My Mnist - 0

TensorFlow + Keras 2.0 Intro
2017-4-1

Goals

- Make a simple neural net on MNIST.

Content made to follow along with the python file.
(taken mostly from:
https://blog.keras.io/keras-as-a-simplified-interface-to-tensorflow-tutorial.html)

Import + start session

```
import tensorflow as tf
from tensorflow.examples.tutorials.mnist import input_data

from keras import backend as K
sess = tf.Session()
K.set_session(sess)
```

Tensorflow quirks

- Make placeholder symbols for the data and labels

```
# mnist images are 28**2 = 784 pixels, and 10 classes
img = tf.placeholder(tf.float32, shape=(None, 784))
labels = tf.placeholder(tf.float32, shape=(None, 10))
```

Our first model

```
# make model
x = Dense(32, activation='relu')(img)
preds = Dense(10, activation='softmax')(x)
```

- note relu, not sigmoid activation

Define loss and performance metrics

```
# define loss
loss = tf.reduce_mean(categorical_crossentropy(labels, preds))
```

- A smooth-ish function to tell the network how learn from its errors

```
# define performance metrics
acc_value = tf.reduce_mean(categorical_accuracy(labels, preds))
```

- How we actually judge its performance, with accuracy

Tensorflow quirks

```
# initialize all variables
init_op = tf.global_variables_initializer()
sess.run(init_op)
```

- Do this before we start evaluating the symbolic functions we defined

The magic

```
for iter in range(num_iter):
  batch = mnist_data.train.next_batch(batch_size)
  train_step.run(feed_dict={
      img: batch[0],
      labels: batch[1]
  })
```

- Training in Tensorflow is much more manual than in Keras

Print our success

```
print('train acc = {:.4f}'.format(
  acc_value.eval(feed_dict={
      img: mnist_data.train.images,
      labels: mnist_data.train.labels
  })
))
```

- Training accuracy after the current batch

```
print('valid acc = {:.4f}'.format(
  acc_value.eval(feed_dict={
      img: mnist_data.test.images,
      labels: mnist_data.test.labels
  })
))
```

- Test accuracy after the current batch

Should reach ~0.85-0.90