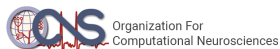


Hands on Keras (with Tensorflow)

Cecilia Jarne

cecilia.jarne@unq.edu.ar



Twitter: @ceciliajarne



Necessary Libraries

Available in the following sites:

<https://numpy.org/>

<https://www.scipy.org/install.html>

<https://www.tensorflow.org/install/>

<https://keras.io/>

if you prefer:

<https://www.anaconda.com/distribution/>

It would be so cool if you create a github account:

<https://github.com/>

Keras: high-level neural networks API, written in Python and capable of running on top of TensorFlow. It was developed with a focus on enabling fast experimentation.

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,)and others via complementary projects).
- Runs seamlessly on CPU and GPU.

The Sequential model is a linear stack of layers.

Three main steps are necessary:

1. Specify the input shape
2. Compile()
3. Fit()

Model compilation also need 3 elements:

- An optimizer. This could be the string identifier of an existing optimizer (such as rmsprop or adagrad), or an instance of the `Optimizer` class.
- A loss function. This is the objective that the model will try to minimize. It can be the string identifier of an existing loss function (such as `categorical_crossentropy` or `mse`), or it can be an objective function.
- A list of metrics.

```
1  
2 model.compile(loss='mean_squared_error', optimizer='sgd', metrics=[metrics.mae, metrics.  
    categorical_accuracy])
```

Keras models are trained on Numpy arrays of input data and labels. For training a model, you will typically use the fit function.

Layer Types: Convolutional layers

- Conv1D
- Conv2D
- Conv3D

Specially fast Convolutional layers:

- CuDNNGRU
- CuDNNLSTM

Layer Types: Recurrent layers

- RNN
- SimpleRNN
- GRU
- LSTM
- ConvLSTM2D
- StackedRNNCells

Losses:

- `mean_squared_error`
- `mean_squared_error(y_true, y_pred)`
- `mean_absolute_error(y_true, y_pred)`
- `mean_absolute_percentage_error(y_true, y_pred)`
- `mean_squared_logarithmic_error(y_true, y_pred)`
- `categorical_crossentropy(y_true, y_pred)`

Metrics

- `accuracy`
- `binary_accuracy`
- `binary_accuracy(y_true, y_pred)`
- `categorical_accuracy(y_true, y_pred)`
- `top_k_categorical_accuracy(y_true, y_pred, k=5)`

Optimizers

```
1 keras.optimizers.SGD(lr=0.01, momentum=0.0, decay=0.0, nesterov=False)
2 keras.optimizers.RMSprop(lr=0.001, rho=0.9, epsilon=1e-08, decay=0.0)
3 keras.optimizers.Adagrad(lr=0.01, epsilon=1e-08, decay=0.0)
4 keras.optimizers.Adadelta(lr=1.0, rho=0.95, epsilon=1e-08, decay=0.0)
5 keras.optimizers.Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
```

Activations:

Example:

```
1 from keras.layers import Activation, Dense
2 model.add(Dense(64))
3 model.add(Activation('tanh'))
```

Our notebooks:

https://github.com/katejarne/Keras_tensorflow_course