3. theano multi-class classification

April 4, 2016

0.1 3. Multi-Class Classification in Theano

The aim of this IPython notebook is to show some features of the Python **Theano** library in the field of machine learning. It has been developped by the LISA group at the *University of Montreal* (see: http://deeplearning.net/software/theano/). The notebook also relies on other standard Python libraries such as *numpy*, *pandas* and *matplotlib*.

To exemplify the use of **Theano**, this notebook solves the assignments of the *Machine Learning* MOOC provided by **Coursera** (see: https://www.coursera.org/learn/machine-learning) and performed in *Stanford University* by **Andrew Ng** (see: http://www.andrewng.org/).

The original MOOC assignments should to be programmed with the **Octave** language (see: https://www.gnu.org/software/octave/).The idea with this notebook is to provide Python developpers with interesting examples programmed using **Theano**.

This notebook has been developped using the *Anaconda* Python 3.4 distribution provided by **Continuum Analytics** (see: https://www.continuum.io/). It requires the **Jupyter Notebook** (see: http://jupyter.org/).

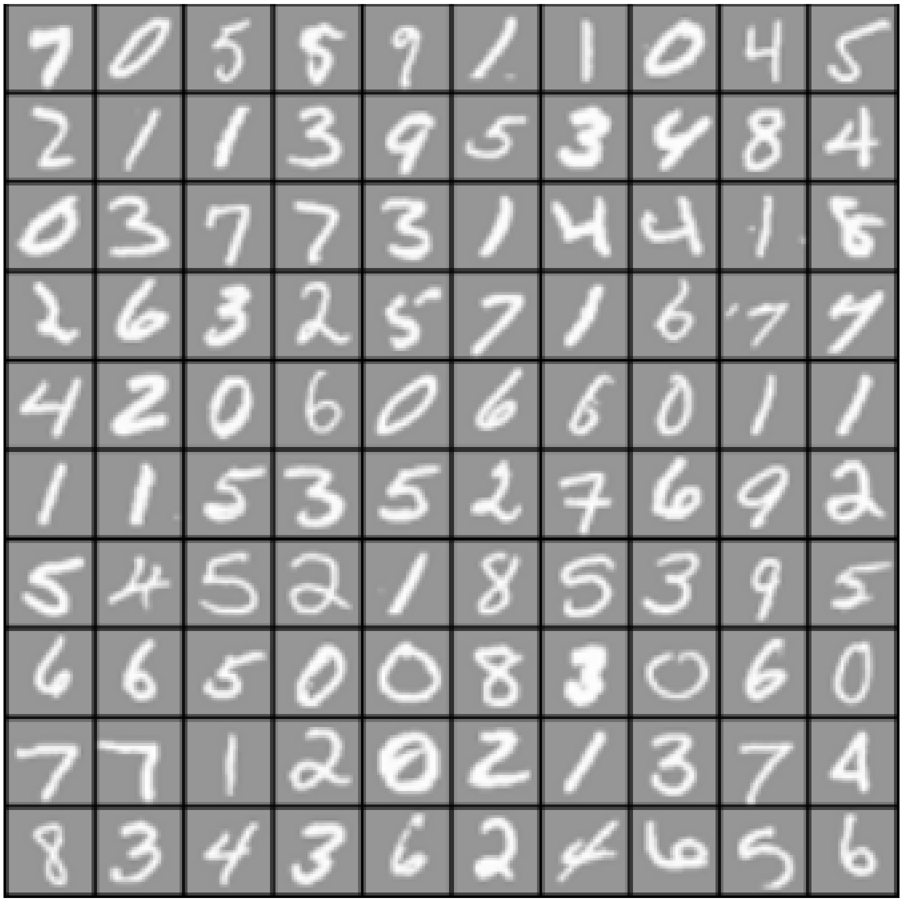
About the author: **Francis Wolinski** has an Engineering Degree From *Ecole des Ponts ParisTech* as well as a MSc. in Artificial Intelligence and a PhD. in Computer Science from *Université Pierre et Marie Curie* (UPMC).

0.1.1 3.1 Logistic Regression

3.1.1 Visualizing the data

```
In [1]: import theano
        import numpy as np
        import theano.tensor as T
        import scipy.io as sio
        data = sio.loadmat('data/ex3data1.mat')
        X, Y = data['X'], data['y'].T[0]
        m = X.shape[0]
        sel = X[np.random.permutation(m)[0:100],:] # 100 random digits
```

```
In [2]: %matplotlib inline
        import matplotlib.pyplot as plt

        def display_data(X):
            example_width = int(np.sqrt(X.shape[1]))

            # Compute rows, cols
            m, n = X.shape
            example_height = int(n / example_width)

            # Compute number of items to display
            display_rows = int(np.floor(np.sqrt(m)))
            display_cols = int(np.ceil(m / display_rows))

            # Between images padding
            pad = 1

            # Setup blank display
            display_array = - np.ones((pad + display_rows * (example_height + pad),\
                                   pad + display_cols * (example_width + pad)))

            # Copy each example into a patch on the display array
            # dataset contains 20 pixel by 20 pixel grayscale images of the digit
            curr_ex = 0
            for j in range(display_rows):
                for i in range(display_cols):
                    if curr_ex >= m:
                        break
                    # Get the max value of the patch
                    max_val = max(abs(X[curr_ex]))
                    display_array[(pad + j * (example_height + pad)):(pad + j * (example_height + pad))+
                                  (pad + i * (example_width + pad)):(pad + i * (example_width + pad))+ex
                                    np.reshape(X[curr_ex], (example_height, example_width)) / max_val
                    curr_ex += 1
                if curr_ex >= m:
                    break

            plt.figure(figsize=(8,6))
            plt.axis('off')
            plt.imshow(display_array.T, cmap="Greys_r")

In [3]: display_data(sel)
```



3.1.2 Logistic Regression

In [4]:
```
X1 = np.c_[np.ones(m), X] # Add intercept term to X
X1 = X1.T

# Choose some alpha and lambda values
alpha = 0.1
lambda_ = 0.1

# to compute theta * theta without first parameter which is not regularized
I = np.eye(X1.shape[0])
I[0,0] = 0

num_labels = 10

all_theta = np.zeros((num_labels, X1.shape[0]))

for j in range(num_labels):
    print('Feature: %i' % (j+1))
    # Init Theta and Run Gradient Descent
```

```
        t = np.zeros(X1.shape[0])
        theta = theano.shared(t,name='theta')

        x = T.matrix('x')
        y = T.vector('y')

        h = 1.0 / (1.0 + T.exp(-T.dot(theta,x)))
        r = lambda_ * T.dot(T.dot(I,theta),T.dot(I,theta)) / 2 / m
        cost = -T.sum(y * T.log(h) + (1.0 - y) * T.log (1.0 - h))/m + r
        grad = T.grad(cost,theta)

        train = theano.function([x,y],cost,updates = [(theta,theta-alpha*grad)])

        num_iters = 500
        for i in range(num_iters):
            costM = train(X1,Y==(j+1))

        all_theta[j] = theta.get_value()

    # number of positive predictions where best prediction == Y-1
    accuracy = np.sum(np.argmax(np.dot(all_theta,X1), axis=0) == (Y-1))/len(Y)
    print('\nTrain Accuracy: %f' % accuracy)
```

```
Feature: 1
Feature: 2
Feature: 3
Feature: 4
Feature: 5
Feature: 6
Feature: 7
Feature: 8
Feature: 9
Feature: 10

Train Accuracy: 0.885000
```

```
In [ ]:
```