In [1]:
```
from keras.models import Sequential
from keras.layers import Dense

import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

from tensorflow.examples.tutorials.mnist import input_data

mnist = input_data.read_data_sets('./', one_hot=True)
```

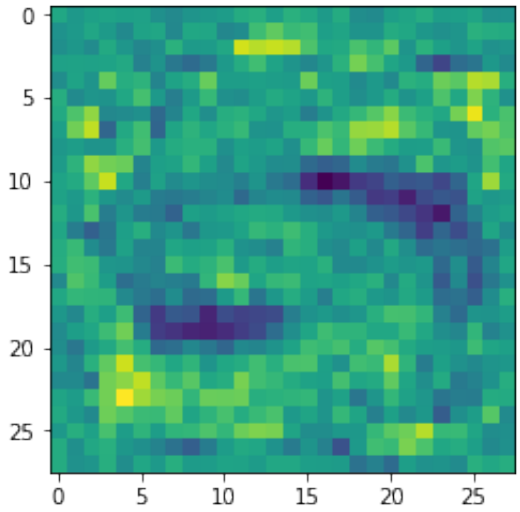
```
Using TensorFlow backend.

Successfully downloaded train-images-idx3-ubyte.gz 9912422 bytes.
Extracting ./train-images-idx3-ubyte.gz
Successfully downloaded train-labels-idx1-ubyte.gz 28881 bytes.
Extracting ./train-labels-idx1-ubyte.gz
Successfully downloaded t10k-images-idx3-ubyte.gz 1648877 bytes.
Extracting ./t10k-images-idx3-ubyte.gz
Successfully downloaded t10k-labels-idx1-ubyte.gz 4542 bytes.
Extracting ./t10k-labels-idx1-ubyte.gz
```

In [10]:
```
x, y = mnist.train.next_batch(20)
x.shape
```

Out[10]:
```
(20, 784)
```

In [3]:
```
model = Sequential() #initiats a layer
model.add(Dense(100, activation='relu', input_dim=784)) # adds a layer
model.add(Dense(10, activation='softmax')) #one layer of softmax
    #required
model.compile(loss='categorial_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
```

In [19]:
```
plt.imshow(model.get_weights()[0][:,40].reshape(28,28))
plt.show()
```



In [4]:
```
def get_batch(dataset, batch_size = 256):
    while (1):
        yield dataset.next_batch(batch_size)
        #continues where left off if called again(new
            #test set trials are different)
```

In [5]:
```
batch_size = 256
test_gen = get_batch(mnist.test, batch_size)
steps_per_epoch = mnist.test.num_examples// batch_size
model.evaluate_generator(test_gen, steps_per_epoch)
#untrained model
```

Out[5]: [2.4107079261388535, 0.12770432692307693]

In [11]:
```
batch_size = 256
data_gen = get_batch(mnist.train, batch_size)
steps_per_epoch = mnist.train.num_examples//batch_size

model.fit_generator(data_gen, steps_per_epoch, epochs=10)
#trains model to data
```

```
Epoch 1/10
214/214 [==============================] - 1s - loss: 0.0541 - acc:
0.9847
Epoch 2/10
214/214 [==============================] - 1s - loss: 0.0490 - acc:
0.9864
Epoch 3/10
214/214 [==============================] - 1s - loss: 0.0430 - acc:
0.9882
Epoch 4/10
214/214 [==============================] - 1s - loss: 0.0391 - acc:
0.9894
Epoch 5/10
214/214 [==============================] - 1s - loss: 0.0351 - acc:
0.9904
Epoch 6/10
214/214 [==============================] - 1s - loss: 0.0317 - acc:
0.9919
Epoch 7/10
214/214 [==============================] - 1s - loss: 0.0285 - acc:
0.9928
Epoch 8/10
214/214 [==============================] - 1s - loss: 0.0255 - acc:
0.9943
Epoch 9/10
214/214 [==============================] - 1s - loss: 0.0229 - acc:
0.9949
Epoch 10/10
214/214 [==============================] - 1s - loss: 0.0213 - acc:
0.9951
```

Out[11]:
```
<keras.callbacks.History at 0x7ffa9411b8d0>
```

In [7]:
```
model.evaluate_generator(test_gen, steps_per_epoch)
```

Out[7]:
```
[0.085014622623675332, 0.97282053154205606]
```

It is really important to be able to reload the model after you've been training it for hours on end (usually).

In [8]:
```
from keras.models import load_model

model.save('my_model.h5')

model2 = load_model('my_model.h5')
```

In [9]:
```
model2.evaluate_generator(test_gen, steps_per_epoch)
```

Out[9]:
```
[0.084644120517317387, 0.97276577102803741]
```

In [ ]:
```

```