به نام خدا

گزارش‌کار پروژه‌ی هوش محاسباتی

عنوان پروژه: پیش‌بینی آب‌وهوا با شبکه عصبی

نام استاد: استاد تورانی

نام اعضای گروه: محمدعلی اسدی، حمیدرضا زارع، امیر مدوی ثابت

DataSet

```
1 21 9 12 8 100 46 1022 1018 2 0 14 0100
1 21 8 12 8 100 49 1020 1015 3 0 14 0100
1 21 9 11 7 100 40 1018 1015 4 0 10 0100
1 21 11 12 7 94 40 1019 1015 4 0 14 0100
1 22 11 11 4 100 31 1019 1016 4 0 10 0100
1 20 9 13 8 100 46 1022 1019 2 0 14 0100
1 20 7 10 3 100 33 1022 1019 5 0 19 0100
1 20 7 13 4 93 35 1024 1020 4 1 13 0100
1 19 7 9 5 100 40 1023 1020 4 1 13 0100
1 19 7 9 4 100 37 1022 1018 4 1 19 0100
1 19 7 8 5 100 43 1022 1019 3 1 13 0100
1 19 8 9 6 93 43 1022 1018 3 0 13 0100
1 18 6 8 4 100 40 1021 1016 3 0 14 0100
1 19 5 8 2 100 32 1020 1016 2 0 10 0100
1 20 7 9 4 93 35 1020 1016 3 0 11 0100
1 14 6 8 6 100 67 1021 1017 1 0 11 0100
1 20 8 9 3 100 33 1020 1016 4 1 19 0100
1 22 7 6 -3 87 19 1020 1015 5 0 10 0100
1 15 11 12 4 100 54 1015 1008 2 1 40 0001
1 16 10 13 9 100 63 1013 1010 2 0 14 0100
1 16 8 11 8 100 63 1019 1014 2 0 6 0100
1 15 6 10 6 100 63 1021 1017 3 0 23 0100
1 16 6 9 6 100 52 1018 1015 4 0 27 0100
1 17 6 9 5 100 45 1017 1013 2 0 23 0100
1 19 5 7 3 100 35 1019 1014 4 1 24 0100
```

دیتاست برنامه به‌صورت time series بوده که متشکل از 13 المِنت است. هرکدام بیانگر یک ویژگی آب‌وهوایی مثل دما، رطوبت، جهت باد و ... بوده و المِنتِ نهایی(سیزدهم) نشان‌دهنده‌ی وضعیت آب‌وهوا است ک از روی 12 المنت ابتدایی نتیجه‌گیری می‌شود. این المنت 4 حالتِ ممکن دارد که به صورت 0001، 0010، 0100 و 1000 هستند.

Required Libraries

```
import numpy as np
import pandas as pd
from keras.utils import np_utils
from sklearn.preprocessing import StandardScaler
```

Numpy: کتاب خانه‌ی آماری برای کار کردن با اعداد و آرایه‌ها

Pandas: برای ورودی گرفتن و اعمال عملیات روی دیتاست و فایل‌های csv

Keras: برای عملیات train کردن برروی دیتا

Sklearn: کتاب‌خانه‌ای ساده‌ترشده نسبت به keras که کار یادگیری روی داده‌ها را انجام می‌دهد.

Separating DataSet

```
x = dataset.iloc[:,:12].values
y = dataset.iloc[:,12:13].values
```

جداسازی 12 فیلد اول از آخرین المنت

Scaling

```
sc= StandardScaler()

x1 = sc.fit_transform(x)
```

ما از این تابع برای طبقه‌بندی داده ها با مقیاس کوچک‌تر استفاده می‌کنیم تا کار با آن‌ها ساده‌تر باشد.

Encoding

```
from sklearn.preprocessing import OneHotEncoder

ohe = OneHotEncoder()
ohe.categories='auto'
z = ohe.fit_transform(y).toarray()
```

تبدیل چهار حالت آب‌وهوایی به یک ماتریس 4*4

Train & Test

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x1,z,test_size=0.1)
```

تقسیم دیتاست به دو قسمت train و test با نسبت 9 به 1

ANN call

```
from model_archi import *
model = model_archi.build(12,4)
model.compile(loss='binary_crossentropy',
 optimizer='adam', metrics=['accuracy'])
```

فراخوانی کلاس شبکه ع صبی و ساخت نمونه‌ای از آن براساس ورودی(تعداد پارامترهای داده‌های ورودی) و خروجی(تعداد لیبل های تعریف‌شده برای وضعیت آب‌وهوا)

Starting Train

```
history = model.fit(x_train, y_train, epochs=100, batch_size=10
                    , validation_data=(x_test,y_test))
```

با استفاده از تابع fit می‌توانیم آموزش شبکه عصبی را شـــروع کنیم. مقـادیر x_train و y_train مقادیر داده شـــده برای اجرا در شـــبکه هســـتند(فاز یادگیری)

epoch : داده ها را در دســـته‌های کوچک ذخیره می‌کند و یادگیری برای کامپیوتر آسان‌تر شود.

Evaluation

```
scores = model.evaluate(x_test, y_test, verbose=0)
```

در این قسمت تابع دو لیست که لیست اول شامل 12 آیتم پیش‌بیی کننده‌ی آب‌وهوا ا ست و لی ست دوم حاوی نتایج ما یعنی و ضعیت آب‌وهوا ست که تابع evaluate مقایسه می‌کند که این پیش‌بینی ها تا چه حد درست بوده‌اند.

Compare

```
y_pred = model.predict(x_test)
pred = list()
for i in range (len(y_pred)):
    pred.append(np.argmax(y_pred[i]))
    test = list()
for i in range (len(y_test)):
    test.append(np.argmax(y_test[i]))
```

در این قسمت قبل از انجام تست و train کردن برروی داده‌ها یک تست پیش‌بینی‌کننده انجام می‌دهیم تا حدس بزنیم نتیجه تست ما تا چه حد قابل قبول خواهد بود.

MatPlotLib

با اســـتفاده از این کتاب‌خانه تســـت و Train خود را در قالب نمودار نمایش می‌دهیم.

```
import matplotlib.pyplot as plt
plt.plot(history.history['acc'])
plt.plot(history.history['val_acc'])
plt.title('Model accuracy')
plt.ylabel('Accuracy')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model loss')
plt.ylabel('Loss')
plt.xlabel('Epoch')
plt.legend(['Train', 'Test'], loc='upper left')
plt.show()
```

JSon

مدل را در قا لب json ذخیره می‌کنیم و وزن نودهای داخل شبکه عصبی را ذخیره می‌کنیم.

```
model_json = model.to_json()
open('ann_model.json', 'w').write(model_json)
model.save_weights('weights.h5',overwrite=True)
```

Model Structure

در این قســمت با اســتفاده از تابع‌های تعریف شـــده در keras ارزش هر نود در شـــبکه عصبی را به‌دست می‌آوریم.

برای لایه اولیه 12 ورودی در نظر می‌گیریم کـه ا ین 12 ورودی بـه 16 ورودی برای نودهای داخلی (لایه میانی) تبدیل می‌شـوند و هم چنیین 4 نود در لایه‌ی آخر وجود دارد.

```
model = Sequential()
model.add(Dense(16,input_dim=input,activation='relu'))
model.add(Dense(12,activation='relu'))
model.add(Dense(classes,activation='softmax'))
```