CSE 546 Homework #1
Fall 2016

Janet Matsen

October 20, 2016

For this assignment, I went to office hours and developed strategies with Serena Liu and David Flemming.

1 Prob 1: Expectations, Variance, & Covariance [5 points]

1.1 $E[X]$

Definition of variance: $Var(X) = E[X^2] - (E[X]))^2$
Use the independence to separate the pdfs of x_1 and x_2. Since they are uniform on $[0, 1]$, the pdfs are 1 over $[0, 1]$.
Solve first term: $E[X] = \int_{\substack{0\\x_2}}^{1} \int_{\substack{x_2\\x_1}}^{1} x_1 dx_1 dx_2 + \int_{\substack{0\\x_1}}^{1} \int_{\substack{x_1\\x_2}}^{1} x_2 dx_2 dx_1$
The two terms have symmetry, so
$E[X] = 2\int_{\substack{0\\x_2}}^{1} \int_{\substack{x_2\\x_1}}^{1} x_1 dx_1 dx_2$
Solve the first integral.
$E[X] = 2\int_{\substack{0\\x_2}}^{1} x_1^2/2\Big|_{x_2}^{1} dx_2$

$E[X] = \int_{\substack{0\\x_2}}^{1} (x_1^2\Big|_{x_2}^{1})dx_2$

$E[X] = \int_{\substack{0\\x_2}}^{1} (1 - x_2^2)dx_2$

$E[X] = (x_2 - x_2^3/3)\Big|_{0}^{1}$
$E[X] = (1 - 1/3) = 2/3$
$E[X] = 2/3$

1.2 $Var(X)$

$Var(X) = E[X^2] - [E[X]]^2$
We know $E[X]$ from part 1. Compute $E[X^2]$.

$E[X^2] = \int_{\substack{0\\x_2}}^{1} \int_{\substack{x_2\\x_1}}^{1} x_1^2 dx_1 dx_2 + \int_{\substack{0\\x_1}}^{1} \int_{\substack{x_1\\x_2}}^{1} x_2^2 dx_2 dx_1$

$E[X^2] = 2\int_{\substack{0\\x_2}}^{1} \int_{\substack{x_2\\x_1}}^{1} x_1^2 dx_1 dx_2$

$E[X^2] = 2\int_{\substack{0\\x_2}}^{1} x_1^3/3\Big|_{x_1}^{1} dx_2$

$E[X^2] = 2/3\int_{\substack{0\\x_2}}^{1} 1 - x_2^3 dx_2$

$E[X^2] = 2/3(x_2 - x_2^4/4)\Big|_0^1$
$E[X^2] = 2/3(1 - 1/4)$
$E[X^2] = 2/3(3/4) = 1/2$

Put these expectations together:
$Var(X) = E[X^2] - [E[X]]^2 = 1/2 - (2/3)^2 = 1/2 - 4/9 = 9/18 - 8/18 = 1/18$
$Var(X) = 1/18$

1.3 $Cov(X, X_1)$

Use general formula: $Cov(Y, Z) = E[YZ] - E[Y]E[Z]$.
So $Cov(X, X_1) = E[XX_1] - E[X]E[X_1]$
We already computed $E[X] = 2/3$.
The expectation of X_1 is $1/2$ because it is uniform on $[0, 1]$
All we need to compute then is $E[XX_1]$.
$E[XX_1] = \int_{\substack{0\\x_2}}^1 \int_{\substack{x_2\\x_1}}^1 x_1^2 dx_1 dx_2 + \int_{\substack{0\\x_1}}^1 \int_{\substack{x_1\\x_2}}^1 x_1 x_2 dx_2 dx_1$
We solved term 1 earlier when solving for $E[X^2]$, and it worked out to be $1/4$.
Now solve $\int_{\substack{0\\x_1}}^1 \int_{\substack{x_1\\x_2}}^1 x_1 x_2 dx_2 dx_1$.

$$\begin{aligned}
\int_{\substack{0\\x_1}}^1 \int_{\substack{x_1\\x_2}}^1 x_1 x_2 dx_2 dx_1 &= \int_{\substack{0\\x_1}}^1 x_1 x_2^2/2\Big|_{x_1}^1 dx_1 \\
&= \int_{\substack{0\\x_1}}^1 x_1(1/2)(1 - x_1^2) dx_1 \\
&= (1/2)\int_{\substack{0\\x_1}}^1 x_1 - x_1^3 dx_1 \\
&= (1/2)(x_1^2/2 - x_1^4/4)\Big|_0^1 \\
&= (1/2)(1/2 - 1/4) \\
&= (1/2)(1/4) \\
&= 1/8
\end{aligned}$$

Put together the $E(XX_1)$:

$$\begin{aligned}
E[XX_1] &= \underset{x_2}{\int_0^1} \underset{x_1}{\int_{x_2}^1} x_1^2 dx_1 dx_2 + \underset{x_1}{\int_0^1} \underset{x_2}{\int_{x_1}^1} x_1 x_2 dx_2 dx_1 \\
&= 1/4 + 1/8 = 2/8 + 1/8 \\
&= 3/8
\end{aligned}$$

So $Cov(X, X_1) = E[XX_1] - E[X]E[X_1] = 3/8 - (2/3)(1/2) = 3/8 - 1/3 = 9/24 - 8/24 = 1/24$
$Cov(X, X_1) = 1/24$

2 Prob 2: MLE for Poisson [5 points]

Poisson Distribution:

$$\Pr(X = k) = \frac{\lambda^k}{k!} e^{-\lambda} \qquad k \in \{0, 1, 2, \dots\}$$

Data G from the Poisson distribution:

Sword Swing	1	2	3	4	5	6	7	8
Orcs Slain	4	1	3	5	5	1	3	8

Question (a): Give the log-likelihood function of G given λ.

For n samples from a Poisson distribution with values G_i:

$$\begin{aligned}
L(\lambda \mid G) = P(G \mid \lambda) &= \prod_{i=1}^{n} P(G_i) \\
&= \prod_{i=1}^{n} \frac{\lambda^{G_i} e^{-\lambda}}{G_i!} \\
&= \frac{\lambda^{\sum_{i=1}^{n} G_i} e^{-\lambda n}}{\prod_{i=1}^{n} G_i!} \\
\ln(L(\lambda \mid G)) &= \ln(\lambda^{\sum_{i=1}^{n} G_i}) + \ln(e^{-\lambda n}) - \ln(\prod_{i=1}^{n} G_i!) \\
&= \sum_{i=1}^{n} G_i \ln(\lambda) - \lambda n - \ln(\prod_{i=1}^{n} G_i!)
\end{aligned}$$

Question (b):
Compute the MLE for λ in the general case.
To get MLE from the likelihood function, we take the derivative and set it equal to zero.

$$\begin{aligned}
\frac{d}{d\lambda} L(\lambda \mid G) &= \frac{d}{d\lambda}[\sum_{i=1}^{n} G_i \ln(\lambda) - \lambda n - \ln(\prod_{i=1}^{n} G_i!)] \\
&= \sum_{i=1}^{n} G_i/\lambda - n - 0
\end{aligned}$$

set the derivative to zero and solve for lambda:

$$0 = (\sum_{i=1}^{n} G_i)/\lambda - n$$

$$\lambda = \frac{\sum\limits_{i=1}^{n} G_i}{n}$$

Question (c):
Compute the MLE for λ using the observed G.
For our particular case: $\lambda = \frac{4+1+3+5+5+1+3+8}{8} = \frac{30}{8} = 3.75$

3 Prob 3: Linear Regression & LOOCV [10 points]

3.1 *[2 points]* What is the time complexity of computing the LOOCV score natively?

To compute the LOOCV score, we must calculate $\hat{Y}$, which requires calculation of $\hat{w}$.
To calculate $\hat{w}$, we nee to compute $(X^TX)^{-1}X^TY$.

- X is $n \times d$, so X^TX is $d \times d$. Inverting a $d \times d$ matrix is $\mathcal{O}(d^3)$.

- After that, we multiply by X^T. We are multiplying a $d \times d$ matrix with a $d \times n$ matrix, so that cost is $\mathcal{O}(d^2n)$; the costs are additive.

- $(X^TX)^{-1}X^T$ is $d \times n$, and Y is $n \times 1$, so we add on cost $\mathcal{O}(nd)$ for the last matrix multiplication.

The costs sum to $\mathcal{O}(d^3 + d^2n + nd) = \mathcal{O}(d^3 + d^2n)$. We can't simplify further without knowing whether d is larger than n. The cost of calculating the LOOCV score without tricks is a function of the number of points, and the dimensionality of each point.

3.2 *[1 point]* Write $\hat{y}_i$ in terms of H and Y.

Since $\hat{Y} = HY$, $\hat{y} = \sum_{j=1}^{d} H_{ij}Y_j$

3.3 *[2 points]* Show that $\hat{Y}^{(-i)}$ is also the estimator which minimizes SSE for Z, where Z = ...

SSE for $Z = \sum_{i \neq j}^{n} (y_i - \hat{y}_i)^2 + (y_j - \hat{y}_j)$

In the second term both y_j and $\hat{y}_j = \hat{y}_j^{(-j)}$, so the term is zero. SSE for $Z = \sum_{i \neq j}^{n} (y_i - \hat{y}_i)^2$; this is just SSE on a matrix with $n-1$ elements and corresponds to the $\hat{Y}^{(-i)}$ SSE.

3.4 *[1 point]* Write $\hat{y}_i^{(-i)}$ in terms of H and Z.

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

3.5 *[2 points]* Show that $\hat{y}_i - \hat{y}_i^{(-i)} = H_{ii}y_i - H_{ii}\hat{y}_i^{(-i)}$

Use $\hat{y}_i = \sum_{j=1}^{n} H_{ij}y_j$ (3.1.2 answer) and $\hat{y}_i^{(-1)} = \sum_{j=1}^{n} H_{ij}Z_j$ (3.1.4 answer).
Also use

$$\sum_{j=1}^{n} H_{ij}y_j = \left(\sum_{i\neq j} H_{ij}y_j\right) + H_{ii}y_i$$

$$\text{and when } i \neq j,\ Z_j = y_j$$

$$= \left(\sum_{i\neq j} H_{ij}Z_j\right) + H_{ii}y_i$$

And $\sum_{j} H_{ij}Z_j = \left(\sum_{i\neq j} H_{ij}Z_j\right) + H_{ii}\hat{y}_i^{(-i)}$

Now plug those into the original expression:

$$\begin{aligned}
\hat{y}_i - \hat{y}_i^{(-i)} &= \sum_{j=1}^{n} H_{ij}y_j - \sum_{j=1}^{n} H_{ij}Z_j \\
&= \left(\sum_{i\neq j} H_{ij}Z_j\right) + H_{ii}y_i - \left(\sum_{i\neq j} H_{ij}Z_j\right) + H_{ii}\hat{y}_i^{(-i)} \\
&= H_{ii}y_i - H_{ii}\hat{y}_i^{(-i)}
\end{aligned}$$

3.6 *[2 points]* Show that $LOOCV = \sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1 - H_{ii}}\right)^2$. What is the algorithmic complexity of computing the LOOCV score using this formula?

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

Get an expression for $\hat{y}_i^{(-i)}$ from $\hat{y}_i - \hat{y}_i^{(-i)} = H_{ii}y_i - H_{ii}\hat{y}_i^{(-i)}$:

$$\begin{aligned}
\hat{y}_i - \hat{y}_i^{(-i)} &= H_{ii}y_i - H_{ii}\hat{y}_i^{(-i)} \\
-H_{ii}\hat{y}_i^{(-i)} + \hat{y}_i^{(-i)} &= -H_{ii}y_i + \hat{y}_i \\
\hat{y}_i^{(-i)}(1 - H_{ii}) &= \hat{y}_i - H_{ii}y_i \\
\hat{y}_i^{(-i)} &= \frac{\hat{y}_i - H_{ii}y_i}{1 - H_{ii}}
\end{aligned}$$

Plug this into the original LOOCV formula:

$$\begin{aligned}
LOOCV &= \sum_{i=1}^{n}(y_i - \hat{y}_i^{(-i)})^2 \\
&= \sum_{i=1}^{n}\left(y_i - \frac{\hat{y}_i - H_{ii}y_i}{1-H_{ii}}\right)^2 \\
\text{and } y_i &= \frac{y_i(1-H_{ii})}{(1-H_{ii})} \\
&= \sum_{i=1}^{n}\left(y_i - \frac{\hat{y}_i - H_{ii}y_i}{1-H_{ii}}\right)^2 \\
&= \sum_{i=1}^{n}\left(\frac{y_i(1-H_{ii})}{(1-H_{ii})} - \frac{\hat{y}_i - H_{ii}y_i}{1-H_{ii}}\right)^2 \\
&= \sum_{i=1}^{n}\left(\frac{y_i(1-H_{ii}) - \hat{y}_i + H_{ii}y_i}{1-H_{ii}}\right)^2 \\
&= \sum_{i=1}^{n}\left(\frac{y_i - y_iH_{ii} - \hat{y}_i + H_{ii}y_i}{1-H_{ii}}\right)^2 \\
LOOCV &= \sum_{i=1}^{n}\left(\frac{y_i - \hat{y}_i}{1-H_{ii}}\right)^2
\end{aligned}$$

In our new LOOCV formula, no terms depend on a particular i being left out. We only need to compute and store H and $\hat{y}_i$, and pick out the values needed or each term. We said earlier that this pair of operations had complexity $\mathcal{O}(d^3 + d^2n)$. Note that this is faster than needing to compute the N leave-one-out models, which would have complexity $\mathcal{O}(nd^3)$.

4 Prob 4: Regularization Constants [16 points]

4.1 How does choosing too small an λ value affect the magnitude of the following?

4.1.1 The error on the training set

When λ is too small, a model can be over-fit to the training data. This means that the training error gets smaller. This is true of both LASSO and ridge regression, as both tend to the ordinary least squares (OLS) solution as $\lambda \to 0$.

4.1.2 The error on the testing set

When λ is too small, then the model was over-fit to the training data. This will lead to a poor fit on the testing set, and thus larger testing set error. The same trend is true for both LASSO and ridge regression.

4.1.3 The elements of $\hat{w}$

When λ is too small, then the lack of penalty on large coefficients leads them to be as large as necessary to minimize the sum of squared errors. The weights for both LASSO and Ridge will be similar because $\lambda \to 0$ means both solutions approach OLS.

4.1.4 The number of nonzero elements of $\hat{w}$

When λ is too small and the model is over-fit, both models will have more nonzero coefficients. Again, both ridge and LASSO will have a similar number of nonzero coefficients because both are approaching the OLS solution.

4.2 How does choosing too large an λ value affect the magnitude of the following?

4.2.1 The error on the training set

When λ is too large, there is a poor fit on the training. The large λ is directly preventing a good fit by limiting how low the sum of squares can be. This is true for both ridge regression and LASSO.

4.2.2 The error on the testing set

As λ increases from "too low" to "two high" it passes through a sweet spot when the model is reasonably well fit but general enough to fit previously unseen data. By definition of "too high" we are saying that the resulting model is so general that it also doesn't have good predictive power on the unseen testing set. Thus the error on the testing set will be high for both ridge and LASSO.

4.2.3 The elements of $\hat{w}$

The elements of $\hat{w}$ are forced to be small with "too large" of λ. This is true for both ride and LASSO.

4.2.4 The number of nonzero elements of $\hat{w}$

When λ is too large, it has different effects on the number of nonzero elements for ridge and LASSO. LASSO's L1 norm is effective at setting small regression coefficients to zero. The large λ is, the more coefficients will be forced to zero. Ridge, on the other hand, will force the coefficients to be very small, but there is no pressure for them to be truly zero.

5 Prob 5: Regression & Agnostic Learning [20 points]

5.1 Bayes Optimal Prediction

5.1.1 *[5 points]* Show $L(f) = E_X[(E[Y|X] - f(X))^2] + E_{X,Y}[(Y - E[Y|X])^2]$

Given: $L(f) = E_{X,Y}[(Y - f(X))^2]$
Key fact: $E_{Y|X}[Y|X] = E[Y|X]$ Inject $E[Y|X] - E[Y|X]$ and reorganize:

$$\begin{aligned} L(f) &= E_{X,Y}[(Y - f(X))^2] \\ &= E_{X,Y}[(Y - E[Y|X] + E[Y|X] - f(X))^2] \end{aligned}$$

Let $a = Y - E[Y|X]$, $b = E[Y|X] - f(X)$.
The stuff inside the squared term above then is $(a + b)^2 = a^2 + b^2 + 2ab$.
Expectations are separable, so we have
$L(f) = E_{X,Y}[a^2 + b^2 + 2ab] = E_{X,Y}[a^2] + E_{X,Y}[b^2] + E_{X,Y}[2ab]$

Note that we are trying to show $L(f) = E_{X,Y}[(Y - E[Y|X] + E[Y|X] - f(X))^2]$, and that in our a, b notation this is equal to $L(f) = E_{X,Y}[a^2] + E_{X,Y}[b^2]$.

Then the proof is complete if we show that $E_{X,Y}[2ab] = 0$.
Note that $E_{X,Y}[2ab] = E_{X,Y}[ab] = E_{X,Y}[(Y - E[Y|X])(E[Y|X] - f(X))]$

$$\begin{aligned} E_{X,Y}[(Y - E[Y|X])(E[Y|X] - f(X))] &= E_{X,Y}[(YE[Y|X] - Yf(X) - (E[Y|X])^2 + f(X)E[Y|X]] \\ &= E_{X,Y}[YE[Y|X] - (E[Y|X])^2] - E_{X,Y}[Yf(X) - f(X)E[Y|X]] \end{aligned}$$

Introduce new notation:
Let $c = E_{X,Y}[(YE[Y|X] - (E[Y|X])^2]$, and let $d = E_{X,Y}[Yf(X) - f(X)E[Y|X]]$
I will show that both c and d are zero to finish the proof.

$$\begin{aligned} c &= E_{X,Y}[YE[Y|X] - (E[Y|X])^2] \\ &= E_X[E_{Y|X}[YE[Y|X] - (E[Y|X])^2]] \\ &= E_X[E_{Y|X}[YE[Y|X]]] - E_X[E_{Y|X}[(E[Y|X])^2]] \\ &\qquad \text{note that } E[Y|X] \text{ is a } f(x) \text{ only.} \\ &= E_X[E[Y|X]E_{Y|X}[Y]] - E_X[E_{Y|X}[(E[Y|X])^2]] \\ &= E_X[E[Y|X]E_{Y|X}[Y]] - E_X[E_{Y|X}[(E[Y|X])^2]] \\ &\qquad \text{use } E_{Y|X}[Y|X] = E[Y|X] \\ &\qquad \text{use } E_{Y|X}[(E[Y|X])^2] = E[Y|X])^2 \text{ because } E[Y|X] \text{ is only } f(X) \\ &= E_X[(E[Y|X])^2] - E_X[(E[Y|X])^2] \\ &= 0 \end{aligned}$$

Now do d:

$$\begin{aligned} d &= E_{X,Y}[Yf(X) - f(X)E[Y|X]] \\ &= E_X[E_{Y|X}[Yf(X) - f(X)E[Y|X]]] \\ &= E_X[E_{Y|X}[f(X)(Y - E[Y|X])]] \\ &= E_X[f(X)E_{Y|X}[Y - E[Y|X]]] \\ &= E_X[f(X)[E_{Y|X}[Y] - E_{Y|X}[E[Y|X]]]] \\ &= E_X[f(X)([E[Y|X] - E[Y|X])]] \\ &= E_X[f(X)(0)]] \\ &= 0 \end{aligned}$$

Thus

$$\begin{aligned} L(f) &= E_{X,Y}[a^2] + E_{X,Y}[b^2] + E_{X,Y}[2ab] \\ &= E_{X,Y}[a^2] + E_{X,Y}[b^2] \\ &= E_{X,Y}[(Y - E[Y|X])^2] + E_{X,Y}[(E[Y|X] - f(X))^2] \\ &= E_{X,Y}[(E[Y|X] - f(X))^2] + E_{X,Y}[(Y - E[Y|X])^2] \\ &= E_X E_{Y|X}[(E[Y|X] - f(X))^2] + E_{X,Y}[(Y - E[Y|X])^2] \\ &= E_X[(E[Y|X] - f(X))^2] + E_{X,Y}[(Y - E[Y|X])^2] \end{aligned}$$

$L(f) = E_X[(E[Y|X] - f(X))^2] + E_{X,Y}[(Y - E[Y|X])^2]$

5.1.2 *[3 points]* The Bayes optimal regressor f_{Bayes} is the function which minimizes L(f) over the class of all functions. Provide a natural expression for f_{Bayes} and write down L(f_{Bayes}).

The loss shown in problem 5.1.1's statement has two terms. The second term doesn't have an expression for the solution ("regressor") f in it, so the choice of regressor cannot minimize it. The first term, however, can be minimized by setting $f_{Bayes} = f(X) = E[Y|X]$. That makes the whole first term zero. Then the loss after that is $L(f_{Bayes}) = E_{X,Y}[(Y - E[Y|X])^2]$. This represents the error due to scatter of points according to the probability distribution.

5.2 Bias-Variance-Noise Error Decomposition

5.2.1 *[5 points]* Show that ...

We aim to show that

$$\begin{aligned} E_T L(\hat{f}) &= E_T E_{X,Y}[(Y - \hat{f}(X))^2] \\ &= E_T E_X[(\bar{f}(X) - \hat{f}(X))^2] + E_X[(\bar{f}(X) - E[Y|X])^2] + E_{X,Y}[(Y - E[Y|X])^2] \end{aligned}$$

I will use a similar trick to problem 5.2.1, but with three terms, a, b, c.
First add $E[Y|X] - E[Y|X] + \bar{f}(X) - \bar{f}(X)$.

From now on, simplify the notation: $\bar{f}(X) == \bar{f}$, and $\hat{f}(X) == \hat{f}$

$$\begin{aligned}
E_T L(\hat{f}) &= E_T E_{X,Y}[(Y - \hat{f}(X))^2] \\
&= E_T E_{X,Y}[(Y - \hat{f})^2] \\
&= E_T E_{X,Y}[(Y - \hat{f} + E[Y|X] - E[Y|X] + \bar{f} - \bar{f})^2] \\
&= E_T E_{X,Y}[((\bar{f} - \hat{f}) + (-\bar{f} + E[Y|X]) + (Y - E[Y|X]))^2]
\end{aligned}$$

I will use these "tricks":

1. We can switch E_T and E_X b/c T is independent of X and Y; it doesn't change when we sample.
2. $\bar{f}$ is no longer a function of X; we have already integrated X out.

Let $a = (\bar{f} - \hat{f})$, $b = (-\bar{f} + E[Y|X])$, $c = (Y - E[Y|X])$
Note that $b^2 = (-\bar{f} + E[Y|X])^2 = (E[Y|X] - \bar{f})^2$
Then $E_T L(\hat{f}) = E_T E_{X,Y}[(a + b + c)^2] = E_T E_{X,Y}[a^2 + b^2 + c^2 + ab + bc + ca]$
Once again, the solution we are looking for is the formula above, without the cross terms.

First show that
$E_T E_{X,Y}[a^2+b^2+c^2] = E_T E_X[(\bar{f}(X)-\hat{f}(X))^2]+E_X[(\bar{f}(X)-E[Y|X])^2]+E_{X,Y}[(Y-E[Y|X])^2]$:

$$\begin{aligned}
E_T E_{X,Y}[a^2 + b^2 + c^2] &= E_T E_{X,Y}[(\bar{f} - \hat{f})^2 + (-\bar{f} + E[Y|X])^2 + (Y - E[Y|X])^2] \\
&= E_T E_{X,Y}[(\bar{f} - \hat{f})^2] + E_T E_{X,Y}[(-\bar{f} + E[Y|X])^2] + E_T E_{X,Y}[(Y - E[Y|X])^2] \\
&= E_T E_X E_{Y|X}[(\bar{f} - \hat{f})^2] + E_X E_{Y|X}[(-\bar{f} + E[Y|X])^2] + E_X E_{Y|X}[(Y - E[Y|X])^2] \\
&\quad \text{the firs two expectations are not } f(Y) \text{ so the } E_{X,Y} \text{ does nothing:} \\
&= E_T E_X[(\bar{f} - \hat{f})^2] + E_X[(-\bar{f} + E[Y|X])^2] + E_X E_{Y|X}[(Y - E[Y|X])^2]
\end{aligned}$$

This exactly matches the question's goal. If we can show that $E_T E_{X,Y}[ab+bc+ca] = 0$, the proof is complete.

First write out the cross terms.

$$\begin{aligned}
E_T E_{X,Y}[ab+bc+ca] &= E_T E_{X,Y}[(\bar{f}-\hat{f})(-\bar{f}+E[Y|X])+ \\
&\quad (-\bar{f}+E[Y|X])(Y-E[Y|X])+ \\
&\quad (Y-E[Y|X])(\bar{f}-\hat{f})] \\
&= E_T E_{X,Y}[-\bar{f}^2+\bar{f}E[Y|X]+\hat{f}\bar{f}-\hat{f}E[Y|X] \\
&\quad -\bar{f}Y+\bar{f}E[Y|X]+YE[Y|X]-(E[Y|X])^2 \\
&\quad +\bar{f}Y-Y\hat{f}-\bar{f}E[Y|X]+E[Y|X]\hat{f}] \\
&\qquad \text{six terms cancel out:} \\
&= E_T E_{X,Y}[-\bar{f}^2+\hat{f}\bar{f}+YE[Y|X]-(E[Y|X])^2+\bar{f}E[Y|X]-Y\hat{f}] \\
&= E_T E_{X,Y}[-\bar{f}^2+\hat{f}\bar{f}]+E_T E_{X,Y}[YE[Y|X]-(E[Y|X])^2]+E_T E_{X,Y}[\hat{f}E[Y|X]-Y\hat{f}]
\end{aligned}$$

Define 3 new variables:
$d == E_T E_{X,Y}[-\bar{f}^2+\hat{f}\bar{f}]$
$e == E_T E_{X,Y}[YE[Y|X]-(E[Y|X])^2]$
$f == E_T E_{X,Y}[\hat{f}E[Y|X]-Y\hat{f}]$
I will show that each of these is zero.

$$\begin{aligned}
d &= E_T E_{X,Y}[-\bar{f}^2+\hat{f}\bar{f}] \\
&\qquad \text{These are only functions of the training set, not } X \text{ or } Y \\
&= E_T[-\bar{f}^2+\hat{f}\bar{f}] \\
&= \quad \bar{f} \text{ is not f}(T) \\
&= -\bar{f}^2+\bar{f}E_T[\hat{f}] \\
&= -\bar{f}^2+\bar{f}^2 \\
&= 0
\end{aligned}$$

$$\begin{aligned}
e &= E_T E_{X,Y}[YE[Y|X]-(E[Y|X])^2] \\
&= E_{X,Y}[YE[Y|X]-(E[Y|X])^2] \\
&= E_{X,Y}[YE[Y|X]]-E_{X,Y}[(E[Y|X])^2] \\
&= E_X E_{Y|X}[YE[Y|X]]-E_X E_{Y|X}[(E[Y|X])^2] \\
&= E_X E[Y|X]E_{Y|X}[Y]-E_X E[Y|X]^2 \\
&= E_X E[Y|X]E[Y|X]-E_X E[Y|X]^2 \\
&= E_X E[Y|X]^2-E_X E[Y|X]^2 \\
&= 0
\end{aligned}$$

$$\begin{aligned}
f &= E_T E_{X,Y}[\hat{f} E[Y|X] - Y\hat{f}] \\
&= E_{X,Y}[\hat{f} E[Y|X]] - E_{X,Y}[Y\hat{f}] \\
&= E_X E_{Y|X}[\hat{f} E[Y|X]] - E_X E_{Y|X}[Y\hat{f}] \\
&= E_X[\hat{f} E[Y|X]] - E_X \hat{f} E_{Y|X}[Y] \\
&= E_X[\hat{f} E[Y|X]] - E_X \hat{f} E[Y|X] \\
&= 0
\end{aligned}$$

5.2.2 *[3 points]* Variance, bias, and noise.

The expected loss over all possible training sets, $E_T L(\hat{f})$, can be decomposed into variance, bias and noise.

The bias is the average error of the model, and is given by term 2. This term aggregates the differences between the average model and the true Y values given the sampled points X. An example of when this term would be high is fitting an exponential curve with lines. You can average many lines, but your average of these predictions is a poor fit to the true exponential curve. Also, underfitting is an example of high bias.

The variance is error from sensitivity to fluctuations in the training set, and is shown in term 1. The difference between the average model ($\bar{f}$) and each particular model ($\hat{f}$) is aggregated. If the different training sets give large differences in particular $\hat{f}(X)$ models, then the magnitude of the terms will be big and the first term will be large.

The noise is given by term 3. In this term we compare the true Y values to the expected Y values given different X distributions. Note that there is no model in this term because it is fundamental to the data, not the model you fit to the data.

5.2.3 *[1 point]* For the ERM, is it the case that

In general $f* \neq \bar{f}$. $\bar{f}$ is the result you get when you aggregate all the models on all possible datasets. If you don't sample enough data when making each model, you can come up with many wrong models that do not aggregate into $f*$.

No. Here is a counter-example. Let $\hat{f}$ be a function that maps X to the set $\{0, 1\}$. Given a finite training sample, some of the optimal $\hat{f}$ values will take the value 1, and some will take 0. When you average the $\bar{f}$s, you get a function that produces Y values in the inclusive range $[0, 1]$. Since these continuous values are not in the set $\{0, 1\}$, $\bar{f}$ is not in the same class of models as f^*.

5.2.4 *[3 points]* For finite F and large N, what do $\bar{f}$ and $\bar{f}$ tend to and why?

In the limit of large N, the emperical risk converges to the square loss: $\frac{1}{N}\sum_{i=1}^{N}(y_i - f(x_i))^2$

This is because we are averaging, which is equivalent to taking the expectation as $N \to \infty$

Since f is finite, the probability of $\hat{f}$ being anything other than f^* approaches zero as $N \to \infty$.

Therefore $\hat{f}$ must also approach f^*

6 Prob 6: Classification using Least Squares [10 points]

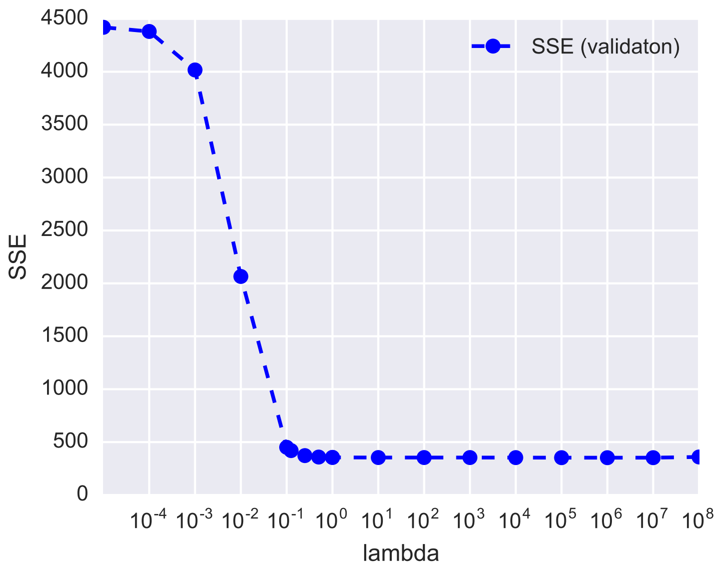
6.1 Binary Classification with Regression

6.1.1 *[4 points]* Chose a reasonable threshold for classification

I implemented the Book formula 7.33: $\hat{w}_{ridge} = (\lambda I_D + X^T X)^{-1} X^T y$ using scipy sparse matrices. The equation ignores bias, but Professor Kakade confirmed this is reasonable given the problem.

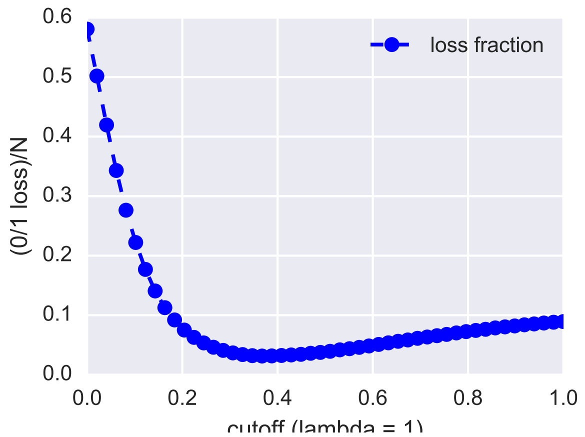
First, I investigated what values of lambda to use. After splitting the training data such that 5/6 of it was for training and 1/6 of it was for validation, I solved for $\hat{w}$ at lambda varying by orders of magnitude. If I had more time, I would be doing k-fold validation for each point on the plot below. For this implementation, I only had time to split the training data into 5/6 training, 1/6 validation.

When I calculated MSE for the held-out data using the training $\hat{w}$, I saw that higher λ led to a better fit.



We want to chose lambda high enough to be general. I chose $\lambda = 10$, which is the smallest λ that gets around the best validation SSE. If I had more time, I would use the same training/validation split of data to chose an optimal cutoff at each of the $\lambda > 1$ values.

I explored the 0/1 loss across different cutoff values. The plot below shows different cutoff values for $\lambda = 10$.



The table corresponding to this graph reveals that my lowest 0/1 loss on the training data happens for cutoff = 0.38. This loss is 1869, which corresponds to getting only 3.1% of the points classified incorrectly. The SSE = 2229, and the RMSE = 0.193.

6.1.2 *[4 points]* On the test set, what is the 0/1 loss? What is the square loss?

With $\lambda = 10$, my weights applied to the test data produced a loss of 334, I got a loss of 1872, corresponding to a 3.3% error rate. The square loss is 389.0, and the RMSE is 0.197. It makes sense that the RMSE is slightly higher than the RMSE of the training set, as the parameters were only fit to the training set and not the test set. I believe that with more time, more explorations of different lambda and threshold values using the train/validation split data would have led to an even better test result.

6.1.3 *[1 point]* Why might linear regression be a poor idea for classification?

Linear regression for classification produces continuous outputs for binary labels. Though the cutoff can sort those continuous output values into bins, the optimization is often penalizing ”correct” answers because the optimization step does not know what cutoff you will be applying.

As an example, say your regression for an output that should be labeled 1 is 0.4, and your cutoff is 0.3. While the point is above the cutoff and will be correctly binned, however, for linear regression this point will still contribute an error of 0.6.

In addition, we are not accounting for the probability that each point was correctly called. Logistic regression would be a better tool, as it estimates probabilities and bins data in the process of minimizing the objective function.

6.1.4 *[1 point]* What is the Bayes optimal predictor using the square loss for classification? (In natural form, in terms of probability)

We said earlier that $f_{Bayes} = E[Y|X]$. For this case, Y can only take the values 0 and 1. We know $E[Y|X] = 0P(Y = 0|X) + 1P(Y = 01|X)$ So $E[Y|X] = P(Y = 1|X)$

7 Lasso [40 points]

7.1 Time complexity & making your code fast

7.1.1 *[1 point]*

$$w_0' \leftarrow \sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{d} w_j X_{ij}\right)/N$$

$$w_0' \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{d} w_j X_{ij}\right)$$

Use the provided definition: $\hat{y}_i = X_i w + w_0$

$$\hat{y}_i = X_i w + w_0 = \left(\sum_{j=1}^{d} X_{ij} w_j\right) + w_0$$

So $\left(\sum_{j=1}^{d} X_{ij} w_j\right) = \hat{y}_i - w_0$

Plug that back into our w_o' equation:

$$w_0' \leftarrow \frac{1}{N}\sum_{i=1}^{N}\left(y_i - (\hat{y}_i - w_0)\right)$$

$$w_0' \leftarrow \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i) + \frac{1}{N}\sum_{i=1}^{N} w_0$$

$$w_0' \leftarrow \left[\frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)\right] + w_0$$

Use $\sum_{j\neq k} X_{ij} w_j = \left(\sum_{j=1}^{d} X_{ij} w_j\right) - X_{ik} w_k$ and the same $\left(\sum_{j=1}^{d} X_{ij} w_j\right) = \hat{y}_i - w_0$ to update the formula for c_k

$$c_k \leftarrow 2\sum_{i=1}^{N} X_{ik}\left(y_i - w_0 - \sum_{j\neq k} X_{ij} w_j\right)$$

$$c_k \leftarrow 2\sum_{i=1}^{N} X_{ik}\left(y_i - w_0 - \left(\sum_{j=1}^{d} X_{ij} w_j - X_{ik} w_k\right)\right)$$

$$c_k \leftarrow 2\sum_{i=1}^{N} X_{ik}\left(y_i - w_0 - (\hat{y}_i - w_0) + w_k X_{ik}\right)$$

$$c_k \leftarrow 2\sum_{i=1}^{N} X_{ik}\left(y_i - \hat{y}_i + w_k X_{ik}\right)$$

7.1.2 *[1 point]*

To compute $\hat{y}$, we need to compute Xw The number of operations we have to do if X and w are non-sparse (all elements filled) is $O(Nd)$, but if there are only $||X||_0$ elements because X is sparse, then we do $||X||_0$ operations. Just computing Xw is O($||X||_0$).

After that, we add w_0, which has N elements. These are sequential operations, so the total complexity is O($||X||_0 + N$).

7.1.3 *[1 point]*

$w_0' \leftarrow \left[\frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i) \right] + w_0$

To update w_0 when $\hat{y}$ is not computed, we have to compute $\hat{y}_i$, which is $X_i w$. We can compute all of the $X_i w$ at once via Xw for cost O($||X||_0$). After that, we can can access single Xw_i rows in the sum. We need to compute N sums, so O($||X||_0$ + N), when $\hat{y}$ isn't computed.

When $\hat{y}$ is already computed, the complexity of computing $\sum_{i=1}^{N} y_i - \hat{y}_i$ is O(N). The addition of w_0 at the end is O(1). O($N+1$) = O(N).

7.1.4 *[1 point]*

When $\hat{y}$ is already computed, the main computational expense is calculating a_k and c_k. This only needs to be done when $X_{ik} \neq 0$, which happens z_j times. Computing a_k has complexity $O(z_j d)$ and computing ck has complexity $O(z_j d)$. Note that computing c_k from λ, a_k is just $O(d)$

So the overall complexity is $O(z_j d)$

7.1.5 *[2 points]*

Use $\hat{y}^{(t)} = X_i w^{(t)} + w_o^{(t)}$, which is the same as $X_i w^{(t)} = \hat{y}^{(t)} + w_o^{(t)}$

$$\begin{aligned}
\hat{y}^{(t+1)} &= X_i w^{(t)} + w_o^{(t+1)} \\
\hat{y}^{(t+1)} &= \hat{y}^{(1)} + w_o^{(t)} + w_o^{(t+1)} \\
\hat{y}^{(t+1)} - \hat{y}^{(1)} &= w_o^{(t)} + w_o^{(t+1)}
\end{aligned}$$

Computing $\hat{y}^{(t+1)}$ when $\hat{y}^{(t)}$ is already computed is $O(N)$, for the N updates we do.

7.1.6 *[2 points]*

$$
\begin{aligned}
\hat{y}^{(t+1)} &= \left(\sum_{j\neq k} w_j^{(t)} X_{ij}\right) + w_k^{(t+1)} X_{ik} + w_o^{(t)} \\
&= \left(\sum_{j=k} w_j^{(t)} X_{ij}\right) - w_k X_{ik} + w_k^{(t+1)} X_{ik} + w_o^{(t)} \\
&= \left(\sum_{j=k} w_j^{(t)} X_{ij}\right) + X_{ik}\left(w_k^{(t+1)} - w_k\right) + w_o^{(t)} \\
&= X_i w^{(t)} + X_{ik}\left(w_k^{(t+1)} - w_k\right) + w_o^{(t)} \\
&\qquad \text{Use } \hat{y}_i^{(t)} = X_i w^{(t)} + w_o^{(t)} \\
&= \hat{y}_i^{(t)} + X_{ik}\left(w_k^{(t+1)} - w_k\right)
\end{aligned}
$$

When $\hat{y}_i^{(t)}$ is pre-computed, we are adding and multiplying a pre-computed value with constants. So each $\hat{y}_i^{(t+1)}$ update is $O(1)$. The whole $\hat{y}_i^{(t)}$ update would have complexity $O(N)$, but when X is sparse, it is $O(z_j)$

7.1.7 *[2 points]*

O($||X||_0 + dN$). Since $||X||_0 \leq dN$, the algorithm is less than O($2dN$). Thus the whole algorithm is O(dN).

7.2 *[0? points]* Implement coordinate descent to solve the Lasso

7.3 *[10 points]* Try out your work on synthetic data

7.3.1 N=50, d=75, k=5, $\sigma = 1$

Data was generated according to $y = Xw* + w_o^* + \epsilon$ in the function generate_random_data of my lasso.py file. The class SyntheticDataRegPath creates random X and Y data for w having $w = [10, -10, 10, -10, 10, 0, 0...]$. When the regularization is strong (large λ), the sparsity pattern of the best-fit w matches nearly perfectly: $w = [[9.67, -9.31, 9.72, -9.41, 9.39, 0, ..., -0.06, 0, ...]$. There is only one coefficient that should be zero but has a small nonzero magnitude.

For plots of both precision and recall, see the section below, where they are plotted alongside those for $\sigma = 10$.

The precision is a metric of the sparsity fit which is low when there are excess nonzero parameters. When we regularize very strongly ($\lambda = 50$), the precision is high: 0.833. It is not 100% this time because there is one nonzero coefficient beyond the 5 expected.

The recall is less sensitive to changes in λ. I found recall = 1 for all values of λ from $\lambda = 50$ to $\lambda = 0.000005$. This makes sense because with low σ the Y values are well predicted by the features in X.

7.3.2 Increase σ from 1 to 10

When σ is increased to 10, the 10-fold higher noise means the Y values are less well predicted by the input X values. This leads to a poorer ability to recover the true weights of the model. As a consequence, recall drops with increasing lambda faster than it did for $\sigma = 1$, without gaining much benefits in precision increases. This is because more of the feature weighting is applied to features other than the 5 that truly influenced Y.

In summary, increasing lambda increases the pressure to have coefficients = 0. When the data isn't too noisy, you can increase precision by increasing lambda. This leads to having less false positive coefficients in w. On the other hand, relaxing (lowering) lambda allows more nonzero features and recall increases.

Figure 1: Q7.3 recall

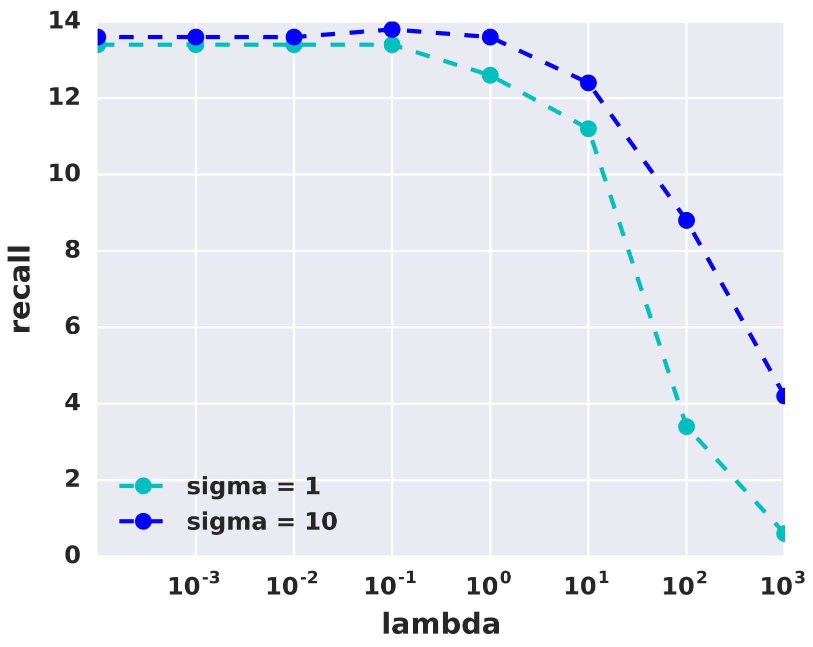
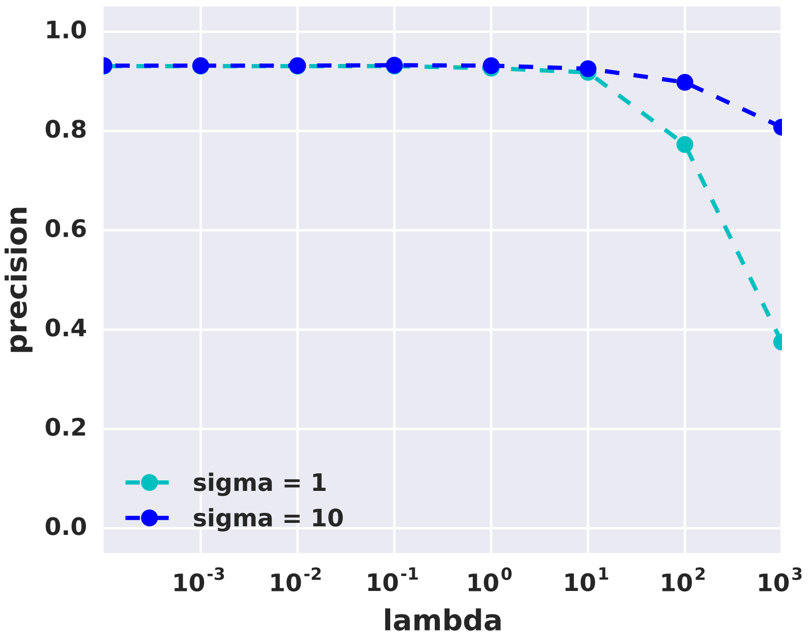


Figure 2: Q7.3 precision



7.4 *[20 points]* Become a data scientist at Yelp

7.4.1 *[6 points]* Regularization Path for Upvotes: best λ, RMSE, and nonzero count

I randomized the data, then normalized each column to have it's L2 norm = 1. The randomization was because there existed some training points with outlier values, and the 4000th to 5000th point was enriched in such points. Both data randomization and data normalization is shown in my Q7-4-1 pdf notebook file. The regression was implemented in lasso.py, which is attached.

Figure 3: RMSE of validation and training data. Note that the data was randomized before splitting into train, validation, and test sets.

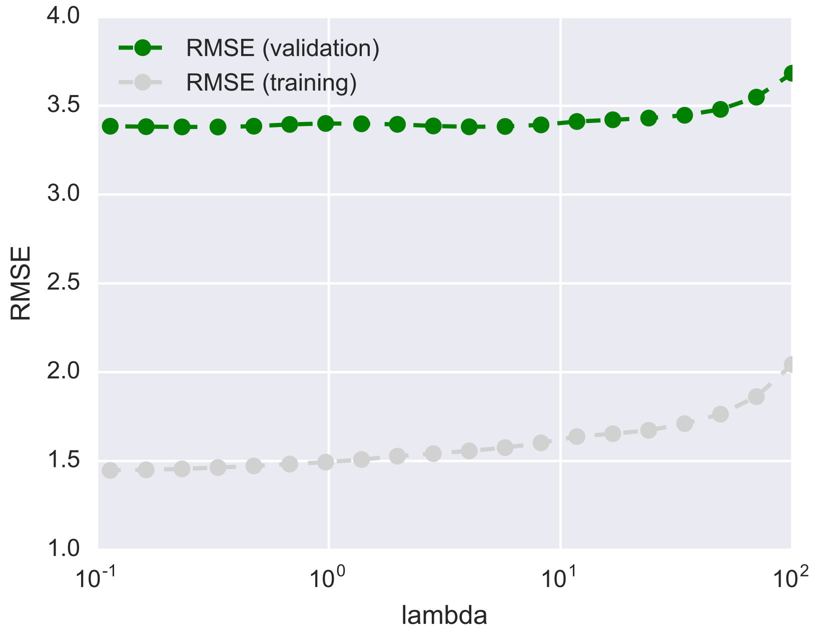
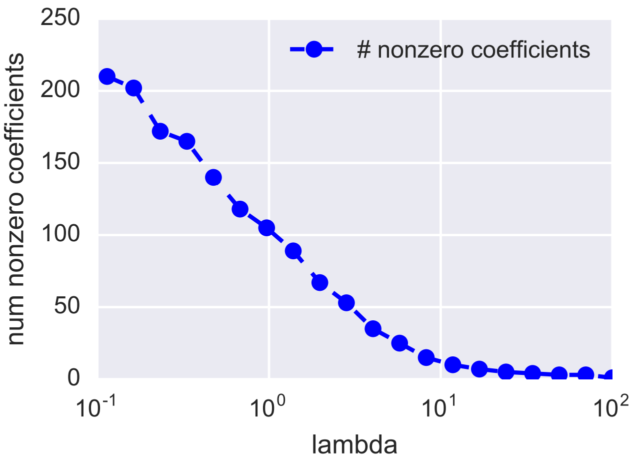


Figure 4: Number of nonzero coefficients versus lambda



As I increased λ, I saw the expected trends:

- As lambda increases, there are fewer and fewer nonzero coefficients
- On my shuffled data, the validation RMSE was higher than the training RMSE.
- As lambda increases, the validation RMSE goes up. I would also expect validation

RMSE to go up again as λ gets even smaller, but I didn't have time to run more λ values.

7.4.2 λ with best performance: Re-train on all the training data and calc RMSE

After re-training RMSE on all of the training data with the best λ (minimized validation RMSE), my RMSE on the never-touched test set was 3.75.

7.4.3 Top 10 features & their weights.

Table 1: Top features and weights for Yelp Upvote data. $\lambda = 0.781250$

Feature	**Regression coefficient**
'log(UserNumReviews)'	-108.57
'sqrt(UserNumReviews*UserFunnyVotes)'	-74.39
'sqrt(UserCoolVotes*BusinessNumStars)'	73.77
'sqrt(ReviewNumCharacters*UserCoolVotes)'	69.28
'sqrt(ReviewNumLineBreaks*UserCoolVotes)'	57.05
'log(UserUsefulVotes)'	50.33
'sqrt(ReviewNumLineBreaks*UserNumReviews)'	-49.11
'sqrt(UserFunnyVotes*InPhoenix)'	47.61
'log(ReviewNumCharacters*UserUsefulVotes)'	47.11
'ReviewNumLineBreaks*UserUsefulVotes'	40.31

Positive weights here indicate that feature's tendency to correlate with higher numbers of useful votes. I wouldn't have expected the log(UserNumReviews) to have negative correlation with up-votes. Some of the positive predictive features make sense: cool reviewers, popular businesses, and review length (probably correlated with line breaks). I'm surprised to see a negative effect from the log of the number of reviews that user has submitted, but maybe people who review a lot don't review in good depth.

7.5 *[10 points]* Regularization Path for Stars: best λ, RMSE, and nonzero count

7.5.1 RMSE, nonzeros on Yelp Stars

Figure 5: RMSE of validation and training data for Yelp star prediction. Note that the data was randomized before splitting into train, validation, and test sets.

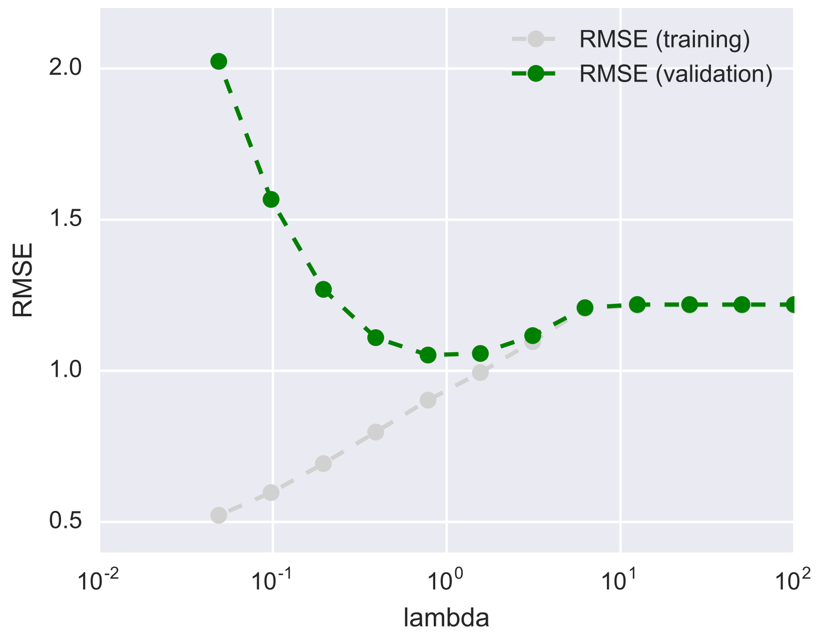
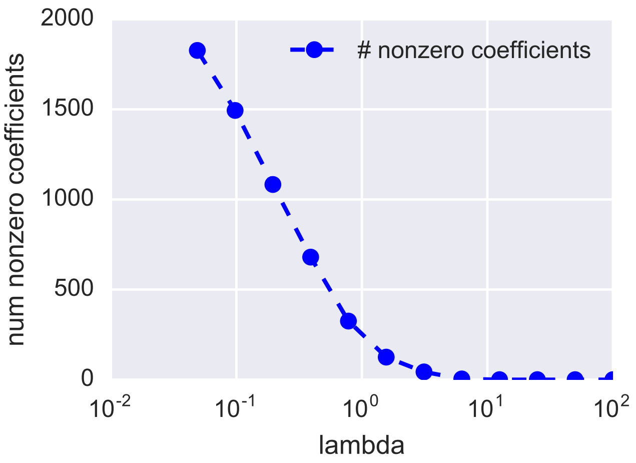


Figure 6: Number of nonzero coefficients versus lambda for LASSO regression on Yelp Stars



7.5.2 RMSE for un-touched test data: Yelp Stars

RMSE = 3.757

7.5.3 Top Coefficients: Yelp Stars

Table 2: Top features and weights for Yelp Star data. $\lambda = 0.78125$

Feature	**Coefficient**
great	24.95
bland	-21.76
best	19.67
the worst	-18.80
should have	-16.79
mediocre	-15.98
terrible	-15.94
love	15.37
awesome	14.42
## minutes	-14.17

It makes sense that these passion-filled words will have a large effect on the score.