Build a Network in Keras

StartOnAI Deep Learning Tutorial 2:

This tutorial explains Keras and shows the process of designing a basic network

1. What is Keras?

Keras is an Open Source Neural Network library written in Python that runs on top of TensorFlow (backend). One big reason for using keras is because this API (Application Programming Interface) is very user-friendly. The Keras Python Library provides a clean and convenient way to create a range of deep learning models and networks. Four guiding principles have been part of the development of Keras:

- Modularity
- Minimalism
- Extensibility
- Python

In simple terms, Keras is a minimalist python library for running deep learning.

stackshare.io

2. Designing a Neural Network

In this tutorial the main objective will be building a neural network using Keras and Python. This procedure is not very code intensive and is relatively easy to follow along.

What Will You Need:

1. Install either python 2 or 3 and have it configured
2. Have SciPy and NumPy installed/configured
3. Have Keras and a backend (Theanos or Tensorflow) installed and configured

Check out the following link for additional help with Keras/Python set-up: Set-Up

Objective:

Developing a Neural Network.

3. Code Walkthrough

1. Create a file: name it **keras_first_network.py**
2. Define the functions and classes needed for this demonstration
3. The imports required are the following:

```
#first neural network with keras tutorial
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense

```

4. This demonstration is going to use the Pima Indians onset of diabetes dataset. This dataset will describe the patient medical record for Pima Indians and display if they had an onset of diabetes within 5 years. This is a binary

classification problem, allowing form numerical input and numerical output values

5. The datasets can be obtained from: (download these - save as *pima-indians-diabetes.csv)*
    a. Dataset CSV File (pima-indians-diabetes.csv)
    b. Dataset Details
6. There should be 8 input variables (X) as well as 1 output variable (y) → y=f(X)
7. This data will then be stored in a 2D array

```
...
# load the dataset
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
# split into input (X) and output (y) variables
X = dataset[:,0:8]
y = dataset[:,8]
...
```

8. The 8 input variables are:
    a. Number of times pregnant
    b. Plasma glucose concentration a 2 hours in an oral glucose tolerance test
    c. Diastolic blood pressure (mm Hg)
    d. Triceps skin fold thickness (mm)

   e. 2-Hour serum insulin (mu U/ml)

   f. Body mass index (weight in kg/(height in m)^2)

   g. Diabetes pedigree function

   h. Age (years)

9. Output Variable - 0 or 1

10. Define the Keras Model

   a. Models are defined as a sequence of layers - fully connected layers are defined using the dense class

   b. Input layers must have the appropriate number of input features = 8 (since there are 8 variables)

   c. The ReLU will be used for the two hidden layers and the sigmoid function will be used for the output layer.

   d. The model must have rows of data with 8 variables (input), then the first hidden layer has 12 nodes, the second hidden layers has 8 nodes

   e. The output layer has 1 node

   f. The following code establishes this:

```
...
# define the keras model
model = Sequential()
```

```
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
...
```

11. Once the model is defined, we must compile it

    a. Numerical libraries such as Tensorflow or Theano are used. These backends automatically choose the best way to represent the network for training and making predictions

    b. *Training a networks means finding the correct number of weights to map inputs to outputs

    c. We have to make sure to define a loss function to use to evaluate a set of weights: cross entropy will be used as the loss function (”binary_crossenripy”)

    d. “Optimizer” will be defined as the effective gradient descent algorithm (adam)

    e. The classification accuracy will be collected and reported using the metrics argument

    f. Take a look at the following code:

```
...
```

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

12. The goal now is execute this model using some data
    a. The fit() function will be used to train the data
    b. Training occurs over epochs (which are split into batches)
    c. The training will then run for a fixed number of iterations
    d. The model needs to be trained good enough so that it knows a good amount of mapping of inputs to outputs
    e. Note: There will be error initially, but after sometime the error will level out (*model convergence*)
    f. The number of epochs/batches are chosen on an experimental basis (we wil be using a relatively small amount)
    g. Example Code:

```
...
# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
...
```

13. Evaluating the Keras Model

   a. The “evaluate()” function is used to test the performance of our model
   b. Specifically, it will show the “loss of the model” for the dataset as well as any other metrics configured such as “the accuracy of the model” for the dataset
   c. Here is a sampling of the code required:

```
...
# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

14. Congratulations!!! You have just made your neural network with a few lines of Python code.
   a. Here is all the code used so far:

```
# first neural network with keras tutorial
from numpy import loadtxt
from keras.models import Sequential
from keras.layers import Dense
# load the dataset
dataset = loadtxt('pima-indians-diabetes.csv', delimiter=',')
# split into input (X) and output (y) variables
```

```
X = dataset[:,0:8]
y = dataset[:,8]
# define the keras model
model = Sequential()
model.add(Dense(12, input_dim=8, activation='relu'))
model.add(Dense(8, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
# compile the keras model
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# fit the keras model on the dataset
model.fit(X, y, epochs=150, batch_size=10)
# evaluate the keras model
_, accuracy = model.evaluate(X, y)
print('Accuracy: %.2f' % (accuracy*100))
```

15. Copy this code into a Python file and label/save it: "keras_first_network.py" in the same directory as the data file
    a. Then you can run this as:

```
python keras_first_network.py
```

    b. There should a be message present for every 150 epochs (prints the loss and accuracy)

c. Note: ideally, we would like the loss to go to 0 and the accuracy to 1, but this is only possible at the high-end level of models.

d. However, the goal should always be to choose a model configuration and training the minimizes the loss and maximizes the accuracy

More Resources:

1. https://towardsdatascience.com/building-our-first-neural-network-in-keras-bdc8abbc17f5
2. https://victorzhou.com/blog/keras-neural-network-tutorial/
3. https://medium.com/@ODSC/building-a-custom-convolutional-neural-network-in-keras-48171163aa7f
4. https://builtin.com/data-science/how-build-neural-network-keras
5. https://www.datasciencecentral.com/profiles/blogs/building-neural-network-in-keras

References:

Brownlee, Jason. "Your First Deep Learning Project in Python with Keras Step-By-Step." *Machine Learning Mastery*, 19 Dec. 2019, machinelearningmastery.com/tutorial-first-neural-network-python-keras/.