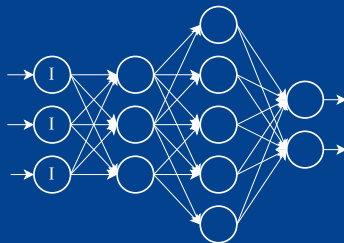




LESSON 6: Neural Networks

CARSTEN EIE FRIGAARD

AUTUMN 2019

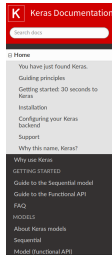
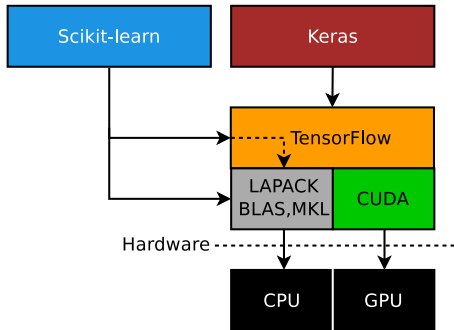




Keras and Tensorflow



Using the Keras API instead of Scikit-learn or TensorFlow



Docs » Home

Edit on GitHub

Keras: The Python Deep Learning library



You have just found Keras.

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user-friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

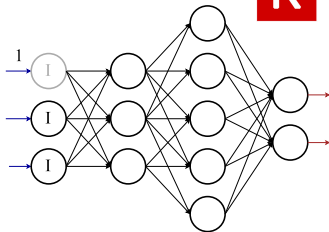
NOTE:

- ▶ documentation: <https://keras.io/>
- ▶ keras provides a `fit-predict`-interface,
- ▶ many similarities to Scikit-learn,
- ▶ but also many differences!

Building Keras MLPs



Using the Keras `Sequential` class,
programatically build up model:



```
1 # Build Keras model
2 model = Sequential()
3 model.add(Dense(input_dim=2, units=3, activation="tanh", ..)
4 model.add(Dense(units=5, activation="relu", ..)
5 model.add(Dense(units=2, activation="softmax"))
6 .
7 .
8 X_train, .. = train_test_split(X, y, .. )
9 .
10 .
11 y_train_categorical = to_categorical(y_train, num_classes=2)
12 y_test_categorical  = to_categorical(y_test,  num_classes=2)
13 .
14 .
15 history = model.fit(X_train, y_train_categorical, ..
16 .
17 .
18 score = model.evaluate(X_test, y_test_categorical)
```

Notes on Keras MLPs

Typical Keras MLP Supervised Classifier setup..

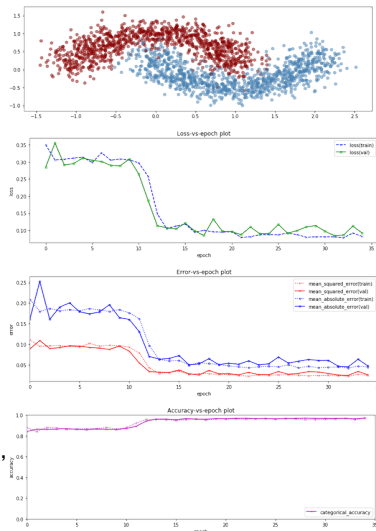
- ▶ loss function

```
loss='categorical_  
crossentropy'
```

- ▶ metrics collected via history

```
metrics=[  
    'categorical_accuracy',  
    'mean_squared_error',  
    'mean_absolute_error'] )
```

- ▶ input lay.: categorical encoding,
- ▶ output lay.: softmax function,



And notice that Keras do *not* provide metrics like

precision, recall, F1

but instead

categorical_accuracy, binary_accuracy

Input Layer: Categorical Encoding



For MLP Classification

One-hot to_categorical(.) encoding in Keras:

- ▶ input layer: one-hot class encoding,
- ▶ output layer: one neuron per output class that fires, and use `softmax` for output neurons
- ▶ beware of misformatted classes.

```
1 import numpy as np
2 from keras.utils.np_utils import to_categorical
3
4 y = np.array([1, 2, 0, 4, -1])
5 y_cat = to_categorical(y)
6
7 print(y_cat)
8
9 #[[0.  1.  0.  0.  0.] => i=0, class 1
10 # [0.  0.  1.  0.  0.] => i=1, class 2
11 # [1.  0.  0.  0.  0.] => i=2, class 0
12 # [0.  0.  0.  0.  1.] => i=3, class 4
13 # [0.  0.  0.  0.  1.] => i=4, also class 4!
14 #                               NOTE: no class 3
```

Output Layer: Softmax Function



For MLP Classification: Assigning a Probability for each Class

Softmax (softargmax/normalized exponential) definition

$$\text{softmax}(\mathbf{x})_i = \frac{e^{x_i}}{\sum_{i=1}^n e^{x_i}}$$

softmax: smooth approx. of **argmax** function.

argmax: the index-of-the-max-value for some data.

```
1 # python demo of softmax/argmax
2 x = np.array([1, 2, -4, 5, 1])
3 i = np.argmax(x)
```

```
4
5 PrintMatrix(x, "x = ")
6 print(f"np.argmax(x) = {np.argmax(x)}")
```

```
7
8 def softmax(x):
9     z = np.exp(x)
10    s = np.sum(z)
11    return z / s
```

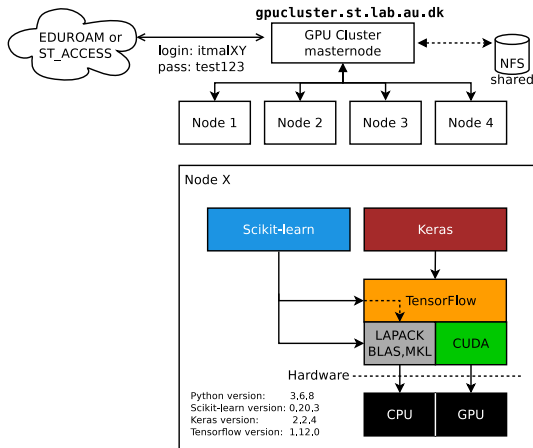
```
12
13 PrintMatrix(softmax(x), "softmax(x) = ")
14 print(f"np.argmax(softmax(x)) = {np.argmax(softmax(x))}")
```

[L06/Extra/softmax.ipynb]

```
1 # output
2 x = [ 1  2 -4  5  1 ]
3 np.argmax(x) = 3
4 softmax(x) = [0.02 0.05 0. 0.92 0.02]
5 np.argmax(softmax(x)) = 3
```

High-Performance-Computing (HPC)

Running on the ASE GPU cluster, your group login=itmalXY



NOTE: manuel GPU hukommelses Garbage Collection...

For keras + GPUkald:

```
StartupSequence_EnableGPU(gpu_mem_fraction=0.1, gpus=1)
```