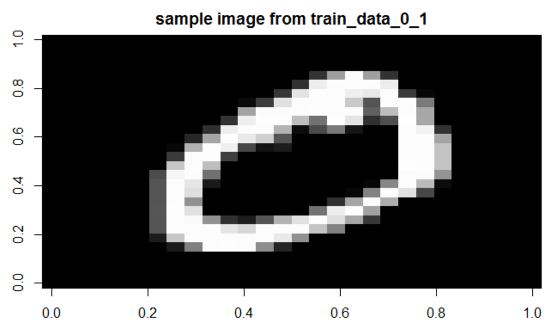
HW3

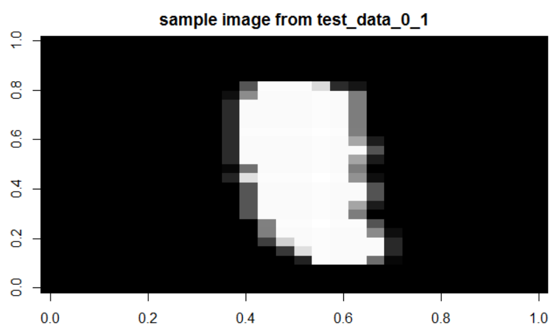
1. Data Preprocessing [20 points]

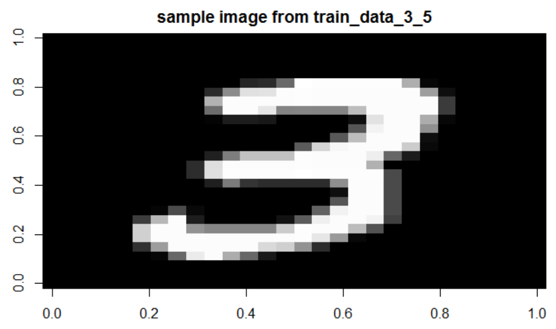
Answer:

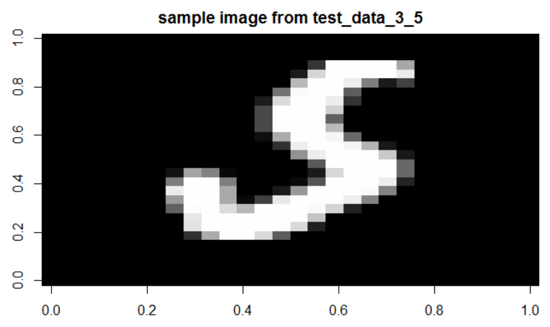
The dimension of train_0_1 are 785 12665. The dimension of train_3_5 are 785 11552. The dimension of test_0_1 are 785 2115. The dimension of test_3_5 are 785 1902.

Images:









2. Theory [20 points]

For a given dataset D of n training example:

$$D = \{< x^{(i)}, y^{(i)} > \ for\ i = 1 \dots n\}$$

Each $x^{(i)}$ is a vector of features with d dimensions:

$$x^{(i)} = < x_1^{(i)}, \quad x_2^{(i)}, \quad x_3^{(i)}, \quad x_4^{(i)}, \quad \dots \quad , \quad x_d^{(i)} >$$

Each $y^{(i)}$ is corresponding to -1 or 1.

The parameter of my model are show as θ with d dimensions:

$$\theta = < \theta_1, \quad \theta_2, \quad \theta_3, \quad \theta_4, \quad \dots \quad , \quad \theta_d >$$

$< x, \theta >$ is the inner product between x and θ

Write down the formula for the loss function used in Logistic Regression, the expression that you want to minimize: $L(\theta)$

$$\overset{\wedge}{\theta_{MLE}} = \underset{\theta}{\arg max}\ p_\theta\left(Y = y^{(1)} \middle| X = x^{(1)}\right) \dots p_\theta\left(Y = y^{(n)} \middle| X = x^{(n)}\right)$$

$$\overset{\wedge}{\theta_{MLE}} = \underset{\theta}{\arg max}\ log\ p_\theta\left(Y = y^{(1)} \middle| X = x^{(1)}\right) + \dots + log\ p_\theta\left(Y = y^{(n)} \middle| X = x^{(n)}\right)$$

We use logistic regression as probabilistic classifier:

$$p_\theta\left(Y = y | X = x\right) = \frac{1}{1 + \exp(y < \theta, x >)}$$

Then,

$$L(\theta) = \overset{\wedge}{\theta_{MLE}} = \underset{\theta}{\arg max} \sum_{i=1}^{n} log\left(\frac{1}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)}\right)$$

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

$$= \underset{\theta}{\arg min} \sum_{i=1}^{n} log\left(1 + \exp\left(y^{(i)} < \theta, x^{(i)} >\right)\right)$$

In order to get θ which shows the minimized value of $L(\theta)$, we would like to use gradient descent:

(1) initialize a set of values for vector θ, (2) use while loop to update $\theta_j < -\ \theta_j - \frac{\partial\ L(\theta)}{\partial\ \theta}$, (3) repeat while loop until the update is below certain threshold.

$$\theta_j <- \quad \theta_j - \frac{\partial\, L(\theta)}{\partial\, \theta} = \theta_j - \alpha \frac{\partial \sum_{i=1}^{n} \, log\left(1 + \exp\left(y^{(i)} < \theta, x^{(i)} >\right)\right)}{\partial \theta_j}$$
$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{\partial log\left(1 + \exp\left(y^{(i)} < \theta, x^{(i)} >\right)\right)}{\partial \theta_j}$$
$$= \ \theta_j - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)} \times \frac{\partial \ \ [\,1 + \exp\left(y^{(i)} < \theta, x^{(i)} >\right)]}{\partial \theta_j}$$
$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)} \times \frac{\partial\, 1 + \ \partial \ \exp\left(y^{(i)} < \theta, x^{(i)} >\right)}{\partial \theta_j}$$
$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{1}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)} \times \frac{\exp\left(y^{(i)} < \theta, x^{(i)} >\right) \partial\, y^{(i)} < \theta, x^{(i)} >}{\partial\, \theta_j}$$
$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{y^{(i)} \exp\left(y^{(i)} < \theta, x^{(i)} >\right)}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)} \times \frac{\partial < \theta, x^{(i)} >}{\partial \theta_j}$$
$$= \theta_j - \alpha \sum_{i=1}^{n} \frac{x^{(i)} y^{(i)} \exp\left(y^{(i)} < \theta, x^{(i)} >\right)}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)}$$

Express the Stochastic Gradient Descent (SGD) update rule that uses a single sample at a time.

For SGD, we need to give a set of random values to every dimensions of vector θ_j $(j = 1, 2, 3, \ldots., k)$, then pick up one set of data ($x^{(i)}$,$y^{(i)}$) randomly, then update θ_j based on :

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

Repeat this step until the update of θ_j became too small.

Pseudocode for training a model using Logistic Regression and SGD.

1. Load the training dataset into R, identify the dimension of $x^{(i)}$

2. For each $x^{(i)}$, find out its corresponding $y^{(i)}$, and assign $y^{(i)}$ either 1 or -1 based on it true label. For instance, the true label for y in train_0_1 is 0 or 1, and I can assign -1 to all y which have true label of 0 and assign 1 to all y which have true label of 1.

3. Generate a vector θ which have the same dimension as $x^{(i)}$, give each element in θ a random value.

4. Randomly pick a pair of $x^{(i)}$, $y^{(i)}$ in the training data set, and then update θ using :

$$\theta \quad <- \quad \theta \quad - \alpha \times \frac{y^{(i)} \exp\left(y^{(i)} < \theta, x^{(i)} >\right)}{1 + \exp(y^{(i)} < \theta, x^{(i)} >)} \times x^{(i)}$$

α could be a random value between 0.001 and 0.5

To update θ, first calculate temp = $\alpha \times \frac{y^{(i)} \exp(y^{(i)} <\theta, x^{(i)}>)}{1+\exp(y^{(i)} <\theta, x^{(i)}>)}$, then calculate $b = \ temp \times x^{(i)}$,

and finally $\theta = \quad \theta - b$. Here $\theta, x^{(i)}, b$ have the same dimension.

Repeat this step 1000 times and check the absolute mean value of θ updates every 1000 times

5. Use while-loop to repeat step 4 until absolute mean value of θ updates is smaller than a threshold

6. Now we get θ which minimize $L(\theta)$.

Estimate the number of operations per epoch of SGD, where an epoch is one complete iteration through all the training samples. Express this is Big-O notation, in terms of the number of samples (n) and the dimensionality of each sample (d).

In each epoch of SGD, the θ will be updated n times from the random selection of training samples and labels. Because each θ has the dimensionality of d, we need to update each individual elements in θ for d times. So, the total number of operation per epoch of SGD will be n x d. Because the vector operation at d dimensions is much faster than for loop, the time for vector operation at d dimension can be neglected compared with θ updates using for loop.

<u>3. Implementation [20 points]</u>

<u>Report:</u>

<u>Initialization method:</u>

I initialized every elements in θ with a random value between 0 and 1 using runif function (theta=runif(n=length(x), min = 0, max = 1)). I set alpha value to be 0.1 for my training.

<u>Convergence criteria:</u>

In one epoch of SGD, I randomly sampled 1000 times and update θ 1000 times. After that, I checked the absolute mean of change of θ in every dimensions during this 1000 times of updates [*abs(mean(sum_update))*]. I stopped the while loop when the absolute mean value of θ updates is below certain threshold (0.0001). Here are part of my code for the Convergence criteria:

```
while (threshold>0.0001){
………
 for(colu in random_col){
        ……
   inerProduct=as.numeric(x%*%theta)+1
   theta=theta-alpha*(y*exp(y*inerProduct)/(1+exp(y*inerProduct)))*x
   sum_update=sum_update+alpha*(y*exp(y*inerProduct)/(1+exp(y*inerProduct)))*x
  }
  threshold=abs(mean(sum_update))
  }
```

<u>Modifications you have made to the input data:</u>

I have converted the labels of 0 and 3 to be -1 and converted 5 to be 1. When I calculate inner product, I introduced a bias term (1). All of my training and prediction use use <x, θ>+1 instead of inner produce itself to add flexibility of my model.

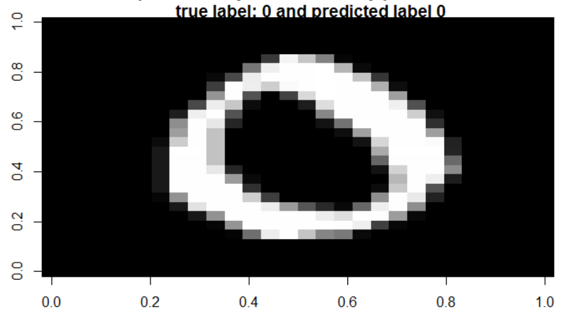
<u>Algorithm:</u>

For training function: First convert labels from 0, 1, 3, 5 to be -1 or 1. Then, initialize random value to theta with dimension equal to $x^{(i)}$. Next, randomly pick a pair of $x^{(i)}$ $y^{(i)}$, calculate the inner product of theta and x with addition of bias term 1. Update theta 1000 times using SGD. Then, check the change of theta before and after 1000 updates, if change is not smaller than the threshold, repeat another epoch of SGD. If the change of theta is very small, break loop and return theta.

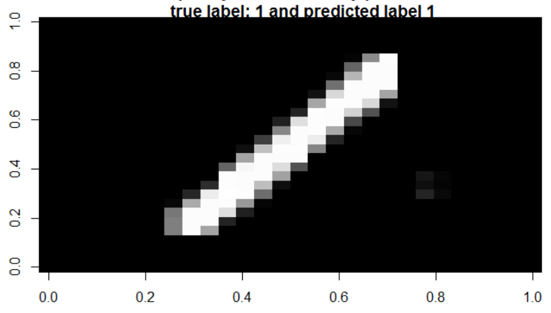
For predict function: apply theta to calculate probability using probability=1/(1+exp(theta%*%data+1)). If (probability>=0.5) return label as 1 for 0/1 sample or return 5 for 3/5 sample; else return label as 0 for 0/1 sample or return 3 for 3/5 sample.

<u>Visualization of 2 correct and 2 incorrect predictions each for 0/1 and 3/5 training sets (a total of 8 samples). For each sample, show the image, true label and predicted label.</u>

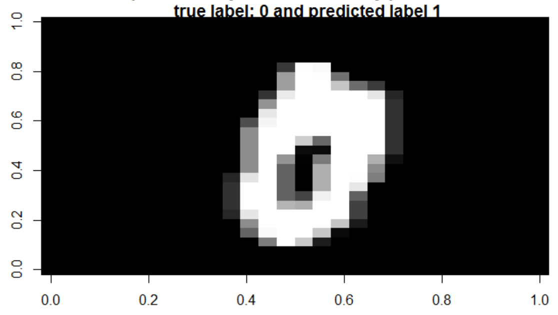
one example where my model correctly predicted the label
true label: 0 and predicted label 0



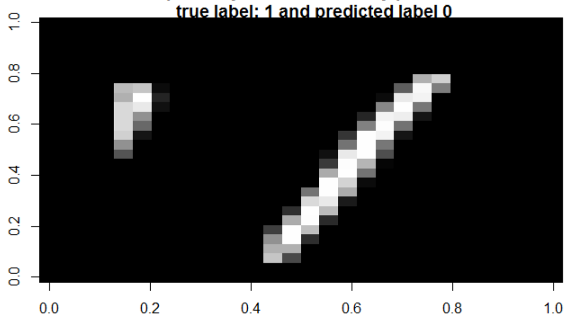
another example my model correctly predicted the label
true label: 1 and predicted label 1



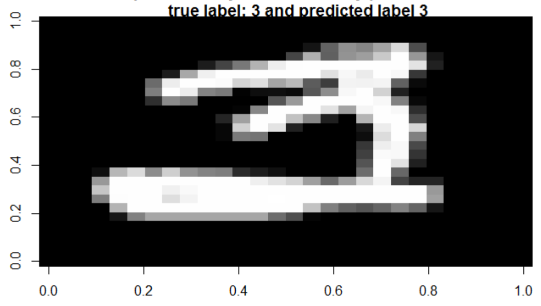
one example where my model incorrectly predicted the label
true label: 0 and predicted label 1



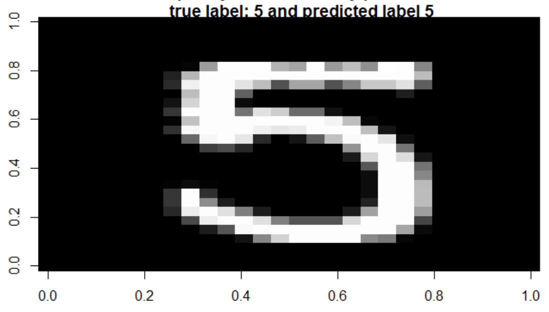
another example in my model incorrectly predicted the label
true label: 1 and predicted label 0



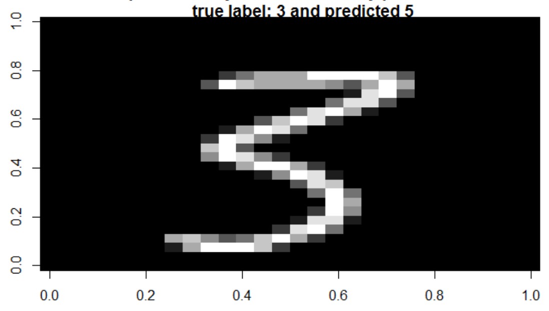
one example where my model correctly predicted the label
true label: 3 and predicted label 3



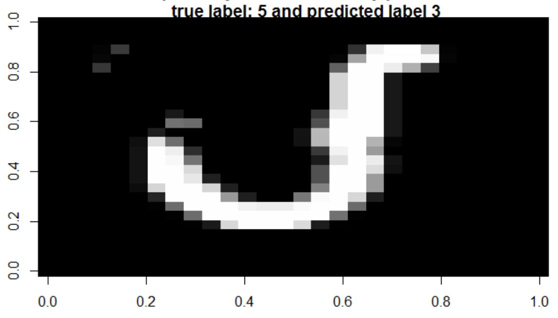
another example my model correctly predicted the label
true label: 5 and predicted label 5



one example where my model incorrectly predicted the label
true label: 3 and predicted 5



another example in my model incorrectly predicted the label
true label: 5 and predicted label 3



4. Modeling [20 points]

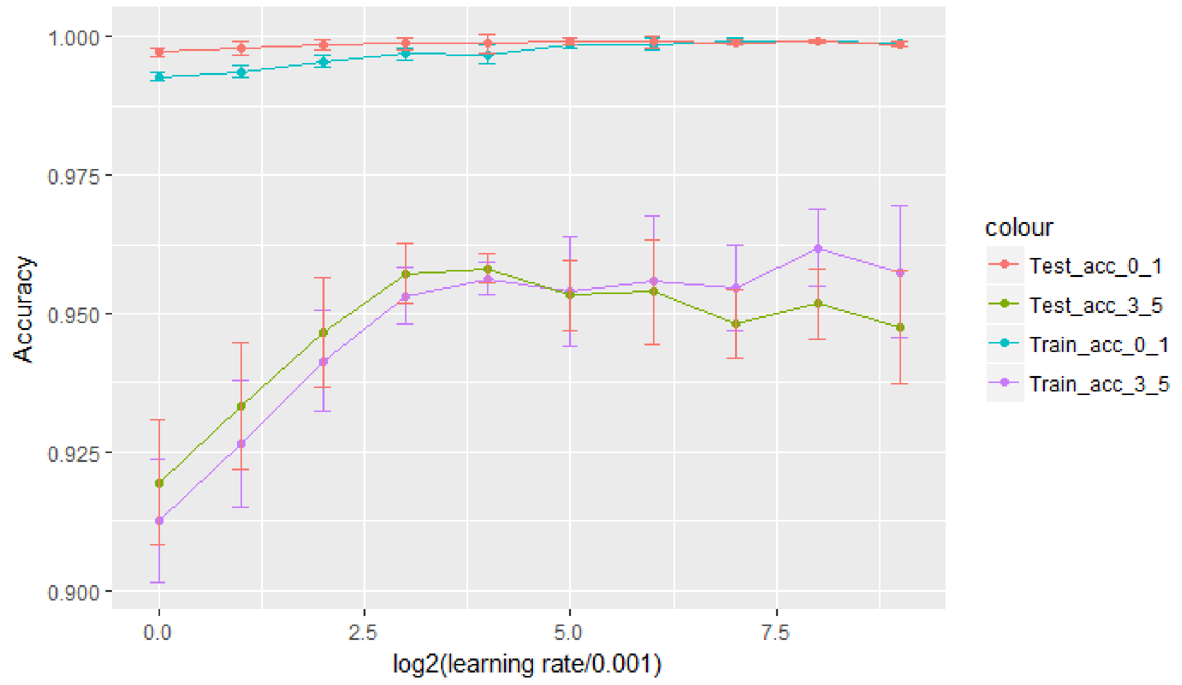
Report:

Train 2 models, one on the 0/1 set and another on the 3/5 set, and compute their training and test accuracies.

I trained 2 models, one on the 0/1 set and another on the 3/5 set. I then calculated their accuracy. The accuracies for 0/1 train and test dataset are 0.9987367 and 0.9990544, respectively. The accuracies for 3/5 train and test dataset are 0.9582756 and 0.955836, respectively.

Repeat the above step 10 times, with varying learning rates. Plot the training and test accuracies against learning rate for 0/1 and 3/5.

Please see my plot below. In this plot, I used ten different learning rates (0.001, 0.002, 0.004, 0.008, 0.016, 0.032, 0.064, 0.128, 0.256, 0.512). I plotted the accuracy of prediction training and testing 0/1, 3/5 datasets against different learning rates.



Did you observe any difference in the accuracies for 0/1 and 3/5 models? If so, explain why do you think the difference might be.

Yes, the accuracy for predicting 0/1 training dataset and test dataset are much higher than that of 3/5 training dataset and test dataset. At optimized learning rates, the accuracies for predicting 0/1 training and testing dataset are above 0.99. At optimized learning rates, the accuracies for predicting 3/5 training and testing dataset are around 0.95.

I think the shape differences between 0 and 1 are bigger than that of 3 and 5. Because 0 and 1 have bigger differences in shape, the accuracy for prediction is high. The differences between 3 and 5 is small. The bottom part of 3 and 5 are the same. The only difference between 3 and 5 is the top part and people could have difference writing styles for the top part of 3 and 5. Because of above reasons, it is harder to predict 3/5 and accuracies for predicting 3/5 datasets are lower than that of 0/1.

Report the best test accuracy you could achieve for 0/1, and the best test accuracy for 3/5. These may be at different learning rate settings.

The best test accuracy for 0/1 is 0.9992908, with learning rate of 0.256.

The best test accuracy for 3/5 is 0.9582019, with learning rate of 0.016.

This assignment deals with binary classification. Explain what you would do if you had more than two classes to classify using Logistic Regression (e.g. with a combined 0/1/3/5 dataset).

I will first label 0 as 1 and label 1/3/5 as -1, apply logistic regression and find optimized theta (theta0) to separate 0 and other numbers. Then, I will label all 1 as 1, labels 0/3/5 as -1, apply logistic regression and find another optimized theta (theta1) to separate 1 and other numbers. Next, I will label all 3 as 1, labels 0/1/5 as -1, apply logistic regression and find optimized theta (theta3) to separate 3 and other numbers. Finally, I will label all 5 as 1, labels 0/1/3 as -1, apply logistic regression and find optimized theta (theta5) to separate 5 and other numbers.

To predict labels of given datasets, I will get inner products between data and theta0 and calculate probability under theta0. If the probability is larger than 0.5, then I will label this data as 0. If the probability is less than 0.5, the label should be 1 or 3 or 5. I will do further analysis using other theta.

So, if the probability based on theta0 is less than 0.5, I will apply thata1 to the same data, and calculate the probability again. If the probability based on theta1 is larger than 0.5, then I will label this data as 1. If the probability based on theta1 is less than 0.5, the label should be 3 or 5. I will do further analysis using other theta.

If the probabilities based on theta0 and theta1 are less than 0.5, I will apply thata3 to the same data, and calculate the probability again. If the probability based on theta3 is larger than 0.5, then I will label this data as 3. If the probability based on theta3 is less than 0.5, I will label this data as 5. To confirm such label, I will calculate the probability of this data based on theta5. If the probability is large than 0.5, we get the correct label. If the probability is still less than 0.5 based on theta5, this data should not be labels as 0, 1, 3 or 5.

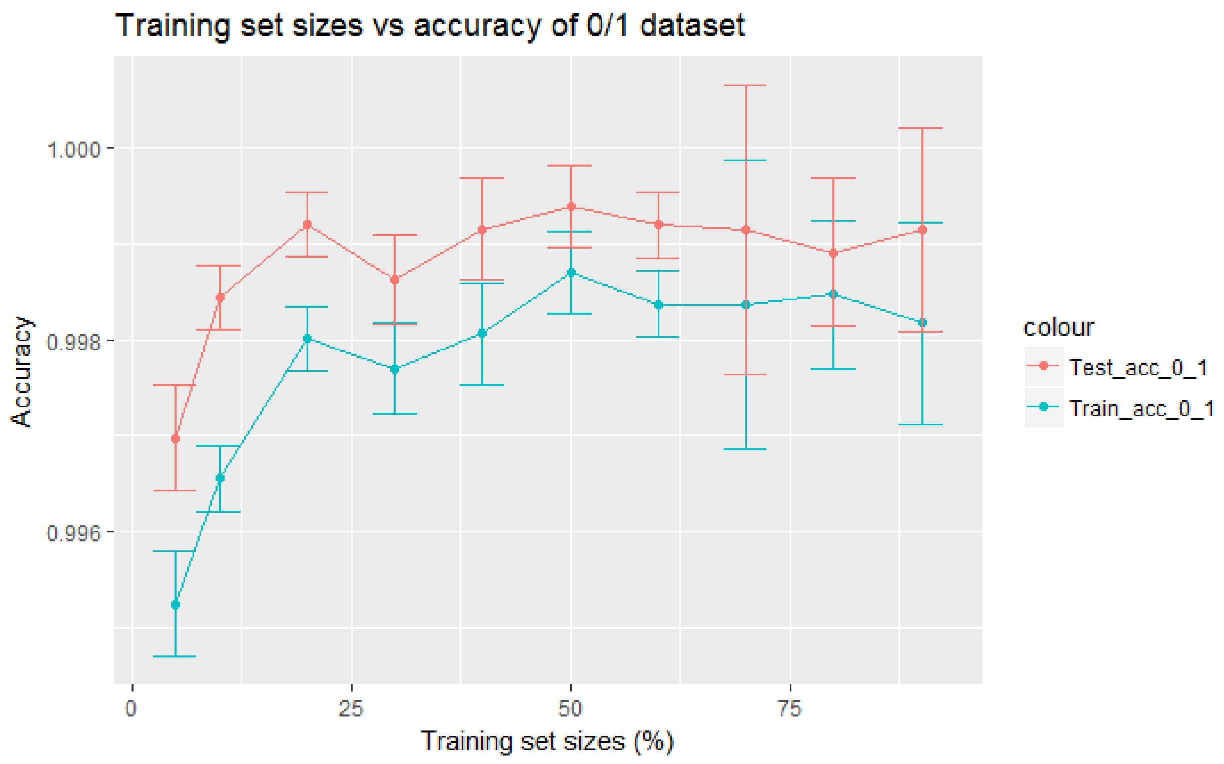
5. Learning Curves [20 points]

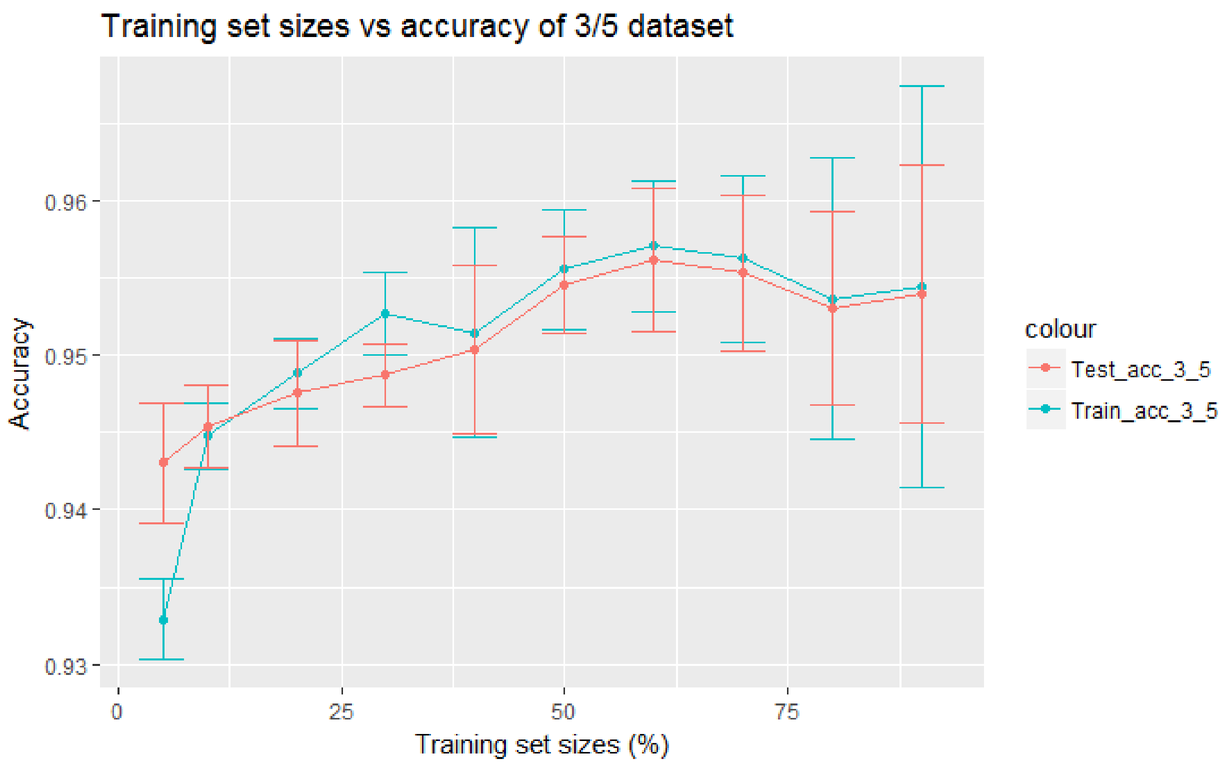
I selected from 5% to 90% of samples for the training set and calculated the accuracy of prediction for train 0/1 data, test 0/1 data, train 3/5 data and test 3/5 data under different training set sizes. Then, I plotted the accuracy against training data size. All the testing use the same alpha = 0.032.

As shown in the plot "Training set sizes vs accuracy of 0/1 dataset", the accuracy increased from 0.996 and 0.997 (for training and testing data) to 0.9986 and 0.9992 (for training and testing data). The accuracy increased to a plateau when the training set size increased to 20% or above.

As shown in the plot "Training set sizes vs accuracy of 3/5 dataset", the accuracy increased from 0.932 and 0.943 (for training and testing data) to 0.956 and 0.956 (for training and testing data). The accuracy increased to a plateau when the training set size increased to 50% or above.

These results suggest that larger sample size is helpful to improve prediction accuracy. However, once the sample size is large than certain threshold, the prediction accuracy cannot be further improved. These results also tell us that the maximum accuracy for different kind of prediction could be different. Also, the harder prediction problem (such as predict 3/5) requires larger sample size to achieve maximum prediction accura





Training set sizes vs accuracy of 3/5 dataset
Accuracy
0.93
0.94
0.95
0.96
0
25
50
75
Training set sizes (%)
colour
Test_acc_3_5
Train_acc_3_5