Reference: http://math.stackexchange.com/questions/945871/derivative-of-softmax-loss-function
Inputs: x_i, where i = 1, 2 ...785
Hidden Layer: output is h_i and input is a_i, where i = 1, 2 ... 201
Output Layer: output is g_i and input is b_i, where i = 1, 2 ... 26
V is 200 x 785 weight matrix, V_{ij} is the weight from x_j to h_i
W is 26 x 201 weight matrix, W_{ij} is the weight from h_j to g_i
$h_i = t(a_i) = t(V_i x) = t(\Sigma_{j=1}^{785} V_{ij} x_j)$ where t is tranh function
$g_i = s(b_i) = s(W_i h) = s(\Sigma_{j=1}^{201} W_{ij} h_j)$ where s is softmax function
$\frac{dt(a)}{da} = 1 - t^2(a)$
$\frac{ds(b)}{db} = s(b) \cdot (1 - s(b))$
$\frac{dL(y,s(b))}{db} = b - y$

W:

$$\frac{dL}{dw_{ij}} = \frac{dL}{db_i} \cdot \frac{db_i}{dw_{ij}}$$
$$= (s(W_i h) - y_i) \cdot h_j$$
$$= (s(W_i h) - y_i) \cdot t(V_j x)$$

matrix form:

$$\frac{dL}{dW} = (g - y) \cdot h^T$$

V:

$$\frac{dL}{dv_{ij}} = \frac{dL}{dh_i} \cdot \frac{dh_i}{dv_{ij}} = \frac{dh_i}{dv_{ij}} \cdot \Sigma_{j=1}^{26} (\frac{dL}{db_j} \cdot \frac{b_j}{h_i})$$
$$= x_j \cdot (1 - t^2(a_i)) \cdot \Sigma_{j=1}^{26} ((s(b_j) - y_j) \cdot w_{ji})$$
$$= x_j \cdot (1 - t^2(V_i x)) \cdot \Sigma_{j=1}^{26} ((s(W_j h) - y_j) \cdot w_{ji})$$

matrix form:

$$\frac{dL}{dV} = W^T (g - y)(1 - h^2) x$$

Stochastic Gradient Descent:

$$w_{ij} = w_{ij} - \epsilon \cdot \frac{dL}{dw_{ij}}$$
$$v_{ij} = v_{ij} - \epsilon \cdot \frac{dL}{dv_{ij}}$$

In [1]:
```
import numpy as np
import pandas as pd
import scipy.ndimage
import pickle
import math
from random import randint
from IPython.display import Image
import scipy.io as sio
from sklearn import preprocessing
import matplotlib.pyplot as plt
```

2. Implementation

Kaggle Score: 0.86221 (yika) ¶

- max steps: 400K
- initial learning rate: 0.01
- regularization lambda: 0.0001

Data Preprocess

In [2]:
```
def shift(X):
    X_shift = np.zeros(X.shape)
    X = np.reshape(np.matrix(X), (28, 28))
    sign = -1
    if (np.random.rand() > 0.5):
        sign = 1
    displacement = np.floor(np.random.rand() / 0.3) * sign
    direction = 0
    if (np.random.rand() > 0.5):
        direction = 1
    X = np.roll(X, int(displacement), axis=direction)
    X_shift = np.reshape(X, (1, 784))

    return X_shift

def rotate(X):
    X_rotated = np.zeros(X.shape)
    X = np.matrix(X).reshape((28, 28))
    degree = np.random.rand() * 10
    if np.random.rand() > 0.5:
        degree = -degree
    X = scipy.ndimage.interpolation.rotate(X, degree, reshape = False, mode
    X = np.reshape(X, (1, 784))
    return X

def one_hot(y):
    hot = np.zeros([len(y), 26])
    for i in range(len(y)):
        hot[i][y[i][0] - 1] = 1
    return hot

def preprocess(train_x, train_y, test_x):

    train_x, test_x = preprocessing.scale(train_x), preprocessing.scale(test
    rand = np.random.permutation(len(train_x))
    train_x, train_y = train_x[rand], train_y[rand]
    split = 4800
    val_x, val_y = train_x[:split], one_hot(train_y[:split])
    train_x, train_y = train_x[split:], one_hot(train_y[split:])
    val_x = np.c_[val_x, np.ones(val_x.shape[0]) ]
    test_x = np.c_[test_x, np.ones(test_x.shape[0]) ]

    return train_x, train_y, val_x, val_y, test_x
```

In [3]:
```
data = sio.loadmat("data/letters_data.mat")
train_x = data['train_x']
train_y = data['train_y']
test_x = data['test_x']
train_x, train_y, val_x, val_y, test_x = preprocess(train_x, train_y, test_x
```

```
/Library/Frameworks/Python.framework/Versions/3.6/lib/python3.6/site-pack
ages/sklearn/utils/validation.py:429: DataConversionWarning: Data with in
put dtype uint8 was converted to float64 by the scale function.
  warnings.warn(msg, _DataConversionWarning)
```

Graph Construction

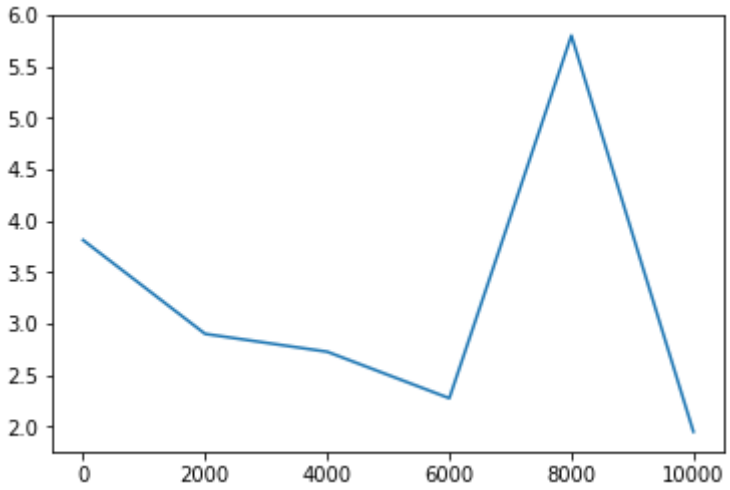
In [3]:
```
def draw_sample(x, y):
    rand = randint(0,len(x)-1)
    xx = x[rand]
    yy = y[rand]
    xx = shift(xx)
    xx = rotate(xx)
    xx = np.c_[xx, np.ones(xx.shape[0])]
    return xx, yy

def tanh(z):
    return np.tanh(z)

def softmax(x):
    return np.exp(x) / np.sum(np.exp(x), axis=0)

def soft_loss(z, y, w, lam):
    z = np.reshape(z, (1, 26))[0]
    z[z < 1e-8] = 1e-8
    regularization = lam * np.linalg.norm(w)
    return -1.0 * np.dot(np.log(z), y) + regularization

def save_weights(step, v, w):
    obj = [step, v, w]
    with open("saved/"+str(step)+".p", "wb") as f:
        pickle.dump(obj, f)
    f.close()

class Graph:
    def __init__(self, n_hidden=400):
        self.n_hidden = n_hidden
        self.V = np.random.normal(loc=0, scale=1.0/math.sqrt(784), size=(sel
        self.W = np.random.normal(loc=0, scale=1.0/math.sqrt(self.n_hidden),
        self.val_accs, self.train_accs, self.losses, self.acc_steps, self.lo
        self.start = 0

    def train(self, x, y, val_x, val_y, max_steps=100001, lr=1e-2, lam=0.1,
        self.lam = lam
        for i in range(self.start, self.start+max_steps):
            xx, yy = draw_sample(x, y)
            hh, zz = self._test(xx) # forward pass 26 x 1

            # stochastic gradient descent
            v_grad, w_grad = self.compute_gradients(xx, yy, hh, zz)
            self.V -= (lr * v_grad)
            self.W -= (lr * w_grad)

            # validation
            if i % stop_step == 0 or i == max_steps - 1:
                val_acc = self.evaluate(val_x, val_y)
                train_acc = self.evaluate(np.c_[train_x, np.ones(train_x.sha
                loss = soft_loss(zz, yy, self.W, lam)
                print("step {}: train acc = {}, validate acc = {}, loss = {}
                self.train_accs.append(train_acc)
                self.val_accs.append(val_acc)
                self.acc_steps.append(i)
                self.losses.append(loss)
```

```
                self.loss_steps.append(i)
                save_weights(i, self.V, self.W)

    def _test(self, xx):
        xx = np.reshape(xx, (785, 1))
        temp1 = np.matmul(self.V, xx)
        h = tanh(temp1) # 401 x 1
        h = np.append(h, [[1]], axis=0)
        temp2 = np.matmul(self.W, h)
        z = softmax(temp2) # 26 x 1
        return h, z

    def test(self, x):
        x = x.T
        temp1 = np.matmul(self.V, x)
        h = tanh(temp1)
        h = np.append(h, [[1]*len(x[0])], axis=0)
        temp2 = np.matmul(self.W, h)
        z = softmax(temp2).T
        return h, z

    def evaluate(self, x, y):
        correct = 0
        h, z = self.test(x)
        labels = np.argmax(z, axis=1)
        y = np.argmax(y, axis=1)
        correct = np.sum(labels == y)
        return 1.0 * correct / len(y)

    def update_weights(self, step):
        with open("saved/" + str(step) + ".p", "rb") as f:
            obj = pickle.load(f)
        f.close()
        self.V, self.W = obj[1], obj[2]
        self.start = step

    def compute_gradients(self, x, y, h, z):
        # W
        y = np.reshape(y, (26, 1))
        w_grad = 1.0 * np.matmul((z - y), h.T)
        w_grad += 2 * self.lam * self.W

        # V
        temp = list(np.reshape(np.matmul(self.W.T, (z - y))[:self.n_hidden],
        temp2 = list(np.reshape(1 - h[:self.n_hidden] ** 2, (1, self.n_hidde
        product = np.array([a*b for a, b in zip(temp, temp2)])
        v_grad = np.outer(product, x)
        v_grad += 2 * self.lam * self.V

        return v_grad, w_grad
```

In [83]:
```
g = Graph()
g.train(train_x, train_y, val_x, val_y, lr=0.001, lam=0.1, max_steps=10000,
```

```
step 0: train acc = 0.044616666666666666, validate acc = 0.04, loss = 3.8
15016606337397
step 2000: train acc = 0.4807166666666667, validate acc = 0.48375, loss =
2.903438655932182
step 4000: train acc = 0.5563, validate acc = 0.5704166666666667, loss =
 2.7297818390040316
step 6000: train acc = 0.5337, validate acc = 0.54875, loss = 2.277983386
6816315
step 8000: train acc = 0.550025, validate acc = 0.554375, loss = 5.803197
032927515
step 9999: train acc = 0.52965, validate acc = 0.5308333333333334, loss =
1.9527718788282125
```

In [84]:
```
plt.plot(g.loss_steps, g.losses)
plt.show()
```



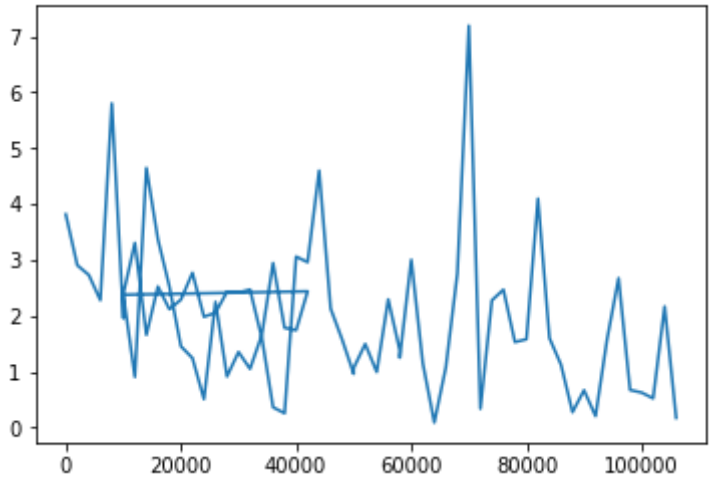
In [86]:
```
g.update_weights(9999)
g.train(train_x, train_y, val_x, val_y, lr=0.001, lam=0.01, max_steps=50000,
```

```
step 10000: train acc = 0.5294333333333333, validate acc = 0.530833333333
3334, loss = 2.374873648594905
step 12000: train acc = 0.5491416666666666, validate acc = 0.550625, loss
= 0.9032761042882205
step 14000: train acc = 0.5847833333333333, validate acc = 0.588125, loss
= 4.645945565034356
step 16000: train acc = 0.6000833333333333, validate acc = 0.601458333333
3333, loss = 3.370690299112182
step 18000: train acc = 0.6094083333333333, validate acc = 0.617291666666
6667, loss = 2.522286042430744
step 20000: train acc = 0.616075, validate acc = 0.6254166666666666, loss
= 1.4509633118147083
step 22000: train acc = 0.62595, validate acc = 0.6277083333333333, loss
 = 1.240316814066879
step 24000: train acc = 0.63425, validate acc = 0.6420833333333333, loss
 = 0.5060589764567509
step 26000: train acc = 0.6381083333333334, validate acc = 0.644375, loss
= 2.2465136616740544
step 28000: train acc = 0.6437583333333333, validate acc = 0.643541666666
6667, loss = 0.9115407745591132
step 30000: train acc = 0.6471416666666666, validate acc = 0.648333333333
3333, loss = 1.350296923045361
step 32000: train acc = 0.65015, validate acc = 0.6564583333333334, loss
 = 1.048026421658969
step 34000: train acc = 0.6562916666666667, validate acc = 0.660833333333
3334, loss = 1.641385957244486
step 36000: train acc = 0.6575083333333334, validate acc = 0.660208333333
3333, loss = 0.35931417137791233
step 38000: train acc = 0.6598833333333334, validate acc = 0.6625, loss =
0.2528256955815294
step 40000: train acc = 0.6534833333333333, validate acc = 0.657083333333
3334, loss = 3.057971061614193
step 42000: train acc = 0.6612, validate acc = 0.66875, loss = 2.95857224
99905396
step 44000: train acc = 0.6606666666666666, validate acc = 0.666041666666
6666, loss = 4.598121066205601
step 46000: train acc = 0.6658833333333334, validate acc = 0.666458333333
3334, loss = 2.1267402971575597
step 48000: train acc = 0.6620583333333333, validate acc = 0.67, loss =
 1.5714437171947337
step 49999: train acc = 0.6648916666666667, validate acc = 0.66625, loss
 = 0.9585315016142746
step 50000: train acc = 0.6645, validate acc = 0.66625, loss = 1.06395792
58840368
step 52000: train acc = 0.6671333333333334, validate acc = 0.668541666666
6667, loss = 1.4932478406500118
step 54000: train acc = 0.669725, validate acc = 0.67375, loss = 0.991970
6164878862
step 56000: train acc = 0.6741166666666667, validate acc = 0.67875, loss
 = 2.29223444085386
step 58000: train acc = 0.6772166666666667, validate acc = 0.683125, loss
= 1.3787377867126647
```

In [89]:
```
g.update_weights(58000)
g.train(train_x, train_y, val_x, val_y, lr=0.001, lam=0.005, max_steps=50000
```

```
step 58000: train acc = 0.6771666666666667, validate acc = 0.683333333333
3333, loss = 1.2477980737142051
step 60000: train acc = 0.673125, validate acc = 0.67625, loss = 3.002169
1029771618
step 62000: train acc = 0.6822333333333334, validate acc = 0.687916666666
6666, loss = 1.1498734700545281
step 64000: train acc = 0.67985, validate acc = 0.6845833333333333, loss
 = 0.08608588298087218
step 66000: train acc = 0.6774583333333334, validate acc = 0.680416666666
6667, loss = 1.0754094464550197
step 68000: train acc = 0.6841166666666667, validate acc = 0.686458333333
3333, loss = 2.755584462053031
step 70000: train acc = 0.6914583333333333, validate acc = 0.694166666666
6667, loss = 7.195007517748422
step 72000: train acc = 0.69085, validate acc = 0.69375, loss = 0.3329105
038364305
step 74000: train acc = 0.689125, validate acc = 0.6985416666666666, loss
= 2.275389730344289
step 76000: train acc = 0.6938166666666666, validate acc = 0.697708333333
3334, loss = 2.4664639910814605
step 78000: train acc = 0.6895833333333333, validate acc = 0.689166666666
6667, loss = 1.5280535916356341
step 80000: train acc = 0.6987, validate acc = 0.705, loss = 1.5782609501
77723
step 82000: train acc = 0.6979333333333333, validate acc = 0.696666666666
6667, loss = 4.0967473536100165
step 84000: train acc = 0.7003, validate acc = 0.705625, loss = 1.5984270
215809706
step 86000: train acc = 0.7019, validate acc = 0.700625, loss = 1.1229814
872993007
step 88000: train acc = 0.70455, validate acc = 0.7110416666666667, loss
 = 0.27270716863438865
step 90000: train acc = 0.7060666666666666, validate acc = 0.705208333333
3333, loss = 0.6648825812725379
step 92000: train acc = 0.7071, validate acc = 0.708125, loss = 0.2025623
707432069
step 94000: train acc = 0.7056833333333333, validate acc = 0.70625, loss
 = 1.575165041729522
step 96000: train acc = 0.7117, validate acc = 0.71125, loss = 2.67242830
85899385
step 98000: train acc = 0.7107583333333334, validate acc = 0.717916666666
6666, loss = 0.6695005721580339
step 100000: train acc = 0.708625, validate acc = 0.7116666666666667, los
s = 0.6230516461024488
step 102000: train acc = 0.71445, validate acc = 0.7097916666666667, loss
= 0.5189181284108003
step 104000: train acc = 0.7155416666666666, validate acc = 0.71604166666
66667, loss = 2.164414997270675
step 106000: train acc = 0.72175, validate acc = 0.72, loss = 0.169892133
3804399
```

In [90]:
```
plt.plot(g.loss_steps, g.losses)
plt.show()
```



In [91]:
```
g.update_weights(106000)
g.train(train_x, train_y, val_x, val_y, lr=0.001, lam=0.001, max_steps=10000
```

```
step 106000: train acc = 0.7218333333333333, validate acc = 0.72083333333
33333, loss = 0.7390005879986374
step 108000: train acc = 0.7182916666666667, validate acc = 0.71979166666
66667, loss = 0.06620859966617908
step 110000: train acc = 0.7240416666666667, validate acc = 0.729375, los
s = 0.3444422937133155
step 112000: train acc = 0.7272416666666667, validate acc = 0.73208333333
33333, loss = 0.207465987444473
step 114000: train acc = 0.7282166666666666, validate acc = 0.72458333333
33334, loss = 0.8483030042178319
step 116000: train acc = 0.733075, validate acc = 0.7372916666666667, los
s = 0.8740848528713372
step 118000: train acc = 0.7312416666666667, validate acc = 0.73354166666
66666, loss = 1.8633922417352156
step 120000: train acc = 0.7330916666666667, validate acc = 0.74166666666
66667, loss = 0.23891038921680305
step 122000: train acc = 0.7379333333333333, validate acc = 0.74, loss =
 0.1093527872271058
step 124000: train acc = 0.736, validate acc = 0.74, loss = 0.21999283461
931796
step 126000: train acc = 0.7395916666666666, validate acc = 0.74291666666
66667, loss = 0.20517781581401698
step 128000: train acc = 0.7407083333333333, validate acc = 0.74104166666
66667, loss = 0.35693790941223286
step 130000: train acc = 0.7457916666666666, validate acc = 0.74291666666
66667, loss = 0.11176174547976127
step 132000: train acc = 0.7487166666666667, validate acc = 0.74791666666
66667, loss = 1.796725829568693
step 134000: train acc = 0.7519, validate acc = 0.754375, loss = 0.955923
168115346
step 136000: train acc = 0.752625, validate acc = 0.7504166666666666, los
s = 0.06549863646423756
step 138000: train acc = 0.7581083333333334, validate acc = 0.75645833333
33333, loss = 2.9343599700534124
step 140000: train acc = 0.7594833333333333, validate acc = 0.756875, los
s = 1.828567321089299
step 142000: train acc = 0.7597833333333334, validate acc = 0.761875, los
s = 0.1890383250844607
step 144000: train acc = 0.7622833333333333, validate acc = 0.76354166666
66667, loss = 1.1668434867562092
step 146000: train acc = 0.7657166666666667, validate acc = 0.76729166666
66667, loss = 0.123360070116587
step 148000: train acc = 0.7657916666666666, validate acc = 0.76520833333
33334, loss = 1.3049155982668785
step 150000: train acc = 0.7657166666666667, validate acc = 0.76708333333
33333, loss = 2.273695036790842
step 152000: train acc = 0.7693166666666666, validate acc = 0.76541666666
66666, loss = 0.9675705158738774
step 154000: train acc = 0.774, validate acc = 0.7733333333333333, loss =
3.4139257373524163
step 156000: train acc = 0.7759, validate acc = 0.7745833333333333, loss
 = 0.06680172690424178
step 158000: train acc = 0.7765416666666667, validate acc = 0.77833333333
33333, loss = 0.33891341058882446
```

```
step 160000: train acc = 0.7733583333333334, validate acc = 0.77520833333
33334, loss = 0.880001202442573
step 162000: train acc = 0.7804833333333333, validate acc = 0.780625, los
s = 0.569501138582996
step 164000: train acc = 0.7836833333333333, validate acc = 0.78375, loss
= 1.4255584180505252
step 166000: train acc = 0.7832166666666667, validate acc = 0.78145833333
33334, loss = 0.71986683408298
step 168000: train acc = 0.7860333333333334, validate acc = 0.78375, loss
= 0.16792151383776846
step 170000: train acc = 0.7831833333333333, validate acc = 0.78166666666
66666, loss = 0.4176867999223109
step 172000: train acc = 0.7897, validate acc = 0.7858333333333334, loss
 = 1.0090603803126292
step 174000: train acc = 0.7898166666666666, validate acc = 0.7875, loss
 = 0.9463176915175089
step 176000: train acc = 0.7905666666666666, validate acc = 0.7875, loss
 = 0.03160247955730109
step 178000: train acc = 0.7896166666666666, validate acc = 0.79041666666
66667, loss = 0.8543417053267914
step 180000: train acc = 0.79535, validate acc = 0.791875, loss = 0.04336
3645302264775
step 182000: train acc = 0.7945416666666667, validate acc = 0.79604166666
66666, loss = 0.8522159147505011
step 184000: train acc = 0.7955583333333334, validate acc = 0.7925, loss
 = 0.33205768697680543
step 186000: train acc = 0.7974083333333334, validate acc = 0.79625, loss
= 0.05276922949437596
step 188000: train acc = 0.80015, validate acc = 0.806875, loss = 0.20867
641092617778
step 190000: train acc = 0.8015333333333333, validate acc = 0.800625, los
s = 0.059307485448253705
step 192000: train acc = 0.8009083333333333, validate acc = 0.799375, los
s = 2.2596058437592945
step 194000: train acc = 0.8007916666666667, validate acc = 0.79666666666
66666, loss = 1.9821854213294376
step 196000: train acc = 0.8037583333333334, validate acc = 0.80458333333
33333, loss = 1.7277997776874194
step 198000: train acc = 0.8056916666666667, validate acc = 0.80791666666
66666, loss = 2.239466943057001
step 200000: train acc = 0.8071583333333333, validate acc = 0.80729166666
66666, loss = 1.276669680601104
step 202000: train acc = 0.8111083333333333, validate acc = 0.81, loss =
 0.16716834631453553
step 204000: train acc = 0.8138666666666666, validate acc = 0.81229166666
66666, loss = 1.4931625125878054
```

In [92]:
```
g.update_weights(204000)
g.train(train_x, train_y, val_x, val_y, lr=0.001, lam=0.001, max_steps=10000
```

```
step 204000: train acc = 0.8138916666666667, validate acc = 0.81270833333
33334, loss = 0.4051470298607587
step 206000: train acc = 0.8098416666666667, validate acc = 0.81375, loss
= 0.058438839411486185
step 208000: train acc = 0.8120083333333333, validate acc = 0.814375, los
s = 0.6358518547468712
step 210000: train acc = 0.81195, validate acc = 0.81625, loss = 0.121275
24156633171
step 212000: train acc = 0.8120166666666667, validate acc = 0.81083333333
33333, loss = 0.06408477371208651
step 214000: train acc = 0.81765, validate acc = 0.8189583333333333, loss
= 0.2354432769572472
step 216000: train acc = 0.8156083333333334, validate acc = 0.81604166666
66667, loss = 0.16240647986737008
step 218000: train acc = 0.8163333333333334, validate acc = 0.815, loss =
0.30993449805764817
step 220000: train acc = 0.8186, validate acc = 0.8122916666666666, loss
 = 0.08789309666580634
step 222000: train acc = 0.81865, validate acc = 0.8202083333333333, loss
= 0.33523823120305324
step 224000: train acc = 0.82115, validate acc = 0.8189583333333333, loss
= 0.5691448263197444
step 226000: train acc = 0.819975, validate acc = 0.8164583333333333, los
s = 1.0915141728705398
step 228000: train acc = 0.8219666666666666, validate acc = 0.82083333333
33333, loss = 1.1004066933334382
step 230000: train acc = 0.823525, validate acc = 0.819375, loss = 2.7245
76322386624
step 232000: train acc = 0.8238, validate acc = 0.8202083333333333, loss
 = 0.326306294426441
step 234000: train acc = 0.82425, validate acc = 0.8227083333333334, loss
= 0.22891196858311652
step 236000: train acc = 0.8246666666666667, validate acc = 0.82416666666
66667, loss = 0.7910601134289318
step 238000: train acc = 0.8254833333333333, validate acc = 0.823125, los
s = 0.3377943781184202
step 240000: train acc = 0.8241416666666667, validate acc = 0.82729166666
66666, loss = 0.742791491257519
step 242000: train acc = 0.8255833333333333, validate acc = 0.83104166666
66667, loss = 0.5040892868012119
step 244000: train acc = 0.8318833333333333, validate acc = 0.82791666666
66666, loss = 0.421159606539701
step 246000: train acc = 0.8250333333333333, validate acc = 0.82291666666
66666, loss = 0.46572749030497884
step 248000: train acc = 0.8245333333333333, validate acc = 0.82520833333
33333, loss = 0.34220272876398905
step 250000: train acc = 0.8333, validate acc = 0.8254166666666667, loss
 = 0.03223097915226828
step 252000: train acc = 0.8326916666666667, validate acc = 0.82791666666
66666, loss = 1.6874495297725103
step 254000: train acc = 0.82965, validate acc = 0.8314583333333333, loss
= 0.685126895255392
step 256000: train acc = 0.835225, validate acc = 0.8327083333333334, los
s = 0.18513291474653698
```

```
step 258000: train acc = 0.83415, validate acc = 0.8310416666666667, loss
= 2.00390492850292
step 260000: train acc = 0.8333416666666666, validate acc = 0.83291666666
66666, loss = 0.382272375250323
step 262000: train acc = 0.8327416666666667, validate acc = 0.83229166666
66667, loss = 0.16255232908602613
step 264000: train acc = 0.8354, validate acc = 0.8345833333333333, loss
 = 0.3187836101264035
step 266000: train acc = 0.8368416666666667, validate acc = 0.83208333333
33333, loss = 7.199189086104267
step 268000: train acc = 0.8366416666666666, validate acc = 0.833125, los
s = 0.013751610501707187
step 270000: train acc = 0.8397083333333333, validate acc = 0.83770833333
33334, loss = 2.4879815782639665
step 272000: train acc = 0.8383916666666666, validate acc = 0.83958333333
33333, loss = 0.4028459488933565
step 274000: train acc = 0.840575, validate acc = 0.8395833333333333, los
s = 0.9540127622840072
step 276000: train acc = 0.8371416666666667, validate acc = 0.83625, loss
= 1.5073287775728452
step 278000: train acc = 0.8355666666666667, validate acc = 0.83583333333
33333, loss = 1.6763938097786346
step 280000: train acc = 0.8402833333333334, validate acc = 0.836875, los
s = 0.2867473966535351
step 282000: train acc = 0.84125, validate acc = 0.8375, loss = 0.2035779
2467450864
step 284000: train acc = 0.8378666666666666, validate acc = 0.83979166666
66667, loss = 2.0615777583250567
step 286000: train acc = 0.8405166666666667, validate acc = 0.83833333333
33334, loss = 0.05589948820871933
step 288000: train acc = 0.8411583333333333, validate acc = 0.83729166666
66667, loss = 5.77784685631859
step 290000: train acc = 0.8437666666666667, validate acc = 0.84020833333
33333, loss = 2.6499848317021493
step 292000: train acc = 0.841675, validate acc = 0.8410416666666667, los
s = 0.24958570910692635
step 294000: train acc = 0.8423666666666667, validate acc = 0.84270833333
33333, loss = 0.0449350414974952
step 296000: train acc = 0.842625, validate acc = 0.841875, loss = 0.3298
4147862389157
step 298000: train acc = 0.842425, validate acc = 0.8370833333333333, los
s = 0.1054164667533456
step 300000: train acc = 0.84535, validate acc = 0.8464583333333333, loss
= 0.2885330720227662
step 302000: train acc = 0.8458416666666667, validate acc = 0.84520833333
33333, loss = 0.44741153856653887
```

In [93]:
```
g.update_weights(302000)
g.train(train_x, train_y, val_x, val_y, lr=0.001, lam=0.001, max_steps=50000
```

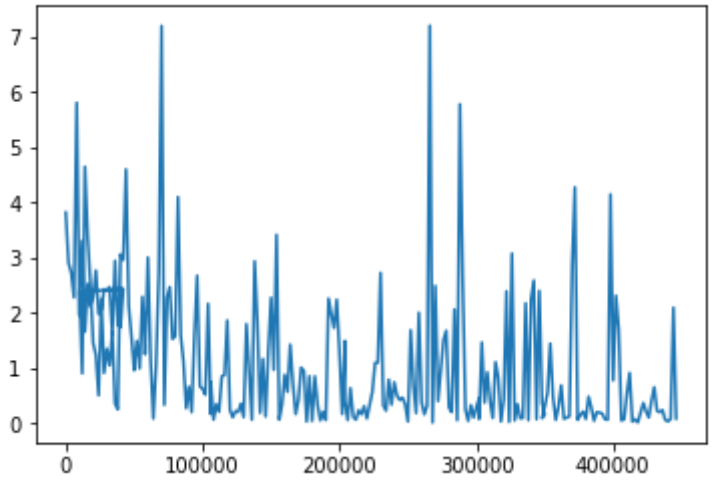
```
step 302000: train acc = 0.8458833333333333, validate acc = 0.845, loss =
0.07808695257481907
step 304000: train acc = 0.8467, validate acc = 0.8439583333333334, loss
 = 1.464771530522225
step 306000: train acc = 0.8468, validate acc = 0.8422916666666667, loss
 = 0.37413649843026187
step 308000: train acc = 0.84745, validate acc = 0.8445833333333334, loss
= 0.9242776101453914
step 310000: train acc = 0.843575, validate acc = 0.8454166666666667, los
s = 0.4001388010740366
step 312000: train acc = 0.8468, validate acc = 0.8464583333333333, loss
 = 0.09620066753392754
step 314000: train acc = 0.8485583333333333, validate acc = 0.84791666666
66667, loss = 1.1079551891894073
step 316000: train acc = 0.8497833333333333, validate acc = 0.85020833333
33333, loss = 0.8261110665439763
step 318000: train acc = 0.8501833333333333, validate acc = 0.84875, loss
= 0.03335114843106739
step 320000: train acc = 0.8462916666666667, validate acc = 0.84041666666
66667, loss = 0.4325780568063292
step 322000: train acc = 0.8510916666666667, validate acc = 0.84729166666
66667, loss = 2.3909510928334132
step 324000: train acc = 0.85205, validate acc = 0.85125, loss = 0.024820
50024778321
step 326000: train acc = 0.8525916666666666, validate acc = 0.85104166666
66667, loss = 3.074239910919517
step 328000: train acc = 0.850075, validate acc = 0.85, loss = 0.05162404
644943425
step 330000: train acc = 0.8520416666666667, validate acc = 0.85083333333
33333, loss = 0.34517637214443514
step 332000: train acc = 0.8495416666666666, validate acc = 0.85, loss =
 0.09253469726217617
step 334000: train acc = 0.8511916666666667, validate acc = 0.85354166666
66666, loss = 0.09579485854921381
step 336000: train acc = 0.8549583333333334, validate acc = 0.849375, los
s = 2.1743062952617636
step 338000: train acc = 0.846725, validate acc = 0.8429166666666666, los
s = 0.05666893159613195
step 340000: train acc = 0.8538, validate acc = 0.8502083333333333, loss
 = 2.2482066336034197
step 342000: train acc = 0.8521666666666666, validate acc = 0.85104166666
66667, loss = 2.58740159967789
step 344000: train acc = 0.853625, validate acc = 0.84875, loss = 0.07429
752723675889
step 346000: train acc = 0.8533333333333334, validate acc = 0.85, loss =
 2.3994663746592173
step 348000: train acc = 0.8552083333333333, validate acc = 0.8525, loss
 = 0.36462991578469456
step 350000: train acc = 0.8507416666666666, validate acc = 0.84833333333
33334, loss = 0.16094804977413465
```

In [94]:
```
g.update_weights(348000)
g.train(train_x, train_y, val_x, val_y, lr=1e-4, lam=0.001, max_steps=100000
```

```
step 348000: train acc = 0.8552083333333333, validate acc = 0.8525, loss
 = 0.09290991384860402
step 350000: train acc = 0.8564583333333333, validate acc = 0.85166666666
66667, loss = 0.30203236605305O7
step 352000: train acc = 0.8568333333333333, validate acc = 0.853125, los
s = 0.5551241014943079
step 354000: train acc = 0.8580333333333333, validate acc = 0.855, loss =
1.441946045079818
step 356000: train acc = 0.858975, validate acc = 0.8564583333333333, los
s = 0.4799132729760404
step 358000: train acc = 0.8594083333333333, validate acc = 0.85354166666
66666, loss = 0.061937561121453086
step 360000: train acc = 0.8601166666666666, validate acc = 0.85583333333
33333, loss = 0.2974307076112502
step 362000: train acc = 0.8609666666666667, validate acc = 0.85708333333
33333, loss = 0.6877345618658223
step 364000: train acc = 0.8615333333333334, validate acc = 0.85833333333
33333, loss = 0.08180936318405577
step 366000: train acc = 0.8625166666666667, validate acc = 0.85729166666
66667, loss = 0.1057837589234257
step 368000: train acc = 0.86265, validate acc = 0.85875, loss = 0.126629
31840125996
step 370000: train acc = 0.862475, validate acc = 0.85875, loss = 2.85395
46040600223
step 372000: train acc = 0.8629666666666667, validate acc = 0.85916666666
66666, loss = 4.272554734843437
step 374000: train acc = 0.8636416666666666, validate acc = 0.858125, los
s = 0.06407904274910421
step 376000: train acc = 0.8639833333333333, validate acc = 0.85833333333
33333, loss = 0.14879291007839857
step 378000: train acc = 0.8639916666666667, validate acc = 0.856875, los
s = 0.2142431197848158
step 380000: train acc = 0.8644916666666667, validate acc = 0.8575, loss
 = 0.07886849693799707
step 382000: train acc = 0.8650583333333334, validate acc = 0.85833333333
33333, loss = 0.48223870272169783
step 384000: train acc = 0.8655666666666667, validate acc = 0.85791666666
66667, loss = 0.28527576653553266
step 386000: train acc = 0.8656, validate acc = 0.8595833333333334, loss
 = 0.046180558766670265
step 388000: train acc = 0.8654666666666667, validate acc = 0.85854166666
66666, loss = 0.20055429335714167
step 390000: train acc = 0.8660666666666667, validate acc = 0.85854166666
66666, loss = 0.19358348160065475
step 392000: train acc = 0.8660666666666667, validate acc = 0.85979166666
66666, loss = 0.1754037961364632
step 394000: train acc = 0.8661916666666667, validate acc = 0.85854166666
66666, loss = 0.07367533749240725
step 396000: train acc = 0.8662666666666666, validate acc = 0.859375, los
s = 0.06638828069528999
step 398000: train acc = 0.8660916666666667, validate acc = 0.86083333333
33333, loss = 4.145237346908687
step 400000: train acc = 0.866025, validate acc = 0.8597916666666666, los
s = 0.7764314249378548
```

```
step 402000: train acc = 0.866475, validate acc = 0.8602083333333334, los
s = 2.3106538438476587
step 404000: train acc = 0.8663083333333333, validate acc = 0.85916666666
66666, loss = 1.7127530602102738
step 406000: train acc = 0.866575, validate acc = 0.860625, loss = 0.0508
3452176535806
step 408000: train acc = 0.8668833333333333, validate acc = 0.86229166666
66667, loss = 0.07272885397645626
step 410000: train acc = 0.8673666666666666, validate acc = 0.85729166666
66667, loss = 0.5329076358044369
step 412000: train acc = 0.8668166666666667, validate acc = 0.86104166666
66667, loss = 0.9134487632917446
step 414000: train acc = 0.8675166666666667, validate acc = 0.86145833333
33333, loss = 0.03182869043892151
step 416000: train acc = 0.8675, validate acc = 0.8608333333333333, loss
 = 0.07268649759146892
step 418000: train acc = 0.8674083333333333, validate acc = 0.861875, los
s = 0.01609598373744728
step 420000: train acc = 0.8669416666666667, validate acc = 0.85854166666
66666, loss = 0.18520887148440937
step 422000: train acc = 0.8671916666666667, validate acc = 0.86145833333
33333, loss = 0.36828944931613944
step 424000: train acc = 0.8676166666666667, validate acc = 0.85895833333
33334, loss = 0.21799722278706174
step 426000: train acc = 0.8674166666666666, validate acc = 0.85979166666
66666, loss = 0.1056803995024262
step 428000: train acc = 0.8679083333333333, validate acc = 0.86, loss =
 0.4069456180746932
step 430000: train acc = 0.8676583333333333, validate acc = 0.86020833333
33334, loss = 0.6531951636487293
step 432000: train acc = 0.8679, validate acc = 0.8604166666666667, loss
 = 0.22779741883151428
step 434000: train acc = 0.86755, validate acc = 0.8610416666666667, loss
= 0.2015035229602315
step 436000: train acc = 0.867225, validate acc = 0.8608333333333333, los
s = 0.2401337332926879
step 438000: train acc = 0.868125, validate acc = 0.8635416666666667, los
s = 0.061614587819352404
step 440000: train acc = 0.868175, validate acc = 0.861875, loss = 0.0355
5303614811221
step 442000: train acc = 0.8681833333333333, validate acc = 0.8625, loss
 = 0.07973556667346243
step 444000: train acc = 0.8690416666666667, validate acc = 0.861875, los
s = 2.094433877028979
step 446000: train acc = 0.8691166666666666, validate acc = 0.86104166666
66667, loss = 0.08625704109885025
```

In [95]:
```
plt.plot(g.loss_steps, g.losses)
plt.show()
```



In [97]:
```
g.update_weights(438000)
g.train(train_x, train_y, val_x, val_y, lr=1e-4, lam=0.001, max_steps=20000,
```

```
step 438000: train acc = 0.8680916666666667, validate acc = 0.86354166666
66667, loss = 1.3905679202807977
step 440000: train acc = 0.868425, validate acc = 0.8610416666666667, los
s = 0.3433082875399257
step 442000: train acc = 0.868775, validate acc = 0.8625, loss = 0.085084
6977937013
step 444000: train acc = 0.8684166666666666, validate acc = 0.863125, los
s = 0.9893703166294514
step 446000: train acc = 0.8686083333333333, validate acc = 0.8625, loss
 = 1.4393931144068401
step 448000: train acc = 0.8689333333333333, validate acc = 0.86125, loss
= 0.58293725136665
step 450000: train acc = 0.869075, validate acc = 0.8616666666666667, los
s = 0.06588222370307836
step 452000: train acc = 0.8691083333333334, validate acc = 0.8625, loss
 = 0.024341330191307346
step 454000: train acc = 0.8687916666666666, validate acc = 0.85979166666
66666, loss = 0.37682573268677977
step 456000: train acc = 0.8692833333333333, validate acc = 0.86041666666
66667, loss = 0.17239669775356486
```

3. Visualization

In [39]:
```
g = Graph()
g.update_weights(438000)
z = g.test(val_x)[1]
z = np.argmax(z, axis=1) + 1
y = np.argmax(val_y, axis=1) + 1
a = z==y
a[:15]
```

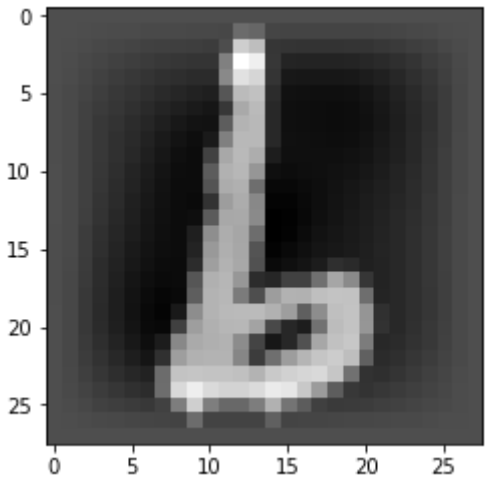
Out[39]:
```
array([False,  True, False, False,  True, False,  True,  True,  True,
       False,  True,  True,  True,  True,  True], dtype=bool)
```

In [51]:
```
def plot_image(i):
    image = val_x[i][:-1].reshape((28, 28))
    print("predicted: {}, true value: {}".format(z[i], y[i]))
    plt.imshow(image, cmap='gray')
    plt.show()
```

Correct Images

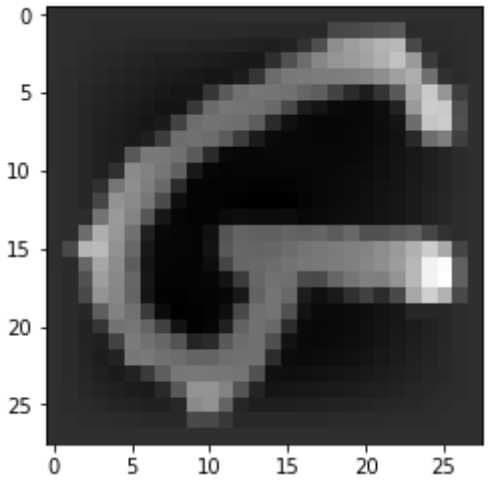
In [52]:
```
plot_image(1)
```

```
predicted: 2, true value: 2
```



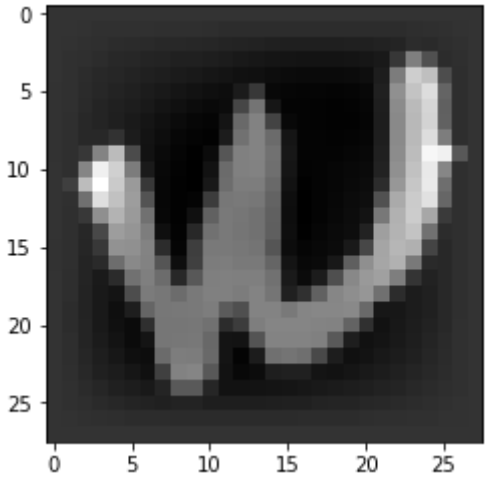
In [53]:
```
plot_image(4)
```

```
predicted: 7, true value: 7
```



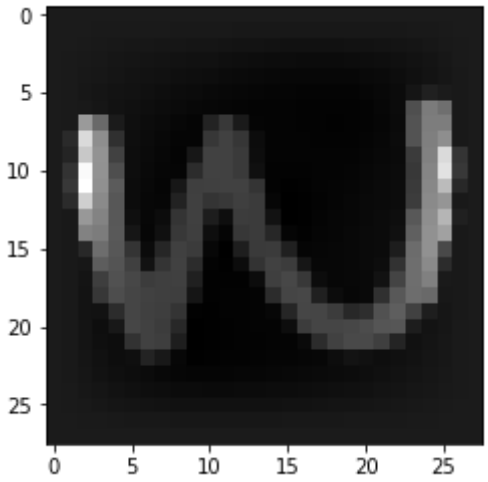
In [54]:
```
plot_image(6)
```

```
predicted: 23, true value: 23
```



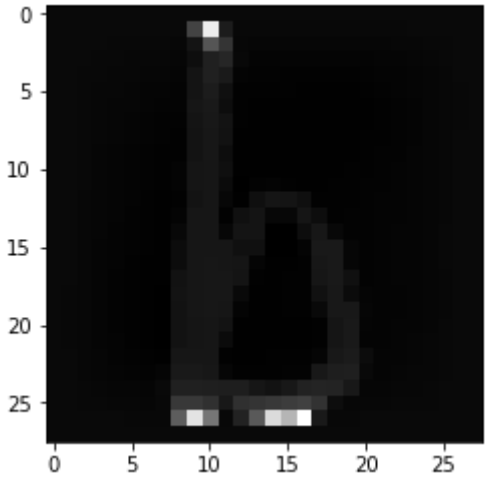
In [55]:
```
plot_image(7)
```

```
predicted: 23, true value: 23
```



In [56]:
```
plot_image(8)
```

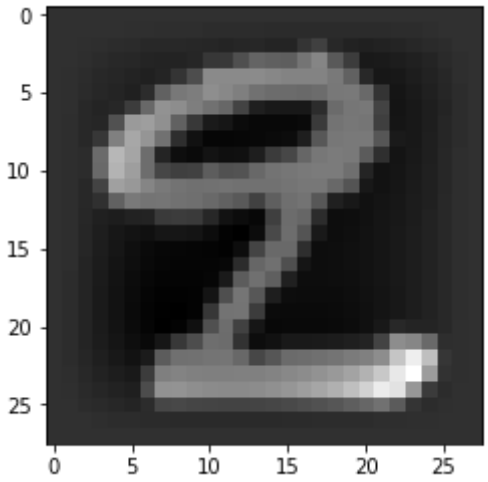
```
predicted: 2, true value: 2
```



Wrong Images

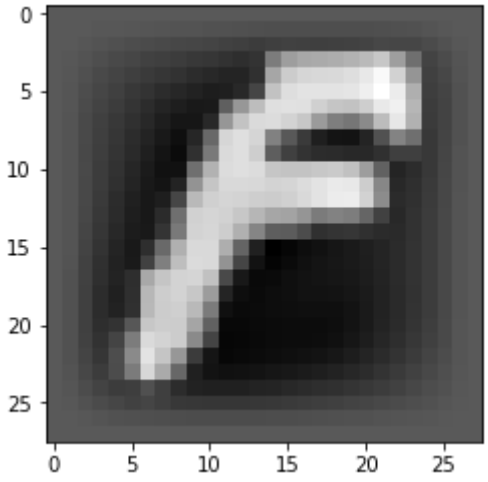
In [57]:
```
plot_image(0)
```

```
predicted: 26, true value: 17
```



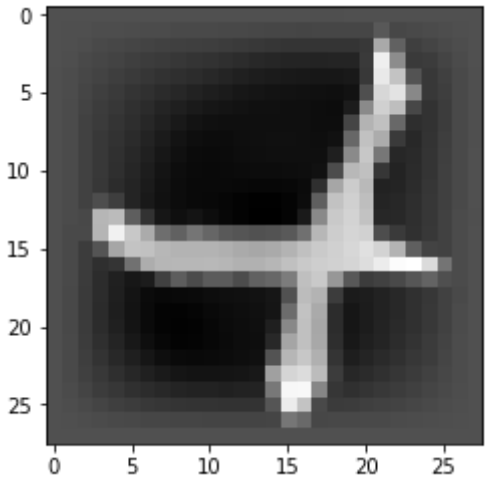
In [58]:
```
plot_image(2)
```

```
predicted: 16, true value: 6
```



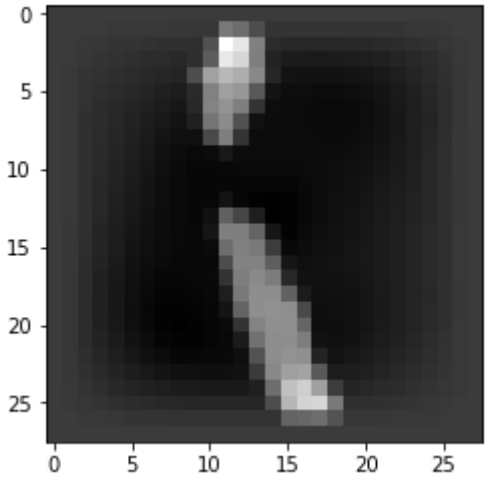
In [59]:
```
plot_image(3)
```

```
predicted: 4, true value: 20
```



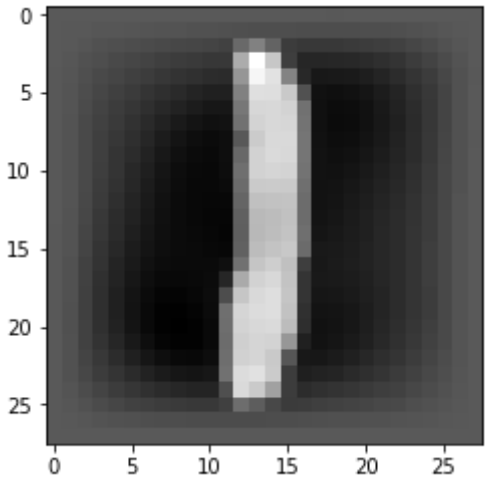
In [60]:
```
plot_image(5)
```

```
predicted: 9, true value: 10
```



In [61]:
```
plot_image(9)
```

```
predicted: 12, true value: 9
```



4. Bells and Whistles

Strategies I used:

- Increase number of hidden units to 400
- Decrease learning rate as the validation accuracy converges
- Add L2 regularization
- During preprocessing, normalize, shift and rotate training data
- Save weights as checkpoints regularly
- stop the training when validation accuracy stop increasing

In [14]:
```
g = Graph()
g.update_weights(438000)
y = g.test(test_x)[1]
y = np.argmax(y, axis=1)
y = y+1
```

In [18]:
```
df = pd.DataFrame(data = y, columns=["Category"])
df.index += 1
df.index.name = "Id"
df.to_csv("result.csv")
```