MNIST BINARY CLASSIFICATION-0 vs 1

Aquino, Patrica Rose
Basallote, Lawrence Andrew
Carag, Stephanie
Jacinto, Dan Emanuel
Lunasco, Jan Osbert

I. Introduction

The MNIST database of handwritten digits, available from this page, has a training set of 60,000 examples, and a test set of 10,000 examples. It is a subset of a larger set available from NIST. The digits have been size-normalized and centered in a fixed-size image. It is a good database for people who want to try learning techniques and pattern recognition methods on real-world data while spending minimal efforts on preprocessing and formatting.

II. Procedure

You will find starter code for this exercise in the ex1ex1b_logreg.m file. The starter code file performs the following tasks for you:

1) Calls ex1_load_mnist.m to load the MNIST training and testing data. In addition to loading the pixel values into a matrix X (so that that jth pixel of the ith example is $Xji = x(i)j$) and the labels into a row-vector y, it will also perform some simple normalizations of the pixel intensities so that they tend to have zero mean and unit variance. Even though the MNIST dataset contains 10 different digits (09), in this exercise we will only load the 0 and 1 digits the ex1_load_mnist function will do this for you.
2) The code will append a row of 1s so that 0 will act as an intercept term.
3) The code calls minFunc with the logistic_regression.m file as objective function. Your job will be to fill in logistic_regression.m to return the objective function value and its gradient.
4) After minFunc completes, the classification accuracy on the training set and test set will be printed out.

III. Data and Results

A. Code

1) logistic_regression.m:

```
function [f,g] = logistic_regression(theta,
X,y)
  %
  % Arguments:
  %   theta - A column vector containing
the parameter values to optimize.
  %   X - The examples stored in a matrix.
  %       X(i,j) is the i'th coordinate of
the j'th example.
  %   y - The label for each example.
y(j) is the j'th example's label.
  %

  m=size(X,2);

  % initialize objective value and
gradient.
  f = 0;
  g = zeros(size(theta));



  %
  % TODO:  Compute the objective function
by looping over the dataset and summing
  %        up the objective values for
each example.  Store the result in 'f'.
  %
  % TODO:  Compute the gradient of the
objective by looping over the dataset and
summing
  %        up the gradients (df/dtheta)
for each example. Store the result in 'g'.
  %
%%% YOUR CODE HERE %%%
%for i = 1:m
   %h=1/(1+2.718^(-theta' * X(:,i)));
   %f = f + sum(-y.*log(h) -
(1-y).*log(1-h));
%end

%f = (1/m)*f;

%for i = 1:m
%    h=1/(1+2.718^(-theta' * X(:,i)));
%    g = g + X(:,i)*(h - y(i));
%end

%g = 1/m*g;

f = -sum(y.*log(sigmoid(theta'*X)) +
(1-y).*log(1 - sigmoid(theta'*X)));
g = X*(sigmoid(theta'*X) - y)';
```

B. Result

```
Iteration 1:
Step Length: 1.20e-006
Function Val: 6.57e+003
Opt Cond: 4.09e+003
Iteration 32:
Step Length: 0.00000e+000
Function Val: 4.84e-001
Opt Cond: 1.36e-001
Optimization took 6.21 seconds.
Training accuracy: 100.0%
Test accuracy: 100.0%
```

IV. Conclusion

The experiment this time was about unsupervised logistic regression, contrary to the previous experiment which was unsupervised linear regression. Since it was unsupervised, the machine did not need us, the programmers, to tell it how to predict the output. Instead, it finds the pattern between the data all by itself, and then formulates an output based on the pattern it found. Furthermore, since we were using logistic regression this time, the output is no longer a predicted value, but instead a probability between 1 and 0. It can observed that more data meant the machine's predicted probability will be much more accurate. It can also be observed from the data that we were able to attain a satisfactory accuracy for the predicted output. By manipulating the theta, we were able to obtain a probability that was either large or small