Programming assignment #2:

Logistic Regression and Softmax Regression

4/17/2015

In a group of 2-3
Due: 04/27/2015 11:59pm

This assignment derives from Stanford UFLDL tutorial (Logistic regression, Gradient check, Softmax regression). You can check the tutorial for more information about problem formulation.

Exercise

For this assignment, you will implement solution for a logistic regression problem, and a softmax problem.

Use the github repository as a starter code. To download it, click"download zip" in the right bottom corner. (**Codes in the repository have changed since last week, so you have to re-download it and not use the starter codes from last assignment's folder. Sorry about that**) The code is in matlab, so you are encouraged to use matlab for this assignment. Using other language is okay but it'll be more complicated since you have to write all the code. If you use other language than matlab, you don't have to implement the minFunc solution (see below)

The data we are using for this assignment is MNIST dataset of handwritten digits. Training set and test set are already separated. You should download all the four files and put it into your *ex1/* folder.

Logistic Regression

The aim of this problem is to classify two handwritten digits: 0 and 1, with logistic regression. Use the *ex1/ex1b_logreg.m* as the starter coder. This file performs most of trivial steps for you:

1. The MNIST data is loaded by *ex1_load_mnist.m*. It will also perform 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 (0-9), in this exercise we will only load the 0 and 1 digits — the *ex1_load_mnist* function will do this for you. An extra '1' feature is added to the dataset so that it will act as the intercept term in the logistic function.

2. It's your job to estimate the parameters . Do:

a. Use matlab optimization package minFunc to decide parameters of our model . Again, minFunc uses line-search to solve the optimization problem given *logistic_regression.m* . You need to compute the objective function value and the gradient with respect to the parameters in *logistic_regression.m*. The *logistic_regression.m* file receives training data , training labels, and current parameters.
b. Write your *own* gradient checking function, and check the gradient you calculated in a). Check Gradient checking for more information.
c. Implement your **stochastic gradient descent method (SGD)** [SGD updates the parameters based on one training example (or a few examples). This will make training procedure faster. You can choose either to use one example a time or use a minibatch a time.] You can use Gradient descent with whole batch as last time we did for extra credits. Note this will take a while. Then draw the plot of convergence (objective function value - iteration) for the method you use.

In your implementation, take advantage of matlab's quick matrix operation. See vectorization for more details. Full credit will be given with vectorization.

3. Finally, evaluate the two methods (three if you use batch gradient descent) by training time and accuracy of test data.

Softmax Regression

This exercise is similar to Logistic Regression, but now the goal is to classify all 10 digits. Use the *ex1/ex1c_softmax.m* as the starter code. Again, this file performs most of the trivial steps for you:

1. Loading procedure is similar to logistic regression. The test and training data is loaded by *ex1_load_mnist.m*. It will also perform normalizations of the pixel intensities so that they tend to have zero mean and unit variance. An extra '1' feature is added to the features.
2. If you looked at the Softmax Regression Tutorial, please don't use the trick in section "Properties of softmax regression parameterization". This means that we optimize all the θ as a d-by-K matrix.
3. It's your job to estimate the parameters . Implement two versions of solutions and check gradient:
   a. Use matlab optimization package minFunc to decide parameters of our model . You need to compute the softmax objective function value and the gradient with respect to the parameters in *softmax_regression.m*. The *softmax_regression.m* file receives training data , training label, and current parameters. Don't forget that minFunc supplies the parameters θ as a vector. The starter code will reshape θ into a d-by-K matrix (d is the dimension of single feature vector, K=10

classes). You also need to remember to reshape the returned gradient g back into a vector using g=g(:).

   b. Use your gradient checker to check the gradient you calculated in a).
   c. Implement your **stochastic gradient descent method (SGD)**. [SGD updates the parameters based on one training example (or a few examples). You can choose either to use one example a time or use a minibatch a time.] You can use gradient descent with whole batch as last time we did for extra credits. Then draw the plot of convergence (objective function value - iteration) for the method you use.

   In your implementation,vectorize your code. Softmax Regression gives some tips for vectorization. Full credit will be given with vectorization.

4. Finally, evaluate the two methods (three, if you use batch gradient descent) by comparing the training time and calculating the accuracy of test data.

Report

From now on, your report should be in NIPS format. (Style in latex)

In your report, include:

1. Description of approach. Make sure you include:
   a. Relevant functions for both exercises (Objective function, gradient in both sum-of-examples form and vector form).
   b. Description of your stochastic gradient descent for both exercises, you can write pseudo codes for this (Specify important parameters, such as stopping criterion, learning rule, tricks you use). If you use batch gradient descent, describe it too.
   c. Description for your gradient checking method.
2. Results:
   a. For each SGD/batch GD, plot for speed of convergence during training procedure (objective function value - # of iteration).
   b. For each method, training time.
   c. For each method, accuracy on training and test set.
3. Your findings and lessons learned.

Submission

1. **Each group submit one report and all your codes to yuw176@ucsd.edu**
2. **Make your title [291Programming_Assignment#2]**