Neural Networks in Keras

Luckily, every time we need to use a neural network, we won't need to code the activation function, gradient descent, etc. There are lots of packages for this, which we recommend you to check out, including the following:

- Keras
- TensorFlow
- Caffe
- Theano
- Scikit-learn
- And many others!

In this course, we will learn Keras. Keras makes coding deep neural networks simpler. To demonstrate just how easy it is, you're going to build a simple fully-connected network in a few dozen lines of code.

We'll be connecting the concepts that you've learned in the previous lessons to the methods that Keras provides.

The general idea for this example is that you'll first load the data, then define the network, and then finally train the network.

Building a Neural Network in Keras

Here are some core concepts you need to know for working with Keras.

Sequential Model

```
from keras.models import Sequential

#Create the Sequential model
model = Sequential()
```

common methods like `compile()`, `fit()`, and `evaluate()` that are used to train and run the model. We'll cover these functions soon, but first let's start looking at the layers of the model.

Layers

The Keras Layer class provides a common interface for a variety of standard neural network layers. There are fully connected layers, max pool layers, activation layers, and more. You can add a layer to a model using the model's `add()` method. For example, a simple model with a single hidden layer might look like this:

```
import numpy as np
from keras.models import Sequential
from keras.layers.core import Dense, Activation

# X has shape (num_rows, num_cols), where the training data are stored
# as row vectors
X = np.array([[0, 0], [0, 1], [1, 0], [1, 1]], dtype=np.float32)

# y must have an output vector for each input vector
y = np.array([[0], [0], [0], [1]], dtype=np.float32)

# Create the Sequential model
model = Sequential()

# 1st Layer - Add an input layer of 32 nodes with the same input shape as
# the training samples in X
model.add(Dense(32, input_dim=X.shape[1]))

# Add a softmax activation layer
model.add(Activation('softmax'))

# 2nd Layer - Add a fully connected output layer
model.add(Dense(1))

# Add a sigmoid activation layer
model.add(Activation('sigmoid'))
```

Keras requires the input shape to be specified in the first layer, but it will automatically infer the shape of all other layers. This means you only have to explicitly set the input dimensions for the first layer.

outputs from the previous layer as inputs and pipes through to the next layer. This chain of passing output to the next layer continues until the last layer, which is the output of the model. We can see that the output has dimension 1.

The activation "layers" in Keras are equivalent to specifying an activation function in the Dense layers (e.g., `model.add(Dense(128)); model.add(Activation('softmax'))` is computationally equivalent to `model.add(Dense(128, activation="softmax")))`), but it is common to explicitly separate the activation layers because it allows direct access to the outputs of each layer before the activation is applied (which is useful in some model architectures).

Once we have our model built, we need to compile it before it can be run. Compiling the Keras model calls the backend (tensorflow, theano, etc.) and binds the optimizer, loss function, and other parameters required before the model can be run on any input data. We'll specify the loss function to be `categorical_crossentropy` which can be used when there are only two classes, and specify `adam` as the optimizer (which is a reasonable default when speed is a priority). And finally, we can specify what metrics we want to evaluate the model with. Here we'll use accuracy.

```
model.compile(loss="categorical_crossentropy", optimizer="adam", metrics = ["accuracy"])
```

We can see the resulting model architecture with the following command:

```
model.summary()
```

The model is trained with the `fit()` method, through the following command that specifies the number of training epochs and the message level (how much information we want displayed on the screen during training).

```
model.fit(X, y, nb_epoch=1000, verbose=0)
```

Note: In Keras 1, `nb_epoch` sets the number of epochs, but in Keras 2 this changes to the keyword `epochs` .

```
model.evaluate()
```

Pretty simple, right? Let's put it into practice.

Quiz

Let's start with the simplest example. In this quiz you will build a simple multi-layer feedforward neural network to solve the XOR problem.

1. Set the first layer to a `Dense()` layer with an output width of 8 nodes and the `input_dim` set to the size of the training samples (in this case 2).
2. Add a `tanh` activation function.
3. Set the output layer width to 1, since the output has only two classes. (We can use 0 for one class an 1 for the other)
4. Use a `sigmoid` activation function after the output layer.
5. Run the model for 50 epochs.

This should give you an accuracy of 50%. That's ok, but certainly not great. Out of 4 input points, we're correctly classifying only 2 of them. Let's try to change some parameters around to improve. For example, you can increase the number of epochs. You'll pass this quiz if you get 75% accuracy. Can you reach 100%?

To get started, review the Keras documentation about models and layers. The Keras example of a Multi-Layer Perceptron network is similar to what you need to do here. Use that as a guide, but keep in mind that there will be a number of differences.

network.py network_solution.py

```
import numpy as np
from keras.utils import np_utils
import tensorflow as tf
# Using TensorFlow 1.0.0; use tf.python_io in later versions
tf.python.control_flow_ops = tf

# Set random seed
np.random.seed(42)

# Our data
X = np.array([[0,0],[0,1],[1,0],[1,1]]).astype('float32')
y = np.array([[0],[1],[1],[0]]).astype('float32')

```

Keras

```python
# Building the model
xor = Sequential()

# Add required layers
xor.add(Dense(8, input_dim=2))

xor.add(Activation("tanh"))

xor.add(Dense(1))

xor.add(Activation("sigmoid"))

```

```
Looks good!
```

RESET QUIZ TEST RUN SUBMIT ANSWER

NEXT