Feed-forward networks in Keras

Max Pumperla - Skymind

Building blocks for MLPs

- `Dense` layers with activations

Building blocks for MLPs

- `Dense` layers with activations
- Use `Dropout` for regularization

Building blocks for MLPs

- `Dense` layers with activations
- Use `Dropout` for regularization
- Build a `Sequential` model from `Dense` and `Dropout` layers

Dense layers

```
from keras.layers import Dense

Dense(units,                    # Number of output neurons
      activation=None,          # Activation function by name
      use_bias=True,            # Use bias term or not
      kernel_initializer='glorot_uniform',
      bias_initializer='zeros')
```

skymind

Dropout layers

```
from keras.layers import Dropout

Dropout(rate,          # Fraction of units to drop
        seed=None)     # Random seed for reproducibility
```

skymind

Imports and loading data

```
from keras.datasets import mnist
from keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Dense, Dropout

batch_size = 128
num_classes = 10
epochs = 20

(x_train, y_train), (x_test, y_test) = mnist.load_data()
```

skymind

Data preprocessing

```
x_train = x_train.reshape(60000, 784)
x_test = x_test.reshape(10000, 784)
x_train = x_train.astype('float32')
x_test = x_test.astype('float32')
x_train /= 255
x_test /= 255

y_train = to_categorical(y_train, num_classes)
y_test = to_categorical(y_test, num_classes)
```

skymind

Defining and compiling your model

```
model = Sequential()
model.add(Dense(512, activation='relu',
          input_shape=(784,)))  # First layer only
model.add(Dropout(0.2))
model.add(Dense(512, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(num_classes, activation='softmax'))

model.summary()

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

skymind

Running and evaluating your model

```
model.fit(x_train, y_train, batch_size=batch_size,
          epochs=epochs, validation_data=(x_test, y_test))

score = model.evaluate(x_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
```