Team: Olympic Gold
Yue Cheng
Chenggang Lou
Saira Rotondo
Sidi Wu

Executive Summary

Question

The question we are trying to solve is to recognize handwritten digits and turning those images into numeric data. It is actually a multi-class classification problem because there are ten possible outputs, the digits from 0 through 9.

Data Description

The data set is consisted of a training data set and a test data set. The training data set includes 60000 training examples and each training example is a 28 pixel by 28 pixel grayscale image of a digit. As an input variable, each pixel is represented by a number from 0~255 indicating the grayscale intensity at that location. The 28 by 28 grid of pixels is "unrolled" into a 784-dimensional vector. Each of these training examples (a vector) is a single row in our data frame. This gives us a 60000 by 784 matrix where every row is a training example for a handwritten digit image. The output variables in the training set is a 60000-dimensional vector whose elements are digits from 0 through 9. The testing data set has 10000 examples.

The data set is collected from the MNIST database of handwritten digits.
URL: "http://yann.lecun.com/exdb/mnist/"

Methodologies

- Load the MNIST digit recognition dataset into R

  Our data set is ubyte file format. Define load_image_file and load_label_file function to read the data set's image data and label data, respectively. The functions convert the data into matrix.

- Visualize the data

  Define show_digit function so that we can view the data in a format of image directly. See Graph-1.

- Normalize the data

  The input variables are the grayscale intensity of each pixel represented by a number from 0 to 255. Normalize the input data so that each element in matrix is divided by 255. Consequently, the input variables are transformed to a scale between 0 and 1.

- Training set and Testing set

  In MNIST database, it has divided the data set into training set and testing set. We don't need to implement cross-validation to split our data.

- Model selection

The multi-class classification is supervised learning. Softmax regression model generalizes logistic regression to classification problems where the class label y can take on more than two possible values. We choose Softmax regression model rather than K binary classifiers. In this case, the classes (number 0~9) are mutually exclusive. A softmax regression classifier would be more appropriate than ten separate logistic regression classifiers. In softmax classifier, the correct class could always have a higher probability and the incorrect classes always a lower probability.

- Running regression in R

  1. First, install package *softmaxreg*.
  2. Choose appropriate arguments.

     The training set input and labels are x and y respectively; maxit is the integer for maximum number of iterations; algorithm is the optimization algorithm; rate is the learning rate; batch is the parameter for mini-batch size. Specifically:

     Hidden: The numeric vector of integers specifying the number of hidden nodes in each layer. The hidden may largely improve the regression result. The node number of neural network without hidden layer is 784 - 10.

     Algorithm: Parameter indicating which gradient descenting learning algorithm to use, including ‘sgd’, ‘adagrad’, ‘rmsprop’, ‘momentum’, ‘nag’. At here, we will try 5 different algorithms and compare their convergence speed.

  3. Fit softmax function with softmax layer.

     We fit 5 models by using ‘sgd’, ‘adagrad’, ‘rmsprop’, ‘momentum’, ‘nag’ algorithm respectively. Then, Fit model by using “500+300 hidden units” neural network.

- Compare the loss of 6 models

  Plot convergence comparison charts between six models. The x-axis is the number of iteration. the y-axis is loss. Then, we can compare loss and convergence speed of six models.

- Make a prediction

  Using testing data, run six models to make prediction. One thing important is that the predict outputs range from 1 to 10, so we deduce 1 from every output.

- Calculate the accuracy

  We use table function to view the contingency table of predicted results and actual results. We want to see whether the predicted results is consistent with actual results. Then use mean function to calculate the prediction accuracy.

Results and Conclusion:

As we can see in the Table-1, in models without hidden layer, model by using RMSprop algorithm has highest accuracy of 92.76% and the lowest loss of 0.2648288. Also, the Grpah-2 shows that in this case RMSprop algorithm converge at fastest speed. Adagrad, momentum and NAG algorithm have similar performance with prediction accuracy above 91%. SGD algorithm works not well with 85% accuracy.

After adding hidden layers, the prediction accuracy is dramatically improved, at 98.19%. Although the convergence speed of this model is lower than other 4 models, it works best in minimizing the loss.

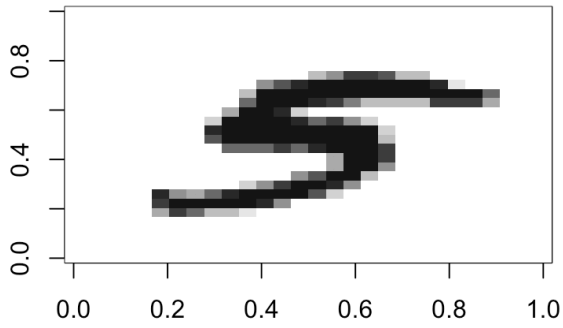
Overall, the softmax regression model does well in out Digits Recognition case. The results is pleasant, but there are much room for improvements. For example, we can try different hidden layers in neural network, add the penalty cost of the L2 regularization term, or increase the max number of iteration.

Appendix

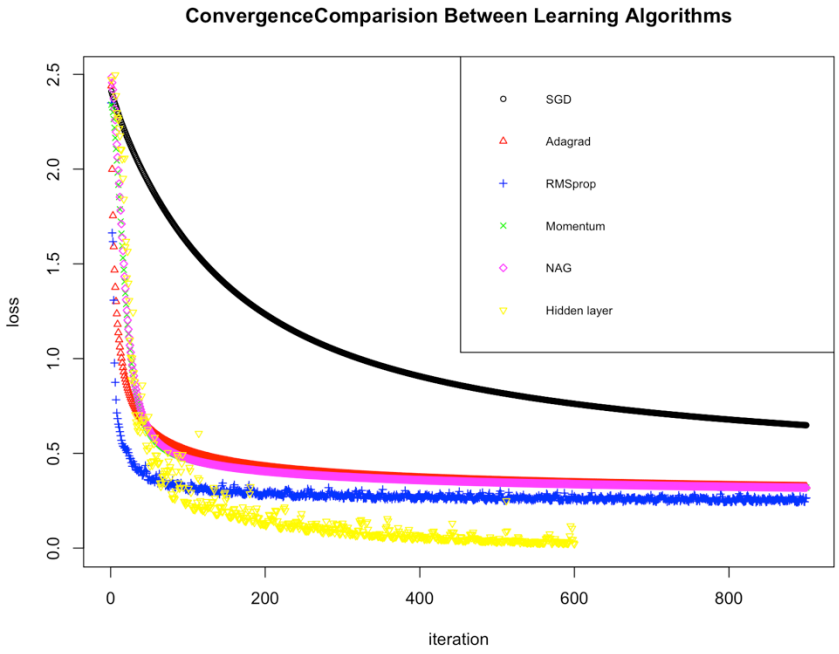
Table-1:

Model	Accuracy	Loss	Algorithm
1	85.76%	0.6485219	sgd
2	91.38%	0.3248247	adagrad
3	92.76%	0.2648288	rmsprop
4	91.54%	0.3192866	momentum
5	91.60%	0.3189832	NAG
Neural Network	98.19%	0.02104953	500+300 HU

Graph-1:



Graph-2:



Reference

Xichen Ding (2016), Package 'softmaxreg'
https://cran.r-project.org/web/packages/softmaxreg/softmaxreg.pdf

Softmax Regression
http://ufldl.stanford.edu/wiki/index.php/Softmax_Regression#Softmax_Regression_vs._k_Binary_Classifiers

THE MNIST DATABASE of handwritten digits
http://yann.lecun.com/exdb/mnist/