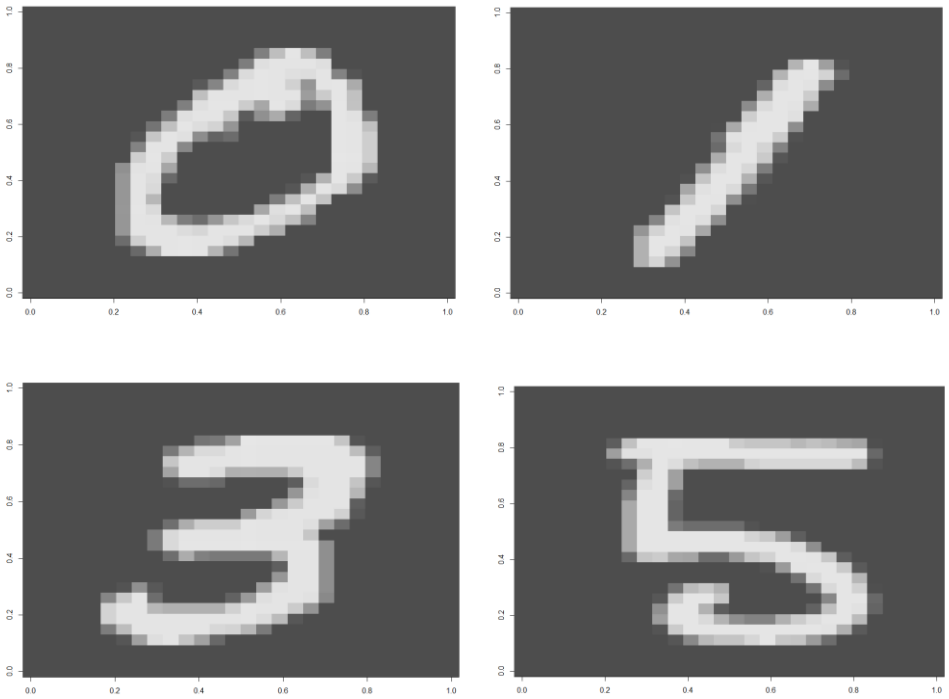
CSE6264 HW3: Logistic Regression

Xiaodong Gu (xgu60)

0. Data Preprocessing

Figure 1. Samples images for 0, 1, 3, 5, respectively.



1. Theory

a. Formula for computing the gradient of the loss function used in Logistic Regression
Logistic Regression cost function:

$$J(\theta) = \sum_{i=1}^{n} \log(1 + exp(y^i < \theta, x^i >))$$

where x is the training example with d dimensions, x = (1, x_1, x_2, ……x_d),

θ is the classifier parameters, $\theta = (\theta_0, \theta_1, ..., \theta_d)$,

n is the training sample number,

i represent i^{th} sample in the training set,

y is the label, y = 1 or -1.

Since y = 1 or -1, the cost function can be re-format as:

$$J(\theta) = \frac{1}{2}\sum_{i=1}^{n}\left(y^i + 1\right)\log(1 + exp(< \theta, x^i >) + \left(1 - y^i\right)\log(1 + \exp\left(-< \theta, x^i >\right))$$

Derivatization steps for Logistic Regression cost function:

$$\frac{\partial}{\partial\theta_j}\left(y^i+1\right)\log\left(1+\exp\left(<\theta,x^i>\right)\right)=\left(y^i+1\right)\frac{x_j^i e^{\theta x^i}}{1+e^{\theta x^i}}$$

$$\frac{\partial}{\partial\theta_j}\left(1-y^i\right)\log\left(1+\exp\left(-<\theta,x^i>\right)\right)=\left(y^i-1\right)\frac{x_j^i e^{-\theta x^i}}{1+e^{-\theta x^i}}$$

$$\frac{\partial}{\partial\theta_j}J(\theta)=\frac{1}{2}\sum_{i=1}^{n}\left(y^i+1\right)\frac{x_j^i e^{\theta x^i}}{1+e^{\theta x^i}}+\left(y^i-1\right)\frac{x_j^i e^{-\theta x^i}}{1+e^{-\theta x^i}}$$

$$\frac{\partial}{\partial\theta_j}J(\theta)=\frac{1}{2}\sum_{i=1}^{n}\left(y^i-\frac{1-e^{\theta x^i}}{1+e^{\theta x^i}}\right)x_j^i$$

Define hypothesis function:

$$h_\theta(x)=\frac{2}{1+e^{\theta x}}-1$$

$$\theta_j:=\theta_j-a\sum_{i=1}^{n}\left(y^i-h_\theta\left(x^i\right)\right)x_j^i,\ a\ is\ the\ learning\ step$$

b. Pseudocode for training a model using Logistic Regression (Batch Gradient descent).
For each iteration:
Initialize θ
#calculate y_exp = (y_exp^1, y_exp^2, …., y_exp^n):

Repeat n times {

#to calculate each y_exp

Repeat d times {

calculate $\theta_i * x_i$
}

}

#gradient descent update theta

#to update each θ

Repeat d times {
cost = 0

Repeat n times {

cumulate cost

}

new_ θ = θ – α * cost

}

c. Based on pseudocode, operations per iteration:
n * (d * c1) + d * ((n * c2) + c3) = O(n*d)

2. Implementation

Code submitted

3. Training

a. The following parameters were selected during the training of training_0_1 set and training_0_3 set.

Learning rate (α)	**α decay rate**	**Convergence Threshold**	**Maximum iterations**
0.6	0.99	1	1000

The convergence threshold was abs($\Delta\Theta$) / length(Θ).
The final accuracies:

	Train accuracy	**Test accuracy**
_0_1 set	0.9968417	0.9995272
_3_5 set	0.9566309	0.9568875

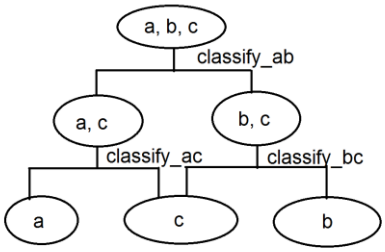
b. Batch Gradient Descent was implemented. Randomly selected 80% of training set was used for training, the training accuracy was measured on the randomly selected set, and test accuracy was measured on whole test set. The reported values are averages of 10 individual runs.

	Train accuracy	**Test accuracy**
_0_1 set	0.9968022	0.9994326
_3_5 set	0.9572232	0.9570452

c. The training and test accuracies of _0_1 set are higher than _3_5 set (training accuracy 0.9968022 vs 0.9572232, test accuracy 0.9994326 vs 0.9570452). Based on Figure 1, there are less overlap between 0 and 1 in _0_1 set versus 3 and 5 in _3_5 set, and the structures of 0, 1 are much simpler compared with the structures of 3, 5. Thus classification of _3_5 set much more difficult.
For both _0_1 set and _3_5 set, the training accuracy is very close to test accuracy, which means there are no overfittings during the training process.

d. Let's assume there are three classes (a, b, c) in my dataset.
For the train set, since I know the actual labels, I can train three classifies: Classify_ab, Classify_ac, Classify_bc.

Then I will use Classify_ab to predict test set. Since Classify_ab has no clue to c class, c class is more likely signed to a or b classes. Then Classify_ac and Classify_bc will be used to isolate a, c and b, c classes. The chart flow should look like:



4. Evaluation

a. Experiment with different initialization of the parameters. All experiments are similar with experiments described in 3b, that randomly selected 80% training set was used for training. All accuracies are averages of ten individual runs.

Theta initialization:

Init_theta	**Train accuracy**	**Test accuracy**
Random numbers 0-1	0.9572232	0.9570452
Random numbers 0-100	0.9572557	0.9567823
Random numbers 0-10000	0.9367817	0.9414301
All 1	0.9572232	0.9570452
All 100	0.9572882	0.9569926
All 10000	0.9331025	0.9365405

Learning rate (α)

α	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**
train	0.9408506	0.9507845	0.9538362	0.9556109	0.9568012
test	0.9495268	0.9531020	0.9552050	0.9560463	0.9567823
	0.6	**0.7**	**0.8**	**0.9**	**1.0**
train	0.9571259	0.9574072	0.9577535	0.9580781	0.9582729
test	0.9565720	0.9568875	0.9570978	0.9568875	0.9569926

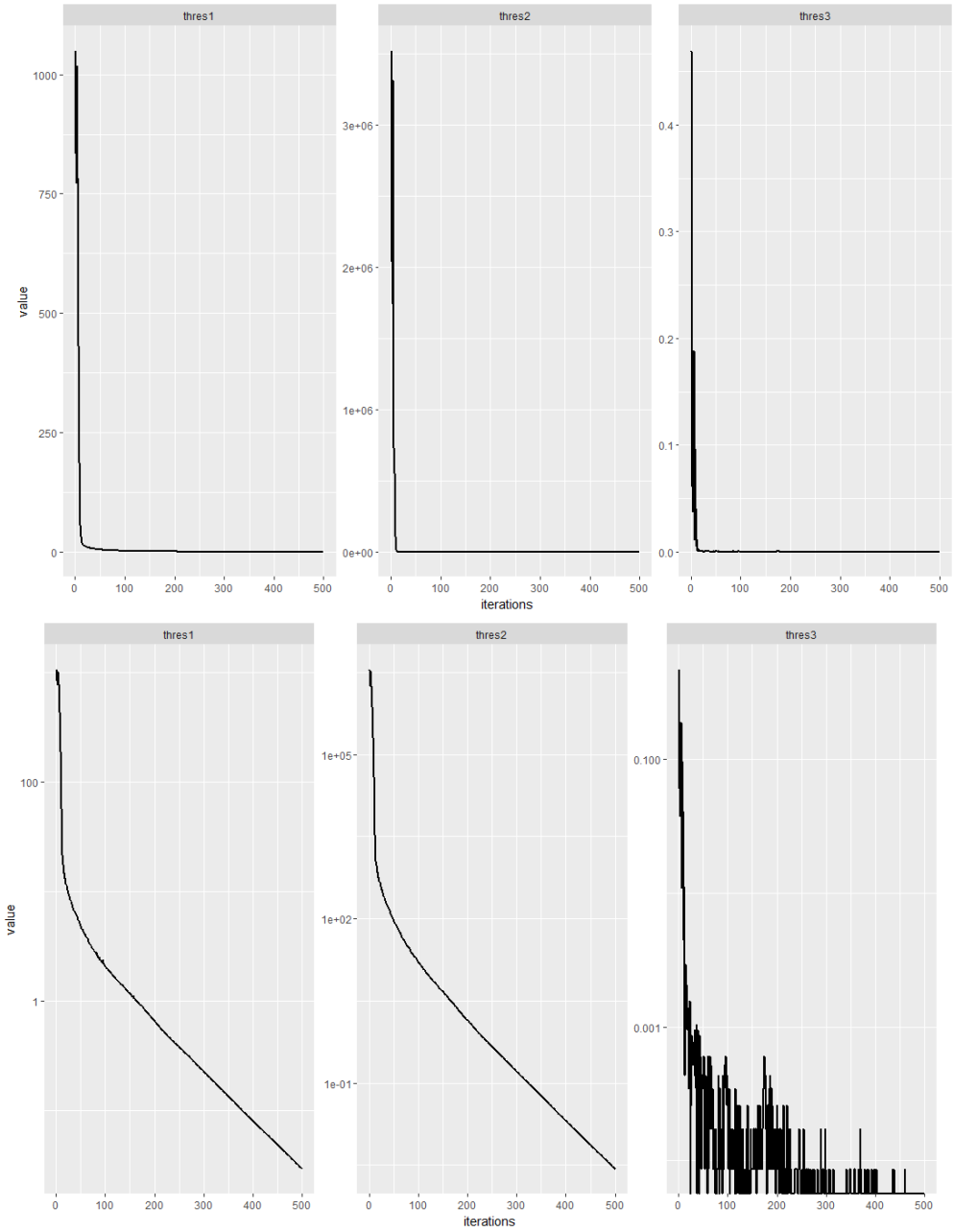
α decay rate

	0.99	**0.98**	**0.95**	**0.90**	**0.80**
train	0.9571259	0.9495076	0.9339682	0.9217834	0.9147711
test	0.9565720	0.9523659	0.9451104	0.9353312	0.9304942

Different initialized parameters were tried. The bench marks were marked in gray. For theta initialization, small thetas (random or consistent) are better than large thetas. For the learning rate, accuracies slightly increase with learning rates. Learning rate decay rates, slow decay (0.99) is better than large decay rates.

b. Experiments with different convergence criteria. Three different convergence criteria were tested. They are: 1. abs($\Delta\Theta$) / length(Θ), 2. $(\Delta\Theta)^2$ / length(Θ), 3. abs(accuracy improvement).

Figure 2: monitor the change of different convergence criteria: thres1. abs($\Delta\Theta$) / length(Θ), thres2. $(\Delta\Theta)^2$ / length(Θ), thres3. abs(accuracy improvement). Upper: normal scale. Lower: log10 scale for y.



Train accuracy and test accuracy of _3_5 set after using different convergence criteria as described:

	Train accuracy	Test Accuracy
Thres1 (1)	0.9572232	0.9570452
Thres2 (1)	0.9585434	0.9576761
Thres3 (0.000001)	0.9438156	0.9514196

Based on results, thres1 and thres2 which use the improvement of theta as convergence criteria have similar train and test accuracies. Thres3 which directly uses the improvement of prediction accuracy as convergence criteria has low train and test accuracies. I thus use thres1 (bench mark) for future experiments.

5. Learning Curves

a. Learning curves for train and test accuracies.

Figure 3: Learning curves for train (red) and test (blue) accuracies (averages of 10 individual runs) for _0_1 set using different randomly selected percentage data for training. Please note that y axis is not start from 0.

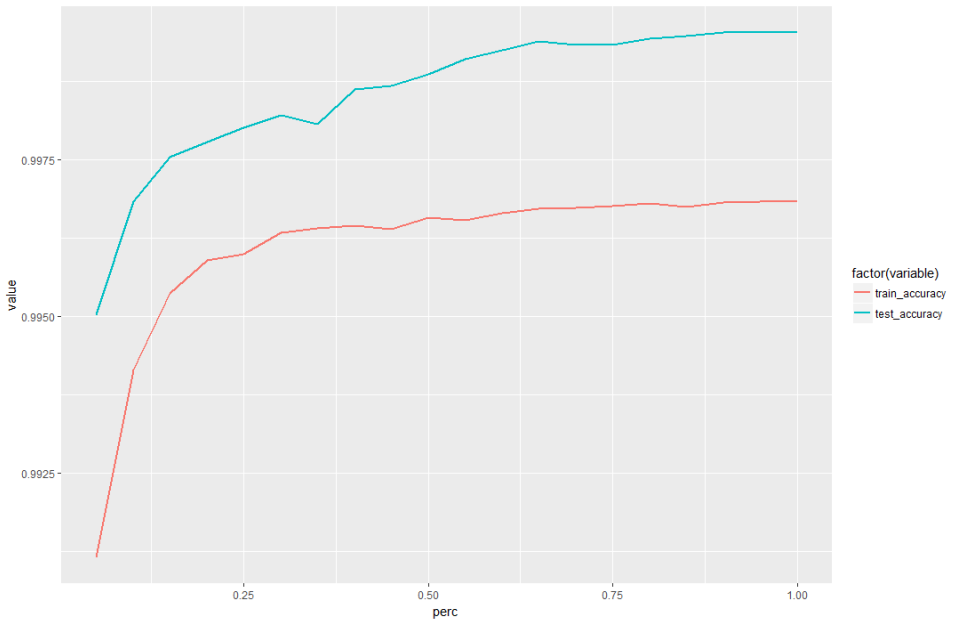
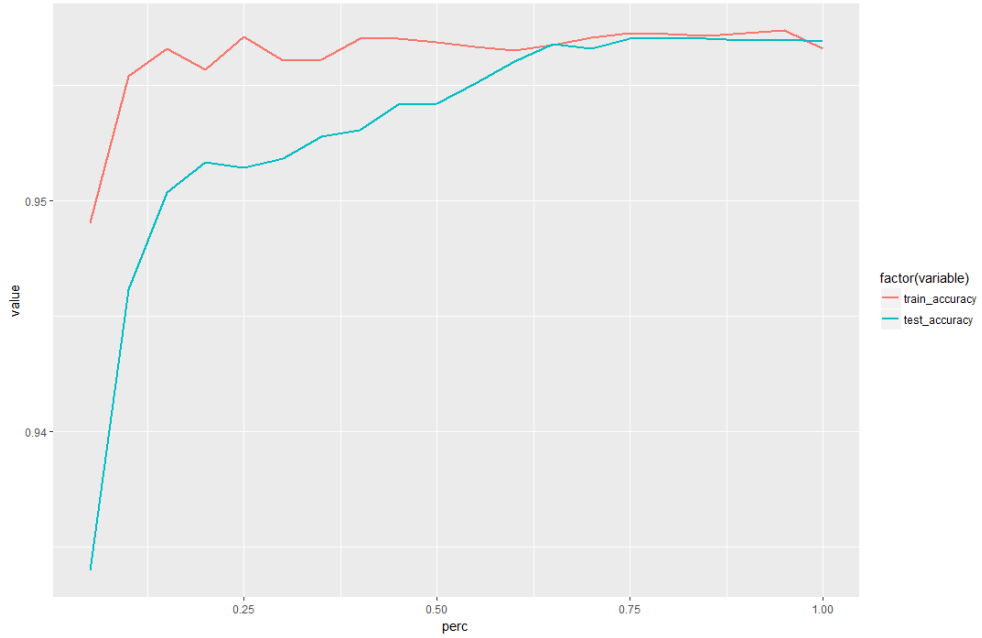


Figure 4: Learning curves for train (red) and test (blue) accuracies (averages of 10 individual runs) for _3_5 set using different randomly selected percentage data for training. Please note that y axis is not start from 0.



Based on data shown in Figure 3 and Figure4, train and test accuracies increase with the size of train set. More dramatic increments are observed when the train sets are small (percentage < 0.2). The learning curves plateaued with the further increment of train set. For _0_1 set, the test accuracies are always larger than the train accuracies. For _3_5 set, the train accuracies are larger than test accuracies when the train sets are small, and the train and test accuracies are close when the train sets are large (percentage > 0.8)

b. Learning curves for logistic loss. The logistic loss function that used during implementation is:

$$\frac{1}{n}\sum_{i=1}^{n}\log(1+exp(y^i<\theta,x^i>))$$

Based on data shown in Figure 5 and Figure 6. The logistic losses for train keep increase with the sample size. For _0_1 set, the logistic loss for test are small, and slightly decrease with sample size. For _3_5 set, the logistic loss for test also increase with sample size. The interesting find for _3_5 set is that these two lines cross over at certain point. At the beginning, logistic loss of test are larger than train, but when the size of train increases (> 0.75), the logistic loss of test are smaller than train.

Figure 5: Learning curves for train (red) and test (blue) logistic loss (averages of 10 individual runs) for _0_1 set using different randomly selected percentage data for training.

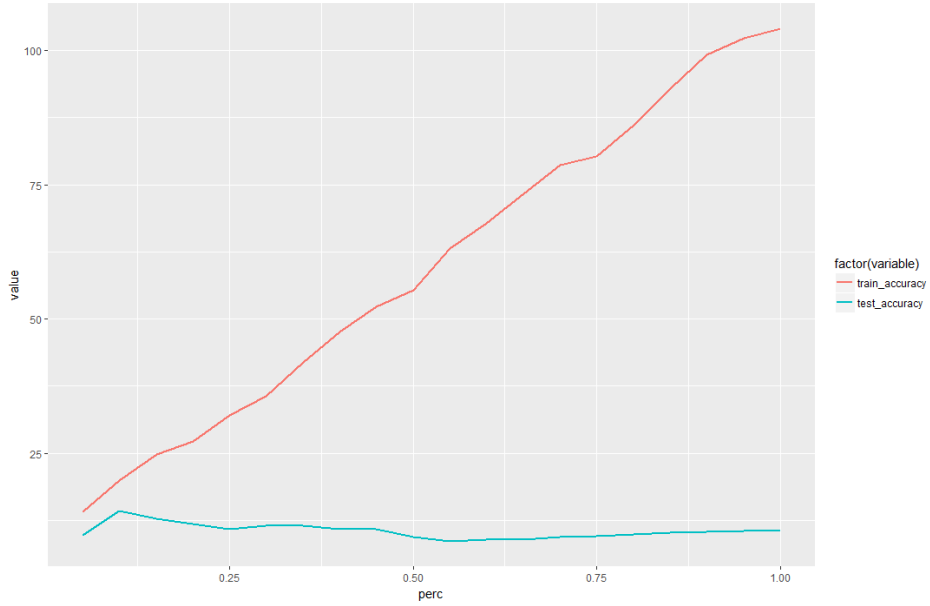


Figure 6: Learning curves for train (red) and test (blue) logistic loss (averages of 10 individual runs) for _3_5 set using different randomly selected percentage data for training.