initial am incarcat datele folosind libraria pandas. Din teste precedente, prin studierea matricei de confuzie am constatat ca datele sunt relative inconsistente pe clasele 5, 6, 7, dar in mod special pe 5 si 7, iar (probabil) aceasta este una dintre cauzele pentru care aveam foarte multe clasificari gresite pe aceste date.

Pentru rezolvarea acestei probleme am triplet datele clasificate cu 5 si 7, iar datele clasificate cu 6 au fost dublate.

Dupa aceasta operatiune am dublat dataset-ul pentru a oferi retelei ocazia sa "invete" mai bine corelarile dintre date, intrucat am aplicat si un "noise". De asemenea, am amestecat datele dupa aceasta operatiune pentru a simula un mediu mai "realist" de invatare.

Plecand de la idea scrisului de mana care difera de la persoana la persoana, am presupus ca datele de test pot fi distorsionate. Astfel am aplicat un "gaussian noise" intre 0 si 0.07 cu ajutorul numpy, de dimensiunea dataset-ului, pe care l-am adaugat datelor mele.

Avand datele procesate, am impartit dataset-ul in date de antrenare (80%) si testare (20%) si voi antrena reteaua de perceptroni pe datele de antrenare si voi testa pe cele de validare.

Pentru reteaua de perceptroni am ales sa construiesc 2 straturi ascunse de 200 respectiv 150, functia de activare "Rectified Linear Unit" (Relu) si algoritmul de compilare Adam (asemanator cu Stochastic Gradient Descent insa cu learning rate adaptive). De asemenea am folosit parametrul "shuffle=True" pentru a nu testa mereu pe aceleasi date, intrucat imi pot face o idee mai clara de performanta

algoritmului, iar toleranta pentru Early Stopping am lasat-o la valoarea de 0.0001(1e-4) pentru a preveni overfitting-ul.

Am calculat **matricile de confuzie** atat pentru datele de antrenare cat si de testare, pentru a imi face o idee in privinta overfitting-ului.

In final am testat prin metoda **3-Fold Cross Validation** folosind functia cross_val_score din sklearn pentru a avea o metrica mai buna in privinta preciziei algoritmului, obtinand o acuratete medie de “0.9711” si o deviatie standard de “0.0050”.

COD SURSA:

```
# Import librariile
import numpy as np
import pandas as pd
from sklearn.utils import shuffle
def get_accuracy(pred, real):
    return len(pred[pred == real]) / len(pred)
# Import dataset-ul
X = pd.read_csv('train_samples.csv', header=None, nrows = 5000)
y = pd.read_csv('train_labels.csv', header=None, nrows = 5000)
print('Dataset loaded')
# Pentru clasele 5 si 7 triplez observatiile, iar pentru clasa 6 dublez
for i in range(len(y)):
    if y.iloc[i][0] == 5 or y.iloc[i][0] == 7:
        y = y.append(y.iloc[i])
        y = y.append(y.iloc[i])
        X = X.append(X.iloc[i])
        X = X.append(X.iloc[i])
    elif y.iloc[i][0] == 6:
        y = y.append(y.iloc[i])
        X = X.append(X.iloc[i])
print ('Classes 5 and 7 tripled, 6 doubled')
X = X.append(X)
```

```
y = y.append(y)
X, y = shuffle(X, y)
print('Dataset doubled and shuffled')
mu, sigma = 0, 0.07
# Creez un noise de maxim 0.07, de acelasi shape ca variabila X pe care il adaug pe date
noise = np.random.normal(mu, sigma, [X.shape[0],X.shape[1]])
X = X + noise
print('Noise generated')
# Impart dataset-ul in 80 - 20
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2)
# Implementez retea de perceptroni
from sklearn.neural_network import MLPClassifier # importul clasei
classifier = MLPClassifier(hidden_layer_sizes=((200, 150)),
                           activation='relu', solver='adam', batch_size='auto',
                           learning_rate='adaptive', max_iter=100, shuffle=True,
                           random_state=None, tol=0.0001,
                           early_stopping=True, validation_fraction=0.2, verbose = True)
classifier.fit(X_train, y_train)
# Prezic pe datele de antrenare si testare
y_pred = classifier.predict(X_test)
x_pred = classifier.predict(X_train)
from sklearn.metrics import confusion_matrix
cm_test = confusion_matrix(y_test, y_pred)
cm_train = confusion_matrix(y_train, x_pred)
y_pred.shape = (len(y_pred), y_test.shape[1])
x_pred.shape = (len(x_pred), y_train.shape[1])
print('Accuracy TEST: ', get_accuracy(y_pred, y_test))
print('Accuracy TRAIN: ', get_accuracy(x_pred, y_train))
# Aplic 3-Fold Cross Validation si afisez media de acuratete si deviatia standard
from sklearn.model_selection import cross_val_score
accuracies = cross_val_score(estimator = classifier, X = X_train, y = y_train, n_jobs = -1, cv = 3)
avg_accuracy = accuracies.mean()
```

```
std_dev = accuracies.std()
print('Avg accuracy: ', avg_accuracy)
print('Std_dev: ', std_dev)
# Creez fisierul csv de output
print('----- CREATING KAGGLE SUBMISSION FORMAT ----')
to_predict = pd.read_csv('test_samples.csv', header=None)
results = pd.DataFrame(columns = ['Id', 'Prediction'])
sample_predictions = classifier.predict(to_predict)
for i in range(len(to_predict)):
    results = results.append({'Id': i+1, 'Prediction':sample_predictions[i]}, ignore_index=True)
results.to_csv('perceptron-15k-noise-07-2-layers.csv', encoding='utf-8', index=False)
```

LINK GITHUB: https://github.com/Katzuno/BlackBoxClassification-IA-Sem2