Multi-class Classification and Neural Networks

DANTING, Jules A. (10934375)

DE VERA, Luis Paolo M. (11113944)

LIM, Alyssa Raphaella (1128941)

RAMOS, Brian Justin S. (11127120)

VALDEZ, Esteen Rae Salvador (1111132)

**1 INTRODUCTION AND MAIN CONCEPTS**

Neural Network is a machine learning algorithm that is based on the brain of the user and is evaluated based on the practical applications a person does like recognition. A neural network is like a processor that has a tendency to store knowledge and keep it until it will be available. It is synonymous to a brain in a way that: knowledge is acquired by the network through a learning process and interneuron connection strengths known as synaptic weights. [1]. The experiment used a multi-class classification that aims to make a function that can make a correct prediction on the class or cluster of the new data set.It uses a One vs. All Classifier that will help in the prediction and have a higher value for accuracy.

**2 METHODOLOGY**

**2.1 Implementation**

1. Download the starter code and unzip its contents to the directory where you wish to complete the exercise.
2. Write the unregularized cost function in the file lrCostFunction.m. A fully vectorized version of lrCostFunction.m should not contain any loops.
3. Complete the code in oneVsAll.m to train one classifier for each class.
4. After you have correctly completed the code for oneVsAll.m, the script ex3.m will continue to use your oneVsAll function to train a multi-class classifier.
5. Complete the code in predictOneVsAll.m to use the one-vs-all classifier to make predictions.
6. Complete the code in predict.m to return the neural network’s prediction.

**3 RESULTS AND DISCUSSION**

**3.1 OUTPUT**

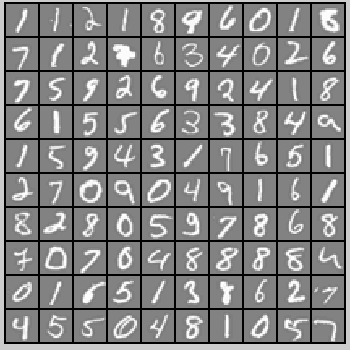


Fig. Data Visualization

Training Set Accuracy: 94.860000

**3.2 MATLAB CODE**

lrCostFunction.m

**function** **[**J**,** grad**]** **=** lrCostFunction**(**theta**,** X**,** y**,** lambda**)**

%LRCOSTFUNCTION Compute cost and gradient for logistic regression with

%regularization

% J = LRCOSTFUNCTION(theta, X, y, lambda) computes the cost of using

% theta as the parameter for regularized logistic regression and the

% gradient of the cost w.r.t. to the parameters.

% 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**));**

% ====================== YOUR CODE HERE ======================

% Instructions: Compute the cost of a particular choice of theta.

% You should set J to the cost.

% Compute the partial derivatives and set grad to the partial

% derivatives of the cost w.r.t. each parameter in theta

%

% Hint: The computation of the cost function and gradients can be

% efficiently vectorized. For example, consider the computation

%

% sigmoid(X \* theta)

%

% Each row of the resulting matrix will contain the value of the

% prediction for that example. You can make use of this to vectorize

% the cost function and gradient computations.

%

% Hint: When computing the gradient of the regularized cost function,

% there're many possible vectorized solutions, but one solution

% looks like:

% grad = (unregularized gradient for logistic regression)

% temp = theta;

% temp(1) = 0; % because we don't add anything for j = 0

% grad = grad + YOUR\_CODE\_HERE (using the temp variable)

%

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

%grad =((sigmoid(theta'\*X) - y)' \* X / m)' + lambda .\* theta .\* [0; ones(length(theta)-1, 1)] ./ m ;

%h = sigmoid(X \* theta);

%J = -1/m\*sum(y.\*log(h) + (1-y).\*log(1-h) + lambda/(2\*m) \*sum(theta(2:end).^2));

%thetap = [0;theta(2:end)];

%grad = X'\*(h-y)/m + lambda/m\*thetap;

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;

% =============================================================

%grad = grad(:);

end

oneVsAll.m

**function** **[**all\_theta**]** **=** oneVsAll**(**X**,** y**,** num\_labels**,** lambda**)**

%ONEVSALL trains multiple logistic regression classifiers and returns all

%the classifiers in a matrix all\_theta, where the i-th row of all\_theta

%corresponds to the classifier for label i

% [all\_theta] = ONEVSALL(X, y, num\_labels, lambda) trains num\_labels

% logisitc regression classifiers and returns each of these classifiers

% in a matrix all\_theta, where the i-th row of all\_theta corresponds

% to the classifier for label i

% 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**];**

% ====================== YOUR CODE HERE ======================

% Instructions: You should complete the following code to train num\_labels

% logistic regression classifiers with regularization

% parameter lambda.

%

% Hint: theta(:) will return a column vector.

%

% Hint: You can use y == c to obtain a vector of 1's and 0's that tell use

% whether the ground truth is true/false for this class.

%

% Note: For this assignment, we recommend using fmincg to optimize the cost

% function. It is okay to use a for-loop (for c = 1:num\_labels) to

% loop over the different classes.

%

% fmincg works similarly to fminunc, but is more efficient when we

% are dealing with large number of parameters.

%

% Example Code for fmincg:

%

% % Set Initial theta

% initial\_theta = zeros(n + 1, 1);

%

% % Set options for fminunc

% options = optimset('GradObj', 'on', 'MaxIter', 50);

%

% % Run fmincg to obtain the optimal theta

% % This function will return theta and the cost

% [theta] = ...

% fmincg (@(t)(lrCostFunction(t, X, (y == c), lambda)), ...

% initial\_theta, options);

%

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

predictOneVsAll.m

**function** p **=** predictOneVsAll**(**all\_theta**,** X**)**

%PREDICT Predict the label for a trained one-vs-all classifier. The labels

%are in the range 1..K, where K = size(all\_theta, 1).

% p = PREDICTONEVSALL(all\_theta, X) will return a vector of predictions

% for each example in the matrix X. Note that X contains the examples in

% rows. all\_theta is a matrix where the i-th row is a trained logistic

% regression theta vector for the i-th class. You should set p to a vector

% of values from 1..K (e.g., p = [1; 3; 1; 2] predicts classes 1, 3, 1, 2

% for 4 examples)

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**];**

% ====================== YOUR CODE HERE ======================

% Instructions: Complete the following code to make predictions using

% your learned logistic regression parameters (one-vs-all).

% You should set p to a vector of predictions (from 1 to

% num\_labels).

%

% Hint: This code can be done all vectorized using the max function.

% In particular, the max function can also return the index of the

% max element, for more information see 'help max'. If your examples

% are in rows, then, you can use max(A, [], 2) to obtain the max

% for each row.

%

C = sigmoid(X\*all\_theta');

[M , p] = max(C , [] , 2);

% =========================================================================

End

predict.m

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

%PREDICT Predict the label of an input given a trained neural network

% p = PREDICT(Theta1, Theta2, X) outputs the predicted label of X given the

% trained weights of a neural network (Theta1, Theta2)

% 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**);**

% ====================== YOUR CODE HERE ======================

% Instructions: Complete the following code to make predictions using

% your learned neural network. You should set p to a

% vector containing labels between 1 to num\_labels.

%

% Hint: The max function might come in useful. In particular, the max

% function can also return the index of the max element, for more

% information see 'help max'. If your examples are in rows, then, you

% can use max(A, [], 2) to obtain the max for each row.

%

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

**3.3 DISCUSSION**

The task is to apply the multi-class classification function which entails the one-vs-all classification and one-vs-all prediction. By using the logistic regression equation the group was able to get the cost function necessary for the succeeding procedures. This is used in the one-vs-all function where the theta is calculated based on the parameters of X, y, num\_labels, and lamda in order to training multiple logistic regression classifiers. The group managed to get a training set accuracy of 94.86 % as shown on the data visualization figure, the machine identified numbers from 0 - 9.

**4 CONCLUSION**

We performed another machine learning algorithm in this laboratory activity where we were able to make a correct prediction on the class of a new data set, and have a 100% training set accuracy or close to 100%. In this experiment we used and combined different functions we used in the previous experiments to be able to have an accurate prediction. Some of this functions are:

function [J, grad] = lrCostFunction(theta, X, y, lambda) to compute for the cost function and to regularize the logistic regression and gradient of the cost.

grad = zeros(size(theta)); for the gradient.

function [all\_theta] = oneVsAll(X, y, num\_labels, lambda) since this is a multi-class classification.

function p = predictOneVsAll(all\_theta, X) for vector predictions.

**REFERENCES**

[1]<http://www.aihorizon.com/essays/generalai/supervised_unsupervised_machine_learning.htm>

[2]<http://ufldl.stanford.edu/tutorial/supervised/LogisticRegression/>