University of Echahid Hama Lakhdar, El-oued
institute of exact sciences
Department of computing
$1^{er} Master$: Artificial Intelligence and Data science
Semester : 1.2024
Mohammed Anouar Naoui

— Deep Learning : First Neural Network With Keras—

1 Keras and tensorflow

Keras is an open-source deep learning framework written in Python. It is designed to be user-friendly, modular, and extensible, allowing both beginners and experts to build and experiment with deep learning models efficiently. Keras was developed with a focus on enabling fast experimentation with deep neural networks. It provides a high-level interface that abstracts away much of the complexity involved in building and training neural networks, making it accessible to a wide range of users. TensorFlow is an open-source machine learning framework developed by Google Brain and released in 2015. It is designed to provide a flexible and efficient platform for building and deploying machine learning models, particularly deep neural networks.

Keras has been the official high-level API for building and training neural networks within TensorFlow.

2 Data :Pima Indians Onset of Diabetes Dataset

The Pima Indians Diabetes dataset is a commonly used dataset in machine learning and statistics for predictive modeling. It originated from a study conducted by the National Institute of Diabetes and Digestive and Kidney Diseases. The dataset contains information about Pima Indian women over the age of 21 from the Gila River Indian Community near Phoenix, Arizona.

1. Number of times pregnant.
2. Plasma glucose concentration a 2 hours in an oral glucose tolerance test.
3. Diastolic blood pressure (mm Hg).
4. Triceps skin fold thickness (mm).
5. 2-Hour serum insulin (mu U/ml).
6. Body mass index.
7. Diabetes pedigree function.
8. Age (years).
9. Class, onset of diabetes within five years.

Target Variable :

- The target variable indicates whether the individual developed diabetes within five years of the observation period.
- It is a binary variable, with 1 representing the onset of diabetes and 0 representing no onset.

Objectif :

- The primary objective of using this dataset is to develop predictive models that can accurately classify individuals as either diabetic or non-diabetic based on the given features.

3 Load data

```
from keras.models import Sequential
from keras.layers import Dense
import numpy
# fix random seed for reproducibility
seed = 7
numpy.random.seed(seed)
# load pima indians dataset

url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/pima-indians-diabetes.data.csv"
names = ['preg', 'plas', 'pres', 'skin', 'insu', 'mass', 'pedi', 'age', 'class']
dataset = pd.read_csv(url, names=names)
# split into input (X) and output (Y) variables
X = iloc.dataset[:,0:8]
Y = iloc.dataset[:,8]
```

The dataset comprises eight input variables and one output variable, which is situated in the final column. After loading, we divided the dataset into the input variables (X) and the output class variable (Y).

4 Define the model

In Keras, the Sequential model is a linear stack of layers. It's one of the simplest types of neural network architectures. It allows you to create models layer-by-layer in a sequential manner, where each layer has weights that correspond to the inputs from the previous layer and outputs to the next layer. The Sequential model is appropriate for most simple architectures where the data flows sequentially from one layer to the next. This makes it easy to understand and implement various neural network architectures, especially when you're getting started with deep learning.

In Keras, a dense layer is a type of neural network layer that's fully connected, meaning each neuron in the dense layer receives input from every neuron in the previous layer. Dense layers are also known as fully connected layers. They are widely used in deep learning models for tasks such as classification and regression

```
# create model
model = Sequential()
model.add(Dense(12, input_dim=8, kernel_initializer= 'uniform' , activation= 'relu' ))
model.add(Dense(8, kernel_initializer= 'uniform' , activation= 'relu' ))
model.add(Dense(1, kernel_initializer= 'uniform' , activation= 'sigmoid' ))
```

Draw the architecture of neural network that illustrate

- input layer include initalization of weights and bias.
- Hiden layer with activation function
- Output layer.

5 Compile Model

- Specify the loss function to use to evaluate a set of weights.In this case we will use logarithmic loss, which for a binary classification problem is defined in Keras as binary crossentropy
- The optimizer used to search through dierent weights for the network and any optional metrics we would like to collect and report during training. . We will also use the efficient gradient descent algorithm adam for no other reason that it is an efficient default.
- it is a classification problem, we will collect and report the classification accuracy as the metric.

```
model.compile(loss= 'binary_crossentropy' , optimizer= 'adam' , metrics=['accuracy'])
```

6 Fit Model

We can proceed to train or fit our model with the loaded data by invoking the fit() function on the model. During the training process, the model iterates through the dataset for a fixed number of epochs, a value we specify using the epochs argument. Additionally, we can determine the number of instances evaluated before updating the weights in the network, referred to as the batch size, using the batch_size argument. In this scenario, we will train for a limited number of epochs (150) and utilize a relatively small batch size of 10. These parameters may be fine-tuned through experimental testing and refinement.

```
history=model.fit(X, Y, epochs=150, batch_size=10)
```

7 Evaluate Model

This action results in predictions for each input-output pair, gathering metrics like the mean loss and any predefined metrics, such as accuracy.

```
    # evaluate the model
scores = model.evaluate(X, Y)
print("%s: %.2f%%" % (model.metrics_names[1], scores[1]*100))
```

PLot accuracy and loss/epochs diagram.

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.plot(history.history['loss'], label='Training Loss')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

8 Question

Perform the result of accuracy by the use of :

— Train/Test split method (Plot accuracy and loss/epochs diagram of training and validation data).

— K-Fold cross validation.(Plot accuracy and loos/epochs diagram of training and validation data).