COSC4337_100-MNIST-Data-Basic-Approach

1 MNIST Data Set - Basic Approach

1.0.1 Get the MNIST Data

[1]:
```
import tensorflow as tf
```

[2]:
```
from tensorflow.examples.tutorials.mnist import input_data
```

[3]:
```
mnist = input_data.read_data_sets("MNIST_data/",one_hot=True)
```

```
WARNING:tensorflow:From <ipython-input-3-758d29429358>:1: read_data_sets (from
tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be
removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from
tensorflow/models.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:260:
maybe_download (from tensorflow.contrib.learn.python.learn.datasets.base) is
deprecated and will be removed in a future version.
Instructions for updating:
Please write your own downloading logic.
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:262:
extract_images (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting MNIST_data/train-images-idx3-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:267:
extract_labels (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.data to implement this functionality.
Extracting MNIST_data/train-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:110:
dense_to_one_hot (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
```

```
deprecated and will be removed in a future version.
Instructions for updating:
Please use tf.one_hot on tensors.
Extracting MNIST_data/t10k-images-idx3-ubyte.gz
Extracting MNIST_data/t10k-labels-idx1-ubyte.gz
WARNING:tensorflow:From C:\Users\RizkN\.conda\envs\tf1\lib\site-
packages\tensorflow_core\contrib\learn\python\learn\datasets\mnist.py:290:
DataSet.__init__ (from tensorflow.contrib.learn.python.learn.datasets.mnist) is
deprecated and will be removed in a future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from
tensorflow/models.
```

** Alternative sources of the data just in case: **

- http://yann.lecun.com/exdb/mnist/
- https://github.com/mrgloom/MNIST-dataset-in-different-formats

[4]:
```
type(mnist)
```

[4]:
```
tensorflow.contrib.learn.python.learn.datasets.base.Datasets
```

[5]:
```
mnist.train.images
```

[5]:
```
array([[0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       …,
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.],
       [0., 0., 0., …, 0., 0., 0.]], dtype=float32)
```

[6]:
```
mnist.train.num_examples
```

[6]:
```
55000
```

[7]:
```
mnist.test.num_examples
```

[7]:
```
10000
```

[8]:
```
mnist.validation.num_examples
```

[8]:
```
5000
```

1.0.2 Visualizing the Data

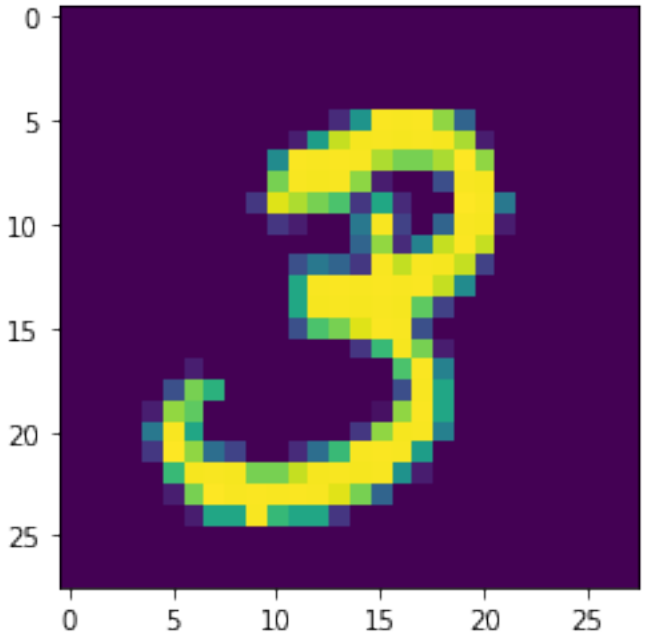
[9]:
```
import matplotlib.pyplot as plt
%matplotlib inline
```

[10]:
```
mnist.train.images[1].shape
```

[10]:
```
(784,)
```

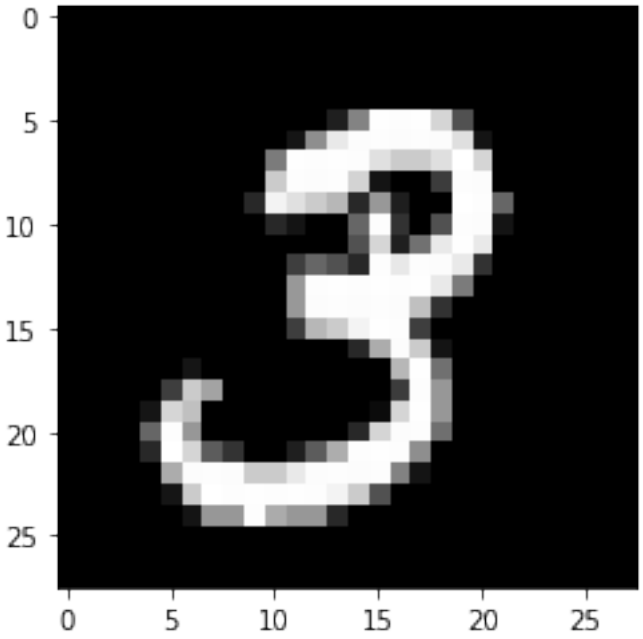
[11]:
```
plt.imshow(mnist.train.images[1].reshape(28,28))
```

[11]:
```
<matplotlib.image.AxesImage at 0x14b0ec77948>
```



[12]:
```
plt.imshow(mnist.train.images[1].reshape(28,28),cmap='gist_gray')
```

[12]:
```
<matplotlib.image.AxesImage at 0x14b0f9cab08>
```

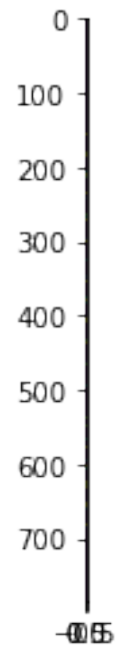


[13]:
```
mnist.train.images[1].max()
```

[13]: 1.0

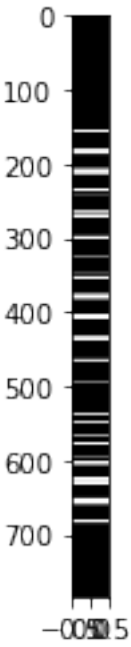
[14]:
```
plt.imshow(mnist.train.images[1].reshape(784,1))
```

[14]: <matplotlib.image.AxesImage at 0x14b0fa30f08>



[15]:
```
plt.imshow(mnist.train.images[1].reshape(784,1),cmap='gist_gray',aspect=0.02)
```

[15]: <matplotlib.image.AxesImage at 0x14b0fa9fd48>



1.1 Create the Model

[16]:
```
x = tf.placeholder(tf.float32,shape=[None,784])
```

[17]:
```
# 10 because 0-9 possible numbers
W = tf.Variable(tf.zeros([784,10]))
```

[18]:
```
b = tf.Variable(tf.zeros([10]))
```

[19]:
```
# Create the Graph
y = tf.matmul(x,W) + b
```

Loss and Optimizer

[20]:
```
y_true = tf.placeholder(tf.float32,[None,10])
```

[21]:
```
# Cross Entropy
```

[22]:
```
cross_entropy = tf.reduce_mean(tf.nn.
↪softmax_cross_entropy_with_logits_v2(labels=y_true, logits=y))
```

[23]:
```
optimizer = tf.train.GradientDescentOptimizer(learning_rate=0.5)
```

[24]:
```
train = optimizer.minimize(cross_entropy)
```

1.1.1 Create Session

[25]:
```
init = tf.global_variables_initializer()
```

[26]:
```
with tf.Session() as sess:
    sess.run(init)

    # Train the model for 1000 steps on the training set
    # Using built in batch feeder from mnist for convenience

    for step in range(1000):

        batch_x , batch_y = mnist.train.next_batch(100)

        sess.run(train,feed_dict={x:batch_x,y_true:batch_y})

    # Test the Train Model
    matches = tf.equal(tf.argmax(y,1),tf.argmax(y_true,1))

    acc = tf.reduce_mean(tf.cast(matches,tf.float32))

    print(sess.run(acc,feed_dict={x:mnist.test.images,y_true:mnist.test.
↪labels}))
```

```
0.917
```

While this may seem pretty good, we can actually do much better, the best models can get above 99% accuracy.

How do they do this? By using other models, such as convolutional neural networks!

[ ]: