MNIST Multi-class Classification - one vs. rest with Neural Network introduction

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

I. Introduction

A training point belong to a certain class. A function is used to determine or predict the class of which a new point belongs to. One vs. rest involves training a single classifier per class with positive and negative samples.

II. Procedure

A. Dataset

You are given a data set in ex3data1.mat that contains 5000 training ex- amples of handwritten digits.1 The .mat format means that that the data 1This is a subset of the MNIST handwritten digit dataset has been saved in a native Octave/Matlab matrix format, instead of a text (ASCII) format like a csvfile.

B. Visualizing the data

You will begin by visualizing a subset of the training set. In Part 1 of ex3.m, the code randomly selects selects 100 rows from X and passes those rows to the displayData function. This function maps each row to a 20 pixel by 20 pixel grayscale image and displays the images together. We have provided the displayData function, and you are encouraged to examine the code to see how it works.

C. Vectorizing Logistic Regression

You will be using multiple one-vs-all logistic regression models to build a multi-class classifier. Since there are 10 classes, you will need to train 10 separate logistic regression classfiers. To make this training ecient, it is important to ensure that your code is well vectorized.

D. Vectorizing the cost function

Your job is to write the unregularized cost function in the file lrCostFunction.m Your implementation should use the strategy we presented above to calcu- late θ^T x(i). You should also use a vectorized approach for the rest of the cost function. A fully.

E. Vectorizing the gradient

The expression above allows us to compute all the partial derivatives without any loops. If you are comfortable with linear algebra, we encourage you to work through the matrix multiplications above to convince yourself that the vectorized version does the same computations. You should now implement Equation 1 to compute the correct vectorized gradient. Once you are done, complete the function lrCostFunction.m by implementing the gradient.

F. Vectorizing regularized logistic regression

After you have implemented vectorization for logistic regression, you will now add regularization to the cost function.

III. Data and Results

A. Code

1) rCostFunction.m:

```
lrCostFunction.m
function [J, grad] = lrCostFunction(theta,
X, y, lambda)

% Initialize some useful values
m = length(y); % number of training
examples

% You need to return the following
variables correctly
J = 0;
grad = zeros(size(theta));

H = sigmoid(X*theta);
T = y.*log(H) + (1 - y).*log(1 - H);
J = -1/m*sum(T) + lambda/(2*m)*
sum(theta(2:end).^2);

ta = [0; theta(2:end)];
grad = X'*(H - y)/m + lambda/m*ta;

end
```

2) OneVsAll.m:

```
oneVsAll.m
function [all_theta] = oneVsAll(X, y,
num_labels, lambda)
% Some useful variables
m = size(X, 1);
n = size(X, 2);
```

```
% You need to return the following
variables correctly
all_theta = zeros(num_labels, n + 1);

% Add ones to the X data matrix
X = [ones(m, 1) X];

for c = 1 : num_labels,
initial_theta = zeros(n + 1 , 1);
options = optimset('GradObj' , 'on' ,
'MaxIter' , 50);
[theta] = ...
fmincg(@(t)(lrCostFunction(t , X , (y ==
c) , lambda)), ...
initial_theta , options);
all_theta(c,:) = theta';
end
```

3) predictOneVsAll.m:

```
function p = predictOneVsAll(all_theta, X)

m = size(X, 1);
num_labels = size(all_theta, 1);

% You need to return the following
variables correctly
p = zeros(size(X, 1), 1);

% Add ones to the X data matrix
X = [ones(m, 1) X];

C = sigmoid(X*all_theta');
[M , p] = max(C , [] , 2);

End
```

4) predict.m:

```
function p = predict(Theta1, Theta2, X)

% Useful values
m = size(X, 1);
num_labels = size(Theta2, 1);

% You need to return the following
variables correctly
p = zeros(size(X, 1), 1);

X = [ones(m , 1) X];
a1 = X;
a2 = sigmoid(a1*Theta1');
a2 = [ones(m , 1) a2];
a3 = sigmoid(a2*Theta2');
[M , p] = max(a3 , [] , 2);

end
```

B. Result

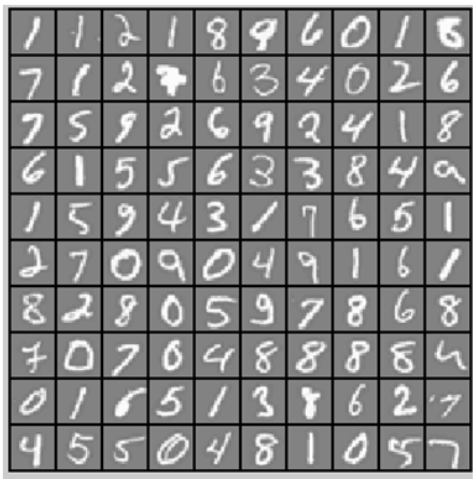

Fig. 1. Data

Accuracy: 92%

IV. Conclusion

In this experiment, we used an algorithm called neural network to scan and identify handwritten numbers. It can be observed that neural network is the same as logistic regression in the sense that it predicts probabilities. However, what makes neural network different from logistic regression is that it can process a much larger data. This algorithm performs almost alike how our brain processes information. Since we were able to obtain an accuracy of almost 100%, it is possible that a machine may be able to become more accurate than a person with the correct algorithm. This is because there are several factors that attribute to the loss of accuracy of a person. However, this loss of accuracy is also the reason why a human brain, as of today, is much more efficient than a machine. A human brain has the capability to obtain 100% accuracy, but it is satisfied with 80% accuracy to avoid performing a very long process. On the other hand, a machine will always attempt 100% accuracy regardless of how much work the process does to the machine. Therefore, the algorithm of a human brain that sacrifices accuracy for work efficiency is much better than today's neural network algorithm.