In [1]:
```
%matplotlib inline
%config InlineBackend.figure_format='retina'

import csv
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

from matplotlib import animation, rc, rcParams
rc('font',**{'family':'serif','serif':['DejaVu Sans']})
rc('text', usetex=True)
rcParams['text.latex.preamble']=[r"\usepackage{amsmath}"]
```

In [2]:
```
def compact_notation(X):
    """ Append 1 to X """
    return np.hstack([np.ones([X.shape[0], 1]), X])

def read_data(path):
    """Read data from a specified path
    Returns:
        X: in compact notation
        Y: a matrix of [Nsamples, 1] where values are from 0 to 9
    """

    loaded_txt = np.loadtxt(path, delimiter=",")
    loaded_imgs = np.asfarray(loaded_txt[:, :-1])
    loaded_labels = np.asfarray(loaded_txt[:, -1:]) - 1
    return compact_notation(loaded_imgs).T, loaded_labels.astype(int)

trainX, trainY = read_data('MNIST_data/MNIST_train_data.csv')
testX, testY   = read_data('MNIST_data/MNIST_test_data.csv')
```

In [3]:

```
def one_versus_all(Y, value):
    """
    generate label Yout,
    where Y == value then Yout would be 1
    otherwise Yout would be -1
    """
    return np.where(Y == value, 1, -1)

def tiny_exp(u):
    s = np.argwhere(u > 100)
    t = np.argwhere(u < -100)
    u[s] = 0
    u[t] = 0
    u = np.exp(u)
    u[t] = 1
    return u

def sigmoid(t):
    """ That young sigmoid function. """
    return 1/(1 + tiny_exp(-t))

def accuracy(w, X, y):
    """ Accuracy of the model """
    # predictions
    all_preds = X.T @ w
    y_preds = np.argmax(all_preds , axis=1).reshape(y.shape)

    # this is equation 4.62
    accuracy = 1 - (1 /y.size) * np.sum(np.sum(np.where(y_preds == y, 0, 1)))

    correct = np.sum(np.where(y_preds == y, 1, 0))
    easy_accuracy = correct / y.size
    return accuracy

def softmax_grad(w, X, y):
    """Return gradient of multiclass softmax
    Utilize softmax function below"""
    t = -y * np.dot(X.T, w)
    r = sigmoid(t) * y
    grad = np.dot(-1.0 * X, r)
    return grad

def gradient_descent(grad, w0, *args, **kwargs):
    """ Gradient descent """
    max_iter = 100
    alpha = 0.001
    eps = 1e-5
    w = w0
    iters = 0
    while True:
        gradient = grad(w, *args)
        cur_eps = np.linalg.norm(gradient)
        w = w - ((alpha) * gradient)

        if iters > max_iter or cur_eps < eps:
            iters -= 1
            print(">> Class {} : Iteration = {}, eps = {}\n".format(i, iters, cur_eps))
            break
        if iters % 5 == 0:
            print(">> Class {} : Iteration = {}, eps = {}".format(i, iters, cur_eps), end="\r", flush=True)
        iters += 1
    return w

def accuracy_plot(w, X, y, label='Training Data', size=12):
    all_preds = X.T @ w
    y_preds = np.argmax(all_preds , axis=1).reshape(y.shape)

    _pltty, _splttys = plt.subplots(2,6, figsize=(12,4))
    suptitle = _pltty.suptitle('{}'.format(label), y=1.05, fontsize=16, fontweight='bold')
    _splttys = _splttys.ravel()
    for splt, indx in zip(_splttys, np.random.randint(low=0, high=len(X), size=size)):
        splt.set_title('Actual: {}, Prediction: {}'.format(y[indx][0], y_preds[indx][0]))
        c_map = 'YlGn'
        if y[indx][0] != y_preds[indx][0]:
            c_map ='Reds'
        splt.imshow(X.T[indx][1:].reshape(28, 28).T, cmap=c_map)#'bone_r')
        splt.axis('off')
    _pltty.tight_layout()
    plt.savefig(label + '.png', bbox_extra_artists=(suptitle,), bbox_inches='tight', dpi=500)
# # iterate through each classifier.
# n_feature = 785
# n_class = 10
# OvA = np.zeros((n_feature, n_class))
# for i in range(n_class):
#     print('> Training for classifier for digit: {}'.format(str(i)))
#     w0 = np.random.rand(n_feature, 1)
#     OvA[:, i:i+1] = gradient_descent(softmax_grad, w0, trainX, one_versus_all(trainY, i))
```

In [4]:

```
from multiprocessing import Pool

n_class = 10
n_feature = 785

def gradient_descent_mp(i, X=trainX, grad=softmax_grad, max_iter=2500):
    """ Gradient descent """
    print('> Training for classifier for digit: {}'.format(str(i)))
    y = one_versus_all(trainY, i)
    max_iter = max_iter
    alpha = 0.001
    eps = 1e-5
    w = np.random.rand(n_feature, 1)
    iters = 0
    while True:
        gradient = grad(w, X, y)
        cur_eps = np.linalg.norm(gradient)
        w = w - ((alpha) * gradient)

        if iters % 100 == 0:
            print(">> Class {} Iteration = {}, eps = {}".format(i, iters, cur_eps))

        if iters > max_iter or cur_eps < eps:
            iters -= 1
            print(">> Class Stopped : {} Iteration = {}, eps = {}".format(i, iters, cur_eps))
            break
        iters += 1
    return w

with Pool(5) as p:
    result = p.map(gradient_descent_mp, [i for i in range(n_class)])


OvA = np.zeros((n_feature, n_class))
for i, res in enumerate(result):
    OvA[:, i:i+1] = res
training_acc = accuracy(OvA, trainX, trainY)
test_acc = accuracy(OvA,  testX,  testY)
print(r"Part(a): Accuracy for training set (N = 60,000) is: %f" % training_acc)
print(r"Part(b):     Accuracy for test set (N = 10,000) is: %f" % test_acc)
```

```
> Training for classifier for digit: 2
> Training for classifier for digit: 0
> Training for classifier for digit: 1
> Training for classifier for digit: 3
> Training for classifier for digit: 4
>> Class 0 Iteration = 0, eps = 63721.70204299138
>> Class 3 Iteration = 0, eps = 63358.86287874765
>> Class 1 Iteration = 0, eps = 62960.839440081414
>> Class 4 Iteration = 0, eps = 63792.756854958076
>> Class 2 Iteration = 0, eps = 63617.17356735969
>> Class 1 Iteration = 100, eps = 52.343969670450335
>> Class 0 Iteration = 100, eps = 72.71917598437747
>> Class 3 Iteration = 100, eps = 2167.646792920893
>> Class 4 Iteration = 100, eps = 186.83116927383895
>> Class 2 Iteration = 100, eps = 1026.2623793285832
>> Class 3 Iteration = 200, eps = 1606.2888157046887
>> Class 0 Iteration = 200, eps = 43.0837582094525
>> Class 1 Iteration = 200, eps = 33.73835046373331
>> Class 4 Iteration = 200, eps = 43.9535956484108
>> Class 2 Iteration = 200, eps = 162.6687370113675
>> Class 3 Iteration = 300, eps = 1312.4478346268197
>> Class 0 Iteration = 300, eps = 31.580444482102127
>> Class 1 Iteration = 300, eps = 26.680907983059473
>> Class 2 Iteration = 300, eps = 34.328213389262224
>> Class 4 Iteration = 300, eps = 33.89017012124615
>> Class 3 Iteration = 400, eps = 1093.671280916831
>> Class 1 Iteration = 400, eps = 22.546953189819504
>> Class 0 Iteration = 400, eps = 25.204655441366107
>> Class 4 Iteration = 400, eps = 28.414484034136763
>> Class 2 Iteration = 400, eps = 28.26746463371671
>> Class 3 Iteration = 500, eps = 910.8165444743089
>> Class 1 Iteration = 500, eps = 19.79070462795235
>> Class 0 Iteration = 500, eps = 21.151842413635475
>> Class 2 Iteration = 500, eps = 24.309390949240985
>> Class 4 Iteration = 500, eps = 24.795430295428908
>> Class 3 Iteration = 600, eps = 747.4268106277378
>> Class 1 Iteration = 600, eps = 17.80298413962927
>> Class 0 Iteration = 600, eps = 18.357027972268458
>> Class 2 Iteration = 600, eps = 21.495757703625234
>> Class 4 Iteration = 600, eps = 22.186559403337704
>> Class 3 Iteration = 700, eps = 593.7513243625391
>> Class 1 Iteration = 700, eps = 16.288706438389422
>> Class 0 Iteration = 700, eps = 16.318476275079753
>> Class 4 Iteration = 700, eps = 20.201140149618332
>> Class 2 Iteration = 700, eps = 19.386534665966902
>> Class 1 Iteration = 800, eps = 15.087699121876316
>> Class 3 Iteration = 800, eps = 442.76372191606544
>> Class 0 Iteration = 800, eps = 14.768511471494605
>> Class 4 Iteration = 800, eps = 18.63085415286049
>> Class 2 Iteration = 800, eps = 17.74307217437201
>> Class 3 Iteration = 900, eps = 291.1626050751128
>> Class 1 Iteration = 900, eps = 14.105686523132865
>> Class 0 Iteration = 900, eps = 13.551540007080662
>> Class 2 Iteration = 900, eps = 16.423035938628573
>> Class 4 Iteration = 900, eps = 17.352510098781625
>> Class 3 Iteration = 1000, eps = 148.62972236682864
>> Class 1 Iteration = 1000, eps = 13.283535029883868
>> Class 0 Iteration = 1000, eps = 12.571130581312174
>> Class 2 Iteration = 1000, eps = 15.336202296670182
>> Class 4 Iteration = 1000, eps = 16.288220103107484
>> Class 3 Iteration = 1100, eps = 48.66503582141033
>> Class 1 Iteration = 1100, eps = 12.582180643854874
>> Class 2 Iteration = 1100, eps = 14.422821406740077
>> Class 0 Iteration = 1100, eps = 11.764447056100332
>> Class 4 Iteration = 1100, eps = 15.386093073067665
>> Class 3 Iteration = 1200, eps = 13.976181803466751
>> Class 1 Iteration = 1200, eps = 11.974688863988991
>> Class 0 Iteration = 1200, eps = 11.088859029247201
>> Class 2 Iteration = 1200, eps = 13.641936060209243
>> Class 4 Iteration = 1200, eps = 14.610066946002174
>> Class 3 Iteration = 1300, eps = 11.201516832590395
>> Class 1 Iteration = 1300, eps = 11.441816085563639
>> Class 0 Iteration = 1300, eps = 10.514468989338212
>> Class 4 Iteration = 1300, eps = 13.934201965917875
>> Class 2 Iteration = 1300, eps = 12.964642881613582
>> Class 3 Iteration = 1400, eps = 10.665079522365845
>> Class 1 Iteration = 1400, eps = 10.969399359068701
>> Class 0 Iteration = 1400, eps = 10.019719355492427
>> Class 4 Iteration = 1400, eps = 13.339303895092938
>> Class 2 Iteration = 1400, eps = 12.370015874415534
>> Class 3 Iteration = 1500, eps = 10.198934184637274
>> Class 1 Iteration = 1500, eps = 10.54675221496159
>> Class 0 Iteration = 1500, eps = 9.588695737885516
>> Class 4 Iteration = 1500, eps = 12.81083310693953
>> Class 2 Iteration = 1500, eps = 11.842550458979344
>> Class 3 Iteration = 1600, eps = 9.780165393244435
>> Class 1 Iteration = 1600, eps = 10.165640003524812
>> Class 0 Iteration = 1600, eps = 9.209409544102169
>> Class 4 Iteration = 1600, eps = 12.337557789352022
>> Class 2 Iteration = 1600, eps = 11.370514308708454
>> Class 3 Iteration = 1700, eps = 9.401401900036538
>> Class 1 Iteration = 1700, eps = 9.819602834070325
>> Class 0 Iteration = 1700, eps = 8.872669255504055
>> Class 4 Iteration = 1700, eps = 11.910656140014432
>> Class 2 Iteration = 1700, eps = 10.944857256855913
>> Class 3 Iteration = 1800, eps = 9.056824049867307
>> Class 1 Iteration = 1800, eps = 9.503494650378567
>> Class 0 Iteration = 1800, eps = 8.571318062704758
>> Class 2 Iteration = 1800, eps = 10.558475166338855
>> Class 4 Iteration = 1800, eps = 11.523099779195169
>> Class 3 Iteration = 1900, eps = 8.741735645585411
>> Class 1 Iteration = 1900, eps = 9.213161087481868
>> Class 0 Iteration = 1900, eps = 8.299706523099765
>> Class 4 Iteration = 1900, eps = 11.169219258484492
>> Class 2 Iteration = 1900, eps = 10.205702940719913
>> Class 3 Iteration = 2000, eps = 8.45229197398235
>> Class 1 Iteration = 2000, eps = 8.945209040557694
>> Class 0 Iteration = 2000, eps = 8.053320055973035
>> Class 2 Iteration = 2000, eps = 9.881958790450588
>> Class 4 Iteration = 2000, eps = 10.844391015815859
>> Class 3 Iteration = 2100, eps = 8.185308853256563
>> Class 1 Iteration = 2100, eps = 8.696838425924156
>> Class 0 Iteration = 2100, eps = 7.828510877598745
>> Class 2 Iteration = 2100, eps = 9.58349013524233
>> Class 4 Iteration = 2100, eps = 10.544807463368075
```

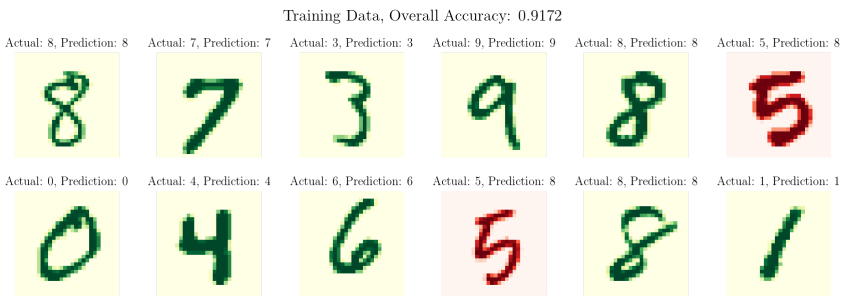
```
>> Class 3 Iteration = 2200, eps = 7.93812317976404
>> Class 1 Iteration = 2200, eps = 8.465717106831828
>> Class 0 Iteration = 2200, eps = 7.6223018718119535
>> Class 2 Iteration = 2200, eps = 9.307188957899442
>> Class 4 Iteration = 2200, eps = 10.267305295362533
>> Class 3 Iteration = 2300, eps = 7.708488382633529
>> Class 1 Iteration = 2300, eps = 8.249886411125564
>> Class 0 Iteration = 2300, eps = 7.432240942733154
>> Class 2 Iteration = 2300, eps = 9.050455392433106
>> Class 4 Iteration = 2300, eps = 10.009235390344603
>> Class 3 Iteration = 2400, eps = 7.494494555143092
>> Class 1 Iteration = 2400, eps = 8.047688742153452
>> Class 0 Iteration = 2400, eps = 7.256291394268165
>> Class 2 Iteration = 2400, eps = 8.811095346035923
>> Class 4 Iteration = 2400, eps = 9.76836295369955
>> Class 3 Iteration = 2500, eps = 7.294506653846369
>> Class Stopped : 3 Iteration = 2500, eps = 7.292572451767633
> Training for classifier for digit: 5
>> Class 5 Iteration = 0, eps = 64328.8826652332
>> Class 1 Iteration = 2500, eps = 7.857711417807655
>> Class Stopped : 1 Iteration = 2500, eps = 7.855869107790744
> Training for classifier for digit: 6
>> Class 0 Iteration = 2500, eps = 7.09274841306582
>> Class Stopped : 0 Iteration = 2500, eps = 7.091170576617158
>> Class 6 Iteration = 0, eps = 63705.55274160594
> Training for classifier for digit: 7
>> Class 7 Iteration = 0, eps = 63399.989494229645
>> Class 2 Iteration = 2500, eps = 8.58724251147603
>> Class Stopped : 2 Iteration = 2500, eps = 8.585076686692117
> Training for classifier for digit: 8
>> Class 4 Iteration = 2500, eps = 9.54278998570718
>> Class 8 Iteration = 0, eps = 63546.1344105567
>> Class Stopped : 4 Iteration = 2500, eps = 9.540605825189845
> Training for classifier for digit: 9
>> Class 9 Iteration = 0, eps = 63565.120248259744
>> Class 5 Iteration = 100, eps = 5603.787864931269
>> Class 6 Iteration = 100, eps = 68.64183851406293
>> Class 7 Iteration = 100, eps = 71.59332400276591
>> Class 8 Iteration = 100, eps = 5091.839721648438
>> Class 9 Iteration = 100, eps = 6311.917694925764
>> Class 5 Iteration = 200, eps = 3471.996863932489
>> Class 6 Iteration = 200, eps = 40.1024197060257
>> Class 7 Iteration = 200, eps = 44.01326342948707
>> Class 8 Iteration = 200, eps = 3766.914278525193
>> Class 9 Iteration = 200, eps = 3670.650759444759
>> Class 5 Iteration = 300, eps = 2950.3713207212713
>> Class 6 Iteration = 300, eps = 29.684827079543734
>> Class 7 Iteration = 300, eps = 32.52819216366873
>> Class 8 Iteration = 300, eps = 3374.355045516738
>> Class 9 Iteration = 300, eps = 3253.283095275879
>> Class 5 Iteration = 400, eps = 2685.192018138175
>> Class 6 Iteration = 400, eps = 24.020628260027905
>> Class 7 Iteration = 400, eps = 26.16283785546571
>> Class 9 Iteration = 400, eps = 3124.556700832254
>> Class 8 Iteration = 400, eps = 3162.224592025277
>> Class 5 Iteration = 500, eps = 2508.0401575692017
>> Class 6 Iteration = 500, eps = 20.38527032930269
>> Class 7 Iteration = 500, eps = 22.126760656813616
>> Class 9 Iteration = 500, eps = 2618.9166997764446
>> Class 8 Iteration = 500, eps = 3027.429872943509
>> Class 5 Iteration = 600, eps = 2373.199753386553
>> Class 6 Iteration = 600, eps = 17.828842863812184
>> Class 7 Iteration = 600, eps = 19.34991774869558
>> Class 9 Iteration = 600, eps = 3512.8798161626464
>> Class 8 Iteration = 600, eps = 2935.026356139503
>> Class 5 Iteration = 700, eps = 2263.502230171338
>> Class 6 Iteration = 700, eps = 15.922786603867236
>> Class 7 Iteration = 700, eps = 17.32612494841718
>> Class 9 Iteration = 700, eps = 3386.872689679067
>> Class 8 Iteration = 700, eps = 2868.4148529690374
>> Class 5 Iteration = 800, eps = 2170.8217226571423
>> Class 6 Iteration = 800, eps = 14.442609662125976
>> Class 7 Iteration = 800, eps = 15.784362496046661
>> Class 9 Iteration = 800, eps = 931.2034324959475
>> Class 8 Iteration = 800, eps = 2818.4183648381286
>> Class 5 Iteration = 900, eps = 2090.560531482198
>> Class 6 Iteration = 900, eps = 13.258108024798975
>> Class 7 Iteration = 900, eps = 14.567352351784212
>> Class 9 Iteration = 900, eps = 572.7229532339226
>> Class 8 Iteration = 900, eps = 2779.5782735026673
>> Class 5 Iteration = 1000, eps = 2019.783926639725
>> Class 6 Iteration = 1000, eps = 12.287939800865244
>> Class 7 Iteration = 1000, eps = 13.578397412861321
>> Class 9 Iteration = 1000, eps = 729.8913619700853
>> Class 8 Iteration = 1000, eps = 2748.500488043908
>> Class 5 Iteration = 1100, eps = 1956.4767562710977
>> Class 6 Iteration = 1100, eps = 11.478362505552195
>> Class 7 Iteration = 1100, eps = 12.755308873728188
>> Class 9 Iteration = 1100, eps = 1411.0550098074284
>> Class 8 Iteration = 1100, eps = 2723.0016184270175
>> Class 5 Iteration = 1200, eps = 1899.1924613510816
>> Class 6 Iteration = 1200, eps = 10.792298920127896
>> Class 7 Iteration = 1200, eps = 12.056569343612257
>> Class 9 Iteration = 1200, eps = 3509.513472817752
>> Class 8 Iteration = 1200, eps = 2701.6337994791443
>> Class 5 Iteration = 1300, eps = 1846.8600947668028
>> Class 7 Iteration = 1300, eps = 11.453553271006614
>> Class 6 Iteration = 1300, eps = 10.203278041326765
>> Class 9 Iteration = 1300, eps = 5689.473405742166
>> Class 8 Iteration = 1300, eps = 2683.408559226977
>> Class 5 Iteration = 1400, eps = 1798.6664994975058
>> Class 7 Iteration = 1400, eps = 10.925952761779769
>> Class 6 Iteration = 1400, eps = 9.691868995063167
>> Class 9 Iteration = 1400, eps = 4386.304390016219
>> Class 8 Iteration = 1400, eps = 2667.6316358288755
>> Class 5 Iteration = 1500, eps = 1753.9804271591513
>> Class 7 Iteration = 1500, eps = 10.45898770369251
>> Class 6 Iteration = 1500, eps = 9.243476800722679
>> Class 9 Iteration = 1500, eps = 2785.1901187288577
>> Class 8 Iteration = 1500, eps = 2653.8018966413174
>> Class 5 Iteration = 1600, eps = 1712.3023064394088
>> Class 6 Iteration = 1600, eps = 8.846923959520872
>> Class 7 Iteration = 1600, eps = 10.041650086382862
>> Class 9 Iteration = 1600, eps = 1620.8701128132848
>> Class 8 Iteration = 1600, eps = 2641.548212958721
```

```
>> Class 5 Iteration = 1700, eps = 1673.230231042134
>> Class 6 Iteration = 1700, eps = 8.493506629334187
>> Class 7 Iteration = 1700, eps = 9.665568311652045
>> Class 9 Iteration = 1700, eps = 1027.2809467031786
>> Class 8 Iteration = 1700, eps = 2630.5892382721404
>> Class 5 Iteration = 1800, eps = 1636.4363657951249
>> Class 6 Iteration = 1800, eps = 8.176348573411005
>> Class 7 Iteration = 1800, eps = 9.324254182854574
>> Class 9 Iteration = 1800, eps = 738.0022095030282
>> Class 8 Iteration = 1800, eps = 2620.707237428875
>> Class 5 Iteration = 1900, eps = 1601.65017367357
>> Class 6 Iteration = 1900, eps = 7.889948078761227
>> Class 7 Iteration = 1900, eps = 9.012592283799622
>> Class 8 Iteration = 1900, eps = 2611.730675271683
>> Class 9 Iteration = 1900, eps = 587.791903363495
>> Class 5 Iteration = 2000, eps = 1568.6462144722957
>> Class 6 Iteration = 2000, eps = 7.62985343771905
>> Class 7 Iteration = 2000, eps = 8.726486508934276
>> Class 9 Iteration = 2000, eps = 504.543086757529
>> Class 8 Iteration = 2000, eps = 2603.522358491851
>> Class 5 Iteration = 2100, eps = 1537.2350762307751
>> Class 6 Iteration = 2100, eps = 7.39242616784794
>> Class 7 Iteration = 2100, eps = 8.462610671861048
>> Class 9 Iteration = 2100, eps = 456.695658422514
>> Class 8 Iteration = 2100, eps = 2595.9711627649435
>> Class 5 Iteration = 2200, eps = 1507.2564932408902
>> Class 6 Iteration = 2200, eps = 7.174665389483379
>> Class 7 Iteration = 2200, eps = 8.218229405646596
>> Class 9 Iteration = 2200, eps = 429.7126680599661
>> Class 8 Iteration = 2200, eps = 2588.986121204791
>> Class 5 Iteration = 2300, eps = 1478.5740158749368
>> Class 6 Iteration = 2300, eps = 6.974075642518009
>> Class 7 Iteration = 2300, eps = 7.99106740503759
>> Class 9 Iteration = 2300, eps = 417.0445432255202
>> Class 8 Iteration = 2300, eps = 2582.4921018298796
>> Class 5 Iteration = 2400, eps = 1451.0708022708814
>> Class 6 Iteration = 2400, eps = 6.788566082299873
>> Class 7 Iteration = 2400, eps = 7.7792124809045555
>> Class 9 Iteration = 2400, eps = 416.43220540712116
>> Class 8 Iteration = 2400, eps = 2576.4265788214225
>> Class 5 Iteration = 2500, eps = 1424.6462393828435
>> Class Stopped : 5 Iteration = 2500, eps = 1424.4481391987097
>> Class 6 Iteration = 2500, eps = 6.616372690539649
>> Class Stopped : 6 Iteration = 2500, eps = 6.614712738280108
>> Class 7 Iteration = 2500, eps = 7.581042640348958
>> Class Stopped : 7 Iteration = 2500, eps = 7.579125184323956
>> Class 9 Iteration = 2500, eps = 428.7437666058314
>> Class 8 Iteration = 2500, eps = 2570.737174265715
>> Class Stopped : 9 Iteration = 2500, eps = 409.7313591161487
>> Class Stopped : 8 Iteration = 2500, eps = 2570.702972247016
Part(a): Accuracy for training set (N = 60,000) is: 0.917183
Part(b):     Accuracy for test set (N = 10,000) is: 0.917500
```

In [5]:
```
training_acc = accuracy(OvA, trainX, trainY)
test_acc = accuracy(OvA,  testX,  testY)
print(r"Part(a): Accuracy for training set (N = 60,000) is: %f" % training_acc)
print(r"Part(b):     Accuracy for test set (N = 10,000) is: %f" % test_acc)
```

```
Part(a): Accuracy for training set (N = 60,000) is: 0.917183
Part(b):     Accuracy for test set (N = 10,000) is: 0.917500
```

In [6]:
```
accuracy_plot(OvA,  trainX,  trainY, label='Training Data, Overall Accuracy: {:.4f}'.format(training_acc))
```



In [7]:
```
accuracy_plot(OvA,  testX,  testY, label='Test Data, Overall Accuracy: {:.4f}'.format(test_acc))
```