Universidade do Minho
Escola de Engenharia/Departamento de Informática

Visão por computador

Uma abordagem Deep Learning

Aulas práticas

IMAGEM MÉDICA

AMBIENTE DE TRABALHO

Para executarem os exercicios que irão ser propostos devem preparar o ambiente de trabalho com o seguinte sw:

python 3.x (versão 64bits)
+numpy + scipy + scikit-learn + scikit-image + h5py + matplotlib

se tiverem o anaconda instalado basta executar: conda install ....
opcional: instalar o opencv
instalar o keras: pip install keras
instalar o theano: pip install theano
instalar o tensorflow: pip install tensorflow

Para os alunos que prefiram ter uma maquina virtual com tudo já instalado coloquei uma maquina virtual (Ubuntu 16.10+opencv+keras + ...) num servidor da DI em:
https://reposlink.di.uminho.pt/uploads/d98e0b93a1b61d7dc996fe03052468fe.file.ubuntu16.10DLalunos.zip
Essa maquina tem o anaconda instalado e está configurada com ambientes.
Para trabalharem no ambiente correcto devem executar: source activate keras-test
Devem fazer o upgrade para o keras2:
pip install git+git://github.com/fchollet/keras.git --upgrade

podem fazer as instalações e upgrades tanto utilizando o conda como o pip mas sempre dentro do ambiente keras-test

REDE LSTM UTILIZANDO KERAS

Vamos desenvolver uma rede recorrente **LSTM – Long Short-Term Memory** para fazer a previsão de uma série temporal utilizando o registo dos valores das acções da Google disponível em .

http://chart.finance.yahoo.com/table.csv?s= GOOGL &a=11&b=15&c=2011&d=29&e=10&f=2016&g=d&ignore=.csv

AULA5 – LSTM

Bibliotecas necessárias para a execução desta aula.

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import math, time
import datetime
from keras.models import Sequential
from keras.layers.core import Dense, Dropout, Activation
from keras.layers.recurrent import LSTM

# fixar random seed para se puder reproduzir os resultados
seed = 9
np.random.seed(seed)
```

AULA5 – LSTM

```
# Etapa 1 - preparar o dataset
'''
fazer o download da sequencias do valor das ações da google GOOGL stock data (fonte yahoo.com)
dataset:
http://chart.finance.yahoo.com/table.csv?s=GOOGL&a=11&b=15&c=2011&d=29&e=10&f=2016&g=d&ignore=.csv
A função get_stock_data é generica para ir buscar dados à yahoo.com
trata-se uma tabela com: ['Date','Open','High','Low','Close','Volume','Adj Close']
Vamos só utilizar os campos ['Open','High','Close']
'''
def get_stock_data(stock_name, normalized=0,file_name=None):
    if not file_name:
        file_name =
'http://chart.finance.yahoo.com/table.csv?s=%s&a=11&b=15&c=2011&d=29&e=10&f=2016&g=d&ignore=.csv' %
stock_name
    col_names = ['Date','Open','High','Low','Close','Volume','Adj Close']
    stocks = pd.read_csv(file_name, header=0, names=col_names) #fica numa especie de tabela
exactamente como estava no csv (1350 linhas,7 colunas)
    df = pd.DataFrame(stocks) #neste caso não vai fazer nada
    date_split = df['Date'].str.split('-').str #não vai servir para nada
    df['Year'], df['Month'], df['Day'] = date_split  #não vai servir para nada
    df["Volume"] = df["Volume"] / 10000  #não vai servir para nada
    df.drop(df.columns[[0,3,5,6, 7,8,9]], axis=1, inplace=True) #vou só ficar com as colunas 1,2,4
    return df
```

AULA5 – LSTM

```
def load_GOOGL_stock_dataset():
    stock_name = 'GOOGL'
    return get_stock_data(stock_name,0, 'table.csv')

def pre_processar_GOOGL_stock_dataset(df):
    df['High'] = df['High'] / 100
    df['Open'] = df['Open'] / 100
    df['Close'] = df['Close'] / 100
    return df

# Visualizar os top registos da tabela
def visualize_GOOGL():
    df = load_GOOGL_stock_dataset()
    print('### Antes do pré-processamento ###')
    print(df.head()) #mostra só os primeiros 5 registos
    df = pre_processar_GOOGL_stock_dataset(df)
    print('### Após o pré-processamento ###')
    print(df.head())
```

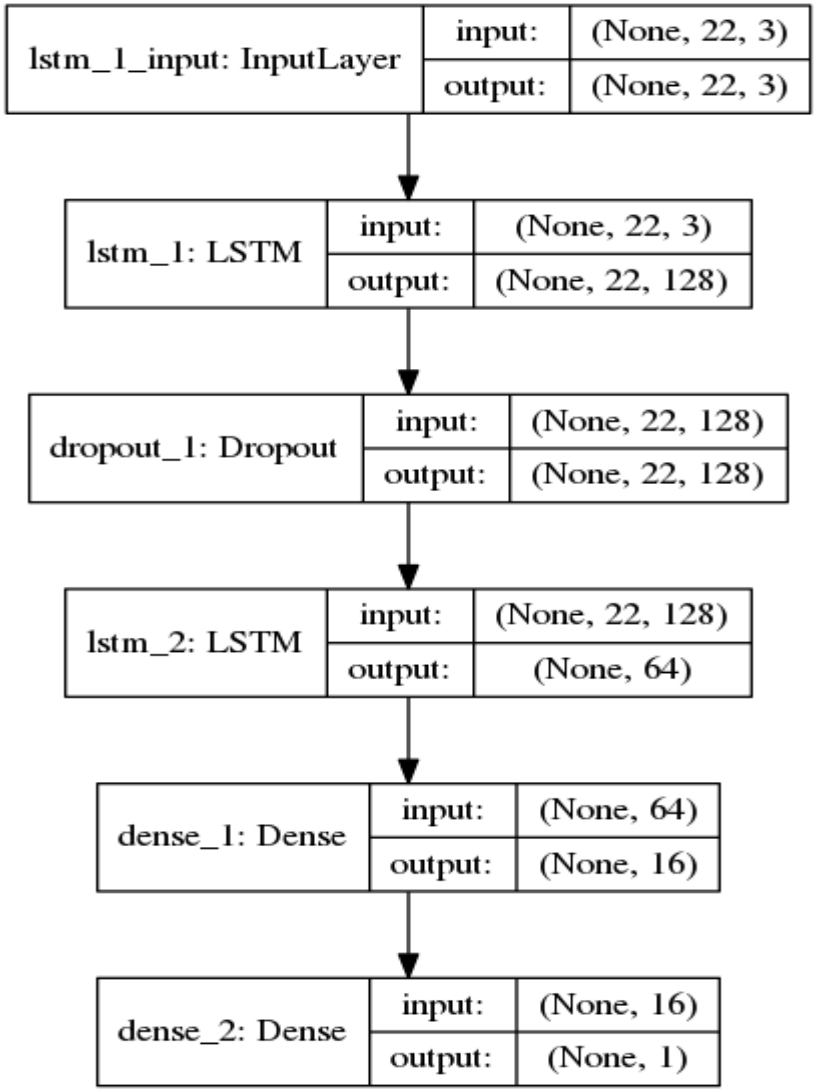
AULA5 – LSTM

```
#função load_data do lstm.py configurada para aceitar qualquer número de parametros
#o último atributo é que fica como label (resultado)
#stock é um dataframe do pandas (uma especie de dicionario + matriz)
#seq_len é o tamanho da janela a ser utilizada na serie temporal
def load_data(df_dados, janela):
    qt_atributos = len(df_dados.columns)
    mat_dados = df_dados.as_matrix() #converter dataframe para matriz (lista com lista de cada registo)
    tam_sequencia = janela + 1
    res = []
    for i in range(len(mat_dados) - tam_sequencia): #numero de registos - tamanho da sequencia
        res.append(mat_dados[i: i + tam_sequencia])
    res = np.array(res) #dá como resultado um np com uma lista de matrizes (janela deslizante ao longo da serie)
    qt_casos_treino = int(round(0.9 * res.shape[0])) #90% passam a ser casos de treino
    train = res[:qt_casos_treino, :]
    x_train = train[:, :-1] #menos um registo pois o ultimo registo é o registo a seguir à janela
    y_train = train[:, -1][:,-1] #para ir buscar o último atributo para a lista dos labels
    x_test = res[qt_casos_treino:, :-1]
    y_test = res[qt_casos_treino:, -1][:,-1]
    x_train = np.reshape(x_train, (x_train.shape[0], x_train.shape[1], qt_atributos))
    x_test = np.reshape(x_test, (x_test.shape[0], x_test.shape[1], qt_atributos))
    return [x_train, y_train, x_test, y_test]
```

AULA5 – LSTM

```
# Etapa 2 - Definir a topologia da rede (arquitectura do modelo) e compilar '''
def build_model2(janela):
        model = Sequential()
        model.add(LSTM(128, input_shape=(janela, 3), return_sequences=True))
        model.add(Dropout(0.2))
        model.add(LSTM(64, input_shape=(janela, 3), return_sequences=False))
        #model.add(Dropout(0.2))
        model.add(Dense(16, activation="relu", kernel_initializer="uniform"))
        model.add(Dense(1, activation="linear", kernel_initializer="uniform"))
        model.compile(loss='mse',optimizer='adam',metrics=['accuracy'])
        return model
```

AULA5 – LSTM


lstm_1_input: InputLayer
input: (None, 22, 3)
output: (None, 22, 3)
lstm_1: LSTM
input: (None, 22, 3)
output: (None, 22, 128)
dropout_1: Dropout
input: (None, 22, 128)
output: (None, 22, 128)
lstm_2: LSTM
input: (None, 22, 128)
output: (None, 64)
dense_1: Dense
input: (None, 64)
output: (None, 16)
dense_2: Dense
input: (None, 16)
output: (None, 1)

AULA5 – LSTM

```
#imprime um grafico com os valores de teste e com as correspondentes tabela de previsões
def print_series_prediction(y_test,predic):
    diff=[]
    racio=[]
    for i in range(len(y_test)): #para imprimir tabela de previsoes
        racio.append( (y_test[i]/predic[i])-1)
        diff.append( abs(y_test[i]- predic[i]))
        print('valor: %f  ---> Previsão: %f   Diff: %f  Racio: %f' % (y_test[i],predic[i], diff[i],
racio[i]))
    plt.plot(y_test,color='blue', label='y_test')
    plt.plot(predic,color='red', label='prediction') #este deu uma linha em branco
    plt.plot(diff,color='green', label='diff')
    plt.plot(racio,color='yellow', label='racio')
    plt.legend(loc='upper left')
    plt.show()
```

AULA5 – LSTM

```
def LSTM_utilizando_GOOGL_data():
    df = load_GOOGL_stock_dataset()
    df = pre_processar_GOOGL_stock_dataset(df)
    print("df", df.shape)
    janela = 22 #tamanho da Janela deslizante
    X_train, y_train, X_test, y_test = load_data(df[::-1], janela)# o df[::-1] é o df por ordem
inversa
    print("X_train", X_train.shape)
    print("y_train", y_train.shape)
    print("X_test", X_test.shape)
    print("y_test", y_test.shape)
    #model = build_model(janela)
    model = build_model2(janela)
    #model.fit(X_train, y_train, batch_size=512, epochs=500, validation_split=0.1, verbose=1)
    model.fit(X_train, y_train, batch_size=512, epochs=500, validation_split=0.1, verbose=1)
    print_model(model,"lstm_model.png")
    trainScore = model.evaluate(X_train, y_train, verbose=0)
    print('Train Score: %.2f MSE (%.2f RMSE)' % (trainScore[0], math.sqrt(trainScore[0])))
    testScore = model.evaluate(X_test, y_test, verbose=0)
    print('Test Score: %.2f MSE (%.2f RMSE)' % (testScore[0], math.sqrt(testScore[0])))
    print(model.metrics_names)
    p = model.predict(X_test)
    predic = np.squeeze(np.asarray(p)) #para transformar uma matriz de uma coluna e n linhas em
um np array de n elementos
    print_series_prediction(y_test,predic)

  '''‘ MSE- (Mean square error), RMSE- (root mean square error) –
  o significado de RMSE depende do range da label. para o mesmo range menor é melhor.
  '''
```

AULA5 – LSTM

```
if __name__ == '__main__':
    #visualize_GOOGL()
    LSTM_utilizando_GOOGL_data()
```