Regularization

Notes for CS 6232: Data Analysis and Visualization
Georgia Tech (Dr. Guy Lebanon), Fall 2016
as recorded by Brent Wagenseller

Lesson Preview

We will learn about modeling in high dimensions
What we have learned breaks down in high dimensions
Overfitting and underfitting can be a problem
Techniques to combat these problems

- Shrinkage
- Ridge Regression
- Lasso Regression

What is Regularization?

Problems with overfitting the data

- Models can over-generalizes / overfits attributes that are too specific
- The classification rule is relatively complex and places too much emphasis on a irrelevant attribute
  - Even though the training data works, it will not generalize well to new data
  - Once we allow complex rules, we can overfit to the training data
  - Sometimes, simpler rules are better

Regularization discourages complexity in the prediction logic that is learned from the training data

- Regularization will increase the mistakes on the training set but decrease them on the test set

Model Complexity

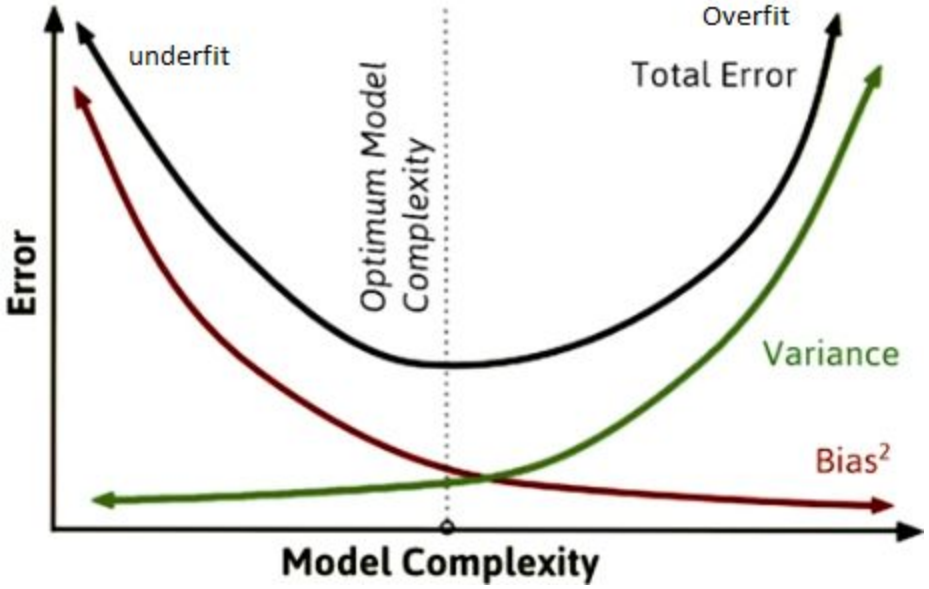
The case of **Mean Squared Error** – the average of the squared error between the true parameter values and the predicted parameter values – breaks down into several smaller parts

- Components
  - **Bias^2** – This is the square of the difference between the true model parameters (ground truth?) and the expected value of the model prediction
    Bias captures the **systematic deviation** from the model
    If the model went through every one of the training points, the bias / systematic deviation will be zero
    Low bias (with high variance) usually means overfitting
  - **Variance** – the variance of the estimator
    Variance captures systematic error
    Low variance (with high bias) is underfitting
    BRENTS NOTE: Think of high variance is your run your model several times for one datapoint and you get inconsistent results
  - Irreducible Error – We will not talk about this much, but know it cant be controlled if we focus on a only a few models
- The sum of Bias^2 and variance these two components is called the **estimation error**
  - Both of these must be low – the model cannot have a large systematic error
    We also want the model to give us consistent values (presumably across features), even if the training data is a bit different
  - We can manipulate this by changing the relative sizes of the bias and the variance and try to find the right trade-off in order to minimize the model error

- Intuitively, the bias captures systematic deviation of the model prediction (or: between the model prediction and ground truth).
    systematic mistakes can happen as you relearn the model (over and over again) based on a new training set, such systematic deviation from the true value you are trying to estimate is captured by the bias intuitively
- The variance captures the variability in the estimation if you were to repeat the machine learning training on multiple instances of the data

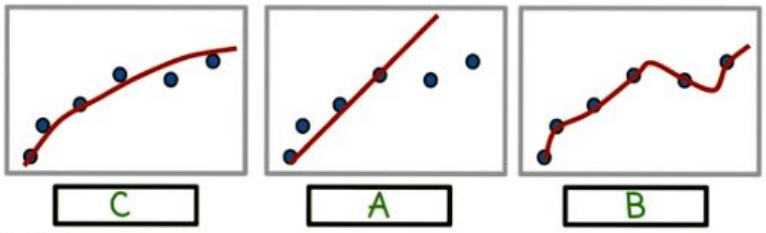
Everything mentioned applies mostly to mean-squared error; that said, other error measures still have a bias in variance component (though they may not breakdown into bias + variance as nicely)

Examine this graph



- As the model complexity increases, the bias decreases (red line)
    - Remember, bias corresponds to systematic deviation from the model
        If the model is more complex, the model is richer and can capture more complicated phenomena
        - That said, it can lead to overfitting
- As the model complexity increases, the variance decreases (green line)
    - Model complexity is usually associated in a more difficult-to-learn model
    - If we keep the training set size constant and yet we keep adding more parameters, we will start to introduce noise that will make it harder to train the model effectively
- What we care about is the sum of the bias and the variance
- In general we want to keep both down, but the sum usually has the shape above
- We want to remain in the perfect trade-off area which is between a model that is too complex and one that is not complex enough

Bias Variation Quiz



A. High bias, low variance
B. Low bias, high variance
C. Middle bias and variance, overall low estimation error

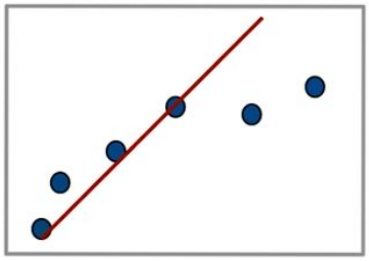
In B, the model is complex (low bias, high variance)

- Again, low bias in this instance means we are overfitting

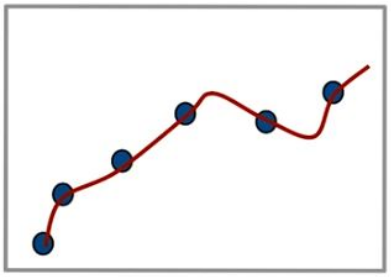
In A, the model is not complex enough (high bias, low variance)

- Again, high bias actually means 'high systematic error'!!!

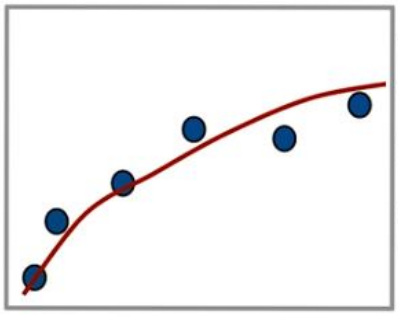
In pictures



- High bias
- Low variance
- Underfitting the training data



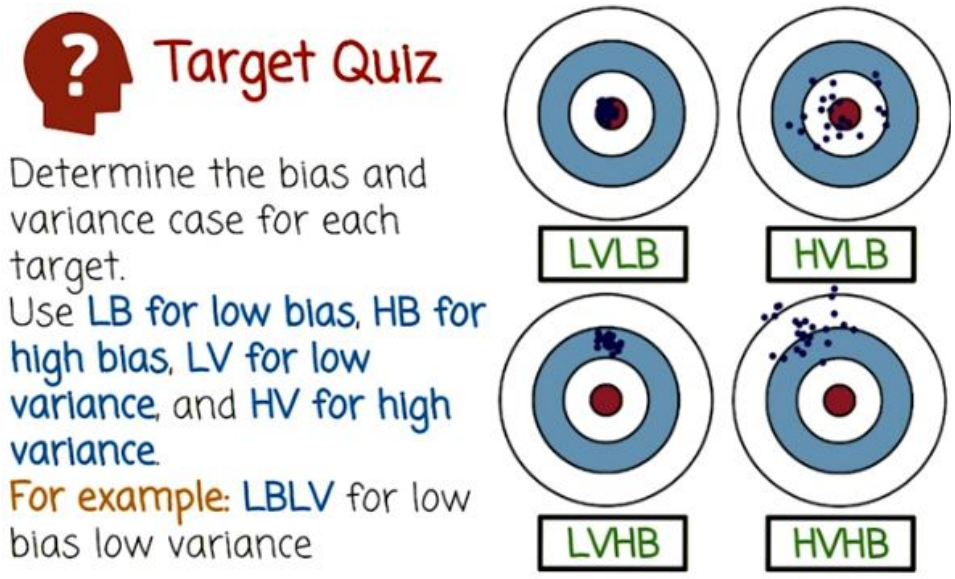
- Low bias
- High variance
- Overfitting training data



Not overfitting training data nor underfitting training data. Small mistakes are made in predicting training data labels, but low error on predicting future data.

- Just right!

Target Quiz



For LVLB
- There is no systematic deviation (from the bullseye; they all appear to be centered) and there is low variance (they are clustered)

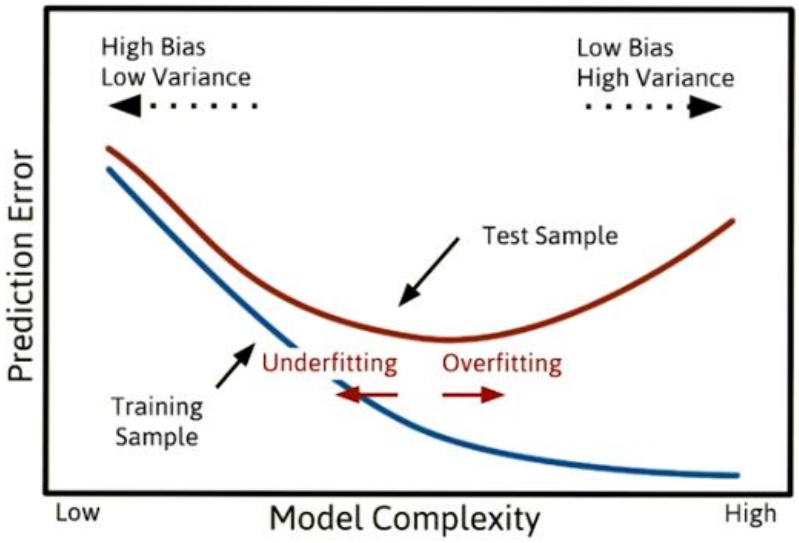
For HVLB
- High variance because the cluster is not as tight; low bias as the center of all of the points are on target
  - If this were an experiment, we would get somewhat drastic results but they would aggregate to close to the right answer (which still isn't helpful without bagging)
  - This is a case of overfitting

LVHB
- This is a case of underfitting

HVHB – at this point these are intuitive

BRENTS NOTE: This quiz seems to be against what we learned above, but its really not. In this case, the targets with 'high bias' are highly biased because the grouping are off target; the high variance selections are labeled as such as the grouping is not as tight

Goldilocks Principle



We wish to minimize the test sample prediction error
Moving to the right means adding more features
- Removing features is a way to minimize prediction error
  - There are other ways too (will be mentioned) but this is simplest

This is assuming the training set is fixed
We can play with the model complexity, but be careful not to complete the complexity too much
If you can increase the training set size, it would be beneficial to attempt to make the model more complex

Overfitting Quiz

Select the **best method for addressing overfitting** when there are a lot of features:

- [x] Reduce the number of model parameters (often implying fewer features and feature transformations)
- [x] Keep all features but reduce the magnitude of some features

Either reducing the feature count OR the magnitude of some features could work
Reducing the magnitude uses regularization
- By making the parameters smaller, each feature is smaller
- The data indicates how much to reduce the magnitude or the important of some features
  - Its completely data driven
- This is an effective an easy way of preventing overfitting

Which method is best (Feature reduction or reduction of magnitude)
- This depends on the machine learning method
- In linear modelling (logistic / linear regression), reduction of magnitude is usually tried first

Feature reduction is popular for other algorithms, such as trees

MLE (Maximum Likelihood Estimator)

Recall that the MLE is a technique for estimating model parameters

- It answers the question, 'which parameters will most likely properly characterize the dataset / best predicts the data?
- Recall that the likelihood function is a product of the probabilities of the training set samples

We search for the parameter that maximizes the likelihood function

Sometimes, instead of looking at the likelihood we look at the log likelihood as its easier to maximize computationally

Traditional statistics usually refers to where the number of parameters (the model dimensionality) is fixed while the training set size increases to infinity

- This is often denoted as d<<n
  - 'd' is the number of parameters
  - 'n' is the training set size
  - '<<' implies 'much smaller'
- In the d<<n case, the MLE performs well
- Asymptotic optimality kicks in
  - **Asymptotic optimality**: as your training set approaches infinity, the model will converge to the ground truth or 'nature' that generates the data, and the convergence happens at the fastest possible rate
  - Asymptotic variance of the MLE is the best possible
  - Asymptotic variance is the **inverse fisher** information

The d<<n turns out to be a huge assumption

- We do not know that d is much smaller than n
- Typically, d is not much smaller than n
  - If this is the case, MLE performs poorly
  - We must control for overfitting
    - Its hard to tell which features are noise and which are important signals
- This is a big problem as many datasets out there have, in some cases, millions of features and not enough records
  - Regularization is important

Mean Squared Error

We are looking at the MSE of this estimator:

$$\mathbf{E}_{p\_true} || \hat{\theta} - \theta^{true} ||^2$$

- The estimator Θ(hat) is trying to estimate the ground truth Θ(true)
  - It can be logistic or linear regression – doesn't matter
- The MSE is the square of the difference of the two parameters, taken in expectation over the distribution that generates the data

Recall that the MSE = Bias^2 + Variance

- The Maximum Likelihood Estimator for linear regression is unbiased (meaning bias == 0)
  - In other words, variance is the only component of the MSE
- Somehow, its still possible to overfit with bias = 0
- We can actually increase the bias here
  - We can use regularization by introducing bias to reduce variance
  - Why would we want to increase the bias via regularization?
    - Because it can reduce the variance to the point where we have a lower overall MSE
    - Bias can reduce variance and lower MSE overall

In the case of linear models, **bias to simplicity** can reduce variance and lower MSE overall

- **Simplicity bias** (another way of saying bias to simplicity) can be thought of as reducing the

parameters of the model towards 0; this does increase the bias but it lowers the estimation variance and thus lowers the MLE, which may result in better overall estimation accuracy and less overfitting of training data

- In other words, increase bias to reduce variance and the MSE overall
- We basically have to reduce | Θ^{MLE} |
  - That is to say, reduce the values of the parameters
  - Note it's the absolute value of the coefficients we learned – we need to reduce these to 0 as close as we possibly can

    <u>This would mean the model places less importance on the corresponding feature, which is the entire point</u>

    Even though the number of features stay the same, the importance of the features are reduced and thus the model becomes simpler and thus less likely to overfit

Regularization Methods to be Reviewed

Methods

- James-Stein Shrinkage
- Breiman's Garrote
- Ridge Estimator
- Lasso Estimator
- Elastic Net Estimator

All of these can be used when the traditional statistical case where d<<n falls apart

James-Stein Shrinkage

Traditional statistical theory states 'no other estimation rule for means is uniformly better than the observed average'

- In traditional statistics theory, there is no better way to estimate the mean or expectation than by computing the average

The **James-Stein Estimator** says: when estimating multiple means: shrink all individual averages towards a grand average

- The James-Stein estimator addresses a situation where we have to estimate multiple means at the same time (means of different, random variables)
- We want to estimate the mean simultaneously

In the James-Stein, we compute all entity's separate means, but then we shrink all individual means to the 'grand average'

- This was shocking at first, but it has been shown to – in some cases – perform better than the traditional way of just computing the average for each individual entry

One point about James-Stein that makes it stick out is the James-Stein Estimator shrinks all the dimensions of Θ(hat) uniformly towards the grand average.

- This can be seen as a limitation, as it does not shrink different averages in different ways, which is something we WANT to do
  - We want to avoid estimating the noise, but NOT the signal

Note that the James-Stein Estimator isn't necessarily the 'best' form of regression, but we talk about it as it has important historical connotations

Examples of estimating the average / proportion of three different quantities

- Quantities
  - People who (will) vote for Clinton
  - Babies in China that are girls
  - Americans that have light-colored eyes
- Kind of bizarre, as why would we want to estimate these things together

- In the case of high-dimensionality, there is reason to think of this as we will have a large number of variables
- The variables may be weakly related

◦ In the James-Stein estimate, the proportion of voters for Hillary Clinton depends on …..Chinese baby data and eye color

◦ A big surprise in the statistical world is: this sort of makes sense

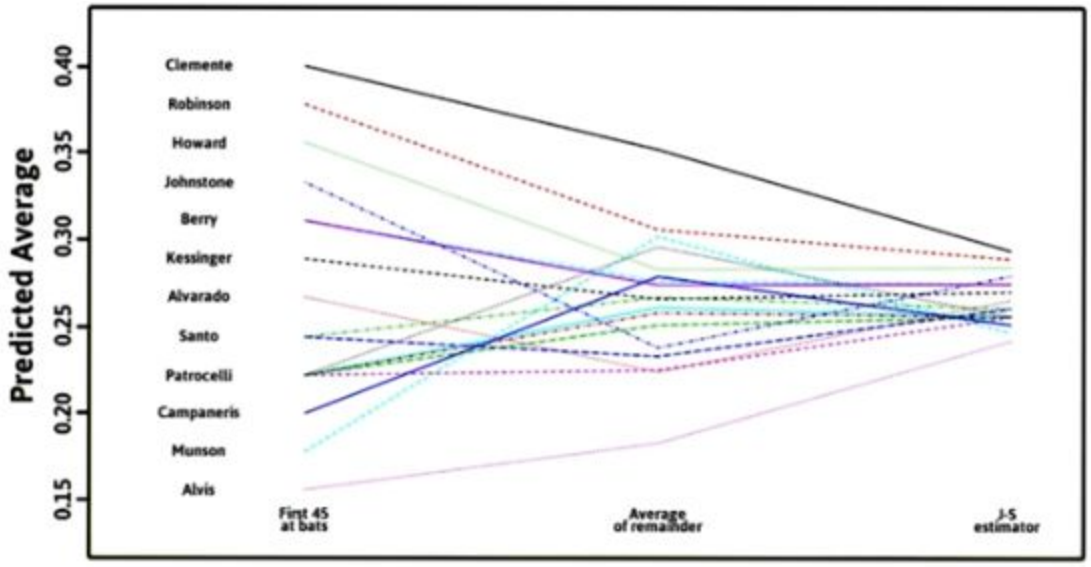
Thinking about the example more in the context of math

$$\hat{\theta}^{\mathrm{JS}} = \frac{1}{1+\tau}\hat{\theta}^{\mathrm{mle}}, \ \ \tau > 0$$

◦ The James-Stein estimator takes the maximum likelihood estimator and shrinks it (for example, towards 0, so it simply reduces it – so
- multiply it by 1/(1+T),where T >0

  Θ(hat) and Θ(JS) are both vectors; they are d separate equations (one for each dimension)
- Each dimension of the MLE we are going to shrink it and then multiply it by 1/(1+T) to get the value of the James-Stein estimator
- Its important to note that the 1/(1+T) does not depend on the dimensionality of the features; they are all multiplied by the same coefficient

James-Stein Weakness Quiz



Source: http://www.r-bloggers.com/example-9-27-baseball-and-shrinkage/

This computes the average batting average for player's using on the first 45 at-bats

What we really want is the season batting average…but we do not have this yet

One way is saying the average we will estimate will be: for each player, simply compute the average over the first 45 at-bats

◦ This is the traditional statistics solution

The James-Stein estimator says we need to also compute the grand mean (the mean of all the means together), shrinking all of the means after 45 days towards the grand mean (resulting in the right-most number on the graph)

The actual ground truth is the average of the remainder (middle number)

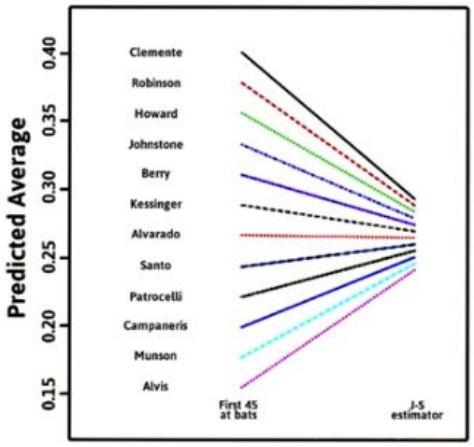
◦ Note that both the traditional AND the James-Stein estimator fails, but the question is: which one is closer

- We do note that the James-Stein estimator is ‘more correct’, as shrinking towards the grand mean is the correct thing to do (even if its still inaccurate)
- Intuitively, this can be explained by ‘there is some extra variability that is just due to the sampling and the fact that we don’t have enough data; the long-term behavior of these baseball players would be closer to the grand mean rather than reflected by the initial performance after the first 45 at-bats

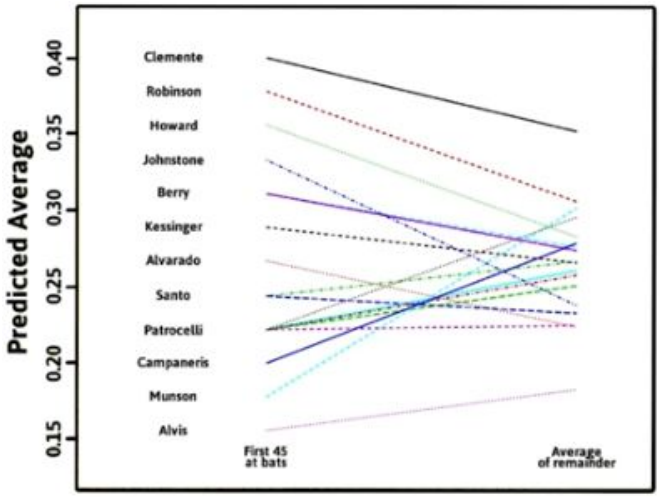
Question: What is the weakness of the James-Stein Estimator?

- Answer
  - Workthrough

    Look at the first 45 at bats, then at the ground truth for the remainder of the season, and then what happens with the James-Stein estimator:



    All of the individual dimensions are shrunk by the same amount (1+(1/T)
    - This is clearly not reality if we look at the slide that shows the actual average remainder:



    - We see that shrinkage DOES indeed happen, but its not uniform
      - Sometimes the shrinkage is weak, sometimes its large
      - Sometimes the ordering between the dimensions or the players gets reversed (that is to say, two lines may cross as they map the initial average to the final average
      - The James-Stein estimator cannot capture this

        It keeps shrinking all players / all parameters / all dimensions the same towards the grand mean (or, if we want to control overfitting in linear regression, towards 0)

        It will not allow the data to show us that the shrinkage should be done in

different amounts for different dimensions
    - For example, if there is a feature that is less relevant, it should be shrunk more than others
- Final answer: It will not allow the data to show us that the shrinkage should be done in different amounts for different dimensions

Breiman's Garrote

Breiman's Gattote brings the maximum likelihood estimator unlike James-Stein shrinkage: each dimension is different. Observe:

$$\hat{\theta}_j^{\text{garrote}} = \hat{\theta}_j^{\text{mle}} \hat{\tau}_j, \quad j = 1, \ldots, d$$

- Note that each individual Θ(hat) has its own T(hat)
- We are going to look at the Jth dimension of the MLE vector and shrink it by a coefficient constant T_j
- Basically different dimensions of the MLE could be more or less

Determining T

- To find Tau we need to solve a separate optimization problem, similar to maximum likelihood:

$$\hat{\theta}_j^{\text{garrote}} = \hat{\theta}_j^{\text{mle}} \hat{\tau}_j, \quad j = 1, \ldots, d$$

$$\text{where} \quad \hat{\tau} = \arg\max_{\tau} \ell(\hat{\theta}_1^{\text{mle}} \tau_1, \ldots, \hat{\theta}_d^{\text{mle}} \tau_d),$$

$$\text{subject to} \quad \sum_{j=1}^{d} \tau_j \leq c$$

- Find the T that maximizes the log likelihood where the vector of parameters instead of Θ_1 to Θ_d is $\Theta_1 * T_1$ to $\Theta_d * T_d$, subject to a constraint that the sum of the are T_j 's are less than or equal to some number c

  When c<d, some or all of the components of $\Theta(\text{hat})^{MLE}$ may be reduced in absolute value towards 0 (in other words, they can approach or actually be 0).

  When we constrain $T_j > 0$ it is called the **non-negative garotte**. The parameter c is a '**tuning parameter**' and several different values are typically considered
    - The non-negative garotte is usually standard
    - We usually try different values of c and test the solution, usually on test / held-out / out-of-sample data, then select the model that did the best on the data
        - Doing this is common in regularization
        - Usually there are 1 or 2 '**hyperparamaters**' or tuning parameters (like c)
            we do not know the best selection, so different values are tried
            typically a 'grid' of values is formed, and the 'garrote' is solved for every value of c in the grid of values evaluated on a test set of data
            - from this, the best model that performed on the test set is chosen

  The shrinkage is learned from the data
    - That is to say, the different theta modifiers (T) are determined by running the data through a minimizer of sorts, finding the argmax l (<u>lower case l here stands for the log likelihood</u>)

<u>One issue is computation can be expensive in high dimensions</u>

- This is because we are solving an optimization problem
- There are usually no problems in low dimensions, but there are in high dimensions
- The extra cost of computing Breiman's Garrote in high dimensions must be considered!

Wrap-up

- In Breiman's Garrote, some dimensions are shrunk more than others on an as-needed basis
- Breiman's Garrote is a poor choice for high-dimensional data due to the computational cost!

Ridge Regularization

James-Stein was proposed for means and garrote was proposed for regression

- In the same vein, Ridge Regression was also proposed for linear regression but the same principle can apply for other models (logistic regression and other methods)

Definition of Ridge Regression

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_2^2 \leq c$$

- Maximize the likelihood (or the log likelihood) subject to a constraint
  - The constraint is the L2 norm of the vector theta is less than some constant c
    Recall that the L2 norm is the sum of squares for Theta!
  - The L2 norm squared is just the sum of the squares of the dimensions of theta
    Keep in mind that Θ is a vector
    In other words, the L2 norm squared is $\Theta_1^2 + \Theta_2^2 + \Theta_3^2 + \ldots.. + \Theta_d^2$
    More generally:

$$\|v\|_p = \left(\sum_{j=1}^{d} |v_j|^p\right)^{1/p}$$

    you can think of the p norm (in this case it's the L2 norm but also the Lp norm) we have the absolute value of each component of the vector raised to the power p (that is to say, $|v_j|^p$), summed up and then taken to the power 1/p.
    If we do the math and substitute 2 for p and we get what we saw earlier (in the 'subject to' screenshot above)
    - Because its raised to the second power it cancels with the square root, so you just get the sum of squares

The problem, again

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_2^2 \leq c$$

- As stated, this is not easy to solve because it's a constraint optimization problem
- This can be reformulated using something called the **Lagrangian**, which is a tool from calculus
  - Lambda is considered to be the Lagrange multiplier
  - BRENTS NOTE: He said to read up on the Lagrangian
- The Lagrangian is defined as the optimization function minus the constraint:

$$\mathcal{L}(\theta, \lambda) = \ell(\theta) - \lambda(\|\theta\|_2^2 - c)$$

  - In our case, it's the L2 norm ($||\Theta||_2^2$) minus c
    This will be less than or equal to 0
  - (again, note that lower case l simply means the log likelihood)
  - Then, the L2 norm is multiplied by the Lagrange multiplier, lambda

- At this point, what the Lagrange multiplier is telling us:

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_2^2 \leq c \quad \Rightarrow \quad \hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) - \lambda\|\theta\|_2^2, \qquad \lambda \geq 0$$

- If we want to solve the original problem (left side of the =>), instead of solving this we can just solve a different problem (on the right) that is simpler
- The right side
  - This is just setting the derivative of the Lagrangian to zero with respect to theta
  - Recall that the Lagrangian is

$$\mathcal{L}(\theta, \lambda) = \ell(\theta) - \lambda(\|\theta\|_2^2 - c)$$

    The log likelihood (of theta) minus the Lagrange multiplier times the constraint
    - Note the constraint c above is a bit different than it is a few lines up as we moved c to the left-hand side, so the constraint is something less than or equal to 0
- The gradient (with respect to theta) must be 0:

$$\nabla_{\theta} \mathcal{L} = 0$$

  - From this we can find the **stationary point**
    When we do this, this will give us the solution to the optimization problem that is also the solution to the original problem!
- Why is the right hand side easier to solve
  - We can take the gradient and do stochastic gradient descent (or even just matrix operation) to find a closed-form solution
  - The equation on the right-hand side has a closed-form solution
    This means we don't even need to do gradient descent (we will see this later)
- More about the stationary point
  - We need to find the stationary point if the Lagrange with respect to lambda or set the gradient with respect to lambda to 0
  - In practice, we do not need to do this because we do not care about achieving the lambda that will correspond exactly to the c threshold (the c in the 'subject to' line) because we do not know what the right threshold is nor do we know what the right lambda is
  - In practice, we solve the optimization problem on the right (again this is:

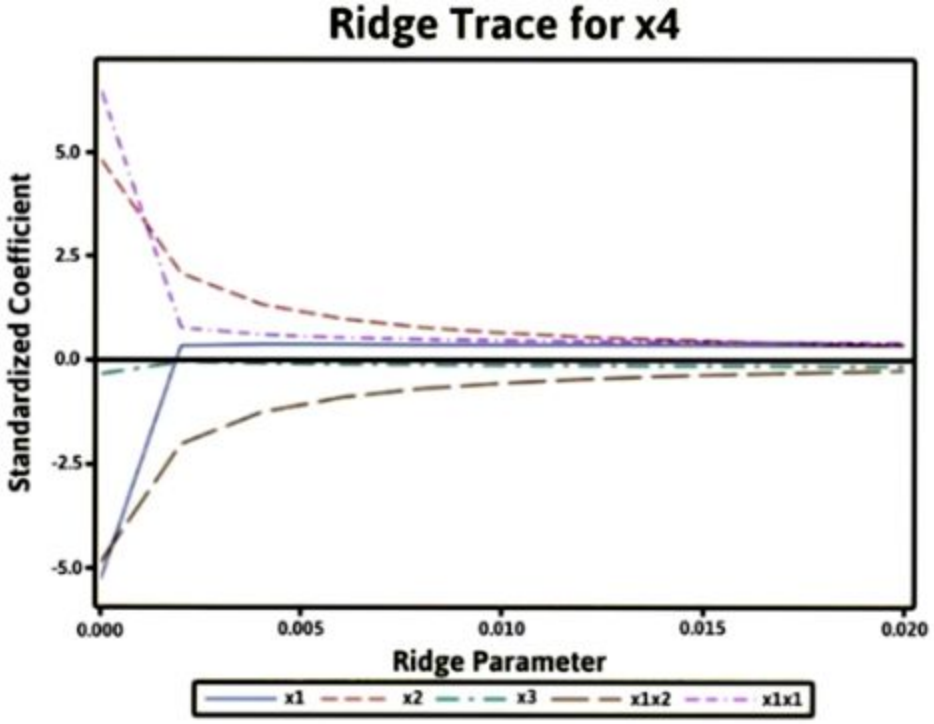
$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) - \lambda\|\theta\|_2^2, \qquad \lambda \geq 0$$

    We solve this for different values of lambda on a grid of values
    We evaluate each solution on a test / held-out set and then select the model or the lambda that performed the best on the held-out set
    This is not that much more expensive than a regular linear regression or logistic regression
    - If we are willing to solve a linear or logistic regression in high dimensions, adding this penalty is almost for free
    - Whether we use SGD (aka gradient descent) for logistic regression, or if its linear regression we can use either SGD or find the closed form solution
      - The cost is the same, and the solution will usually perform much better than just the MLE, especially when we are using high dimensions
      - In particular, this will be better if we are careful about evaluating for different values of lambda on a specific grid that is sufficiently fine-spaced AND we pick the best model

Ridge Regularization penalizes large values of Theta

Ridge Trace

This is an example of a **ridge trace**



This is also called a parameter trace

The x axis corresponds to a ridge parameter (OR: the lambda)

As we increase the lambda, we penalize large values of theta more

As we increase the ridge parameter, the coefficient goes down towards 0 – this is shrinkage

- By doing so, we reduce the model complexity and overfitting

Looking at the graph, we can see the dependency of the coefficient size on lambda

- As we go to the right the coefficients get closer to 0
  - To see at what pace the model complexity is contained is insightful to see

Estimator Comparison Programming Quiz

We will look at an example of ridge regression (using linear regression)

The model:

$$E(Y) = 0 + X1 + X2 + e \; ; \; e \sim N(0,1)$$

- 'Y' is generated by X1 plus X2 plus Gaussian noise
- X1 and X2 are uniform between 0 and 1 (aka U(0,1))
- X3 is 10*X1 + unif(0,1)
  - Unif(0,1) is a uniform random variable between 0 and 1
  - Notice that the correlation between X2 and X3 = sqrt(100/101) = .995

Quiz: estimate the linear regression model based on X1, X2, and X3and examine the model summary

Code:

```
#Create a model summary given the following information.
```

```
library(MASS)
N = 20      # Sample size
x1 = runif(n=N)
x2 = runif(n=N)
x3 = runif(n=N)
x3c = 10*x1 + x3
ep = rnorm(n=N)
y = x1 + x2 + ep

#TODO: add the commands to
## OLS fit of 3-variable model using correlated x3.
#ANSWER BELOW#####
olsc <- lm(y~x1+x2+x3c)
summary(olsc)
```

Output:

```
Call:
lm(formula = y ~ x1 + x2 + x3c)

Residuals:
     Min       1Q   Median       3Q      Max
-1.40669 -0.72876 -0.02766  0.65424  2.07918

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  -0.8779     0.6333  -1.386  0.18468
x1           11.1823    11.0083   1.016  0.32484
x2            2.8920     0.7569   3.821  0.00151 **
x3c          -0.8402     1.0395  -0.808  0.43082
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.9907 on 16 degrees of freedom
Multiple R-squared:  0.5599,        Adjusted R-squared:  0.4774
F-statistic: 6.786 on 3 and 16 DF,  p-value: 0.003662
```

Ridge Regression Quiz

Question: Using the same base code as above (everything above ‘TODO’), use ridge regression in R and find the summary

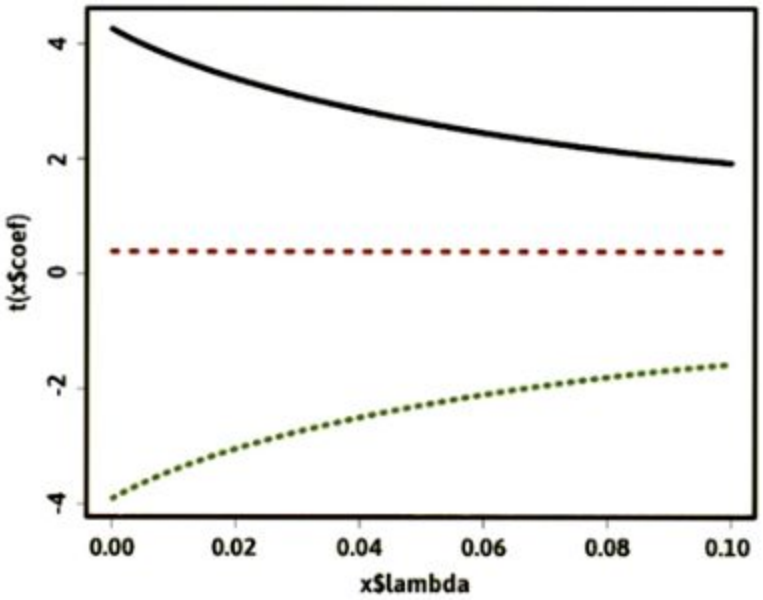
Answer:

```
ridgec <- lm.ridge(y~x1 + x2 +x3c,lambda=seq(0,.1,.001))
summary(ridgec)

#####OUTPUT#############
       Length Class  Mode

coef   303    -none- numeric

scales   3    -none- numeric

Inter    1    -none- numeric

lambda 101    -none- numeric
```

```
ym       1    -none- numeric
xm       3    -none- numeric
GCV    101    -none- numeric
kHKB     1    -none- numeric
kLW      1    -none- numeric
```

- Note that lambda is set to a range of 0 to .1, with increments of .001

If we plot the Ridge coefficient as a function of Lambda:



- There are three coefficients, and they generally shrink towards 0 (even the red line which appears to be flat)

Lasso Estimator

Lasso = Least Absolute Shrinkage and Selection Operator

Lasso is similar to ridge in the fact that its a penalized Maximum Likelihood solution; the difference is it uses L1 instead of L2

Intuitively, Ridge will include all coefficients / parameters of the MLE, but the parameters will still be present (aka nonzero)

- Conversely, Lasso does parameter estimation AND variable selection (parameters shrunk to 0)
- Note that Ridge CAN set ALL parameters to 0 if the penalty lambda is high, but realistically this does not happen

Recall the Lagrange version of Ridge Regression

$$\hat{\theta}^{\text{ridge}} = \arg\max_{\theta} \ell(\theta) - \lambda\|\theta\|_2^2, \qquad \lambda \geq 0$$

- Lasso is similar:

$$\hat{\theta}^{\text{lasso}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_1 \leq c$$

  - Instead of having a constraint of L2 norm we use L1 norm

    The L1 norm corresponds to the sum of the absolute values of the components of theta

    BRENTS NOTE: Dr. Lebanon said to go back a few videos to review the definition of

a lp norm

- Note that this is the primal formulation (aka the constraint optimization); it is NOT easy to optimize in this form; again, just like Ridge, we have to change it via Lagrange multipliers by solving the dual problem:

$$= \arg\max_{\theta} \ell(\theta) - \lambda\|\theta\|_1, \qquad \lambda \geq 0$$

This maximizes the Lagrangian minus some sort of penalty

- Again, the penalty is the L1 penalty, so this is the sum of absolute values instead of the sum of squares

This is very similar to Ridge: we still maximize the likelihood, but we add a penalty of the sum of absolute values (L1) instead of the sum of squares (L2)

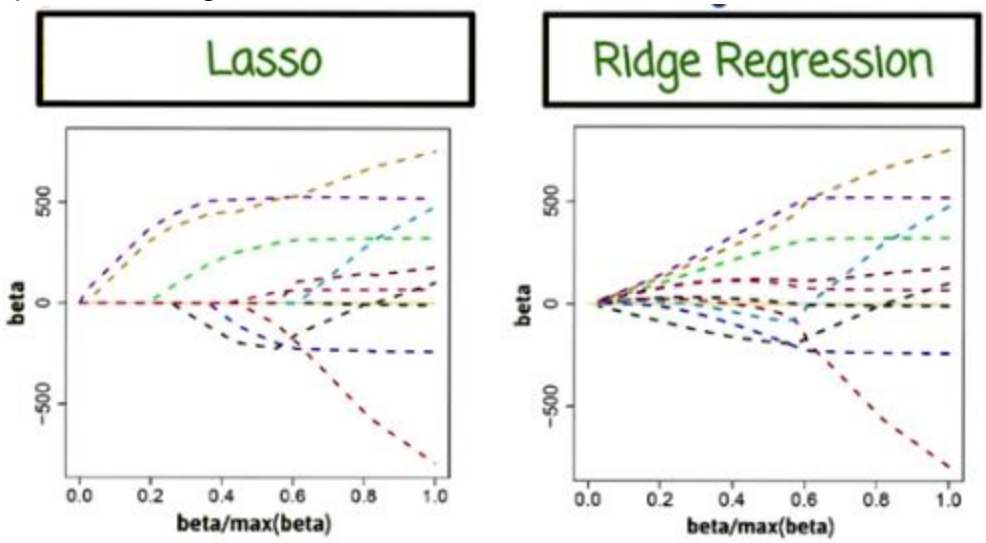
- That said, there is a fundamental difference to L1 and L2: L1 encourages sparse solutions, meaning some components will be 0
- L2 does NOT encourage sparsity; the coefficients of ridge may be close to zero but not precisely zero

◦ Taking some coefficients / weights / thetas makes the model simpler and computationally faster to evaluate later on in the query phase

- Lasso can eliminate up to 90% of features in large datasets (as an example)
- When most coefficients are 0, its far easier to perform the prediction
- Sparsity has significant advantages in high dimensions in terms of being computationally easier as well as the model being more interpretable
- Again, it should be stated that not only is feature selection done, we are ALSO reducing the values in the coefficients just like Ridge

Lasso Ridge Quiz

Which Graph is of a Ridge solution, and which is a Lasso solution?



Lasso has coefficients that are 0 where Ridge does not

Linear Regression

The methods in this lesson generally apply to maximum likelihood problems

The linear regression setting enables derivation of closed forms which provide insight into the

differences between different regularizers

- In this sense, a closed form is something like linear regression where we can easily reason theoretically about; this may not work with harder models

However, in practice ridge and lasso can also be applied for other models like logistic regression (and they can be very successful)

- Regression is a general way of controlling for model complexity

Lets see how the math simplifies in the case of linear regression

- We wont be able to get the nice math in other models (like logistic regression), but the same intuition applies

In the case of linear regression, remember that

- X is a matrix of training data $X^{(1)}$, ….., $X^{(n)}$
- Y is a vector whose entries are training labels $Y^{(1)}$, ….., $Y^{(n)}$

Then, the conditional log likelihood is

$-(Y - X\Theta)^T(Y- X\Theta) + c$

- The squared error between the predicted values of Y and the actual values (represented in matrix form above)

If we want to maximize the log likelihood, we set its gradient to 0:

Setting its gradient to zero: $-X^TX\theta + X^TY = 0$ Implies: $\hat{\theta}^{mle} = (X^TX)^{-1}X^TY$

- Setting it to zero means taking the derivative and setting it to 0, but it's tricky to do this in matrix form

Reasoning

- <Dr Lebanon goes through the math quickly in 'Linear Regression Part 1', from 2:10 to 3:05; he goes from the 'conditional log likelihood (few lines above) and ends at the 'implies' (also above)>
- The 'implies' form is the closed-form solution of linear regression

Special case of orthonormal training examples:

$$X^TX = I : \hat{\theta}_j^{mle} = X^TY$$

- The orthonormal projection of the columns of X and Y
- When we take inner products of different training samples are 0, unless it's the same example and in which case its 1
- In this case, $X^TX = I$
  - I = identity
- This will typically never happen, but it allows us to see the equation simplified, where we can understand intuition and theory

In the case of ridge regularization: setting the penalized log likelihood gradient (the log likelihood plus the L2 norm) to zero

$$-\mathbf{X}^\top\mathbf{X}\boldsymbol{\theta}+\mathbf{X}^\top\mathbf{Y} - \lambda\boldsymbol{\theta} =$$

- The gradient of the L2 norm is just $\lambda\Theta$, with λ being the regularization parameter
- When we set it to 0 we get the matrix equation and get something similar to what we had before:

$$\hat{\boldsymbol{\theta}}^{\mathbf{ridge}} = (\mathbf{X}^\top\mathbf{X} + \lambda I)^{-1}\mathbf{X}^\top\mathbf{Y}$$

  - The only difference is, with the matrix X^TX we add λI (remember I is the identity matrix)

this is insightful as it helps in cases where the matrix X^TX is not invertible
- If its not invertible, we cannot compute its inverse
  - that said if we add a diagonal matrix times some value – even if it's a small value – we turn the non-invertible matrix into an invertible one, becoming coming more stable
  - This is a different way to think about high dimensional linear regression in an algebraic sense (rather than a statistical sense of overfitting)
    In an algebraic sense, the matrix X^TX becomes non-invertible and we regularize / simplify the model by adding some small number to the diagonal in order to make it invertible and thus more stable

- Recall the orthogonal case (where the samples are orthogonal)
  - In this case we can get a simple expression of ridge if we follow the derivation we had before and add ridge to it
  - The closed-form solution of the ridge in the orthogonal case is

$$\hat{\theta}^{\text{ridge}} = \frac{1}{1+\lambda}\hat{\theta}^{\text{mle}}$$

  - We take each component of the MLE and we shrink it towards zero by the same constant factor:

$$\hat{\theta}^{\text{ridge}} = \hat{\theta}^{\text{JS}}$$

(This is the same as the James-Stein estimator!)
<u>In the orthogonal case, ridge regression is the same as James-Stein for linear regression!</u>
<u>That said, this ONLY holds true for the orthogonal case using linear regression! We no longer have a closed-form so we cannot follow the derivation for more complex models!</u>

In the case of Lasso regression
- Things are more complex; we can still get the closed-form solution in some cases (notably, again, the orthogonal case)
- Start with:

$$0 = \nabla(\ell(\theta) - \lambda\|\theta\|_1)$$

  - Take the log likelihood penalized by the L1 norm (this is the objective function we want to maximize in the case of Lasso
    We compute its gradient (above) and we set it to zero; this leads us to:

$$0 = \nabla(\ell(\theta) - \lambda\|\theta\|_1) \Rightarrow -\mathbf{X}^\top\mathbf{X}\theta + \mathbf{X}^\top\mathbf{Y} - \lambda s(\theta) = 0$$

    's' is the sign of theta (for each individual Θ)
    - The derivative of the L1 norm is the sum of the absolute values; the derivative of the absolute value is just the sign of the number EXCEPT at zero (where the absolute value is not differentiable)
      - Assuming we are not at zero , the derivative is $s\Theta$
        This turns out to be the notation of a vector of +1/-1 depending on the sign of each coefficient
- In the orthonormal case, the penalized log likelihood is:

$$-(\mathbf{Y}-\mathbf{X}\theta)^{\top}(\mathbf{Y}-\mathbf{X}\theta)/2-\lambda\|\theta\|_1$$

- This can be reduced (using algebra) to:

$$(\mathbf{Y}^{\top}\mathbf{X}\theta+\theta^{\top}\mathbf{X}^{\top}\mathbf{Y}-\|\theta\|_2^2)/2-\lambda\|\theta\|_1$$

After the algebra, the simplification we get is because X^TX is the identity
Also recall that X^TY is the MLE solution for the orthonormal case, so we will replace that by theta (hat) (MLE):

$$\theta^{\top}\hat{\theta}^{\mathbf{mle}}-\|\theta\|_2^2/2-\lambda\|\theta\|_1$$

Solving this vector equation (setting it to 0) it decomposes into 'd' separate maximization problems:

$$\theta_j\hat{\theta}_j^{\mathbf{mle}}-\theta_j^2/2-\lambda|\theta|_j$$

- These can be solved independently
- Remember that, like all of these orthogonal cases, these are VERY special cases and CANNOT be used in general

For each j, if $|\hat{\theta}_j^{\mathbf{mle}}| < \lambda_j$ the objective function will be negative unless $\theta_j = 0$ then the objective function = 0

<Dr. Lebanon skips how these are solved independently as the absolute value is not differentiable>

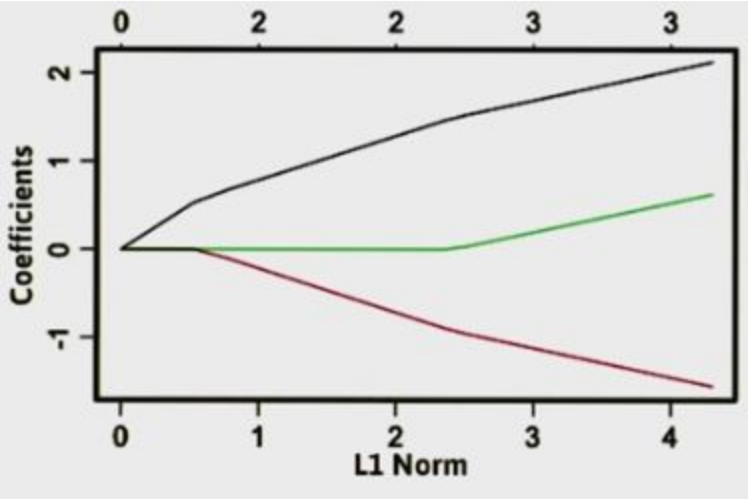
- In the case of lasso regression, combining the two cases and solving for the individual 'd' we get:

$$\hat{\theta}_j^{\mathbf{lasso}} = \mathbf{sign}(\hat{\theta}_j^{\mathbf{mle}})(|\hat{\theta}_j^{\mathbf{mle}}| - \lambda)_+, \quad \text{where} \quad A_+ = \max(A, 0)$$

- The 'd' component is the sign of the jth component of the MLE times (the absolute value of the (jth component of the MLE) minus lambda), positive part, where positive part is the argument of the positive part or 0 (whichever is greater)
- The above is the closed-form solution to the lasso regression in the case where the data is orthonormal
- What is happening in the parenthesis with the absolute value and the sub plus sign?
  We take the absolute MLE coefficient and reduce it by lambda towards zero
  If it goes beyond zero, the sub '+', together with the where 'A_+ =', means 'just make it 0 instead of being negative'
  - Note that this uses the absolute value, so this is basically saying if it crosses zero, just zero it out entirely
  - This is the sparsity encouraging aspect of Lasso
  - If the expression is greater than zero, we need to give it the right sign (hence the 'sign')
  - <u>This helps to show that while ridge will only reduce a coefficient towards 0, Lasso will zero certain coefficients out, ignoring the variable</u>
- Therefore, the sparsity encouraging threshold nature of lasso as it zeros out small coefficients

Lasso Plots

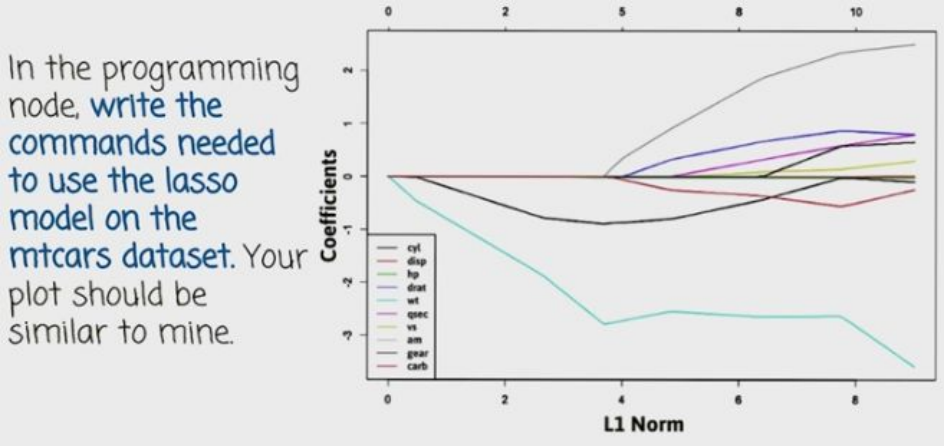
Install 'glmnet' in R
This is the Lasso graph in the uncorrelated case:



- We have features that, as we increase the penalty, they go down
  - Once they get to zero, they stay there
- In the other direction (as we decrease the penalty), we can think of the coefficients as entering the model
  - They enter the model in the order of the magnitude of the linear regression coefficients

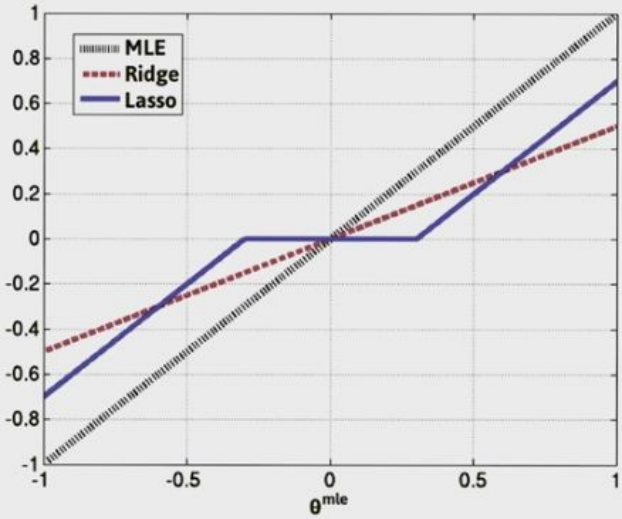
The predictors go into the model in the order of their magnitude of the true linear regression coefficient

MTCars and Lasso Quiz



<see code document>

MLE Comparisons



We will be comparing the MLE graphs in the orthonormal case, as it will give us an intuition as to what happens in them
The X axis maps to the MLE parameter and the Y axis maps to the parameter of the MLE
The MLE maps to itself using the identity (y=x) function; it maps to itself
Ridge maps the MLE into a shrunk version of the MLE (similar to James-Stein)
The Lasso graph shows

- the shrinkage is subtractive (instead of multiplicative)
  - we simply subtract lambda from the MLE value
- the shrinkage is the same UNTIL we hit a point where subtracting the lambda would take it below 0, at which point it flattens at zero (at which point the variable is ignored)

Ridge shrinkage is multiplicative shrinkage while Lasso is subtractive shrinkage
This graph – of the orthogonal case – shows why Lasso has more sparse solutions

Elastic Net Estimator

The **elastic net estimator** is a linear regression model trained with L1 and L2 regularization (Ridge and Lasso)
This method is very common in high dimensional modelling in industry

Formulation:

$$\hat{\theta}^{\text{elnet}} = \arg\max_{\theta} \ell(\theta) \quad \text{subject to} \quad \|\theta\|_1 \leq c_1, \|\theta\|_2^2 \leq c_2$$

- We maximize the likelihood subject to a constraint on the L1 norm and another on the L2 norm

We then form the Lagrangian (as we do not want to solve the optimization problem):

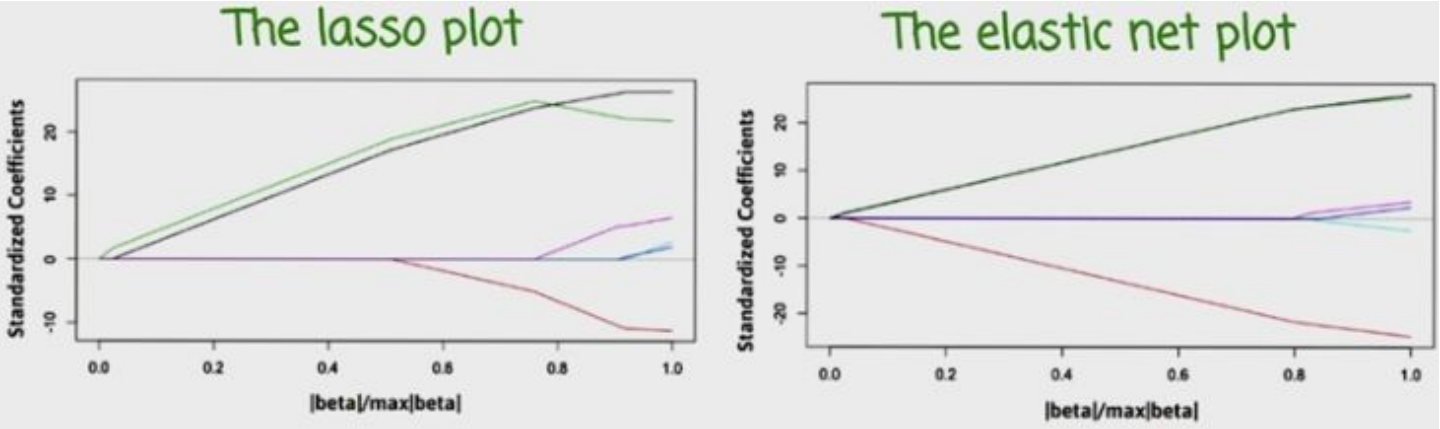
$$= \arg\max_{\theta} \ell(\theta) - \lambda_1\|\theta\|_1 - \lambda_2\|\theta\|_2^2, \qquad \lambda \geq 0.$$

- This is a dual-optimization problem with two penalties and two lambdas
  - We can play with the lambdas in order to get different types of regularization
  - We can also get different amounts of sparsity
    If we want more sparsity, we can increase lambda 1
    If we do not want sparsity, we can make lambda 1 smaller and concentrate on lambda 2 (make it larger)

In general, since the elastic net includes both L1 and L2 as special cases, if we set lambda 1 to zero we get Ridge and setting lambda 2 to zero we get Lasso
The elastic net is considered to be 'more general' than Ridge or Lasso and its not much more computationally expensive

- Basically, if we use Lasso we might as well use elastic net
  - BRENTS NOTE: Presumably because Lasso may be a bit more computationally expensive over Ridge

Elastic Net Graph



- The graph (right is Elastic Net, left is Lasso) is something between Lasso and Ridge
- Note that it takes the coefficients longer to reach 0 (this is due to the settings of L1)
- Sparsity is not as aggressive; that said, we still get sparsity as well as reduction in magnitude

Lesson Summary

Overfitting is a significant problem in high dimensions
It can be handled using simple adaption to linear regression and logistic regression: adding L1 (Lasso) or L2 (Ridge) penalty

Course Summary

We should be comfortable with R, data visualization, and analytics. Good luck in the future!