Inleiding Machine Learning

Exercises

1. Bishop Ex 4.13.
2. Bishop Ex 4.15.
3. Write a computer program to classify the 3's and the 7's from the MNIST data using logistic regression. Get the data from `http://www.cs.nyu.edu/~roweis/data/mnist_all.mat`. Use gradient descent using the expression 4.91 in Bishop. Monitor E in each learning step and ensure that the learning rate η is as large as possible, but small enough so that E decreases in each step. Hint: Make sure that $0 < y < 1$ so that $\log y, \log(1-y)$ well defined.

Here is some Python code to get you started.

```
## Name: ......

#%%

import numpy as np
import scipy.io  # use scipy.io module to read in .mat files



data_file = scipy.io.loadmat('mnist_all.mat')  # load mnist_all.mat (if in same folder.!)
x3 = np.transpose(data_file['train3'])  # fetch and transpose training 3's
x7 = np.transpose(data_file['train7'])
p3 = int(np.size(x3, 1))  # number of 3 patterns
p7 = int(np.size(x7, 1))  # number of 7 patterns
n = int(np.size(x3, 0))  # number of pixels
x3 = np.float32(x3) / 256.0
x7 = np.float32(x7) / 256.0
x3 = np.vstack([x3, np.squeeze(np.ones((np.size(x3, 1), 1)))])  # extra input dimension for thres
x7 = np.vstack([x7, np.squeeze(np.ones((np.size(x7, 1), 1)))])
x = np.transpose(np.hstack([x3, x7]))  # create variable x with all training data
t = np.zeros(p3 + p7)  # target labels
t[:p3] = 0.0
t[p3:] = 1.0  # labels 0,1 for two classes
x = np.float32(x)
p = int(np.size(x, 0))  # total number of samples/patterns
n = int(np.size(x, 1))  # number of dimensions
w = np.zeros(n) # weights w(1:n-1) threshold w(n)

#%%

eta = 0.1  # learning rate
dw_max = 1.0  # max change in w in one learning step (update!)
i = 0  # learning iteration counter
```

```
maxiter = 1000 # maximum number of iterations
while (dw_max > 1e-5) and (i < maxiter):
#      your gradient descent code
 i = i + 1

print('convergence after {} iterations : E = {}, dw_max = {} \n'.format(i, E, dw_max))

#%%

# classification performance on training set
error3 = np.sum(np.dot(w, x3) >= 0)
error7 = np.sum(np.dot(w, x7) <= 0)
print('Training set performance: \n')
print('{} out of {} images of 3s misclassified \n'.format(error3, p3))
print('{} out of {} images of 7s misclassified \n'.format(error7, p7))

# classification performance  on test set

x3t = np.transpose(data_file['test3'])  #  input test data of 3's
x7t = np.transpose(data_file['test7'])  # input test data of 7's
p3t = int(np.size(x3t, 1))  # number of test images of class 3
p7t = int(np.size(x7t, 1))  # number of test images of class 7
n = int(np.size(x3t, 0))  # number of pixels
x3t = np.float32(x3t) / 256.0
x7t = np.float32(x7t) / 256.0
x3t = np.vstack([x3t, np.squeeze(np.ones((np.size(x3t, 1), 1)))])  # extra input dimension for th
x7t = np.vstack([x7t, np.squeeze(np.ones((np.size(x7t, 1), 1)))])
error3t = np.sum(np.dot(w, x3t) >= 0) # count errors in test data
error7t = np.sum(np.dot(w, x7t) <= 0)
print('Test set performance: \n')
print('{} out of {} images of 3s misclassified \n'.format(error3t, p3t))
print('{} out of {} images of 7s misclassified \n'.format(error7t, p7t))

%%
```

And here the Matlab

```
clear all

load mnist_all.mat
x3 = train3'; %  fetch and transpose training 3's
x7 = train7';
p3 = size(x3, 2);  % number of 3 patterns
p7 = size(x7, 2);  % number of 7 patterns
n = size(x3, 1);  % number of pixels
x3 = double(x3)/256.0;
x7 = double(x7)/256.0;
x3(n+1, :) = 1; % extra input dimension for thresholds
x7(n+1, :) = 1;
x = [x3, x7]';  % create variable x with all training data
t = zeros(1, p3 + p7);  % target labels
t(1:p3) = 0;
```

```
t(p3 + 1:p3 + p7) = 1; % labels 0,1 for the two classes
x = double(x); % matlab does not like the mnist data format
p = size(x, 1); % total number of samples/patterns
n = size(x, 2); % number of dimensions
w = zeros(1, n); % weights w(1:n-1) threshold w(n)

%%
eta = 0.1 % learning rate
dw_max = 1; % max change in w in one learning step (update!)
i = 0; % learning iteration counter
maxiter = 1000;  % maximum number of iterations
while (dw_max > 1e-5) && (i < maxiter):
%     your gradient descent code

    i = i+1;
end;



fprintf('convergence after %d iterations: E = %f, dw_max = %f\n', i, E, dw_max)

%%

% classification performance on training set

error3 = sum(w*x3 >= 0);  % count errors
error7 = sum(w*x7 <= 0);
fprintf('Training set performance\n');
fprintf('%d out of %d images of 3s misclassified\n', error3, p3)
fprintf('%d out of %d images of 7s misclassified\n', error7, p7)

% classification performance on test set

x3t = test3'; % input test data of 3's
x7t = test7'; % input test data of 7's
p3t = size(x3t, 2); % number of test images of class 3
p7t = size(x7t, 2); % number of test images of class 7
n = size(x3t, 1);
x3t = double(x3t) / 256.0;
x7t = double(x7t) / 256.0;
x3t(n+1, :) = 1; % extra input dimension for thresholds
x7t(n+1, :) = 1;
error3t = sum(w*x3t >= 0);  % count errors in test data
error7t = sum(w*x7t <= 0);
fprintf('test set performance\n');
fprintf('%d out of %d images of 3s misclassified\n', error3t, p3t)
fprintf('%d out of %d images of 7s misclassified\n', error7t, p7t)
```