Cover page for answers.pdf

CSE512 Fall 2018 - Machine Learning - Homework 2

Your Name: Sriram Reddy Kalluri

Solar ID: 111878857

NetID email address: skalluri@cs.stonybrook.edu sriram.kalluri@stonybrook.edu

Names of people whom you discussed the homework with:

1.1 Since the amount of delay in minutes is Poisson distributed. (iid) with parameter λ.

Let $X = (x_1, x_2 \dots x_n)$ be a random vector where.

X_i is the number of delay minutes of your ith trip.

the likelihood function is

$$L = P(X|\lambda) = \prod_{i=1}^{n} \frac{e^{-\lambda} \lambda^{x_i}}{x_i!} = \frac{e^{-n\lambda} \lambda^{\sum_{i=1}^{n} x_i}}{\prod_{i=1}^{n} x_i!}$$

This can be written by using the property of independent variables i.e $P(x_1 \dots x_n|\lambda) = P(x_1|\lambda) \dots P(x_n|\lambda)$.

1) The loglikelihood function of X given λ is.

Applying log on both sides,

$$\ln L = \sum_{i=1}^{n} \ln\left(\frac{e^{-\lambda} \lambda^{x_i}}{x_i!}\right)$$

$$= \sum_{i=1}^{n} \left[-\lambda + x_i \ln\lambda - \ln(x_i!)\right]$$

$$= -n\lambda + \ln\lambda \sum_{i=1}^{n} x_i - \sum_{i=1}^{n} \ln(x_i!).$$

2). MLE for λ.

$$\hat{\lambda} = \operatorname{argmax}_{\lambda} P(X|\lambda) = \operatorname{argmax}_{\lambda} \ln P(X|\lambda).$$

λ is the parameter,

which can be found by taking the derivative and equating it to zero.

$$\frac{\partial \ln L}{\partial \hat{\lambda}} = -n + \frac{1}{\hat{\lambda}} \sum_{i=1}^{n} x_i = 0.$$

$$\hat{\lambda} = \frac{\sum_{i=1}^{n} x_i}{n}.$$

3). MLE for λ. i.e. $\hat{\lambda}$ using observed x.

$$\hat{\lambda} = \frac{4+5+3+5+6+9+10}{7}$$

$$= 6.$$

1.2. 1). MAP can be written as.

$$\underset{\lambda}{\operatorname{argmax}}\, P(\lambda|x) \propto \underset{\lambda}{\operatorname{argmax}}\, P(x|\lambda)\, P(\lambda).$$

proportional to.

Posterior distribution over λ.

$$= \frac{P(x|\lambda) \cdot P(\lambda)}{P(x)}$$

independent of λ; can be removed when ln is applied

$$= \left[\frac{e^{-n\lambda} \lambda^{\sum_{i=1}^{n} x_i}}{\prod_{i=1}^{n} (x_i)!}\right] \left[\frac{\beta^{\alpha} \lambda^{\alpha-1} e^{-\beta\lambda}}{\Gamma(\alpha)}\right] \Big/ P(x).$$

$$\Rightarrow P(\lambda|x) \propto \left(\frac{\beta^{\alpha}}{\Gamma(\alpha) \prod_{i=1}^{N} x_i!}\right) \lambda^{\alpha-1+\sum_{i=1}^{n} x_i} e^{-(n+\beta)\lambda}.$$

Let's take.

$$\hat{\alpha} = \alpha + \sum_{i=1}^{n} x_i \qquad \hat{\beta} = \beta + n.$$

$$P(\lambda/x) \propto \lambda^{\hat{\alpha}-1} e^{-\hat{\beta}\lambda}$$

This is the posterior distribution.

From the law of total probability, we can calculate the normalizing constant : K

$$\sum_{\lambda} K \lambda^{\hat{\alpha}-1} e^{-\hat{\beta}\lambda} = 1.$$

This looks similar as gamma distribution. So, using normalizing constant of gamma distribution,

$$K = \frac{(\hat{\beta})^{\hat{\alpha}}}{\Gamma(\hat{\alpha})}.$$

So,

$$P(\lambda/x) = \frac{(\hat{\beta})^{\hat{\alpha}}}{\Gamma(\hat{\alpha})} \lambda^{\hat{\alpha}-1} e^{-\hat{\beta}\lambda}$$

2). Deriving the MAP estimate of λ,

$$= \underset{\lambda}{\arg\max}\ P(\lambda/x).$$

$$= \underset{\lambda}{\arg\max}\ \ln P(\lambda/x)$$

$$= \underset{\lambda}{\arg\max}\left[(\hat{\alpha}-1)\ln\lambda - \hat{\beta}\lambda\right]$$

Differentiating with respect to λ and equating it to zero.

$$\frac{(\hat{\alpha}-1)}{\lambda} - \hat{\beta} = 0 \Rightarrow \hat{\lambda} = \frac{\hat{\alpha}-1}{\hat{\beta}}$$

$$= \frac{\sum_{i=1}^{n} x_i + \alpha - 1}{n+\beta}$$

1.3 Estimator Bias.

MLE estimate of η.

$$\eta = e^{-2\lambda}.$$

$$\lambda = -\tfrac{1}{2}\ln\eta.$$

$$\text{MLE of } \eta = \underset{\eta}{\arg\max}\ P(x|\lambda(\eta)) = \underset{\eta}{\arg\max}\ \ln P(x|\lambda(\eta)).$$

$$P(x|\lambda(\eta)) = \frac{(\lambda(\eta))^{x}\, e^{-\lambda(\eta)}}{x!}$$

$$= \frac{1}{x!}\left[\left[\frac{-1}{2}\ln\eta\right]^{x} e^{\frac{+1}{2}\ln\eta}\right.$$

Taking log on both sides,

$$\ln\left[P(x|\lambda(\eta))\right] = x\ln\left(\frac{-1}{2}\ln\eta\right) + \frac{1}{2}\ln\eta - \ln x!$$

Differentiating w.r.t η,

$$x\left(\frac{1}{\frac{-1}{2}\ln\hat{\eta}}\right)\frac{-1}{2\hat{\eta}} + \frac{1}{2\hat{\eta}} = 0$$

$$\frac{x}{\hat{\eta}\ln\hat{\eta}} = \frac{-1}{2\hat{\eta}}$$

$$-2x = \ln\hat{\eta}$$

$$\hat{\eta} = e^{-2x}$$

---

2). $E[\hat{\eta}] - \eta$ = Bias of an estimate

$$E[\hat{\eta}] = \sum_{x\geq 0}\hat{\eta}\, P(x)$$

$$= \sum_{x\geq 0} e^{-2x}\frac{\lambda^{x} e^{-\lambda}}{x!}$$

$$= e^{-\lambda}\sum_{x\geq 0}\frac{e^{-2x}\lambda^{x}}{x!}$$

$$\boxed{\sum_{x\geq 0}\frac{\lambda^{x}e^{-\lambda}}{x!} = 1}$$

$$\Rightarrow \quad = e^{-\lambda}\left[e^{\lambda e^{-2}}\right]$$

$$= e^{(e^{-2}-1)\lambda}$$

The bias of the estimator

$$\text{bias}(\hat{\eta}) = E[\hat{\eta}] - \eta$$

$$= e^{(e^{-2}-1)\lambda} - e^{-2\lambda}.$$

$$= e^{-(1-e^{-2})\lambda}. - e^{-2\lambda}.$$

$$= e^{-(1-1/e^2)\lambda} - e^{-2\lambda}.$$

3).

$$E[\hat{\eta}] = E[(-1)^x]$$

$$= \sum_{x\geq 0} (-1)^x \frac{\lambda^x e^{-\lambda}}{x!}$$

$$= e^{-\lambda} \sum_{x\geq 0} \frac{(-1)^x \lambda^x}{x!} = e^{-\lambda} \sum_{x\geq 0} \frac{(-\lambda)^x}{x!}$$

$$= e^{-\lambda} e^{-\lambda} = e^{-2\lambda}.$$

The bias of the estimator is

$$E[\hat{\eta}] - \eta = e^{-2\lambda} - e^{-2\lambda} = 0.$$

So, it is an unbiased estimator.

The main character of this estimate is that it oscillates between 1 and (-1) depending on whether x is even / odd.

Since the bias is extremely dependent on x and oscillates between 1 and -1; moreover it takes negative values, it is not a good estimator to use. That's a bad estimator.

It doesn't fit good for this scenario.

2). Ridge Regression and LOOCV

$$\underset{w,b}{\text{minimize}} \; \lambda \|w\|^2 + \sum_{i=1}^{n} (w^T x_i + b - y_i)^2$$

Based on piazza suggestion, ~~I'm adding~~ a ~~penalty term~~ . ~~b^2~~ ~~to it. That bias additional term.~~ I had a discussion with professor

$$= \underset{w,b}{\text{minimize}} \; \lambda \|w\|^2 + \cancel{\lambda b^2} + \sum_{i=1}^{n} (w^T x_i + b - y_i)^2$$

	Dimensions	Formula's
$\bar{w} = [w; b]$	$(k+1) \times 1$	$\frac{\partial}{\partial x}[a^T x] = a$
$\bar{X} = [X ; 1_n^T]$	$(k+1) \times n$	$\frac{\partial}{\partial x}[x^T A x] = (A + A^T) x$
$\bar{I} = [I_k ; 0_k ; 0_k^T ; 0]$	$(k+1) \times (k+1)$	

(This matrix ensures that there won't be λb^2 terms as multiplication with zero)

I can restructure the terms by using above terms.

$$= \underset{\bar{w}}{\text{minimize}} \; \lambda(\underbrace{\bar{w}^T \bar{I} \bar{w}}_{\text{(This doesn't contain } \lambda b^2)}) + [y - \bar{X}^T \bar{w}]^T [y - \bar{X}^T \bar{w}]$$

Differentiating w.r.t $\bar{w}$ and equating it to zero.

Using those formulas and ensuring the dimensions to $(k+1) \times 1$ for each term.

$$\underbrace{(\bar{I} + \bar{I}^T)\lambda \bar{w}}_{(k+1)\times 1} + \underbrace{2\underbrace{[-\bar{X}]}_{(k+1)\times n} \underbrace{[y - \underbrace{\bar{X}^T \bar{w}}_{n\times(k+1)\times(k+1)\times 1}]}_{n \times 1}}_{(k+1) \times 1} = 0$$

$$2 \lambda \bar{I} \bar{w} + 2(-\bar{X})(y - \bar{X}^T \bar{w}) = 0$$

$$\bar{w}[\bar{X}\bar{X}^T + \lambda \bar{I}] = \bar{X} y.$$

Scanned by CamScanner

$$\bar{w} = [\bar{x}\bar{x}^T + \lambda \bar{I}]^{-1} [\bar{x} y].$$

$$= \bar{C}^{-1} d. \qquad \text{where} \quad C = (\bar{x}\bar{x}^T + \lambda \bar{I})$$
$$d = [\bar{x} y]$$

2.2).

$$C = \bar{x}\,\bar{x}^T + \lambda \bar{I}$$

$$(k+1) \times n \quad \times n \times (k+1)$$

$(k+1) \times (k+1)$ dimension.

I'm highlighting the terms for x_i which is a column vector of $(k+1)$ terms, i.e. $[x_{i1} \dots x_{ik} \; 1]^T$.

$$C = \begin{bmatrix} x_{i1} \\ \vdots \\ x_{ik} \\ 1 \end{bmatrix} \begin{bmatrix} - & - & \dots & - \\ x_{i1} & \dots & x_{ik} & 1 \\ \dots & & & \end{bmatrix}^T +$$

$$= \underbrace{\begin{bmatrix} x_{i1}^2 & x_{i1}x_{i2} & \dots & x_{i1}x_{ik} & x_{i1} \\ x_{i2}x_{i1} & x_{i2}^2 & & & \\ & & & & x_{ik} \\ x_{i1} & x_{i2} & \dots & x_{ik} & 1 \end{bmatrix}}_{\downarrow} + \underbrace{\text{terms from other input}}_{C_i}$$

Let $\bar{x}_i = [x_i ; 1]$.
$K+1 \times 1$

This term can be generated from $\bar{x}_i \bar{x}_i^T$.

$$\Rightarrow C_i = C - \bar{x}_i \bar{x}_i^T$$

2.2.

$$d = \bar{X} y \quad \underset{(k+1)\times n}{\bar{X}} \quad \underset{n \times 1}{y}$$

$$= \begin{bmatrix} & x_{i1} & \\ & \vdots & \\ & x_{ik} & \\ & 1 & \end{bmatrix} \begin{bmatrix} 1 \\ \vdots \\ y_i \\ \vdots \\ y_n \end{bmatrix}$$

$$d_{(i)} = d - \underset{(k+1)\times 1}{\bar{x}_i}\, y_i \quad \text{scalar}$$

$$= d - \bar{x}_i y_i$$

2.3.

$$C_{(i)}^{-1} = (C - \bar{x}_i \bar{x}_i^T)^{-1}$$

Using the Sherman-Morrison formula,

$$(A + uv^T)^{-1} = A^{-1} - \frac{A^{-1} u v^T A^{-1}}{1 + v^T A^{-1} u}$$

$$= C^{-1} - \frac{C^{-1}(-\bar{x}_i \bar{x}_i^T) C^{-1}}{1 - \bar{x}_i^T C^{-1} \bar{x}_i}$$

$$= C^{-1} + \frac{C^{-1} \bar{x}_i \bar{x}_i^T C^{-1}}{1 - \bar{x}_i^T C^{-1} \bar{x}_i}$$

2.4.

$$\bar{w}_{(i)} = C_{(i)}^{-1} d_{(i)}$$

$$\bar{\omega}_{(1)} = \left(C^{-1} + \frac{C^{-1}\bar{x}_i\bar{x}_i^T C^{-1}}{1-\bar{x}_i^T C^{-1}\bar{x}_i}\right)(d - \bar{x}_i y_i)$$

$$= C^{-1}d - C^{-1}\bar{x}_i y_i + \left(\frac{C^{-1}\bar{x}_i\bar{x}_i^T C^{-1}}{1-\bar{x}_i^T C^{-1}\bar{x}_i}\right)(d - \bar{x}_i y_i)$$

$$= C^{-1}d - C^{-1}\bar{x}_i y_i + \frac{C^{-1}\bar{x}_i y_i \bar{x}_i^T C^{-1}\bar{x}_i + C^{-1}\bar{x}_i\bar{x}_i^T C^{-1}d - C^{-1}\bar{x}_i\bar{x}_i^T C^{-1}\bar{x}_i y_i}{(1-x_i^T C^{-1}\bar{x}_i)}$$

$$= \bar{\omega} + (C^{-1}\bar{x}_i)\left[\frac{-y_i + \bar{x}_i^T C^{-1}d}{1-\bar{x}_i^T C^{-1}\bar{x}_i}\right]$$

$$= \bar{\omega} + C^{-1}\bar{x}_i\left[\frac{-y_i + \bar{x}_i^T \omega}{1-\bar{x}_i^T C^{-1}\bar{x}_i}\right]$$

2.5). Leave one out error for the ith training data is:

$$\omega_i^T \bar{x}_i - y_i = \Bigg[\bar{\omega} + C^{-1}\bar{x}_i\underbrace{\left[\frac{-y_i + \bar{x}_i^T\omega}{1-\bar{x}_i^T C^{-1}\bar{x}_i}\right]}_{\text{Scalar } = K}\Bigg]^T \bar{x}_i - y_i$$

$$= \left[\bar{\omega}^T + \bar{x}_i^T (C^{-1})^T K\right]\bar{x}_i - y_i$$

$$= \bar{\omega}^T\bar{x}_i + \bar{x}_i^T (C^{-1})^T \bar{x}_i\left[\frac{-y_i + \bar{x}_i^T\omega}{1-\bar{x}_i^T C^{-1}\bar{x}_i}\right] - y_i$$

$$= \frac{\bar{\omega}^T\bar{x}_i - \bar{\omega}^T\bar{x}_i\,\bar{x}_i^T C^{-1}\bar{x}_i - \bar{x}_i^T(C^{-1})^T\bar{x}_i y_i + \bar{x}_i^T(C^{-1})^T\bar{x}_i\bar{x}_i^T\omega - y_i + \bar{x}_i^T(C^{-1})^T\bar{x}_i y_i}{(1-\bar{x}_i^T C^{-1}\bar{x}_i)}$$

$$= \frac{\bar{\omega}^T \bar{x}_i - y_i - \cancel{\bar{\omega}^T \bar{x}_i \bar{x}_i^T C^{-1} \bar{x}_i} + \cancel{\bar{x}_i^T (C^{-1})^T \bar{x}_i \bar{x}_i^T \omega}}{1 - \bar{x}_i^T C^{-1} \bar{x}_i}$$

C^{-1} is a symmetric matrix.

$C^{-1} = (C^{-1})^T$

$$= \frac{\bar{\omega}^T \bar{x}_i - y_i}{1 - \bar{x}_i^T C^{-1} \bar{x}_i}$$

2.6) Algorithmic complexity of computing LOOCV error using the above formula is

$O(k^3)$ Time complexity of C^{-1}. +

$O(k^2 n)$ Computation of matrix multiplication i.e. Efficient way

Matrix multiplication

Sum over n examples

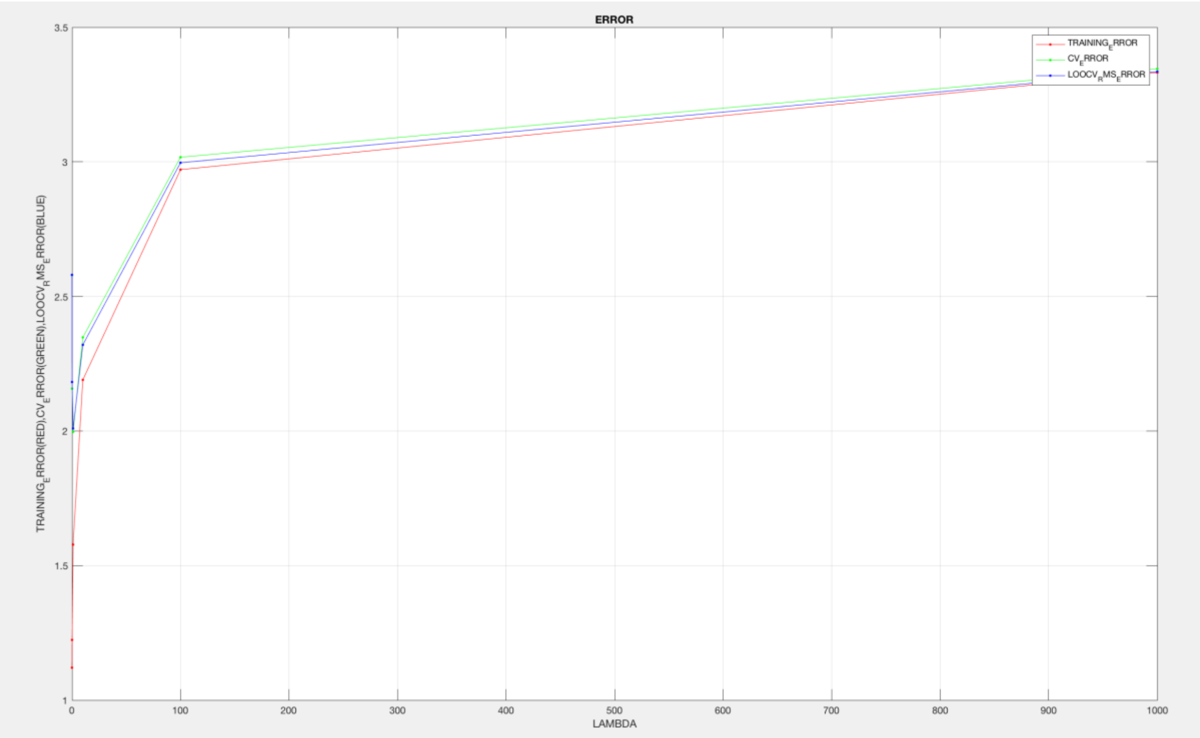
This gives rise. $O(k^3) + O(nk^2)$.

when compared with respect to usual way, it is.

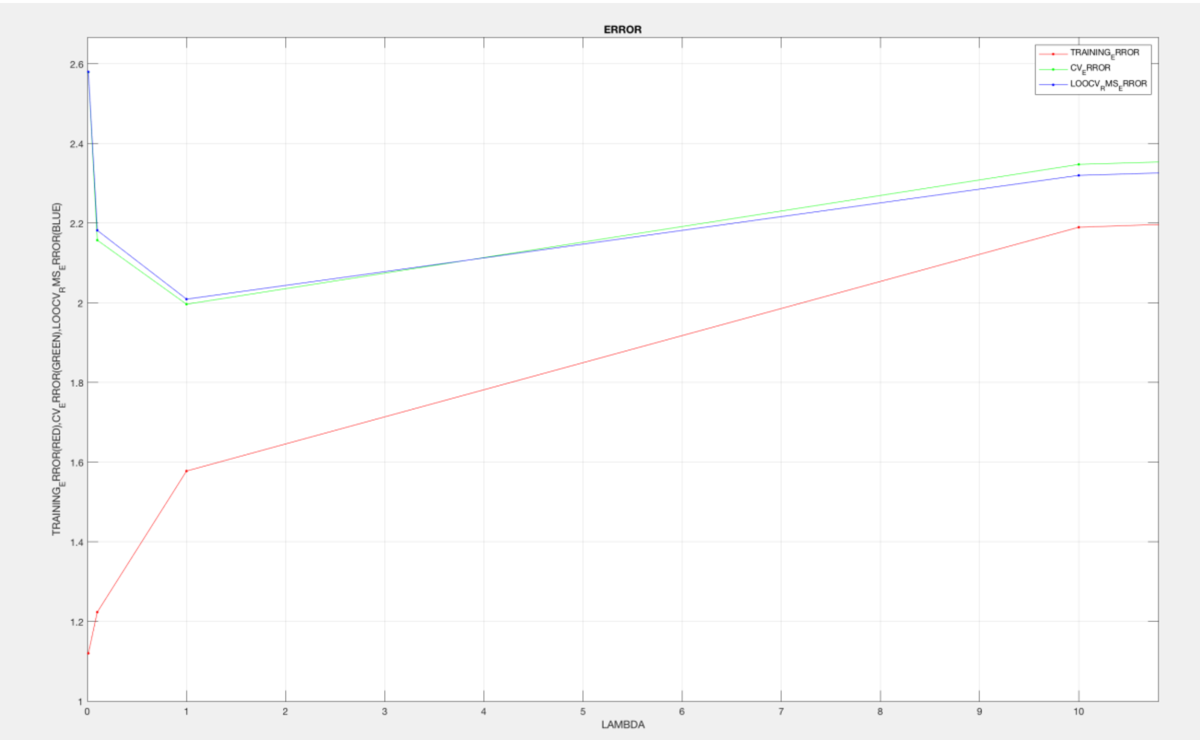
$O(k^3 n)$. which is very inefficient way.

3.2.1)

Please find the below plots for the train, validation and leave-one-out-cross-validation RMSE values together on a plot against lambda.



I have zoomed in the plot to check which lambda gives best LOOCV error.



3.2.2)
As you can observe in the above plot, cross validation is less for lambda=1 out of these 6 lambda values. As discussed in the class, the lambda with least LOOCV error gives the optimum lambda to fit on the test data. So, the objective function, regularization term and sum of squared errors for the lambda=1 are given below:

Objective function	2.43E+04
Sum of square error	1.25E+04
Regularization term	1.18E+04

Infact, I have validated the LOOCV errors for different values of lambda close to 1. The optimal lambda for the whole range is 0.65. I have calculated the above terms with that lambda as well. Please find the values below:

Objective function	2.45E+04
Sum of square error	1.11E+04
Regularization term	1.34E+04

3.2.3)

To get the 10 most important features and the top 10 least important features, we can look at the top 10 values and bottom 10 values with sorted weights in “w” matrix.
The below values are captured for lambda 1 without normalization of input features.

Bottom features	**Weight value**
offers	-0.0001423
light body	-0.0004945
franc petit verdot	-0.0011453
framed	-0.0017861
tannins frame	-0.0018536
tannins finish	-0.0026979
flavors black cherry	-0.0037469
oakville	-0.0044925
wine	-0.005215
picked	-0.0055786
Top features	**Weight value**
price dry	4.6908
cocktail	4.736
currant cola	4.7868
future	4.8483
new french	5.0789
little heavy	5.1285
sweet black	5.1947
red	5.6365
pineapple orange	5.6633

infused	6.999

The below values are captured for lambda 0.65 without normalization of input features.

Bottom features	Weight value
wine tastes	-0.0008685
balanced crisp	-0.0012982
inspired	-0.0020526
bodied wine	-0.002257
lacking	-0.0024739
appeals	-0.0034691
lightly	-0.006492
mainly	-0.0071661
underneath	-0.0079262
hints	-0.0080826
Top features	**Weight value**
wine great	5.2313
alcohol high	5.263
future	5.3009
add	5.3262
new french	5.3526
lifesaver	5.3736
red	5.7426
sweet black	6.024
pineapple orange	6.3451
infused	7.4578

If we take the decision based on above features, that may be misleading. Weight sorting approach will offer the best results if the input features are normalized. So, its better to normalize the features, train them and check the parameters. There are so many feature scaling methods. Out of them, I picked “standard normalization” method i.e (x-mean/standard-deviation).

Bottom features	Weight value
away	-9.728E-05
moderately	-0.000194
bing	-0.0004084
wine just	-0.000535
feels bit	-0.0005434
red wine	-0.0005504
good wine	-0.0005879
flat	-0.0007901
long time	-0.0008456

clear	-0.0008729
Top features	**Weight value**
oak adds	0.31693
date	0.32416
dishes	0.36732
lingering finish	0.36922
blackberry pie	0.3927
winemaking	0.43762
huge	0.4407
just touch	0.50431
green mint	1.6064
spectrum	2.7585

Since the above values are inherent to particular set of features and normalization/feature scaling affects the features, there can be changes in the top and bottom features every time you evaluate as the optimization parameter changes.

The values for 3.2.1 with lambda 1 after applying the feature normalization parameters.

Objective function	1.37E+04
Sum of square error	6.68E+03
Regularization term	7.03E+03

The bottom features like “away”, “moderately”, “feels bit” are something used for negative review. The features like “lingering finish”, “huge”, “just touch” are something used for positive review. So, these features make sense.

3.2.4)
To increase the accuracy, I have tried the below things:
Feature normalization, addition of features by adding the five extra features i.e by taking the sum of all features, sum of squares of all features and sum of squares of sum of all features which reduces my RMSE error on the test data (Adding non-linear terms helped performance). Testing data on Kaggle is bit skewed to Training data and the same RMSE error on training set is analogous to the test data error on Kaggle. If you are receiving better LOOCV errors, you can submit in Kaggle and check it improved the results.

2	new	**sriramreddy**	1.85678	6	17h