Python For Data Science Cheat Sheet

Keras

Learn Python for data science Interactively at www.DataCamp.com



Theano and TensorFlow that provides a high-level neural **Keras** is a powerful and easy-to-use deep learning library for

networks API to develop and evaluate deep learning models.

A Basic Example

```
>>> predictions = model.predict(data)
                                                                                                                                                                                               >>> model.add(Dense(1, activation='sigmoid'))
                                                                                                                                                                                                                                                                                                                    >>> model.add(Dense(32,
                                                                                                                                                                                                                                                                                                                                                           >>> model = Sequential()
                                                                                                                                                                                                                                                                                                                                                                                           >>> labels = np.random.randint(2,size=(1000,1))
                                                                                                                                                                                                                                                                                                                                                                                                                                  >>> data = np.random.random((1000,100))
                                                                                                                                                                                                                                                                                                                                                                                                                                                                             >>> from keras.layers import Dense
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                >>> from keras.models import Sequential
                                          >>> model.fit(data,labels,epochs=10,batch_size=32)
                                                                                                                                                            model.compile(optimizer='rmsprop',
                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       import numpy as np
                                                                                     metrics=['accuracy'])
                                                                                                                     loss='binary_crossentropy',
                                                                                                                                                                                                                                      input_dim=100))
                                                                                                                                                                                                                                                                                 activation='relu',
```

Also see NumPv. Pandas & Scikit-Learn

ally, you split the data in training and test sets, for which you can also resort to the train_test_split module of sklearn.cross_validation. rour data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ide-

```
>>> (x_train,y_train),(x_test,y_test) = mnist.load_data()
>>> (x_train2,y_train2),(x_test2,y_test2) = boston housing.load_data()
>>> (x_train3,y_train3),(x_test3,y_test3) = cifar10.load_data()
>>> (x_train4,y_train4),(x_test4,y_test4) = imdb.load_data(num_words=20000)
>>> num_classes = 10
                                                                                                                                                                                                                                                                                                                                                   >>> from keras datasets import boston_housing,
                                                                                                                                                                                                                                                              cifar10,
                                                                                                                                                                                                                                                                                                           mnist,
```

Other

```
>>> from urllib.request import urlopen
>>> data = np.loadtxt(urlopen("http://archive.ics.uci.edu/
>>> X = data[:,0:8]
>>> y = data [:,8]
                                                                                                    nl/machine-learning-databases/pima-indians-diabetes,
                                                                ima-indians-diabetes.data"),delimiter=",")
```

Preprocessing

Sequence Padding

```
>>> x_train4 = sequence.pad_sequences(x_train4,maxlen=80)
>>> x_test4 = sequence.pad_sequences(x_test4,maxlen=80)
                                                                                                  from keras.preprocessing import sequence
```

One-Hot Encoding

```
Ÿ
                                                                            Ÿ
Y train = to categorical(y train, num classes)
Y test = to categorical(y test, num classes)
Y train3 = to categorical(y train3, num classes)
Y test3 = to categorical(y test3, num classes)
```

Model Architecture

Sequential Model

```
>>> model3 = Sequential()
                            >>> model2 = Sequential()
                                                           >>> model = Sequential()
                                                                                           >>> from keras.models import Sequential
```

Binary Classification Multilayer Perceptron (MLP)

```
>>> model.add(Dense(8, kernel_initializer='uniform', activation='relu'))
>>> model.add(Dense(1, kernel_initializer='uniform', activation='sigmoid'))
                                                                                                                                                                                                                             >>> model.add(Dense(12,
                                                                                                                                                                                                                                                                      >>> from keras.layers import Dense
                                                                                              activation='relu'))
                                                                                                                                   kernel_initializer='uniform',
                                                                                                                                                                                input dim=8,
```

Multi-Class Classification

```
\
\
\
                                                                                                                           V
V
                                 \
\
\
                                                             \
\
\
                                                                                                                                                       >>> from keras.layers import Dropout
>>> model.add(Dense(10,activation='softmax');
                                                             model.add(Dense(512,activation='relu'))
                                                                                       model.add(Dropout(0.2))
                                                                                                                   model.add(Dense(512,activation='relu',input_shape=(784,))))
                              model.add(Dropout(0.2))
```

Regression

model.add(Dense(1) model.add(Dense(64,activation='relu',input_dim=train_data.shape[1]))

```
>>> model2.add(Conv2D(32,(3,3),padding='same',input_shape=x_train.shape[1:]))
                                                          >>> from keras.layers import Activation, Conv2D, MaxPooling2D, Flatten
                                                                                                                                       Convolutional Neural Network (CNN)
```

```
\
\
\
                                                                                      >>> model2.add(Activation('relu'))
       \
\
\
                                                         mode12.add(Conv2D(32,(3,3)))
                            model2.add(Activation('relu'))
model2.add(MaxPooling2D(pool_size=(2,2)))
```

```
model2.add(Activation('relu'))
mode 12. add (Dropout (0.25))
                                    model2.add(MaxPooling2D(pool_size=(2,2)))
                                                                                model2.add(Activation('relu'))
                                                                                                                     model2.add(Conv2D(64,(3, 3)))
                                                                                                                                                                                                  model2.add(Conv2D(64,(3,3), padding='same'))
```

\ \ \ V \ \ \ × × \ \ \

mode 12.add (Dropout (0.25))

```
mode 12.add (Flatten())
mode 12.add (Dropout (0.5))
                               mode12.add(Activation('relu'))
                                                               mode 12.add (Dense (512))
```

\ \ \

\ \ \ V V \ \ \

\ \ \ V V

Recurrent Neural Network (RNN)

model2.add(Activation('softmax') model2.add(Dense(num_classes))

```
>>> model3.add(LSTM(128,dropout=0.2,recurrent_dropout=0.2))
>>> model3.add(Dense(1,activation='sigmoid'))
                                                                                                                                                   >>> from keras.klayers import Embedding,LSTM
                                                                                                      >>> model3.add(Embedding(20000,128))
```

```
Ÿ
                                                                   Train and Test Sets
from sklearn.model_selection import train_tes
X_train5, X_test5, y_train5, y_test5 = train_test_split (X,
                     import train_test_split
```

```
from keras.utils import to_categorical
```

```
test size=0.33,
random_state=42)
```

Standardization/Normalization

```
>>> from sklearn.preprocessing import StandardScaler
>>> scaler = StandardScaler().fit(x train2)
>>> standardized X = scaler.transform(x train2)
>>> standardized_X_test = scaler.transform(x_test2)
```

Inspect Model

```
>>> model.output_shape
>>> model.summary()
>>> model.get_config()
>>> model.get_weights()
  Model configuration 
List all weight tensors in the model
                                                    Model summary representation
                                                                           Model output shape
```

Compile Mode

```
>>> model.compile(optimizer='rmsprop',
                                                                                                                                                                                    >>> model.compile(optimizer="adam",
                                            MLP: Regression
                                                                                                                       MLP: Multi-Class Classification
                                                                                                                                                                                                             MLP: Binary Classification
                                                                             metrics=['accuracy'])
                                                         metrics=['accuracy'])
                                                                                                                                                                loss='binary_crossentropy',
.oss='mse
                                                                             crossentropy',
```

Recurrent Neural Network

metrics=['mae'])

```
Ÿ
                                 model3.compile(loss='binary
metrics=['accuracy'])
                optimizer='adam
                                 crossentropy',
```

Model Training

```
>>> model3.fit(x_train4, y_train4,
verbose=1,
validation_data=(x_test4,y_test4))
                                 epochs=15,
                                                    batch size=32
```

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
<u>y</u>test,
batch_size=32)
```

Prediction

```
>>> model3.predict(x_test4, batch_size=32)
>>> model3.predict_classes(x_test4,batch_size=32)
```

Save/ Reload Models

```
>>> from keras.models import lo
>>> model3.save('model_file.h5')
                             load model
```

```
>>> my_model = load_model('my_
      model.h5')
```

Model Fine-tuning

```
Optimization Parameters
```

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=le-6)
>>> model2.compile(loss='categorical_cro')
                           optimizer=opt,
metrics=["accuracy"])
                                             rical_crossentropy',
```

Early Stopping

```
>>> early_stopping_monitor =
>>> model3.fit(x_train4,
                                                                                                                    from keras.callbacks import EarlyStopping
                          batch size=32,
epochs=15,
                                                 y_train4,
                                                                                              EarlyStopping (patience=2)
```

•

validation_data=(x_test4,y_test4),
callbacks=[early_stopping_monitor])