Q1. Given $X = \min(X_1, X_2)$

Let $g(x)$ be the probability distribution function for X. and $G(x)$ be the corresponding cummulative distribution function.

Also, Let $F(x)$ be the cummulative distribution function for X_1 and X_2.

$$G(x) = P(X \le x) = P(\min(X_1, X_2) \le x)$$

$$= 1 - P((X_1 > x) \text{ and } (X_2 > x))$$

$$= 1 - P(X_1 > x) \cdot P(X_2 > x) \quad \text{// since } X_1 \text{ and } X_2 \text{ are independent}$$

$$= 1 - (1 - F(x)) \cdot (1 - F(x))$$

$$G(x) = 1 - (1 - F(x))^2$$

We know that for the range $[0, 1]$

$$\boxed{g(x) = G'(x) = \frac{d}{dx}(1 - (1 - F(x))^2)} \quad 0 \le x \le 1$$

Also since X_1 and X_2 are continous uniformly distributed random variable over $[0, 1]$

$$F(x) = \begin{cases} 0, & x < 0 \\ x, & 0 \le x \le 1 \\ 1, & x > 1 \end{cases}$$

therefore $g(x) = \frac{d}{dx}(1 - (1 - x)^2) = 2(1 - x)$ for $0 \le x \le 1$

(i) $E(X) = \int_0^1 x\, g(x)\, dx$

$= \int_0^1 x \cdot 2(1-x)\, dx$

$= \int_0^1 (2x - 2x^2)\, dx$

$\Rightarrow \left[x^2 - \frac{2x^3}{3} \right]_0^1 \Rightarrow 1 - \frac{2}{3} = \boxed{1/3}$. Ans

(ii) $Var(X) = E((X-\mu)^2) = \int_0^1 (x-\mu)^2 g(x)\, dx$

$= \int_0^1 (x - 1/3)^2 2(1-x)\, dx$

$\Rightarrow \int_0^1 \frac{2}{9}(3x-1)^2(1-x))\, dx \Rightarrow$

$\Rightarrow \int_0^1 \frac{2}{9}(9x^2 + 1 - 6x)(1-x)\, dx$

$\Rightarrow \int_0^1 \frac{2}{9}(9x^2 - 9x^3 + 1 - x - 6x + 6x^2)\, dx$

$\Rightarrow \int_0^1 \frac{2}{9}(-9x^3 + 15x^2 - 7x + 1)\, dx$

$\Rightarrow \frac{2}{9}\left(-\frac{9x^4}{4} + \frac{15x^3}{3} - \frac{7x^2}{2} + x \right)\Big|_0^1$

$\Rightarrow \frac{2}{9}\left(-\frac{9}{4} + \frac{15}{3} - \frac{7}{2} + 1 \right) \Rightarrow \frac{2}{9}(1/4) = \boxed{\frac{1}{18}}$. Ans

(iii) For $Cov(X, X_1)$

we know by the Law of total probability that

$$Cov(X_1, \min(X_1, X_2))$$

$$\Rightarrow Cov(X_1, X_1)\, P[X_1 = \min(X_1, X_2)] + Cov(X_1, X_2)\, P[X_2 = \min(X_1, X_2)] \longrightarrow (1)$$

$$\Rightarrow Cov(X_1, X_1) \times P[X_1 \leq X_2] + \underbrace{Cov(X_1, X_2)\, P(X_2 < X_1)}$$

this term is 0 as X_1 and X_2 are independant

$$\Rightarrow Cov(X_1, X_1)\, P[X_2 > x \mid X_1 = x] \quad \text{for } x \in (0, 1]$$

$$\Rightarrow Var(X_1) \cdot [(1 - P(X_2 \leq x)) \cdot P(X_1 = x)] \quad // X_1 \text{ and } X_2 \text{ are independe}$$

$$\Rightarrow (E[X_1^2] - (E[X_1])^2) \cdot \left[\int_0^1 (1-x)\, dx\right]$$

$$\Rightarrow \left[\int_0^1 x^2\, dx - (1/2)^2\right] \cdot \left[\int_0^1 (1-x)\, dx\right]$$

$$\Rightarrow (1/3 - 1/4) \cdot [1 - 1/2] \Rightarrow 1/12 : 1/2$$

$$\boxed{\Rightarrow 1/24}$$ Ans

Q2.

1.) Number of eggs laid ~~and the~~ follows a poisson distribution and number of ~~turtles~~ hatchlings reaching is independent.

P(k eggs laid and m reach shore)

$$= \frac{\lambda^k}{k!} e^{-\lambda} \cdot P(m \text{ reach shore} \mid k \text{ eggs laid})$$

$$= \frac{\lambda^k}{k!} e^{-\lambda} \cdot p^m \cdot (1-p)^{k-m}$$

now for N turtles joint likelihood of E and H

$$= \prod_{i=1}^{N} \frac{\lambda^{k_i}}{k_i!} e^{-\lambda} \cdot p^{m_i} (1-p)^{k_i - m_i}$$

As the turtles are independent

$\therefore$ Log-likelihood of E and H

$$= \sum_{i=1}^{N} \ln\left(\frac{\lambda^{k_i}}{k_i!} e^{-\lambda} \cdot p^{m_i} (1-p)^{k_i - m_i} \right) \quad \cdots (1)$$

$$= \sum_{i=1}^{N} \left(\ln\left(\frac{\lambda^{k_i}}{k_i!} e^{-\lambda} \right) + m_i \ln p + (k_i - m_i) \ln(1-p) \right)$$

$$= \sum_{i=1}^{N} \left(k_i \ln \lambda - \lambda - \ln(k_i!) + m_i \ln p + (k_i - m_i) \ln(1-p) \right)$$

Q2.2. $\lambda_{MLE} = \arg\max_{\lambda}$ (log-likelihood)

$\Rightarrow$ λ_{MLE} when $\frac{d}{d\lambda}$ (log-likelihood) $= 0$

$$\frac{d}{d\lambda} (\text{log-likelihood}) = \sum_{i=1}^{N} \left(\frac{k_i}{\lambda} - 1 \right) = 0$$

$$\Rightarrow \boxed{\lambda_{MLE} = \frac{\sum_{i=1}^{N} k_i}{N}}$$

where k_i is the number of eggs i-th turtle laid

$$p_{MLE} = \arg\max_{p} (\text{log-likelihood})$$

$\Rightarrow$ p at $\frac{d}{dp}$ (log-likelihood) $= 0$

$$\Rightarrow \frac{d}{dp} (\text{log-likelihood}) = \frac{\sum_{i=1}^{N} m_i}{p} + \frac{-\sum_{i=1}^{N} (k_i - m_i)}{1-p} = 0$$

$$\Rightarrow \left(\sum_{i=1}^{N} m_i \right) (1-p) = p \left(\sum_{i=1}^{N} k_i - \sum_{i=1}^{N} m_i \right)$$

$$\Rightarrow \sum_{i=1}^{N} m_i - p \cancel{\sum_{i=1}^{N} m_i} = p \sum_{i=1}^{N} k_i - p \cancel{\sum_{i=1}^{N} m_i}$$

$$\Rightarrow \boxed{p_{MLE} = \frac{\sum_{i=1}^{N} m_i}{\sum_{i=1}^{N} k_i}}$$

where m_i is the number of hatchling which made it to shore when i-th turtle laid k_i eggs.

Q2.3. λ_{MLE} for observed values

$$\Rightarrow \frac{\sum_{i=1}^{10} k_i}{10} \Rightarrow \frac{(8+9+6+4+1+5+2+12+9+7)}{10}$$

$$\Rightarrow \frac{63}{10} = 6.3$$

$$p_{MLE} = \frac{\sum_{i=1}^{10} m_i}{\sum_{i=1}^{10} k_i} = \frac{(5+6+4+3+0+5+2+9+8+6)}{63}$$

$$\Rightarrow \frac{48}{63} = 0.7619$$

Q. 3 ①

The naive algorithm can be pseudo-coded as

Let X be of dimension N and d
(number of data points) features

```
set LOOCV = 0
FOR i in Range (0, N)
    ŵ = (X^(-i)T  X^(-i))^-1  X^(-i)T . Y^(-i)
    ŷ_i^(-i) = x_i ŵ
    LOOCV += (y_i - ŷ_i^(-i))^2
endFOR.
```

where $X^{(-i)} \Rightarrow X$ after removing the i^{th} row

$Y^{(-i)} = Y$ after removing the i^{th} row

each iteration:
- Complexity calculating $X^T X = O(d^2 N)$
- Complexity of calculating $(X^T X)^{-1} = O(d^3)$
- " " " $(X^T X^{-1}) X^T = O(d^2 N)$
- " " " $(X^T X^{-1}) X^T Y = O(dn)$
- " " " $x_i \hat{w} = O(dn)$

∴ Total Complexity $= N \left(O(d^2 N) + O(d^3) + O(d^2 N) + O(dn) + O(dn) \right)$
for N iteration

$\Rightarrow O(d^2 N^2) + O(d^3 N)$

for $n \approx d = O(N^3)$

for small $d \Rightarrow O(N^2 d)$

for large $d = O(d^3 N)$

} Ans.

Q3 (2). $\hat{y}_i$ in terms of H and Y

we know $\Rightarrow \hat{Y} = HY$

$$\hat{Y} = \begin{bmatrix} H_{11} & \cdots & H_{1n} \\ & & \\ H_{n1} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} Y_{11} \\ \vdots \\ Y_{n1} \end{bmatrix}$$

$$\hat{y}_i = \sum_{j=1}^{N} H_{ij} \cdot Y_{j1}$$

Q3 (3)

$$z_j = \begin{cases} y_j, & j \neq i \\ \hat{y}_i^{(-i)}, & j = i \end{cases}$$

SSE for $z_j \Rightarrow \sum_{i=1}^{N} (z_i - \hat{y}_i)^2$

$\hat{Y}^{(-i)}$ minimize $\sum_{j \neq i} (y_j - \hat{y}_j^{(-i)})^2$

↑

By definition.

SSE z_j can be broken down as $\Rightarrow \sum_{j \neq i} (y_j - \hat{y}_j)^2 + (\hat{y}_i^{(-i)} - \hat{y}_i)^2$

$\hat{Y}^{(-i)}$ minimizes $\sum_{j \neq i} (y_j - \hat{y}_j)^2$

and sets $(\hat{y}_i^{(-i)} - \hat{y}_i)^2$ to zero

hence $\hat{Y}^{(-i)}$ also minimizes z_j.

Q 3. (4) $\hat{y}_i^{(-i)}$ in terms of H and Z

from part three we know that $\hat{Y}^{(-i)}$ minimizes Z.

hence $\hat{Y}^{(-i)} = HZ$ (from question text)

$$\therefore \hat{y}_i^{(-i)} = \begin{bmatrix} H_{11} & \cdots & H_{1n} \\ & & \\ H_{n1} & \cdots & H_{nn} \end{bmatrix} \begin{bmatrix} y_1 \\ \vdots \\ \hat{y}_i^{(-i)} \\ \vdots \\ y_n \end{bmatrix}$$

$$\boxed{\hat{y}_i^{(-i)} \Rightarrow \sum_{j=1}^{N} H_{ij} \cdot Z_j}$$

Q 3. (5) we know

$$\hat{y}_i = \sum_{j=1}^{N} H_{ij} \cdot Y_j$$

$$\hat{y}_i^{(-i)} = \sum_{j=1}^{N} H_{ij} \cdot Z_j$$

$$\hat{y}_i - \hat{y}_i^{(i)} = \sum_{j=1}^{N} H_{ij} \cdot Y_j - \sum_{j=1}^{N} H_{ij} \cdot Z_j$$

$\Rightarrow$ since $y_i = z_i$ for $\forall j \neq i$ $\Rightarrow$ all these terms cancel out

$$\hat{y}_i - \hat{y}_i^{(-i)} \Rightarrow H_{ii} \cdot y_i - H_{ii} \cdot Z_i$$

$$\boxed{\hat{y}_i - \hat{y}_i^{(-i)} \Rightarrow H_{ii}\, y_i - H_{ii} \cdot \hat{y}_i^{(-i)}} \quad Q.E.D.$$

Q3. (6)

$$\text{LOOCV} = \sum_{i=1}^{n} (y_i - \hat{y}_i^{(-i)})^2 \quad \hookrightarrow (1)$$

from the result in part (5) we know that

$$\hat{y}_i - \hat{y}_i^{(-i)} = H_{ii} y_i - H_{ii} \hat{y}_i^{(-i)}$$

$$\hat{y}_i - H_{ii} y_i = (1 - H_{ii}) \hat{y}_i^{(-i)}$$

$$\hat{y}_i^{(-i)} = \left(\frac{\hat{y}_i - H_{ii} y_i}{(1 - H_{ii})} \right) \quad \hookrightarrow (2)$$

Combining eq^n (2) and (1) we get

$$\text{LOOCV} = \sum_{i=1}^{n} \left(y_i - \frac{\hat{y}_i - H_{ii} y_i}{1 - H_{ii}} \right)^2$$

$$\Rightarrow \sum_{i=1}^{n} \left(\frac{y_i - y_i H_{ii} - \hat{y}_i + H_{ii} y_i}{1 - H_{ii}} \right)^2$$

$$\boxed{\text{LOOCV} \Rightarrow \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - H_{ii}} \right)^2} \quad \text{Q.E.D.}$$

Complexity of calculating LOOCV:

⇒ Complexity of H + complexity of $\sum_{i=1}^{N}\left(\frac{y_i-\hat{y}_i}{1-H_{ii}}\right)^2$
+ complexity of $\hat{Y}$

⇒ Complexity of $\underbrace{X(X^TX)^{-1}X^T}_{H}$ ~~+ O(N)~~
+ complexity HY

⇒ + $O(N)$

⇒ Complexity of matrix inversion $(X^TX)^{-1}$

$O(Nd^2) + O(N^2) + O(N)$

$+ O(N^2d)$

⇒ $O(d^3) + O(Nd^2) + O(N^2d)$

⇒ for small $d \Rightarrow O(N^2d)$
for large $d \Rightarrow O(d^3)$
for $d \cong N \Rightarrow O(N^3)$

Ans

Q4 ① (a). The e_{train} will decrease (as we are overfitting)

(b) The e_{test} will increase (" we trained to hard for

(c) $\hat{w}$ becomes more dense. (magnitude increases)

(d) number of non zero elements increase in $\hat{w}$.

② (a) The e_{train} will increase (as we have a biased solution)

(b) The e_{test} will also increase ↓ biased solution

(c) $\hat{w}$ becomes sparser. (magnitude ~~increase~~ decreases)

(d) number of non zero elements decrease in $\hat{w}$

Q.4 ②

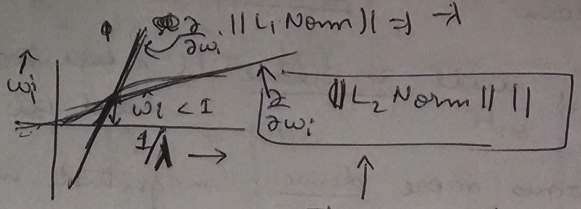
① $\frac{\partial}{\partial \hat{w}_i}(\lambda \|\hat{w}\|_1) \Rightarrow \begin{cases} -\lambda & \text{for } \hat{w}_i < 0 \\ \lambda & \text{for } \hat{w}_i > 0 \end{cases}$

② $\frac{\partial}{\partial \hat{w}_i} \lambda \|\hat{w}\|_2^2 = \lambda \frac{d}{d\hat{w}_i} \sum_{i=1}^{d} (\hat{w}_i)^2 = 2\lambda \hat{w}_i$

③ assume to part ① and part ② suggest that when the values of w_i are very small, the L_1 regularization pushes them towards 0 at a constant rate $|\lambda|$, while the L_2 regularizations pushes them towards 0 at the rate $2\lambda \hat{w}_i$. since $\hat{w}_i$ is small (≤ 1), the product $2\lambda \hat{w}_i$ reduces and the rate at which $\hat{w}_i$ are penalized decreases.

Graphically:

This would mean that



This slope is lower for smaller $\hat{w}_i$ in case of L_2 Regularization (Ridge)

This in turn means that for larger λ, L_1||Norm|| would push a lot faster than L_2||Norm|| hence the weights would reach zero a lot faster.

As a answer to the question:

⇒ Values of w_i for which behaviours differ ⇒ $\hat{w}_i < 1$ (or even smaller)

⇒ what does this mean of estimates of w_i for large λ ?

This means that for Lasso $\hat{w}_i \Rightarrow 0$

but for Ridge $\hat{w}_i << 1$ but not 0

Ques 5.1 Code attached.

Ques.5.2
1.

Overall the lasso is a really good predictor of non-zero features.

Noticing the values of Precision and Recall across the regularization path, we notice the following sparsity pattern:

As λ_{max}decreases by a constant factor, the Recall of the parameters increases, as decreasing λ makes more and more feature weights non zero, until all the relevant features are recalled, after which decreasing lambda further starts making all the other feature weights non-zero and the overfitting of our solution increases.

The Precision on the other hand, starts out as near to one because the initial features are identified are a subset of true features, but as λ keeps on decreasing, more and more features which are not the part of true features start becoming non zero due to overfitting and precision keeps dropping and approaches 0 and we reach the naïve least squared solution.

The graph below generated using synthetic data summarizes the same .

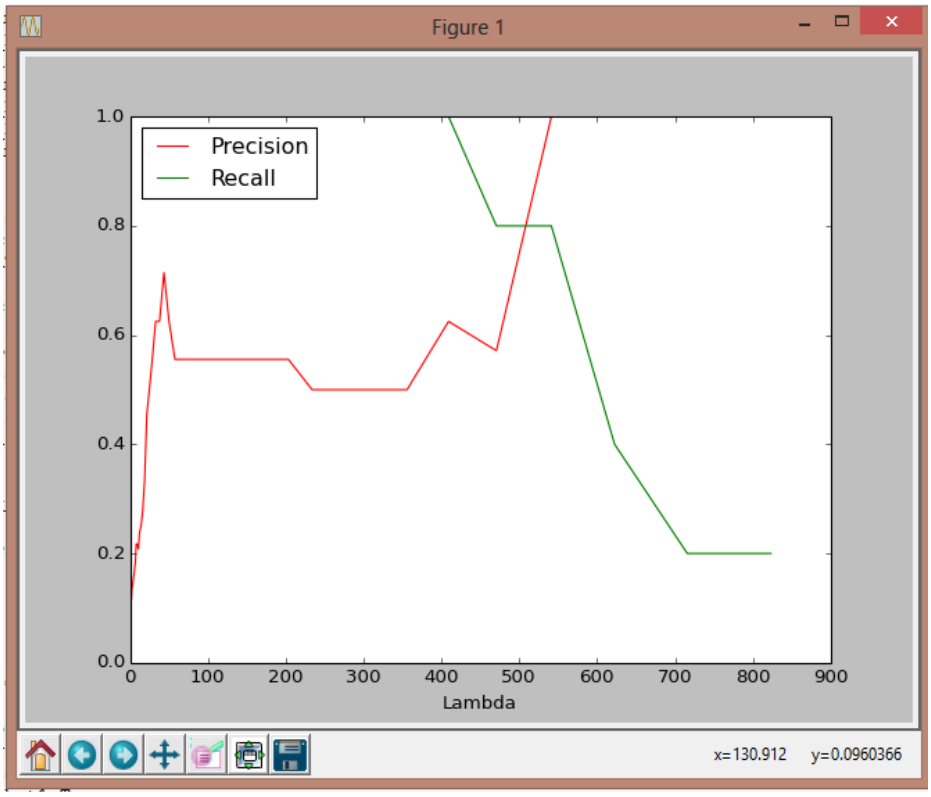


Figure 1:Graph1.

Ques 5.2

2.

When we use σ = 10, we are essentially increasing noise in the actual data a lot. If we use the values of λ for which the solution in part 1 worked well, we notice that the recall and precision of the solution both decrease (graph 2), this is because more variance in data requires a more complex model solution, but if we use the λ from the previous part, we do not account for the increased complexity requirement and the "bias error" of our solution increases.

Bias error occurs because we have a very simple solution (penalization of weights higher), to reduce the bias error we need to decrease the regularization term λ. Figure 3 shows the same graph with reduced λ.

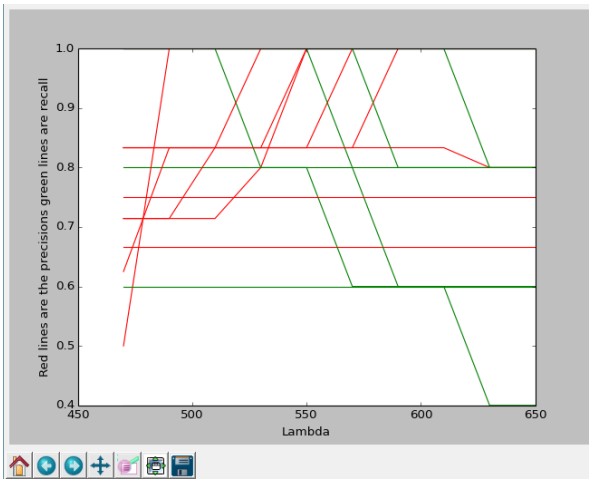


Figure 2 Recall and precision for lambdas: [650,630,610,590,570,550,530,510,490,470]. ***sigma = 10.***

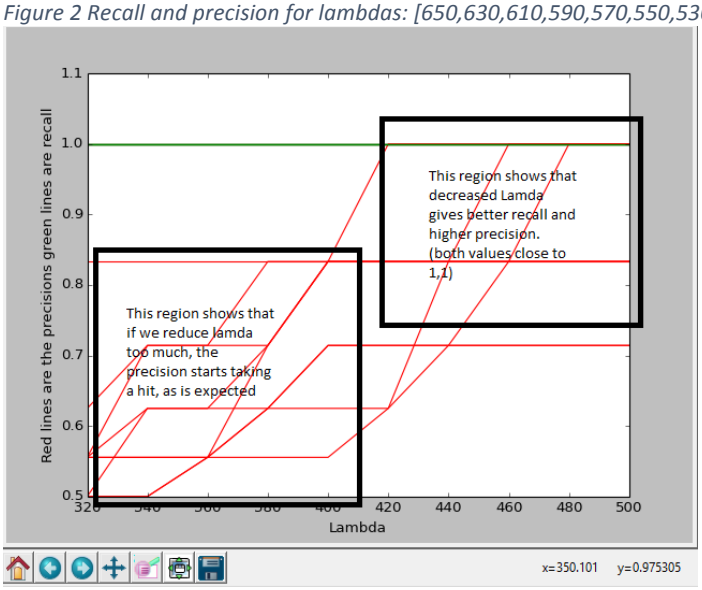


Figure 3 Recall and precision for lambdas: [500,480,460,440,420,400,380,360,340,320]. ***sigma = 10.***

Ques 5.2

3.

We solve lasso at several possible λ, Initializing at λ_{max}, and decreasing by constant ratio in every turn. This experiment is repeated multiple times for several combination of (N,d). we notice that as d is increased and N is kept constant, the range [lambda-precision, lambda- recall] starts increasing. Here lambda-recall is the minimum lambda at which recall is 1 and lambda-precision is the maximum lambda at which precision is zero. This range governs the number of iterations in the regularization path to reach lambda-precision.

The graphs below show the same.

Also, ideally we need to find a lambda which has recall = 1 and precision =1 and also penalizes the weights just as much is required so to avoid baising of the solution. The experiment shows that such a least biasing occurs at lambda-precision because the Sum(|W*-Wactual|) is minimum at lambda-precision.

This means that sooner we reach to lambda-precision, the better the solution is.

The results also show that if we double the magnitude of d, the change in N needed is far lesser than a double, hence, increase in N is sublinear in terms of D.

Therefore, N = O(log(d)). This is a significant result because the complexity of lasso is directly dependent on N and d. Hence if N has to increase with the same rate as D, the computational complexity of lasso takes a lot more hit. (To get similar predictions)

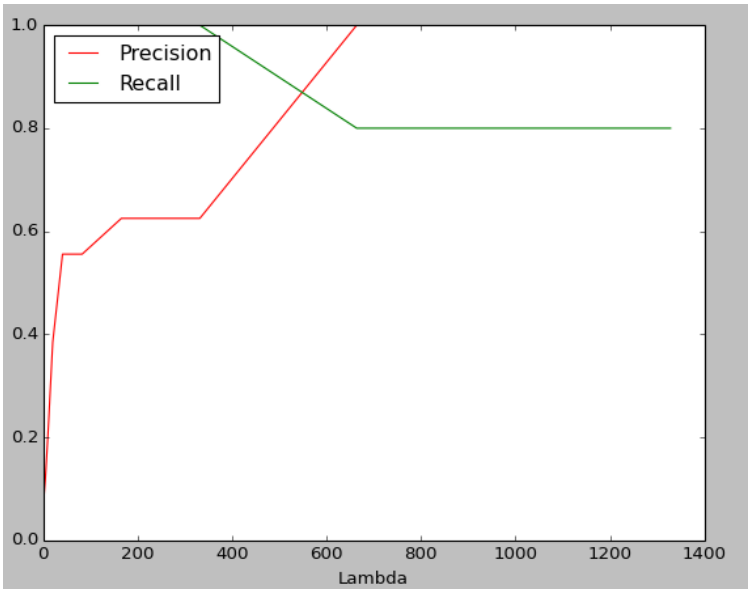


Figure 4 N=50, D=75

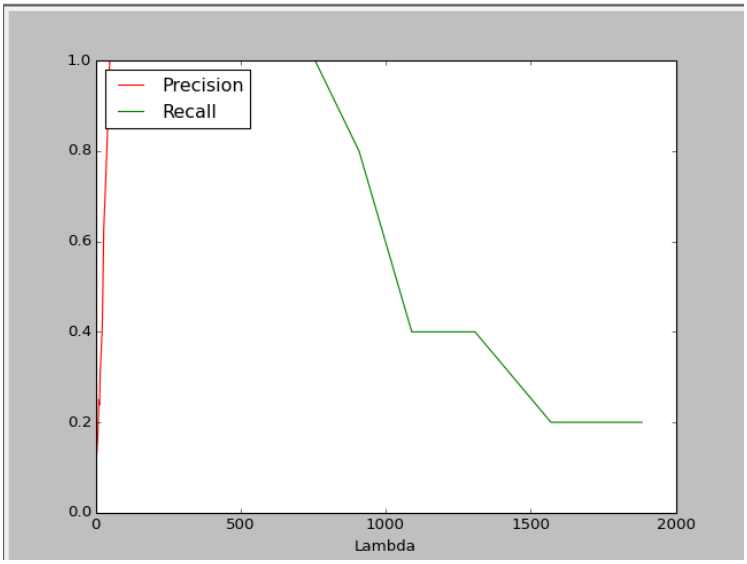
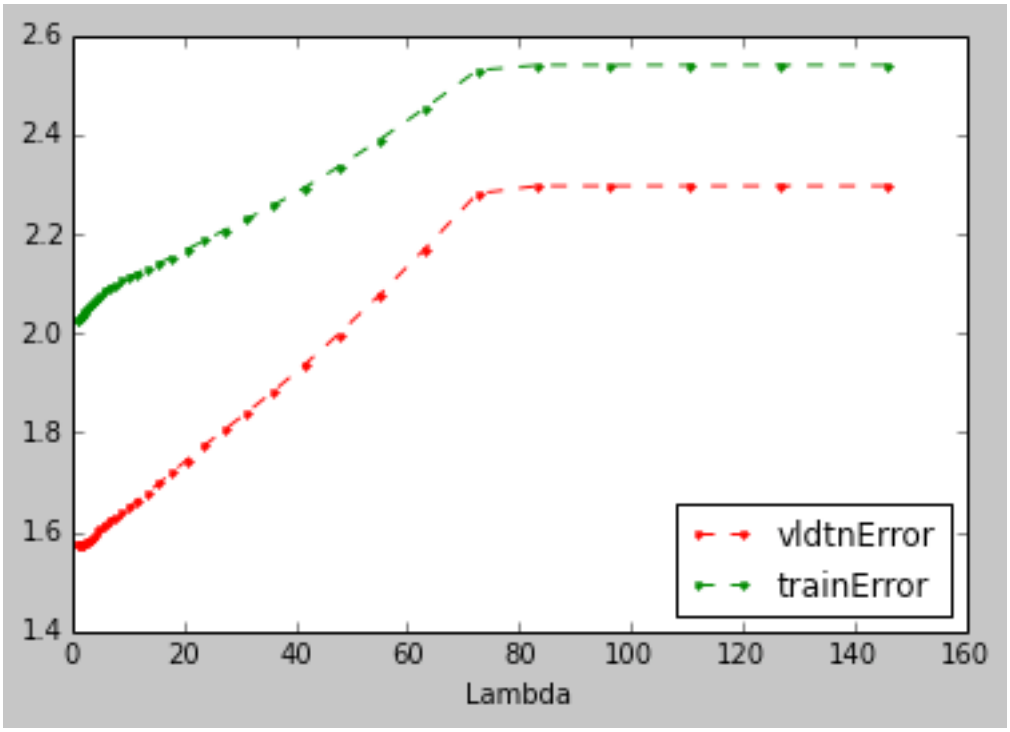
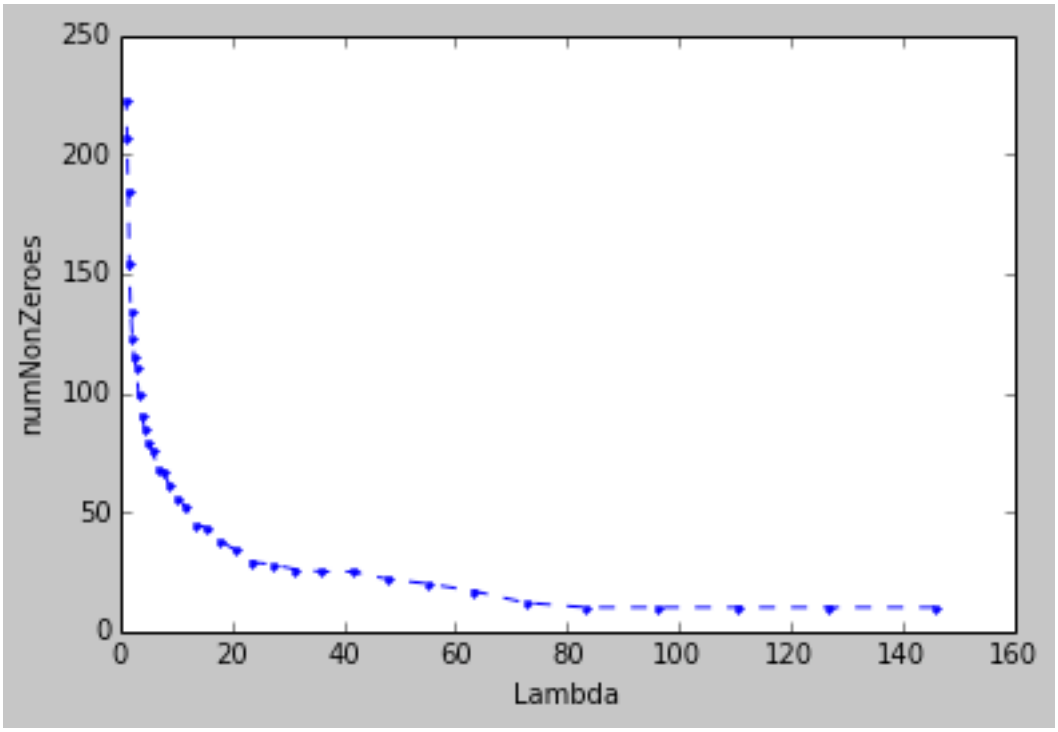


Figure 5 Figure 1N= 50, d = 150

Ques 5.3

1. The graphs for the solutions is attached, Lambda vs validationError and training Error.
   As expected, the number of non zero features increase with decreasing lambda. Also, the validation error decreases as we decrease lambda and then starts increasing (at which point we stop)





Ques 5.3.2

lambda for best performance: 1.26010270711

RMSE on test data at best-lambda 2.38053186025

Ques 5.3.3

The following features have most weight:

'sqrt(ReviewNumCharacters*UserFunnyVotes)': weight 8.0148826067628249

'sqrt(ReviewNumCharacters*UserCoolVotes)': weight 7.2929345265564418

'sqrt(ReviewNumWords*UserFunnyVotes)': weight 6.9001542974389576

'sqrt(UserFunnyVotes*InPhoenix)': weight 6.5672050646532867

'sqrt(ReviewNumLineBreaks*UserCoolVotes)' : weight 6.1711985118884041

'ReviewInFall*InGlendale': weight 6.0726282186118565

'sqrt(ReviewNumWords*UserUsefulVotes)': weight 5.8907850534816166

'sqrt(UserFunnyVotes*BusinessNumStars)': weight 5.7585623533524251

'sqrt(UserCoolVotes*BusinessNumStars)': weight 5.6680134630023469

'sq(UserFunnyVotes*BusinessNumStars)': weight 5.6512991661792285

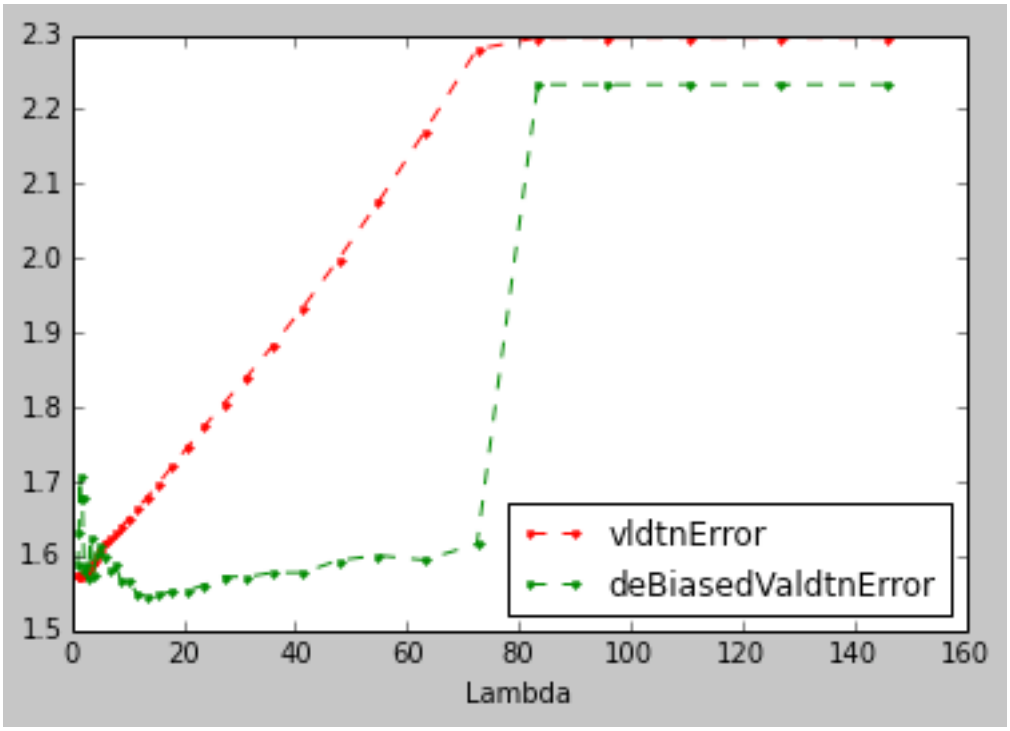
These features suggest that the length of the review and the funny/cool/useful rating of the reviewer is important for upvotes. Also, location of the user is another important factor to take into account.

These features do make sense intuitively as popular user with a funny review is expected to get more upvotes!!

Ques 5.3.4

Following is the graph for debaised RMSE vs biased RMSE, as expected the debiased RMSE is lesser until it starts increasing with more overfitting when lambda becomes too less.
As Lambda decreases: debiased error decreases, until it starts increasing due to overfitting at very low lambda.

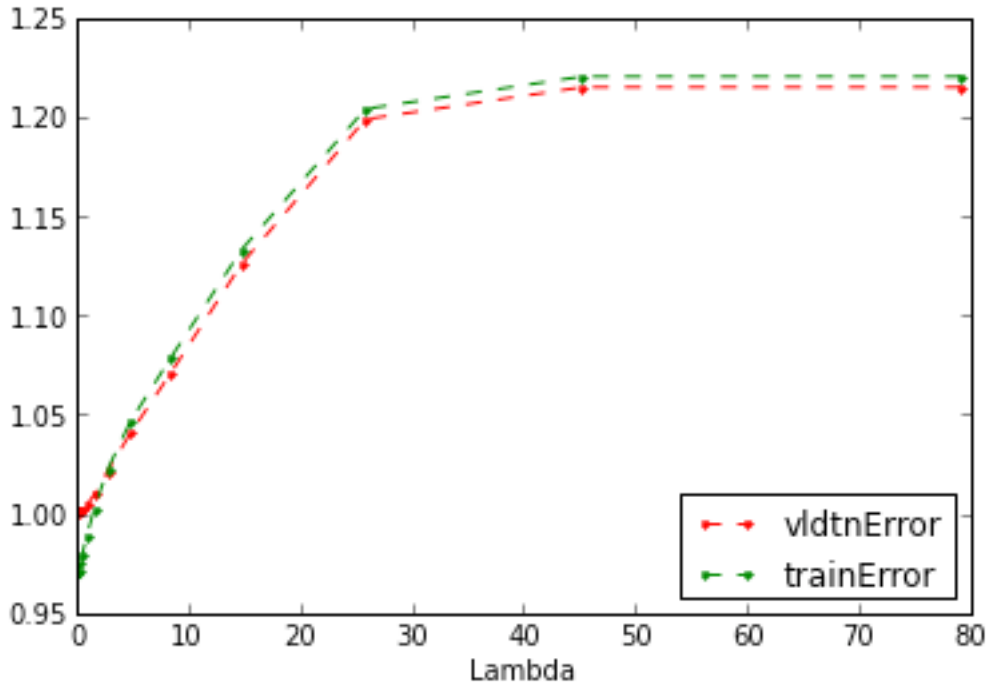
Therefore the performance of the algorithm improves by de-biasing (except for very low lambda)!

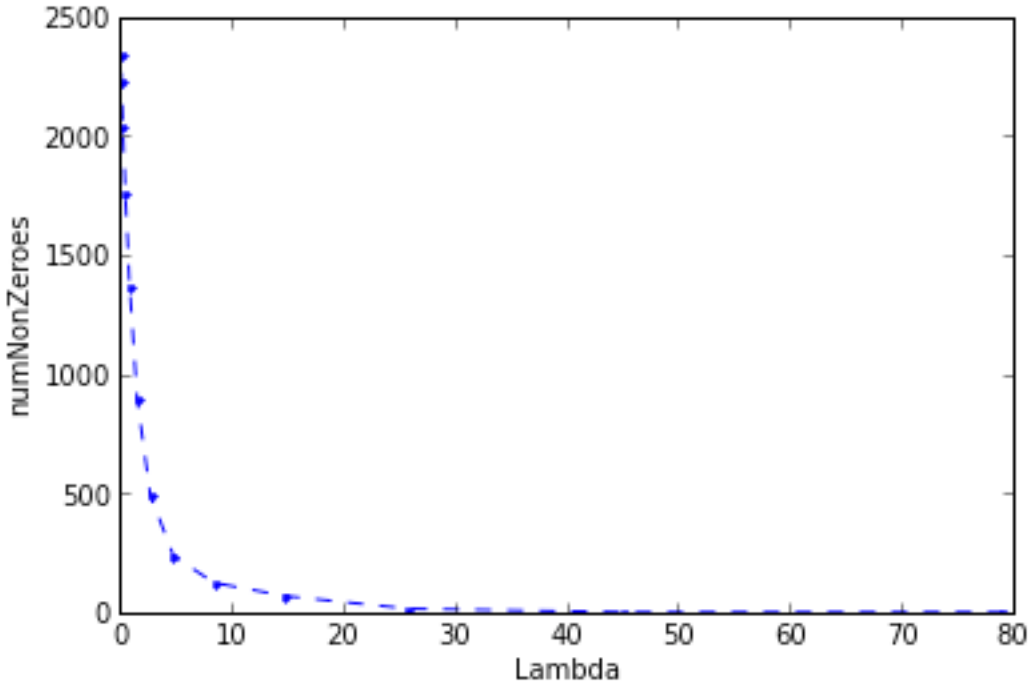


Ques 5.3.5

(I did not use sparse matrices because my original code is worked without it and converged in 15-20 mins)

Again, as expected, the number of non zero features increase with decreasing lambda. Also, the validation error decreases as we decrease lambda and then starts increasing (at which point we stop) Graphs below:





2.

lambda for best performance: **0.16790672529**

RMSE on test data at best-lambda: **1.13957559005**

3.

The following features have most weight (red ones are negative):

'amazing': 7.9387968057802611
'great': 7.8475835952245045
'the worst': -7.3738142437904362
'rude': -7.1002985179855873
'horrible': -6.8906649936158688
'delicious': 6.4122554715741851
'not': -6.3403662759500632
'awesome': 6.3316597574725559
'best': 6.2818064399034386
'terrible': -5.954674663399433
'love': 5.8354419381872722

We notice that the keywords like amazing/great/awesome/best/love have a lot of positive weight and **increase the number of stars for a review.**
Also the negative keywords like 'the worst'/rude/horrible/not/terrible have a negative weight and **decrease the number of predicted stars for a review.**

This is very much near to reality as review with positive adjectives is expected to have more stars!!