Progress Report 2

* Completed first 4 weeks of the course.
* Learnt logistic regression and implemented (for multiclass classification too).
* Also regularization to prevent over fitting.
* One vs. all method for multiclass classification.
* Implemented a neural network, with weights already given(only forward propagation).
* cost function for logistic regression:

*J*(*θ*)=−*m*1​∑*i*=1*m*​[*y*(*i*) log(*hθ*​(*x*(*i*)))+(1−*y*(*i*)) log(1−*hθ*​(*x*(*i*)))]+2*mλ*​∑*j*=1*n*​*θj*2​

with lambda=0,will be the cost function without regularization.

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

%COSTFUNCTIONREG Compute cost and gradient for logistic regression with regularization

% J = COSTFUNCTIONREG(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

addreg=theta;

addreg(1)=0;

sq=addreg.^2;

J=(-1/m)\*sum(y.\*log(sigmoid(X\*theta))+(1-y).\*log(1-sigmoid(X\*theta)))+(lambda/(2\*m))\*sum(sq);

grad=(1/m)\*(X'\*(sigmoid(X\*theta)-y))+(lambda/m)\*addreg;

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

end

* One vs all creating classifiers:

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

% logistic 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 you

% 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:

%

%

for c=1:num\_labels

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);

all\_theta(c,:)=theta';

end

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

end

* Predicting results for multiple classes:

You run the given values of all the features through all the classifiers and pick the one with the greatest value, the one which produces that is the class.

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.

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

[a,p]=max(k,[],2);

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

end

* Predicting results with neural networks:

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];

hidden=sigmoid(X\*Theta1');

hidden=[ones(m,1) hidden];

output=sigmoid(hidden\*Theta2');

[a,p]=max(output,[],2);

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

end