**Introduction to Dense Net with Keras**

The most basic neural network architecture in deep learning is the dense neural networks consisting of fully connected layers. In the dense net, all the inputs and outputs are connected to all the neurons in each layer. [Keras](https://keras.io/) is the high-level APIs that runs on TensorFlow (and CNTK or Theano) which makes coding easier. Writing code in the low-level TensorFlow APIs is difficult and time-consuming. When I build a deep learning model, I always start with Keras so that I can quickly experiment with different architectures and parameters. Then, move onto TensorFlow to further fine tune it. When it comes to the first deep learning code, I think Dense Net with Keras is a good place to start. So, let’ get started.

**Dataset**

Deep learning 101 dataset is the classic MNIST, which is used for hand-written digit recognition. With the code below, you can certainly use MNIST.

In this example, I am using the machine learning classic Iris dataset. The dataset will be imported from a csv file. This gives you an idea on how to import csv into the deep learning model, rather than porting example data from the build-in package.

Deep learning on Iris certainly feels like cracking a nut with a sledge hammer. However, you can apply the knowledge and the same code to more appropriate datasets once you understand how it works.

There are many ways to get a csv version of Iris. I got it from R.

[cc lang="text" tab\_size="4" lines="-1" theme="mac-classic" line\_numbers="false"]

path = '/tmp/iris.csv'

data(iris)

write.csv(iris, path,row.names=FALSE)

[/cc]

**Steps**

1. **Import required modules**

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Data preparation**

Both Keras and TensorFlow takes numpy arrays as features and classes. When the prediction is categorical, the outcome needs to be one-hot encoded (see [one-hot encoding](https://www.kaggle.com/dansbecker/using-categorical-data-with-one-hot-encoding) explanation from the Kaggle’s website). For one-hot encoding, the class needs to be indexes (starting from 0). Once they are transformed, you can use keras.utils.to\_categorical() for conversion.

It uses sklearn.model\_selection.train\_test\_split to create training and test dataset.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Design Network**

I am using the sequential model with 2 fully-connected layers. ReLU is more popular in many deep neural networks, but I am using Tanh for activation because it actually performed better. You almost never use Sigmoid because it is slow to train. Softmax is used for the output layer.

Adding the 3rd layer degrades the performance. This makes sense as the data set is fairly simple.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Compile the model**

You need to define the loss function, optimizer and evaluation metrics. Cross-entropy is the gold standard for the cost function. You will almost never use quadratic. On the other hand, there are many options for optimisers. In this example, I have Adam as well as SGD with learning rate of 0.01. Both works fine.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

1. **Run the model**

The testing accuracy goes up to 96.7% after 120 epochs. With this dataset, a regular machine learning algorithm like random forest or logistic regression can achieve the similar results. The first rule of deep learning is that if the simpler machine learning algorithm can achieve the same outcome, use machine learning and look for a more complicated problem. Here, the purpose is to learn the actual programming process so that you can apply it to more complex problems.

[cc lang="python" tab\_size="4" lines="-1"]

[/cc]

**Next Step**

1. **Try using MNIST dataset on this code.**

MNIST is included in Keras and you can imported it as keras.datasets.mnist. It’s already split into training and test datasets. In preprocessing, you need to flatten the data (from 28 x 28 to 784) and convert y into one-hot encoded values. Here is the code to process the data.

1. **Replicate the same code with low-level TensorFlow code.**

TenorFlow is much more complicated than Keras. The way to code is quite unique. It will be difficult at first, but it will be worthwhile.

See