Solutions 3

Jumping Rivers

Training Neural Networks

The fashion MNIST data is similar to the MNIST digits but instead of images of digits it is made up of images of 10 fashion items. Otherwise it's format is similar. Each image consists of a 28 by 28 pixel, single colour channel image. For the purposes of our example we will treat each of the 784 pixels as independent inputs. There are 60,000 training images and 10,000 test images. The *jrpytensorflow* package provides a function for downloading the data (the first time) and preprocessing it into an appropriate numpy array structure as below

```
import jrpytensorflow

X_train,X_test,y_train,y_test,labels = jrpytensorflow.datasets.load_fashion_mnist()
```

For this example, to begin with, we will consider optimiser performance. Since we want to get a feel for how the different optimisers effect training we will focus on training loss.

- Create a model similar to that we used in the notes for the MNIST digits example. (Hint: think about the shape of the data…)

```
from sklearn.preprocessing import LabelBinarizer
import tensorflow as tf

prep = LabelBinarizer()
y_train_bin = prep.fit_transform(y_train)
y_test_bin = prep.transform(y_test)

def model():
    model = tf.keras.models.Sequential([
        tf.keras.layers.Flatten(input_shape=(28,28)),
        tf.keras.layers.Dense(20,
                                    activation='relu'),
        tf.keras.layers.Dense(10, activation='softmax')
    ])

    return model

model = model()
model.compile(optimizer='sgd',
              loss = 'categorical_crossentropy',
```

```
            metrics = ['accuracy'])
model.fit(X_train, y_train_bin, epochs = 50)
```

- Try different optimisers and varying the hyperparameters. How do these affect the model fitting process? Remember to recreate your model for each run as this stores the weights as internal state.

```
model = model()
opt = SGD(lr=0.05)
model.compile(optimizer=opt,
            loss = 'categorical_crossentropy',
            metrics = ['accuracy'])
model.fit(X_train, y_train_bin, epochs = 25)
```

- Using TensorBoard can you visualise the architecture of the model you have built and its training progress?

```
%load_ext tensorboard

tensorBoardCallback = keras.callbacks.TensorBoard(
        log_dir = "logs/fit/",
        histogram_freq = 1)

model = model()
model.compile(optimizer=opt,
            loss = 'categorical_crossentropy',
            metrics = ['accuracy'])
model.fit(X_train, y_train_bin, epochs=100,
      validation_data=(X_test, y_test_bin),
      callbacks = [tensorBoardCallback])
```