

- <https://github.com/kailashahirwar/cheatsheets-ai/blob/master/Keras.jpg>

Python For Data Science Cheat Sheet

Keras

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



Keras

Keras is a powerful and easy-to-use deep learning library for Theano and TensorFlow that provides a high-level neural networks API to develop and evaluate deep learning models.

A Basic Example

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

Data

Also see NumPy, Pandas & Scikit-Learn

Your data needs to be stored as NumPy arrays or as a list of NumPy arrays. Ideally, 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`.

Keras Data Sets

```
>>> from keras.datasets import boston_housing,
>>>                               mnist,
>>>                               cifar10,
>>>                               imdb
>>> (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
```

Other

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

Preprocessing

Also see NumPy & Scikit-Learn

Sequence Padding

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

One-Hot Encoding

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

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

Multilayer Perceptron (MLP)

Binary Classification

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

Multi-Class Classification

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

Regression

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

Convolutional Neural Network (CNN)

```
>>> from keras.layers import Activation, Conv2D, MaxPooling2D, Flatten
>>> model2.add(Conv2D(32, (3,3), padding='same', input_shape=x_train.shape[1:]))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(32, (3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Conv2D(64, (3,3), padding='same'))
>>> model2.add(Activation('relu'))
>>> model2.add(Conv2D(64, (3,3)))
>>> model2.add(Activation('relu'))
>>> model2.add(MaxPooling2D(pool_size=(2,2)))
>>> model2.add(Dropout(0.25))
>>> model2.add(Flatten())
>>> model2.add(Dense(512))
>>> model2.add(Activation('relu'))
>>> model2.add(Dropout(0.5))
>>> model2.add(Dense(num_classes))
>>> model2.add(Activation('softmax'))
```

Recurrent Neural Network (RNN)

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

Train and Test Sets

```
>>> from sklearn.model_selection import train_test_split
>>> X_train5,X_test5,y_train5,y_test5 = train_test_split(X,
>>>                                                         y,
>>>                                                         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 output shape
Model summary representation
Model configuration
List all weight tensors in the model

Compile Model

MLP: Binary Classification

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

MLP: Multi-Class Classification

```
>>> model.compile(optimizer='rmsprop',
>>>               loss='categorical_crossentropy',
>>>               metrics=['accuracy'])
```

MLP: Regression

```
>>> model.compile(optimizer='rmsprop',
>>>               loss='mse',
>>>               metrics=['mae'])
```

Recurrent Neural Network

```
>>> model3.compile(loss='binary_crossentropy',
>>>                optimizer='adam',
>>>                metrics=['accuracy'])
```

Model Training

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

Evaluate Your Model's Performance

```
>>> score = model3.evaluate(x_test,
>>>                          y_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 load_model
>>> model3.save('model_file.h5')
>>> my_model = load_model('my_model.h5')
```

Model Fine-tuning

Optimization Parameters

```
>>> from keras.optimizers import RMSprop
>>> opt = RMSprop(lr=0.0001, decay=1e-6)
>>> model2.compile(loss='categorical_crossentropy',
>>>                optimizer=opt,
>>>                metrics=['accuracy'])
```

Early Stopping

```
>>> from keras.callbacks import EarlyStopping
>>> early_stopping_monitor = EarlyStopping(patience=2)
>>> model3.fit(x_train4,
>>>            y_train4,
>>>            batch_size=32,
>>>            epochs=15,
>>>            validation_data=(x_test4,y_test4),
>>>            callbacks=[early_stopping_monitor])
```

DataCamp

Learn Python for Data Science interactively



관련 문서

Plugin Backlinks: 아무 것도 없습니다.

From:

<https://jace.link/> - Various Ways

Permanent link:

<https://jace.link/open/keras-%EA%B8%B0%EB%B3%B8%EB%AC%B8%EB%B2%95>

Last update: 2020/06/02 09:25

