Advanced Machine Learning

Bilel GUETARNI, PhD

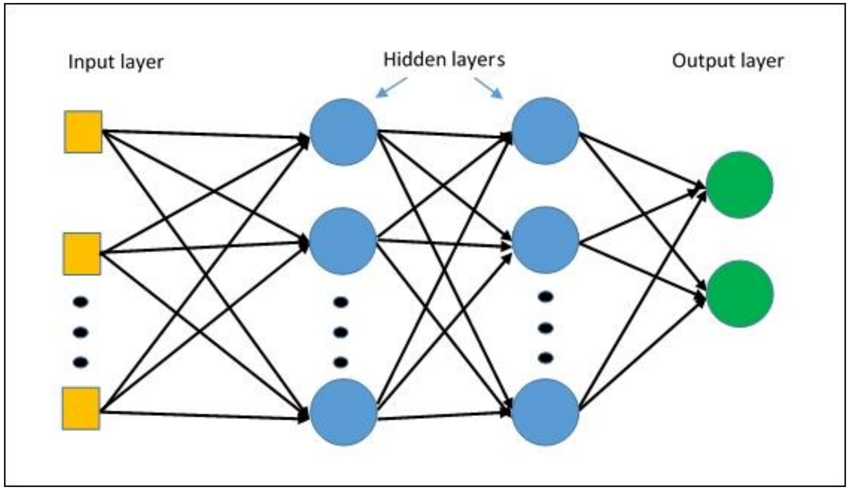
bilel.guetarni@junia.com

JUNIA ISEN

PowerPoint slides from **Halim Benhabiles, PhD** and **Farès BOUGOURZI, PhD**

Properties of the MLP

- Number of layers and neurons affect highly the algorithm
- More layers and neurons improve the capacity of the network but
  - Slow training speed
  - Risk of overfitting particularly with small datasets
- How to deal with these parameters?



MLP example (Multi Layer Perceptron)

Going faster with Keras - https://keras.io/

- Python Deep learning library
  - High-level neural networks API
  - Running on backend platforms

mxnet

TensorFlow

Keras

PyTorch

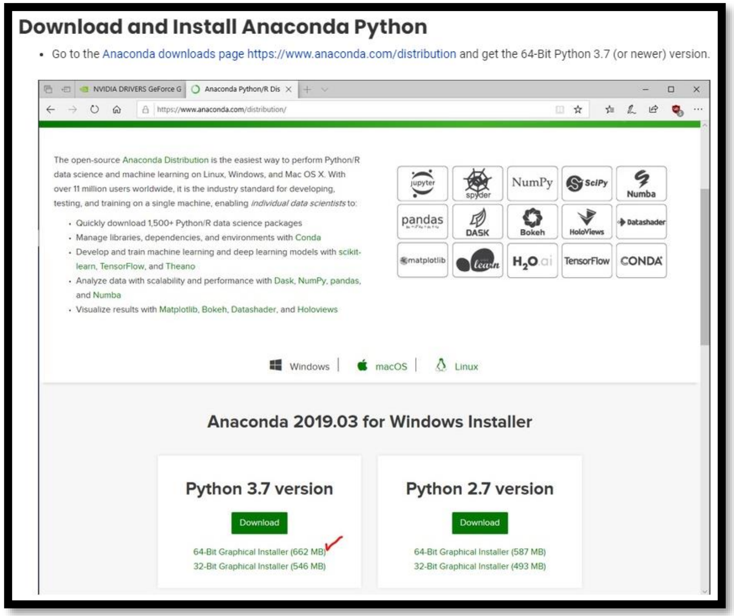
  - Fast prototyping
  - Use **CPU** or **GPU**

Installing keras and TensorFlow

How to Install TensorFlow with GPU Support on Windows 10 (Without Installing CUDA) UPDATED!

Written on April 26, 2019 by Dr Donald Kinghorn

https://www.pugetsystems.com/labs/hpc/How-to-Install-TensorFlow-with-GPU-Support-on-Windows-10-Without-Installing-CUDA-UPDATED-1419/



Download and Install Anaconda Python

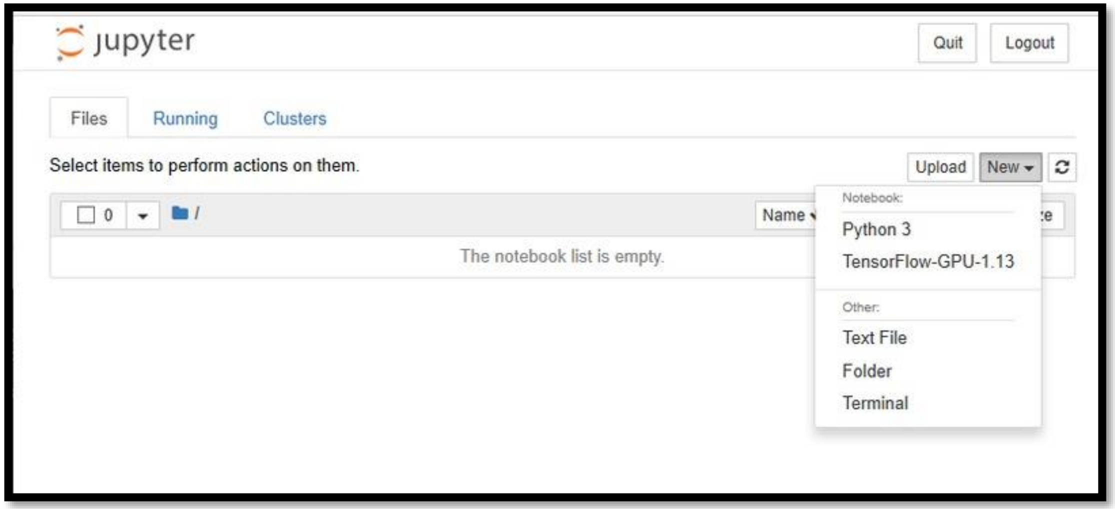
- Go to the Anaconda downloads page https://www.anaconda.com/distribution and get the 64-Bit Python 3.7 (or newer) version.

The open-source Anaconda Distribution is the easiest way to perform Python/R data science and machine learning on Linux, Windows, and Mac OS X. With over 11 million users worldwide, it is the industry standard for developing, testing, and training on a single machine, enabling *individual data scientists* to:

- Quickly download 1,500+ Python/R data science packages
- Manage libraries, dependencies, and environments with Conda
- Develop and train machine learning and deep learning models with scikit-learn, TensorFlow, and Theano
- Analyze data with scalability and performance with Dask, NumPy, pandas, and Numba
- Visualize results with Matplotlib, Bokeh, Datashader, and Holoviews

Anaconda 2019.03 for Windows Installer

Python 3.7 version	Python 2.7 version
Download	Download
64-Bit Graphical Installer (662 MB) ✓	64-Bit Graphical Installer (587 MB)
32-Bit Graphical Installer (546 MB)	32-Bit Graphical Installer (493 MB)



Select items to perform actions on them.

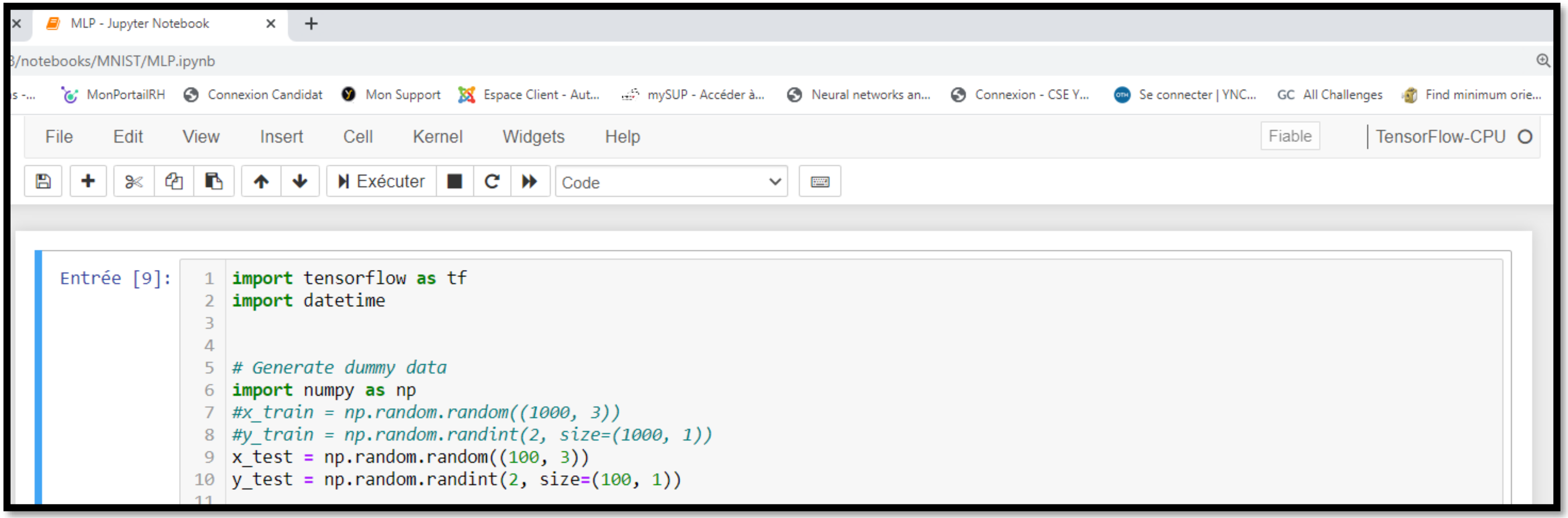
The notebook list is empty.

Installing keras and TensorFlow

How to Install TensorFlow with GPU Support on Windows 10 (Without Installing CUDA) UPDATED!

Written on April 26, 2019 by Dr Donald Kinghorn

https://www.pugetsystems.com/labs/hpc/How-to-Install-TensorFlow-with-GPU-Support-on-Windows-10-Without-Installing-CUDA-UPDATED-1419/



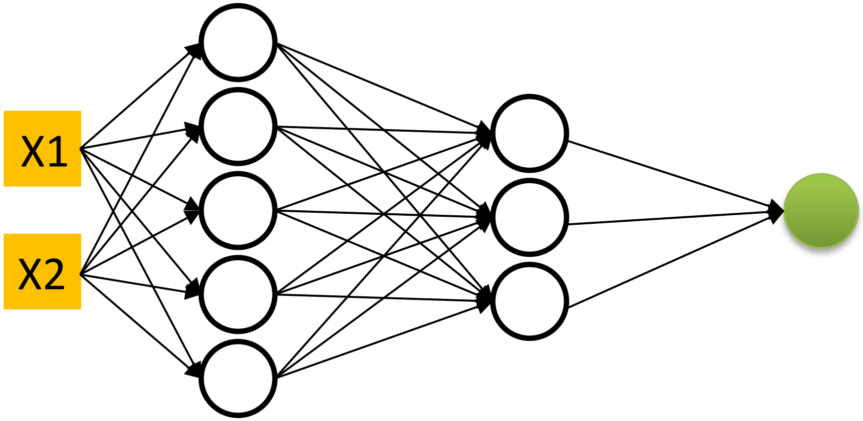
Entrée [9]:

```python
import tensorflow as tf
import datetime


# Generate dummy data
import numpy as np
#x_train = np.random.random((1000, 3))
#y_train = np.random.randint(2, size=(1000, 1))
x_test = np.random.random((100, 3))
y_test = np.random.randint(2, size=(100, 1))
```

Getting started with Keras – TensorFlow core

- Designing the architecture of your MLP model to solve XOR, OR, AND functions

MLP example (input: 2 units, hidden layers: 2 (5 units, 3 units), output: 1 unit)

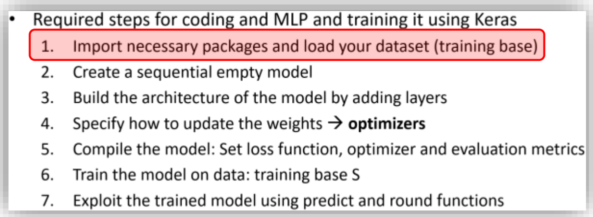


Getting started with Keras – TensorFlow core

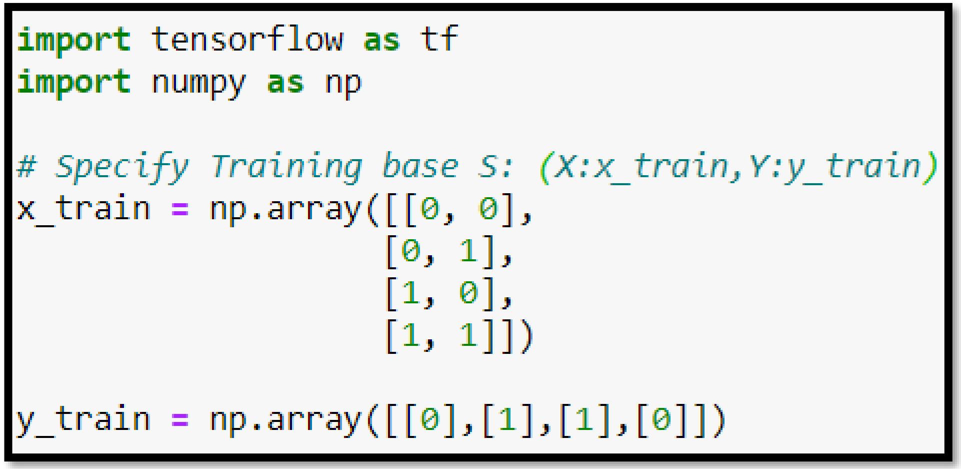
- Required steps for coding and training MLP using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

Getting started with Keras – TensorFlow core

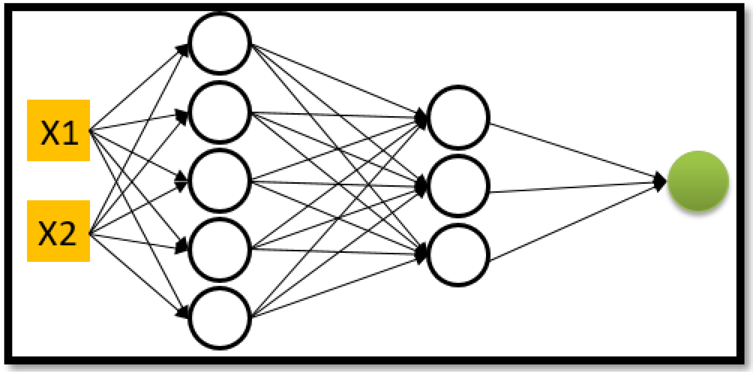
- Import necessary packages and load your dataset



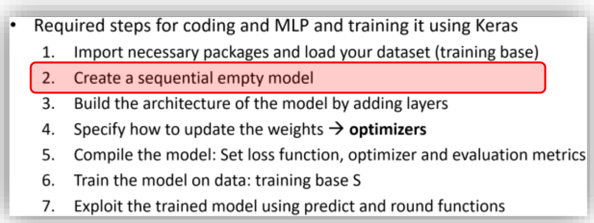
- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions



```
import tensorflow as tf
import numpy as np

# Specify Training base S: (X:x_train,Y:y_train)
x_train = np.array([[0, 0],
                    [0, 1],
                    [1, 0],
                    [1, 1]])

y_train = np.array([[0],[1],[1],[0]])
```

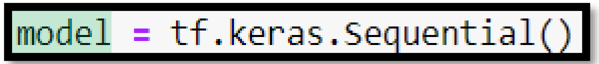


Getting started with Keras – TensorFlow core



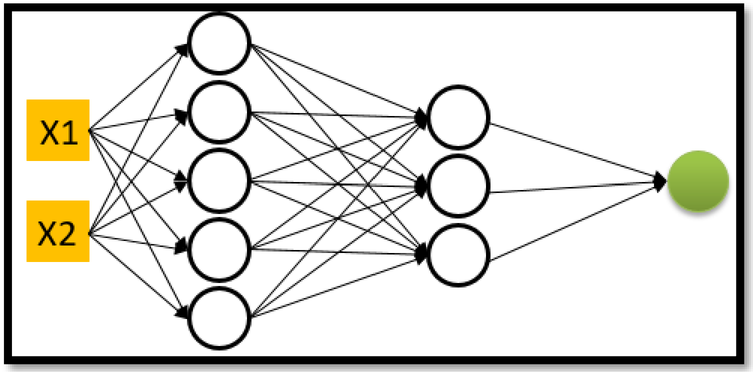
- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

- Create a sequential model (neural network architecture)

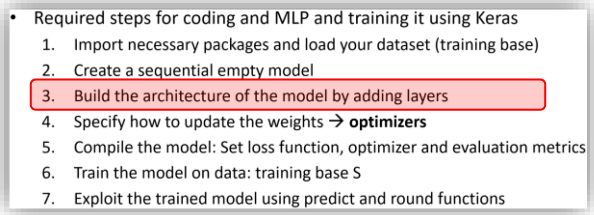


```
model = tf.keras.Sequential()
```

At this stage of the code, the model is empty

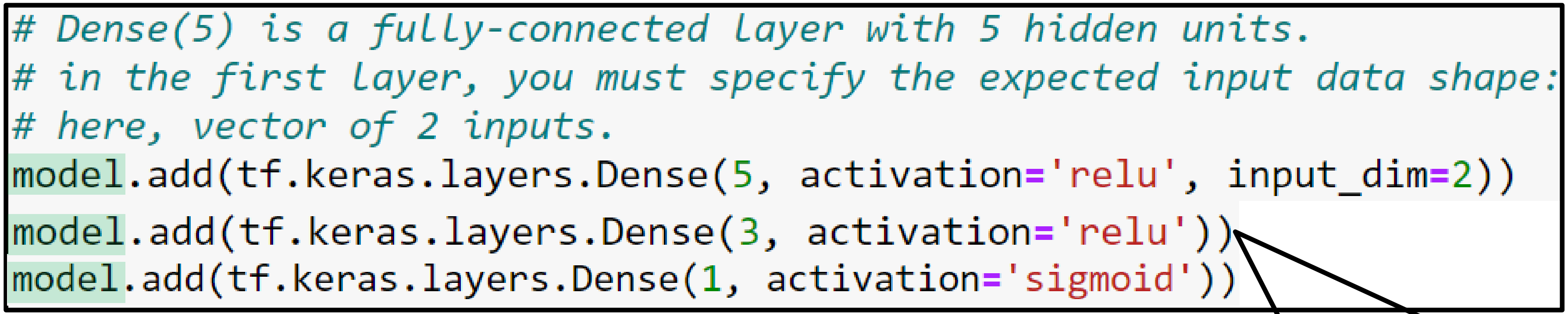


Getting started with Keras – TensorFlow core

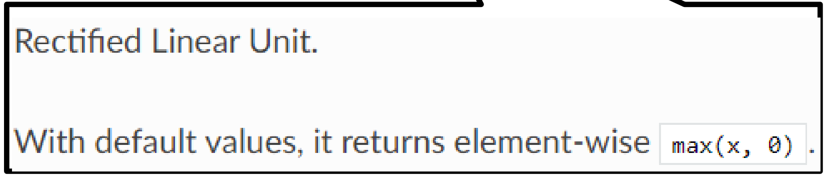


- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

- Building the architecture of our model by adding layers



```
# Dense(5) is a fully-connected layer with 5 hidden units.
# in the first layer, you must specify the expected input data shape:
# here, vector of 2 inputs.
model.add(tf.keras.layers.Dense(5, activation='relu', input_dim=2))
model.add(tf.keras.layers.Dense(3, activation='relu'))
model.add(tf.keras.layers.Dense(1, activation='sigmoid'))
```

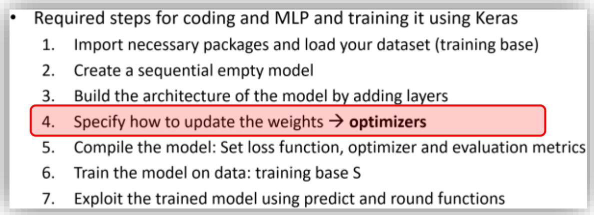


Rectified Linear Unit.

With default values, it returns element-wise `max(x, 0)`.

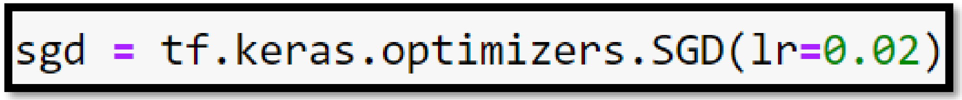
Getting started with Keras – TensorFlow core

- Specifying how to update the weights
  → **optimizers**

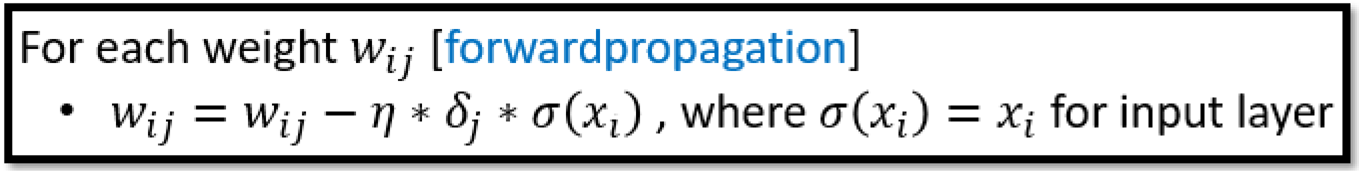


- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

we want to use stochastic gradient descent with learning rate = 0.02



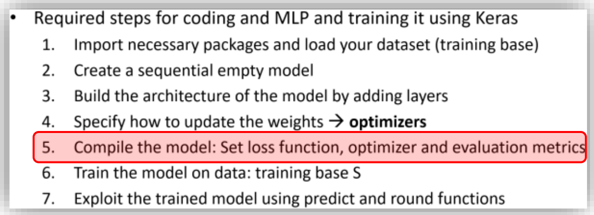
```
sgd = tf.keras.optimizers.SGD(lr=0.02)
```



For each weight w_{ij} [forwardpropagation]

- $w_{ij} = w_{ij} - \eta * \delta_j * \sigma(x_i)$, where $\sigma(x_i) = x_i$ for input layer

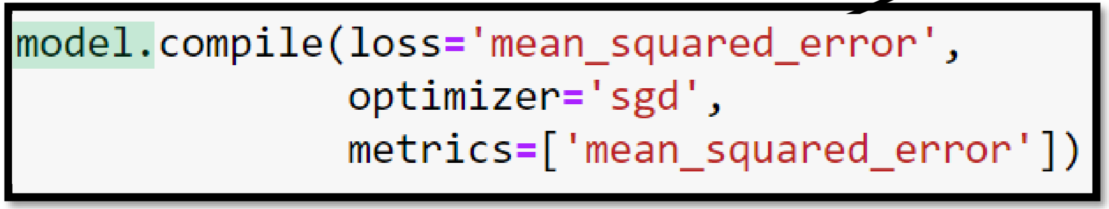
Getting started with Keras – TensorFlow core



- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

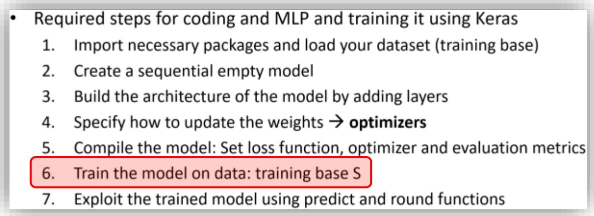
- Compile the model: Set loss function, optimizer and evaluation metrics

$$\text{MSE}(w) = \frac{1}{n}\sum_{(X,c)\ in\ S}(c - o)^2$$



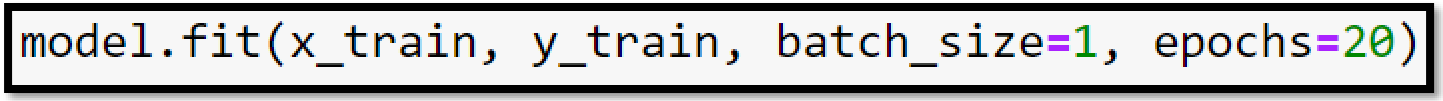
```
model.compile(loss='mean_squared_error',
              optimizer='sgd',
              metrics=['mean_squared_error'])
```

Getting started with Keras – TensorFlow core

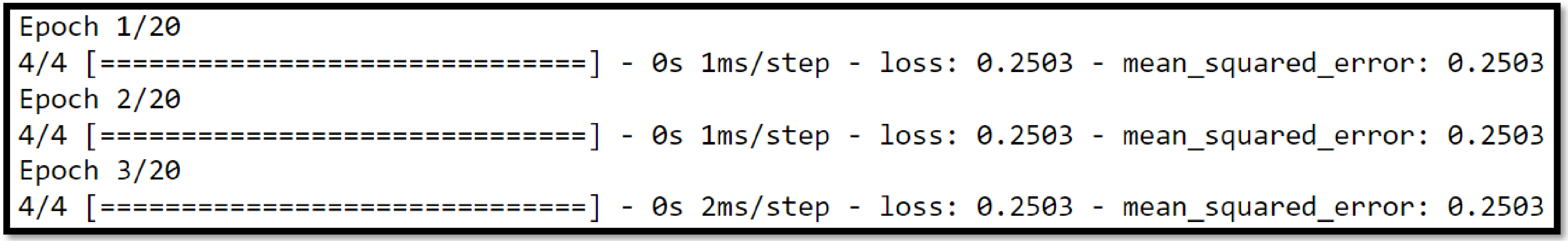


- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

- Train the model on data: training base S

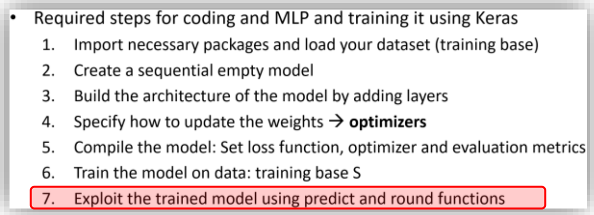


```
model.fit(x_train, y_train, batch_size=1, epochs=20)
```



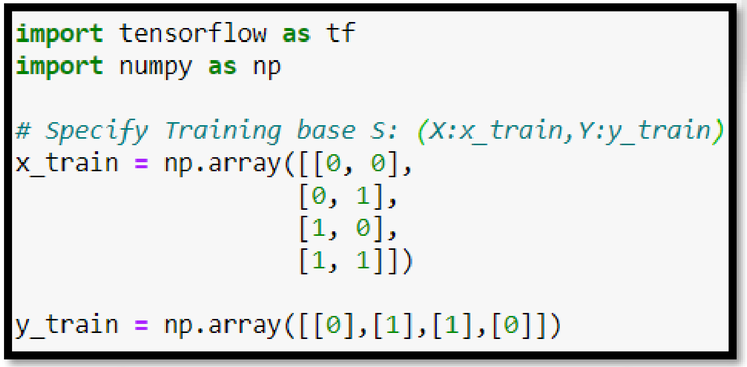
```
Epoch 1/20
4/4 [==============================] - 0s 1ms/step - loss: 0.2503 - mean_squared_error: 0.2503
Epoch 2/20
4/4 [==============================] - 0s 1ms/step - loss: 0.2503 - mean_squared_error: 0.2503
Epoch 3/20
4/4 [==============================] - 0s 2ms/step - loss: 0.2503 - mean_squared_error: 0.2503
```

Getting started with Keras – TensorFlow core

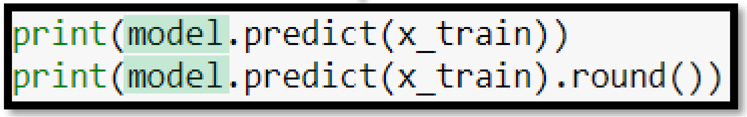


- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

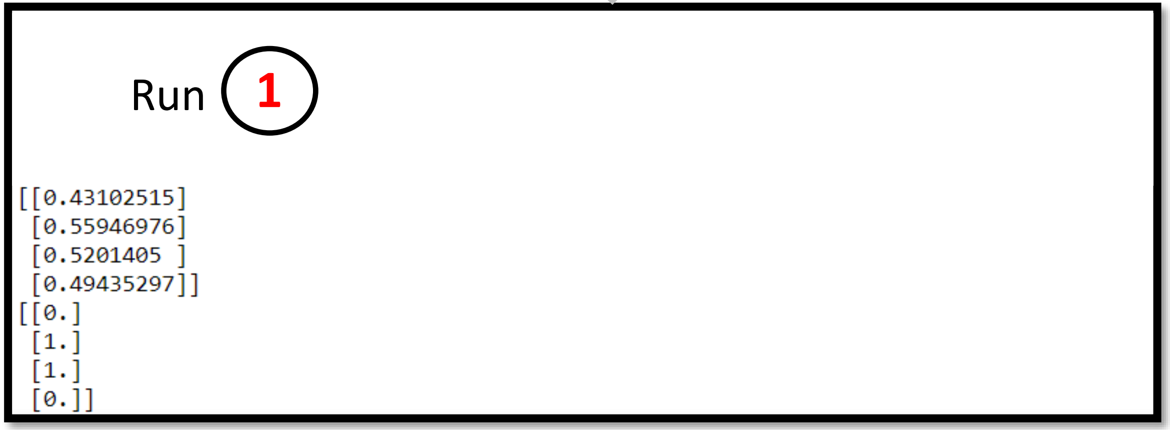
- Use predict and round functions of your model



```python
import tensorflow as tf
import numpy as np

# Specify Training base S: (X:x_train,Y:y_train)
x_train = np.array([[0, 0],
                    [0, 1],
                    [1, 0],
                    [1, 1]])

y_train = np.array([[0],[1],[1],[0]])
```



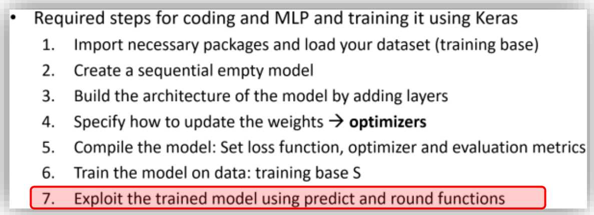
```python
print(model.predict(x_train))
print(model.predict(x_train).round())
```



Run 1

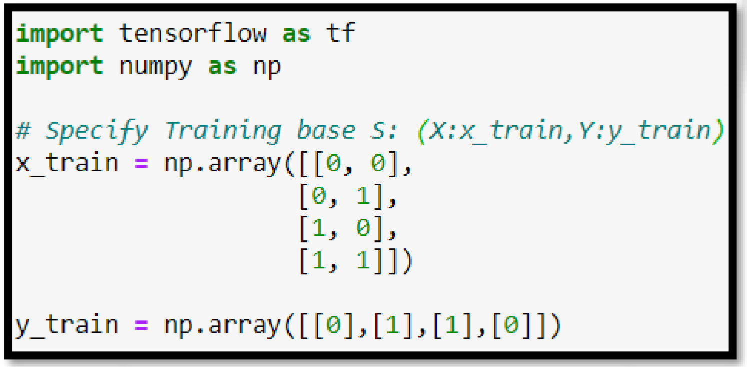
```
[[0.43102515]
 [0.55946976]
 [0.5201405 ]
 [0.49435297]]
[[0.]
 [1.]
 [1.]
 [0.]]
```

Getting started with Keras – TensorFlow core

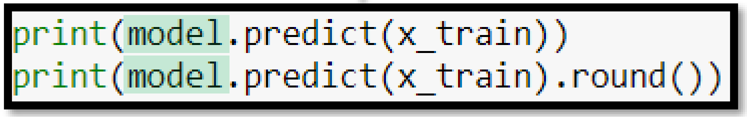


- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

- Use predict and round functions of your model

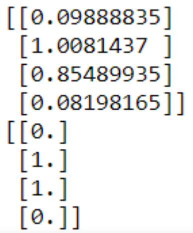


```python
import tensorflow as tf
import numpy as np

# Specify Training base S: (X:x_train,Y:y_train)
x_train = np.array([[0, 0],
                    [0, 1],
                    [1, 0],
                    [1, 1]])

y_train = np.array([[0],[1],[1],[0]])
```



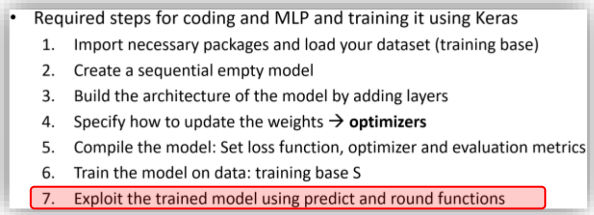
```python
print(model.predict(x_train))
print(model.predict(x_train).round())
```

Run 2



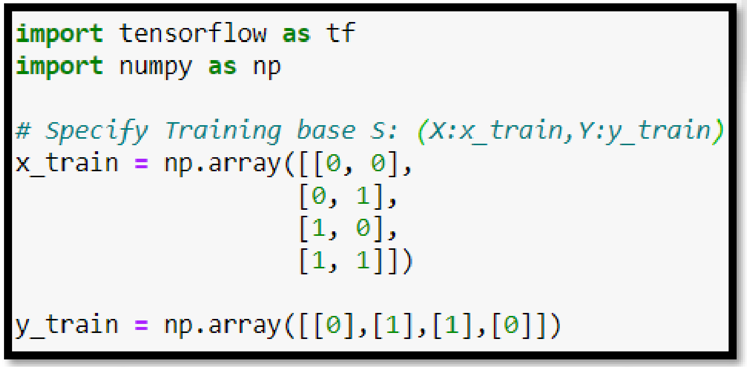
```
[[0.09888835]
 [1.0081437 ]
 [0.85489935]
 [0.08198165]]
[[0.]
 [1.]
 [1.]
 [0.]]
```

Getting started with Keras – TensorFlow core

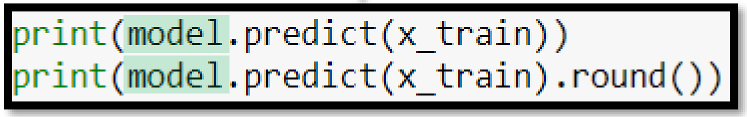


- Required steps for coding and MLP and training it using Keras
  1. Import necessary packages and load your dataset (training base)
  2. Create a sequential empty model
  3. Build the architecture of the model by adding layers
  4. Specify how to update the weights → **optimizers**
  5. Compile the model: Set loss function, optimizer and evaluation metrics
  6. Train the model on data: training base S
  7. Exploit the trained model using predict and round functions

- Use predict and round functions of your model

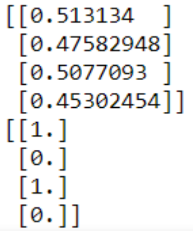


```
import tensorflow as tf
import numpy as np

# Specify Training base S: (X:x_train,Y:y_train)
x_train = np.array([[0, 0],
                    [0, 1],
                    [1, 0],
                    [1, 1]])

y_train = np.array([[0],[1],[1],[0]])
```



```
print(model.predict(x_train))
print(model.predict(x_train).round())
```

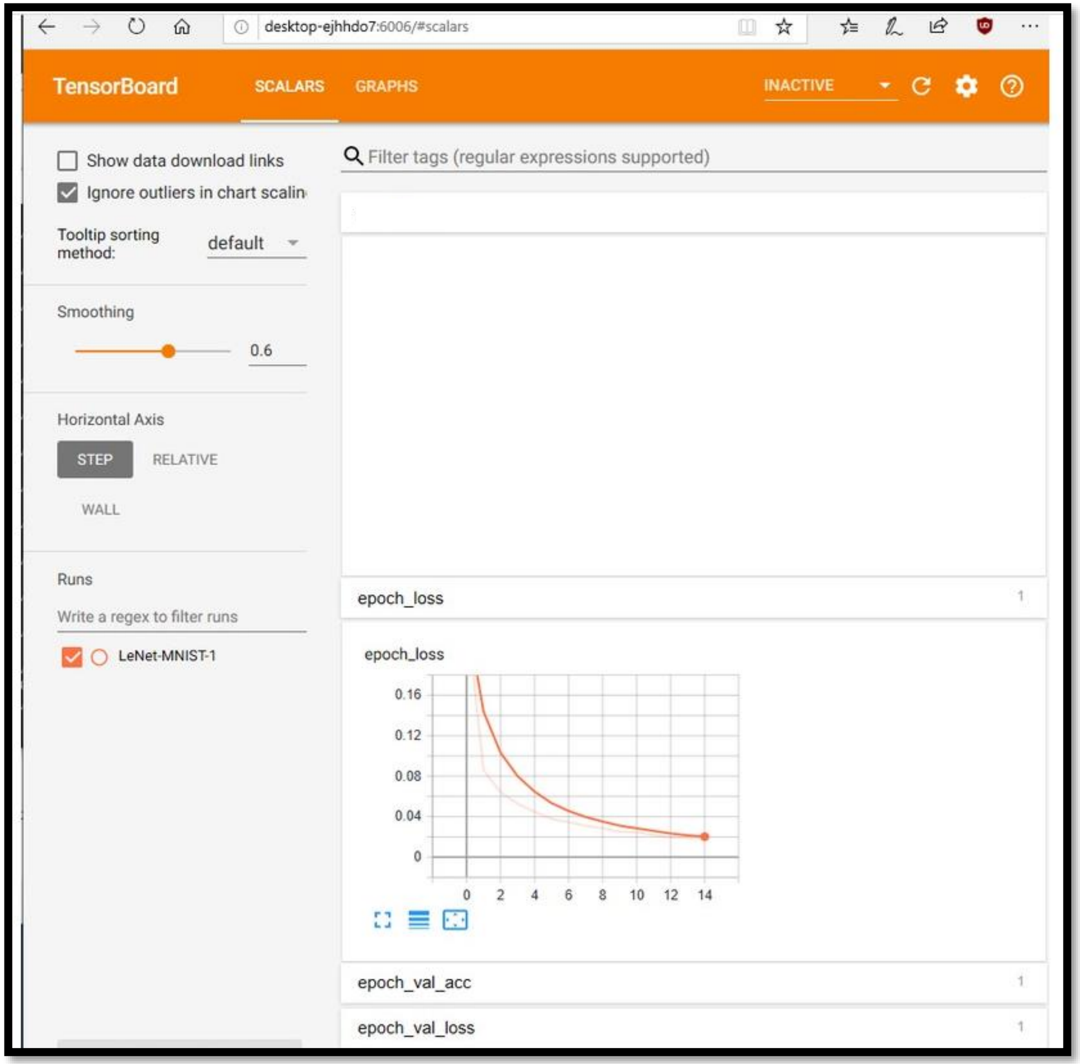
Run 3



```
[[0.513134  ]
 [0.47582948]
 [0.5077093 ]
 [0.45302454]]
[[1.]
 [0.]
 [1.]
 [0.]]
```

Getting started with Keras – TensorFlow core

- Use Tensorboard to display loss curves over epochs



☐ Show data download links

☑ Ignore outliers in chart scalin

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ LeNet-MNIST-1

Filter tags (regular expressions supported)

epoch_loss 1

epoch_loss

epoch_val_acc 1

epoch_val_loss 1

Lab session Keras – TensorFlow core

- Using Keras and Tensorflow backend
  - Build your MLP to solve logic functions
  - Set several architectures (depth, and hidden layer size) and test them by
    - Selecting activation functions studied in the course (sigmoid and pureline function)
    - Starting several runs
    - Plotting for each run the curves showing the evolution of your loss function and your accuracy metric – use TensorBoard
- **Look for the function that permit to output the weights of your best model and display the weights**