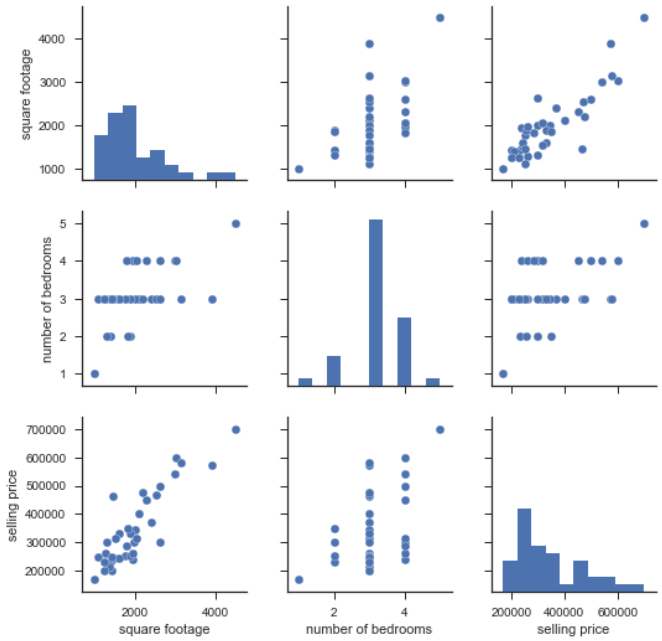
ECE194N HW1
Deren Lei
derenlei@umail.ucsb.edu

**Question 1 (Linear Regression)**

(a) Data Visualization by using scatterplot matrix




As image shows, there exists a strong positive linear relationship between selling price and square footage and there also exists a moderately strong positive linear relationship between selling price and number of bedrooms. In addition, the only two feature: number of bedrooms and square footage seems exists a moderately strong positive linear relationship. After some experiment using stepwise regression, it turns out that using only the square footage in the linear regression model is the best choice. However, for this question, I used the full first order linear regression model to train the data.

(b)

Loss Function chosen is the mean square divided by 2.

$$\frac{1}{2n}\sum (Y - W^T X + b)^2$$

The update rule is:

$$W = W - \eta\delta X$$
$$b = b - \eta\delta$$

Where

$$\delta = \frac{1}{n}\sum (W^T X - Y)$$

(c)

The linear regression model learnt is:

W = [151.61235868563773, -502.71999653132275]

b = 44181.93

I use r-square to measure the performance of the model on the testing data. The result is 0.3667004.

(d)

After doing normalization on trainning data and testing data, the linear regression model learnt is(note: the trainning data is normalized using the mean and standard deviation of the training data):

W = [0.8745967807781843, -0.0028532698380676095]

b = -2.77e-16

The r-square on the testing data is 0.3667004.

(e)

The model is different because the feature space is transformed after normalization. However, the r-square value on the training data is exactly the same. It means that both two method reached the global minima after gradient descent. However, doing normalization make the number of training iterations decreased significantly

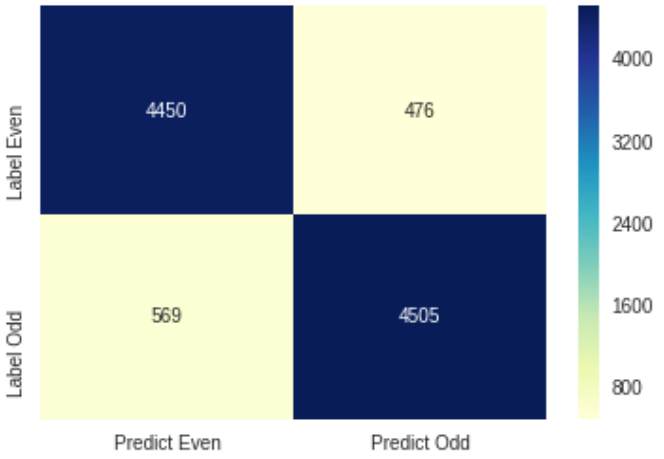
(f)

The final learning rate for trainning the above model without normalization is 0.0000003. After a few experiments, if the learning rate is bigger than 0.0000005, the global minima cannot be reached. Instead, the global minima was skiped during each iteration and the loss was increasing dramatically during each iteration. If the learning rate is smaller than 0.00000001, the global minima can still be reached; however, it takes significantly more iterations, and the learning time is extremely slow.

**Question 2 (Classification)**

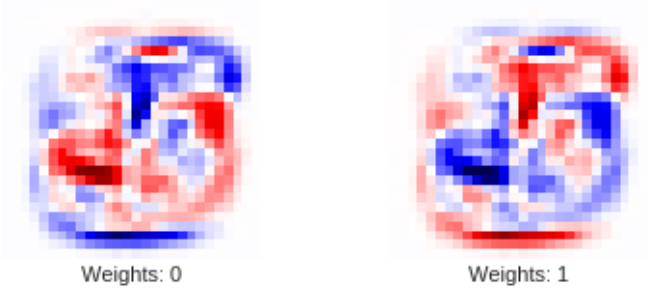
In order to classify even and odd numbers, I modified the output space from 1*10 to 1*2 where index 0 stands for even number and index 1 stands for odd number using one-hot representation. Changing the size of the weight from [784,10] to [784,2] and b from [10] to [2] to mach the shape of output. The training data labels should also be changed during each iteration. Same thing is done for testing data labels. The accuracy of prediction for the testing data is around 0.896.

Confusion matrix:




(a)

Images in the first row are 10 odd numbers that incorrectly predicted and the second row are the 10 even numbers that incorrectly classified. The majority of misclassified numbers are 3,4,8,9. As the misclassified images shown, we can clearly see that 3 looks similar to 8 and it's also possible that the model misclassified 8 to 3. 3 and 8 has different parity. Similarly, 4 and 9 looks similar and has different parity. The number 2 in the image could be interpreted as 7 or 3 since they look similar. Also, 5 could be interpreted as 6 and 7 as 2. Here's a visualization of the weight learned when trainning the model.

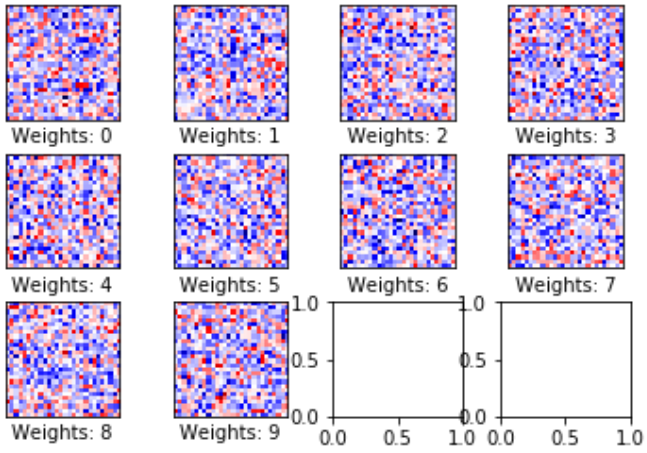



(b)

In order to change the output space from 1 to 2 by adding one extra layer without modifying the previous layers, I add a fully connected layer with two neurons to change the output size from 10 to 2. I also changed the activation function for the first layer from softmax to sigmoid for better accuracy. The output space for trainning data and testing data are modified in the same way as in part a. The accuracy is 0.90 for the testing data which is slightly better than part a.

(c)

There are differences between the method of modifying the last layer and adding a new layer. By modigying the layer, the number of neurons decreased from 10 to 2. Adding a new layer keeps the previous 10 neurons and use add an extra fully connected layer with 2 neurons to get the correct output shape. There are two hidden layers instead of one. Each neuron is a representation of the input's transformation via one feature(w) learnt from gradient descent. In other words, each layer will learn one kind of feature. By modifying neurons from 10 to 2 in part a, each neurons was originally supposed to learn 0 to 9 but changed to learn even and odd. One neuron learns even, the other learns odd. By adding new layers in part 3, the task of identifying even and odd was divided into two subtasks. First layer learn one feature representation and the second layer learn another. The combination of two features can help achieve the goal of identifying even and odd. For example, ideally, the first layer can identidy 0 to 9 and the second layer identify which number is even (0,2,4,6,8) and which number is odd (1,3,5,7,9). Unfortunately, by initializing each variable to a randomly the feature learnt in layer one is not as what I expected. However, the accuracy increased to 0.90. It's reasonable since there are more layers and more neurons.




**Question 3 (K-Nearest Neighbors)**

(a)

The error rate of my classification when k = 1 is 0.6460646064606461

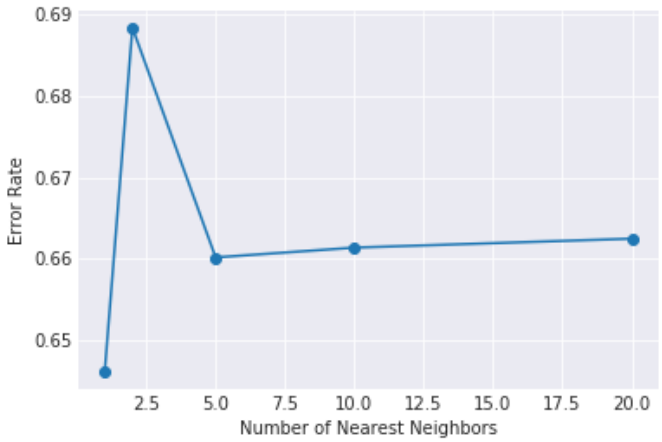
(b)

K = 2: 0.6883688368836883

K = 5: 0.6601660166016602

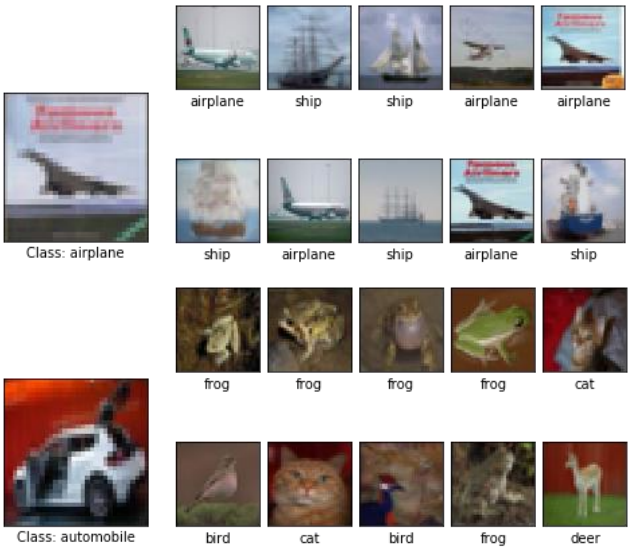
K = 10: 0.6601660166016602

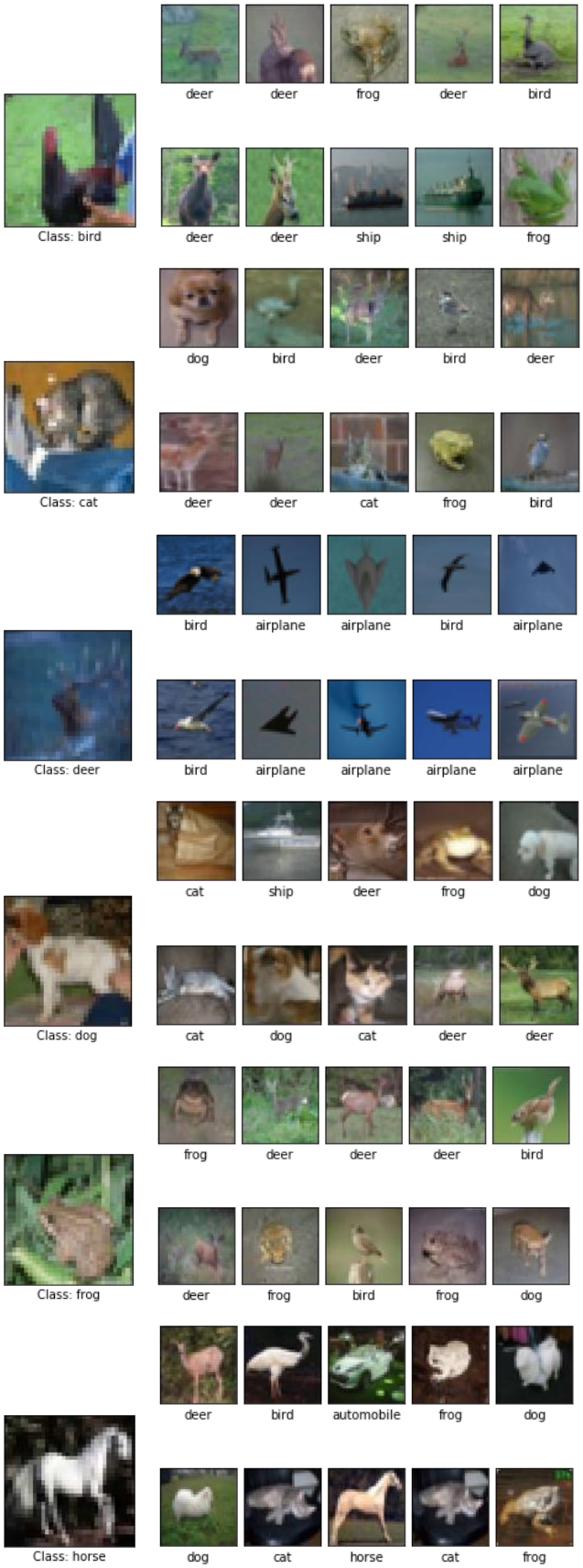
K = 20: 0.6624662466246625




The error rate does not always decreses with k. It increase at first, then decrease and at last increase. The error rate should not always decrease with k. When k is relatively small, calculating more neighbors may include more same class data insdead of data that is not from the same class and the error rate will decrease. However, if k is significantly small like 1 or 2, the error rate is unpredictable. It may either increase or decrease. If k is relatively big, calculating more neighbors may include more data that are not from the same class than data that from the same class. Thus the error rate will increase eventually as the number of nearest neighbors goes up.

(c)




airplane
automobile
ship
ship
ship


Class: ship
ship
ship
ship
ship
ship


ship
ship
ship
ship
airplane


Class: truck
ship
ship
ship
ship
horse