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 MLP-IMAGEM UTILIZANDO KERAS

Vamos desenvolver uma rede **MLP – MultiLayer Perceptron** para fazer classificação de imagens utilizando o dataset mnist:

https://s3.amazonaws.com/img-datasets/mnist.npz

AULA2 – MLP

Bibliotecas necessárias para a execução desta aula.

```
import numpy as np
#from keras.datasets import mnist
from keras.models import Sequential
from keras.layers import Dense
from keras.utils import np_utils
import matplotlib.pyplot as plt

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

AULA2 – MLP

```
# Etapa 1 - preparar o dataset
'''
fazer o download do MNIST dataset com imagens de digitos escritos à mão para fazer a
sua classificação (já pré-preparados)
dataset: https://s3.amazonaws.com/img-datasets/mnist.npz
O ficheiro já tem tudo separado nos ficheiros {x_test.npy, x_train.npy, y_test.npy,
y_train.npy}
Os atributos de entrada estão com matrizes 3D(imagem, largura,altura) e os atributos de
saída é uma lista com o número correspondente
'''
def load_mnist_dataset(path='mnist.npz'):
    #path = get_file(path, origin='https://s3.amazonaws.com/img-datasets/mnist.npz')
    f = np.load(path)
    x_train = f['x_train']
    y_train = f['y_train']
    x_test = f['x_test']
    y_test = f['y_test']
    f.close()
    return (x_train, y_train), (x_test, y_test)
```

AULA2 – MLP

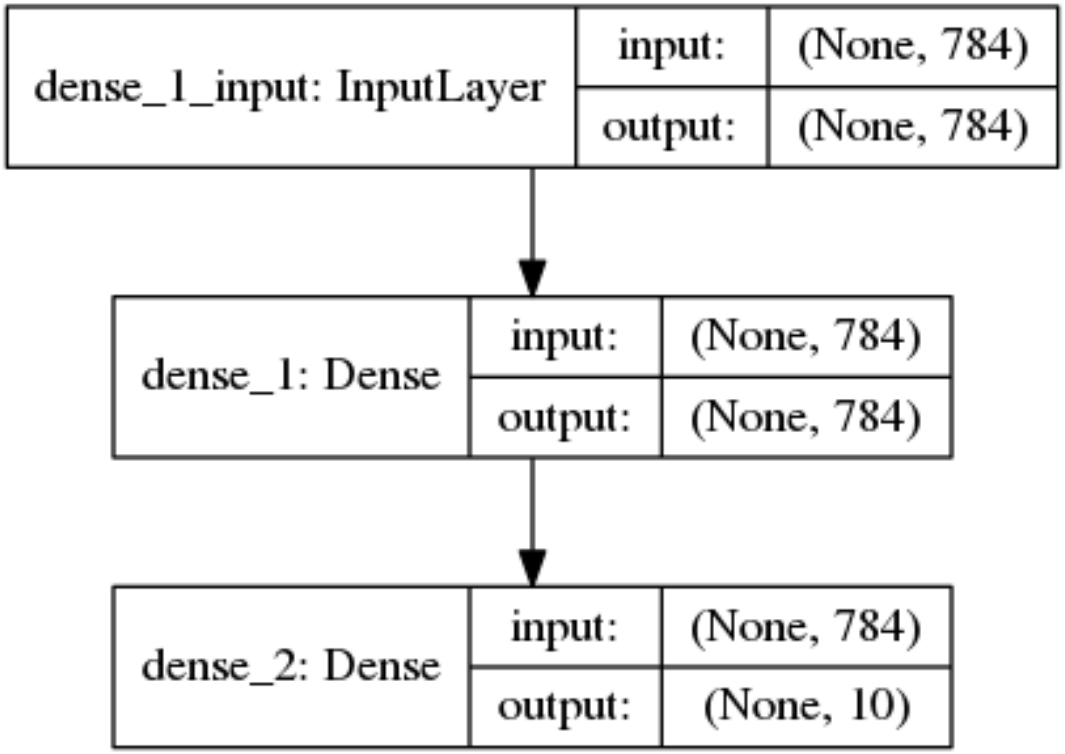
```
# Visualizar 6 imagens do mnist numa escala de cinzentos
def visualize_mnist():
    (X_train, y_train), (X_test, y_test) = load_mnist_dataset('mnist.npz')
    plt.subplot(321)
    plt.imshow(X_train[0], cmap=plt.get_cmap('gray'))
    plt.subplot(322)
    plt.imshow(X_train[1], cmap=plt.get_cmap('gray'))
    plt.subplot(323)
    plt.imshow(X_train[2], cmap=plt.get_cmap('gray'))
    plt.subplot(324)
    plt.imshow(X_train[3], cmap=plt.get_cmap('gray'))
    plt.subplot(325)
    plt.imshow(X_train[4], cmap=plt.get_cmap('gray'))
    plt.subplot(326)
    plt.imshow(X_train[5], cmap=plt.get_cmap('gray'))
    plt.show()
```

AULA2 – MLP

```
# Etapa 2 - Definir a topologia da rede (arquitectura do modelo) e compilar (multilayer perceptrons)
'''
cria-se um modelo sequencial e vai-se acrescentando camadas (layers)
vamos criar uma rede simples com uma camada escondida
Dense class significa que teremos um modelo fully connected
    o primeiro parametro estabelece o número de neuronios na camada (num pixeis na primeira)
    input_dim=num_pixeis indica o número de entradas do nosso dataset (num pixeis atributos neste caso)
    kernel_initializer indica o metodo de inicialização dos pesos das ligações
    'nomal' sigifica com small number generator from Gaussion distribution
    "activation" indica a activation fuction
    'relu' rectifier linear unit activation function com range entre 0 e infinito
    'softmax' foi utilizada para garantir uma percentagem (valor entre 0 e 1) a totalizar entre todas as saidas o valor de 1
Compile - loss - funcão a ser utilizada no calculo da diferença entre o pretendido e o obtido
vamos utilizar logaritmic loss para classificação binária: 'categorical_crossentropy'
o algoritmo de gradient descent será o "adam" pois é eficiente
a métrica a ser utilizada no report durante o treino será 'accuracy' pois trata-se de um problema de classificacao
'''
def create_compile_model_mlp(num_pixels, num_classes):
    model = Sequential()
    model.add(Dense(num_pixels, input_dim=num_pixels, kernel_initializer='normal', activation='relu'))
    model.add(Dense(num_classes, kernel_initializer='normal', activation='softmax'))
    model.compile(loss='categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
    return model
```

AULA2 – MLP

```
#util para visualizar a topologia da rede num ficheiro em pdf ou png
def print_model(model,fich):
    from keras.utils import plot_model
    plot_model(model, to_file=fich, show_shapes=True, show_layer_names=True)
```



AULA2 – MLP

```
#utils para visulaização do historial de aprendizagem
def print_history_accuracy(history):
    print(history.history.keys())
    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()

def print_history_loss(history):
    print(history.history.keys())
    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()
```

AULA2 – MLP

```
def mnist_utilizando_mlp():
    (X_train, y_train), (X_test, y_test) = load_mnist_dataset('mnist.npz')
    # transformar a matriz 28*28 das imagens num vector com 784 atributos para cada imagem (porque é
multilayer-perceptron)
    num_pixels = X_train.shape[1] * X_train.shape[2]
    X_train = X_train.reshape(X_train.shape[0], num_pixels).astype('float32')
    X_test = X_test.reshape(X_test.shape[0], num_pixels).astype('float32')
    # normalizar os valores dos pixeis de 0-255 para 0-1
    X_train = X_train / 255
    X_test = X_test / 255
    # transformar o label que é um inteiro em categorias binárias, o valor passa a ser o
correspondente à posição
    # o 5 passa a ser a lista [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
    y_train = np_utils.to_categorical(y_train)
    y_test = np_utils.to_categorical(y_test)
    num_classes = y_test.shape[1]
    # definir a topologia da rede e compilar
    model = create_compile_model_mlp(num_pixels, num_classes)
    print_model(model,"model.png")
    # treinar a rede
    history=model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=10, batch_size=200,
verbose=2)
    print_history_accuracy(history)
    #print_history_loss(history)
    # Avaliação final com os casos de teste
    scores = model.evaluate(X_test, y_test, verbose=0)
    print('Scores: ', scores)
    print("Erro modelo MLP: %.2f%%" % (100-scores[1]*100))
```

AULA2 – MLP

```
if __name__ == '__main__':
    #visualize_mnist()
    mnist_utilizando_mlp()
```