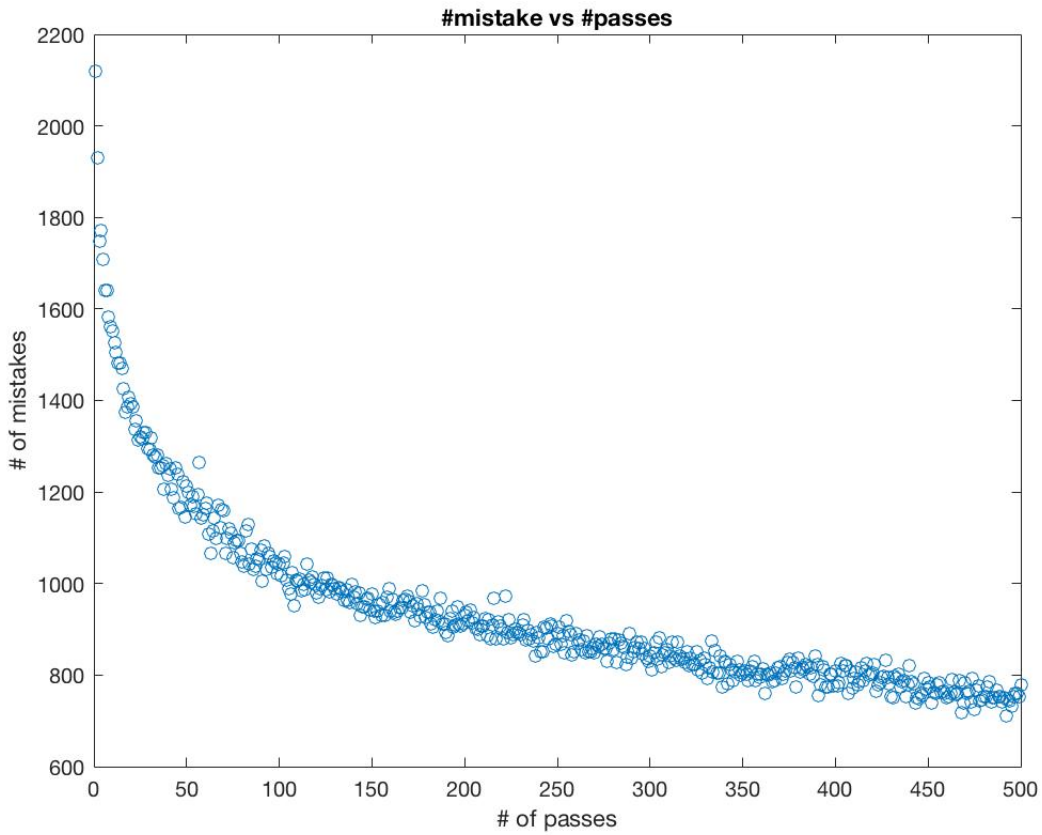
Assignment 1

Ronghao Yang ID:20511820
Session: CS698, 4:00pm-5:20pm

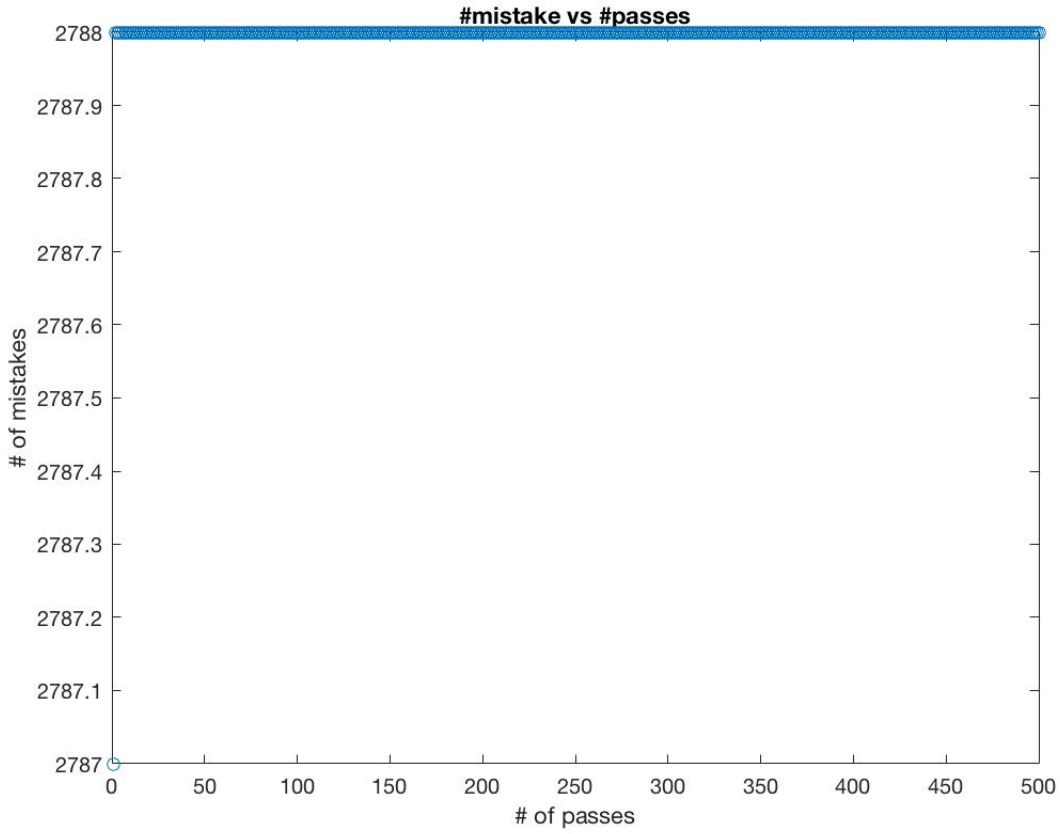
September 26, 2017

1 Exercise 1

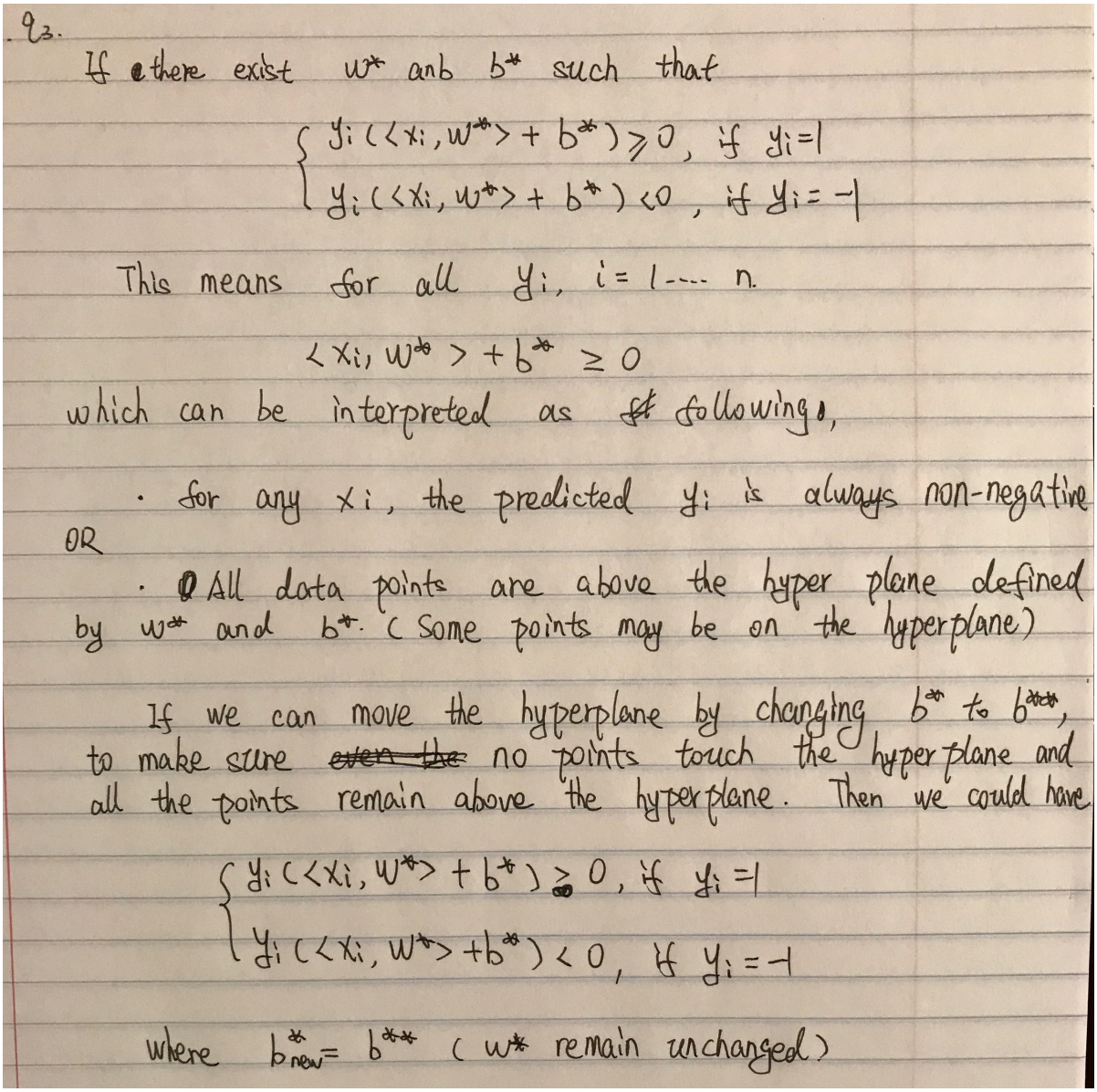
1.1 question 1



1.2 question 2



1.3 question 3



q3.

If there exist w^* and b^* such that

$$\begin{cases} y_i(\langle x_i, w^* \rangle + b^*) \geq 0, & \text{if } y_i = 1 \\ y_i(\langle x_i, w^* \rangle + b^*) < 0, & \text{if } y_i = -1 \end{cases}$$

This means for all y_i, $i = 1 \dots n$.

$$\langle x_i, w^* \rangle + b^* \geq 0$$

which can be interpreted as following,

- for any x_i, the predicted y_i is always non-negative

OR

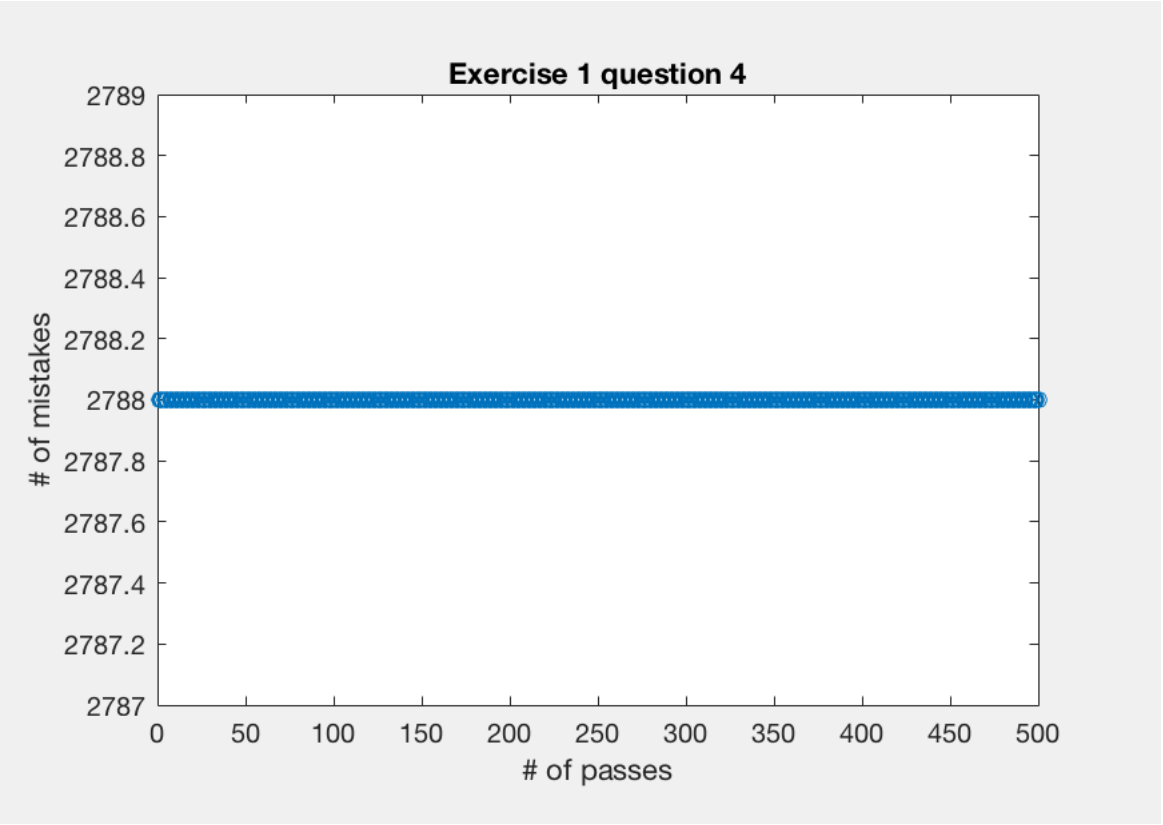
- All data points are above the hyper plane defined by w^* and b^*. (Some points may be on the hyperplane)

If we can move the hyperplane by changing b^* to b^{**}, to make sure ~~even the~~ no points touch the hyperplane and all the points remain above the hyperplane. Then we could have

$$\begin{cases} y_i(\langle x_i, w^* \rangle + b^*) \geq 0, & \text{if } y_i = 1 \\ y_i(\langle x_i, w^* \rangle + b^*) < 0, & \text{if } y_i = -1 \end{cases}$$

where $b^*_{new} = b^{**}$ (w^* remain unchanged)

1.4 question 4



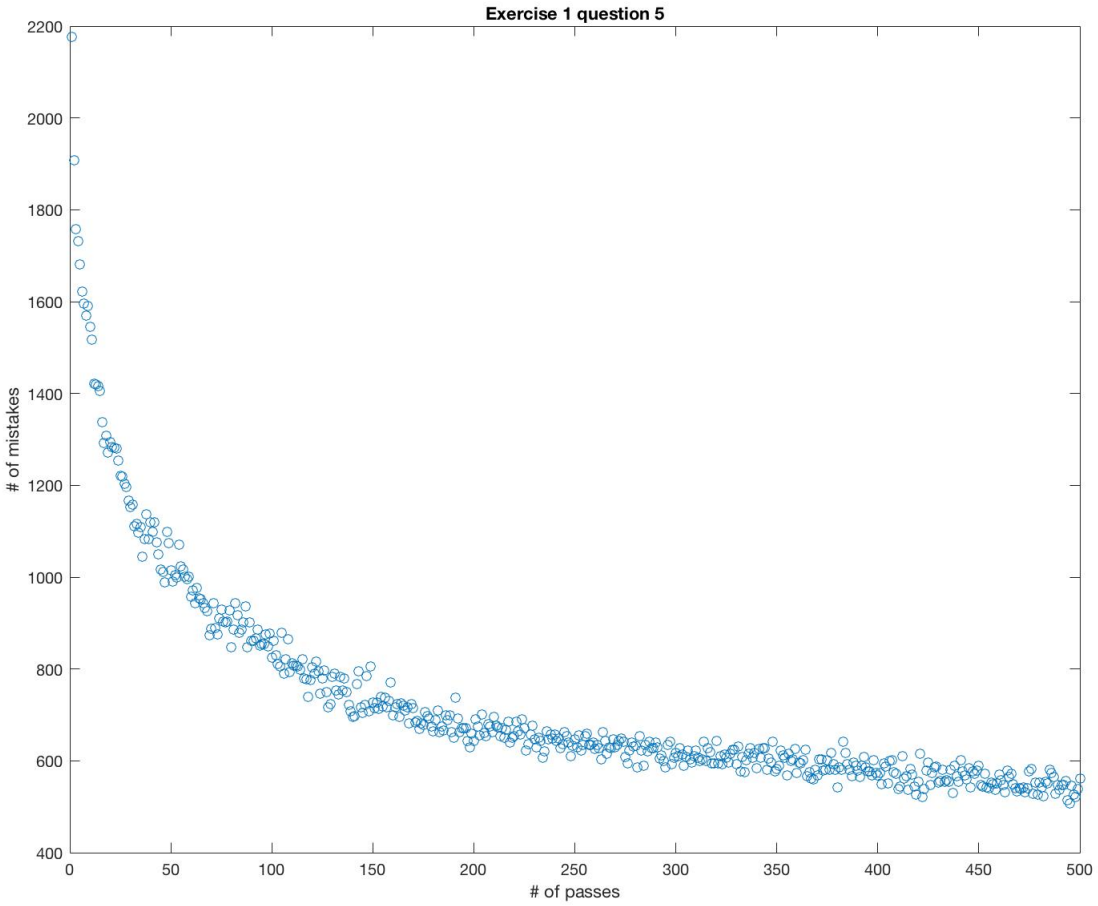
StepSize:
As what has been discussed on Piazza, both W and b are always positive, since all the values in X are also possitive, $< W, X > +b$ is always evaluated to positive. This means, whenever y_i is -1, $y_i(< x_i, w^* > +b^*)$ is negative, which is reported as a mistake. Therefore, the total number of mistake in each pass is always the number of -1s in y. Tuning the step_size wouldn't reduce the number of mistakes made by the algorithm. [please refer to question 5 for calculating step size using cross validation]

1.5 question 5

To transform the data matrix X so that the non-negativity assumption can be ignored, we do the following:
$X_{new} = [X\ -X]$, eq: if $X = [x_1;\ x_2; ....; x_n]$, then $X_{new} = [x_1\ -x_1;\ x_2\ -x_2; ....; x_n\ -x_n]$
Regarding the data set we have, the original data set has dimension 4601×57, after transformation, X_{new} has dimension 4601×114. If we append ones to X[then w is (w;b)] before we do the transformation, our new data matrix has dimension 4601×116.
By doing so, the prediction can still be negative even no negative numbers show in the weight vector w. We know that w = {max(w,0)− max(-w,0)}

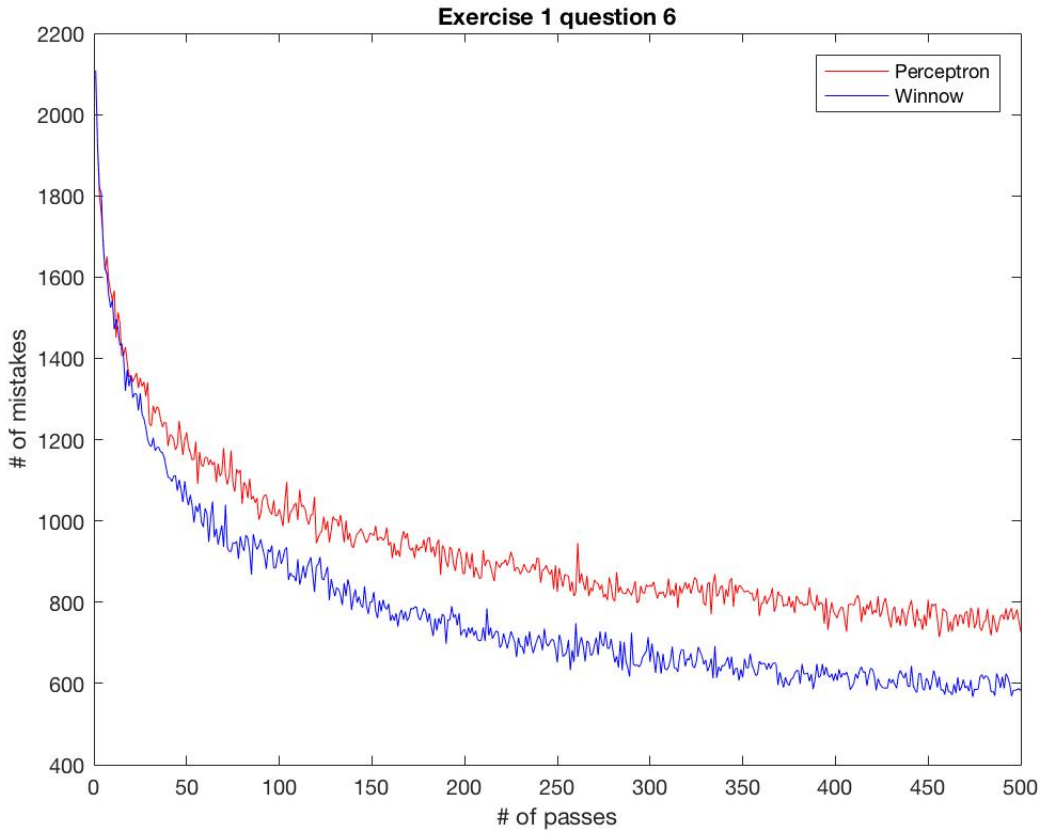
$$Y_{predictions} =< x, max(w, 0) > + < -x, max(-w, 0) > \tag{1}$$

The new weight vector w in the winnow algorithms is (max(w,0),max(w,0)).
By running cross validation (step size from $\frac{1}{m}$ to $\frac{10}{m}$, where m is the maximum value in X) on the new transformed matrix, the optimal is found to be $\frac{2}{m}$.
Using the optimal step size, the graph is the following:



1.6 question 6

Perceptron algorithm uses additive update, while Winnow algorithm uses multiplicative update. Both algorithms are computationally efficient, and guaranteed to find a hyperplane for linearly separable data. However, perceptron algorithm run in $O(n)$ time, winnow algorithm runs in $O(logn)$ time, where n is the number of features. After adding 1000 random features to the data set, we can see that winnow algorithm converges faster than perceptron algorithm. When a data set has a large number of features, winnow algorithm is preferred.



2 Exercise 2

2.1 question 1

```
lambda 0, validation error   10.56228, training error    9.69430, test error  370.21575, density of w 1
lambda 10, validation error   11.40233, training error   10.52252, test error   98.71393, density of w 1
lambda 20, validation error   12.09603, training error   10.91801, test error  111.94803, density of w 1
lambda 30, validation error   12.69878, training error   11.38163, test error  127.01162, density of w 1
lambda 40, validation error   12.96539, training error   11.90295, test error  139.58307, density of w 1
lambda 50, validation error   13.92639, training error   12.46094, test error  149.21522, density of w 1
lambda 60, validation error   14.62812, training error   13.03911, test error  156.24080, density of w 1
lambda 70, validation error   15.18557, training error   13.62560, test error  161.14018, density of w 1
lambda 80, validation error   15.78968, training error   14.21206, test error  164.36497, density of w 1
lambda 90, validation error   16.36429, training error   14.79268, test error  166.29362, density of w 1
lambda 100, validation error   17.46053, training error   15.36347, test error  167.22905, density of w 1
```

2.2 question 2

1. Multiplying y by 10^6

```
lambda 0, validation error   712031.33876, training error   654506.23393, test error  17564374.58755, density of w 1
lambda 10, validation error   704380.70837, training error   656929.56410, test error   7834585.55848, density of w 1
lambda 20, validation error   700156.89141, training error   657815.92055, test error   5609891.91516, density of w 1
lambda 30, validation error   698541.79256, training error   658638.67766, test error   4242654.78139, density of w 1
lambda 40, validation error   697181.10210, training error   659413.31282, test error   3317889.46179, density of w 1
lambda 50, validation error   695676.19677, training error   660129.47541, test error   2660447.58799, density of w 1
lambda 60, validation error   695766.59876, training error   660784.45790, test error   2176338.38325, density of w 1
lambda 70, validation error   694013.70178, training error   661380.70034, test error   1809958.84493, density of w 1
lambda 80, validation error   693648.51447, training error   661922.90554, test error   1526399.82703, density of w 1
lambda 90, validation error   693325.41456, training error   662416.45670, test error   1302774.80983, density of w 1
lambda 100, validation error   692633.61120, training error   662866.66173, test error   1123556.93155, density of w 1
```

2. Multiplying x by 10^3

```
lambda 0, validation error  4562079.43058, training error       11.49341, test error       534.11532, density of w 1
lambda 10, validation error  2116817.15832, training error       15.87316, test error       144.72009, density of w 1
lambda 20, validation error  2137938.43680, training error       20.84814, test error       212.82442, density of w 1
lambda 30, validation error  2203371.88787, training error       26.00264, test error       270.77528, density of w 1
lambda 40, validation error  2230101.02113, training error       30.94108, test error       319.94698, density of w 1
lambda 50, validation error  2232001.65961, training error       35.57569, test error       361.14004, density of w 1
lambda 60, validation error  2288041.96715, training error       39.91076, test error       395.54599, density of w 1
lambda 70, validation error  2305568.20209, training error       43.97491, test error       424.37785, density of w 1
lambda 80, validation error  2329332.81144, training error       47.80002, test error       448.69825, density of w 1
lambda 90, validation error  2348397.39057, training error       51.41501, test error       469.37968, density of w 1
lambda 100, validation error  2437187.58193, training error       54.84455, test error       487.11745, density of w 1
```

2.3 question 3

```
lambda 0, validation error   780265.36124, training error        0.00000, test error    362389.65963, density of w 1
lambda 10, validation error       49.77098, training error        0.00916, test error       126.50265, density of w 1
lambda 20, validation error       47.64196, training error        0.03488, test error       125.22026, density of w 1
lambda 30, validation error       50.16064, training error        0.07482, test error       124.02490, density of w 1
lambda 40, validation error       50.04115, training error        0.12706, test error       122.90805, density of w 1
lambda 50, validation error       46.60753, training error        0.18994, test error       121.86231, density of w 1
lambda 60, validation error       46.51264, training error        0.26208, test error       120.88119, density of w 1
lambda 70, validation error       45.73973, training error        0.34230, test error       119.95897, density of w 1
lambda 80, validation error       44.91099, training error        0.42957, test error       119.09059, density of w 1
lambda 90, validation error       49.76781, training error        0.52303, test error       118.27156, density of w 1
lambda 100, validation error       46.92833, training error        0.62191, test error       117.49789, density of w 1
```

2.4 question 4

```
lambda 0, validation error       12.10293, training error       10.76902, test error        76.70950, density of w 1
lambda 10, validation error       12.42085, training error       11.09634, test error       189.70738, density of w 8.571429e-01
lambda 20, validation error       12.78808, training error       11.75834, test error       316.67972, density of w 7.857143e-01
lambda 30, validation error       13.28065, training error       12.34622, test error       387.14554, density of w 7.857143e-01
lambda 40, validation error       14.12547, training error       12.85613, test error       406.56687, density of w 7.857143e-01
lambda 50, validation error       14.39393, training error       13.27241, test error       391.46662, density of w 7.857143e-01
lambda 60, validation error       14.93735, training error       13.64487, test error       357.17488, density of w 7.857143e-01
lambda 70, validation error       15.22417, training error       13.94502, test error       315.71344, density of w 7.857143e-01
lambda 80, validation error       15.39130, training error       14.21881, test error       271.62432, density of w 7.857143e-01
lambda 90, validation error       15.96225, training error       14.51600, test error       221.80226, density of w 7.857143e-01
lambda 100, validation error       16.21604, training error       14.77973, test error       177.72648, density of w 7.857143e-01
```

2.5 question 5

```
lambda 0, validation error      537.07889, training error        0.03930, test error     49565.19361, density of w 1
lambda 10, validation error      416.93488, training error        0.32935, test error     44672.85182, density of w 7.700000e-01
lambda 20, validation error      295.20777, training error        1.16204, test error     32478.49695, density of w 5.240000e-01
lambda 30, validation error      215.36297, training error        2.65830, test error     15717.65011, density of w 3.950000e-01
lambda 40, validation error      156.51500, training error        4.46587, test error      5111.27085, density of w 3.080000e-01
lambda 50, validation error      123.80871, training error        6.41977, test error      1181.98763, density of w 2.870000e-01
lambda 60, validation error      106.86365, training error        8.21383, test error       218.94074, density of w 2.480000e-01
lambda 70, validation error       99.53219, training error       10.47893, test error       211.73736, density of w 2.180000e-01
lambda 80, validation error       93.18151, training error       12.78934, test error       200.47883, density of w 2.000000e-01
lambda 90, validation error       80.56465, training error       14.87098, test error       176.31721, density of w 1.800000e-01
lambda 100, validation error       69.76385, training error       16.81678, test error       159.36938, density of w 1.530000e-01
```

Comparing the results with the result from 2.3, we can see that validation errors in 2.3 do not differ much, the *lambda* that produces the smallest validation error does not guarantee the smallest test error. However, in this question, we can see that validation errors for each lambda differ much from each other, the lambda value that produces the smallest validation error also produces the smallest test error.
And also, the percentage of $0s$ of w in lasso algorithm is greater than 0%, which means in lasso algorithm, not all features are used for prediction, some minor features are ignored. However, in ridge algorithm, the density of w is always 1, which means all features are considered in the process of prediction.