Data, Environment and Society: Lecture 19: Model Selection and Regularization

Instructor: Duncan Callaway
GSI: Salma Elmallah

October 17, 2019

Introduction

We've been working a lot with this model:

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon$$

And we know well how to fit it with least squares regression.

Today's topic: how can we improve on the "standard" strategy of choosing coefficients via a one step application of least squares regression?

Stop using OLS? Wait, why improve on a good thing?

Let n = number of observations, and p = number of features.

1. When n close to p in size, least squares fit can have high variance $\rightarrow$ overfitting.
2. When p is big, if every variable gets a coefficient it can be hard to interpret models.
   - A subset of model selection known as "feature selection" helps get rid of some features.
   - This lets you "throw in the kitchen sink" with the faith that the feature selection process will get rid of it if it's not helping in prediction.
   - Note, interpreting coefficients is ok with something called subset selection (which is essentially OLS, and is next up), but not a good idea with most other machine learning methods

Ok, how do you do it?

We'll look at two kinds of methods in Ch 6 (today and next Tuesday):

- Subset selection – today
- Shinkage – today and tuesday

Today's Objectives

- Refine our understanding of model identification as an optimization problem
- Learn how to adjust your errors to compare models with different numbers of predictors
- Understand what "regularization" is and why we do it
- Understand the tradeoffs between subset selection, ridge regression, and lasso

Aside: n choose k

To aid in discussing the process it helps to understand the formula for binomial coefficients.

When we are choosing k things from a group of n, this formula tells us how many unique combinations of k things there are.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

Simple examples:

$$\binom{3}{2} =$$
$$\binom{4}{2} =$$

Aside: n choose k

To aid in discussing the process it helps to understand the formula for binomial coefficients.

When we are choosing k things from a group of n, this formula tells us how many unique combinations of k things there are.

$$\binom{n}{k} = \frac{n!}{k!(n-k)!} \tag{1}$$

Simple examples:

$$\binom{3}{2} = \frac{3!}{2!(1!)} = 3, \text{ where the set is } (1,2),(1,3),(2,3)$$
$$\binom{4}{2} = \frac{4!}{2!(2!)} = 6, \text{ where the set is } (1,2),(1,3),(1,4),(2,3),(2,4),(3,4)$$

How many models?

Say you have p features and want to construct as many unique linear models as you can with just k features.

How many models are there?

How many models?

Say you have p features and want to construct as many unique linear models as you can with just k features.

How many models are there?

$$\binom{p}{k} = \frac{p!}{k!(p-k)!}$$

"Best" subset selection

1. Find $\mathcal{M}_0 =$ model with $p = 0$ (just a constant term...will be equal to the mean)

2. Define $\mathcal{M}_k =$ the best model with k predictors, chosen from among the $\binom{p}{k}$ models with k predictors.

- For each k, choose "best" via something like R^2.
- Find $\mathcal{M}_k$ for all k, using all the training data.

3. Then use a model selection method like k-fold cross validation (or other adjusted error approaches like AIC) to choose from among $\mathcal{M}_0, \cdots \mathcal{M}_p$.

"Best" subset selection

1. Find $\mathcal{M}_0 =$ model with $p = 0$ (just a constant term...will be equal to the mean)

2. Define $\mathcal{M}_k =$ the best model with k predictors, chosen from among the $\binom{p}{k}$ models with k predictors.
   - For each k, choose "best" via something like R^2.
   - Find $\mathcal{M}_k$ for all k, using all the training data.

3. Then use a model selection method like k-fold cross validation (or other adjusted error approaches like AIC) to choose from among $\mathcal{M}_0, \cdots \mathcal{M}_p$.

Question: Why could we just use R^2 with all the training data for step two, but not step three?

When to cross-validate or use an adjusted error?

Question: Why could we just use R^2 with all the training data for step two, but not step three?

When to cross-validate or use an adjusted error?

Question: Why could we just use R^2 with all the training data for step two, but not step three?

Answer: Error adjustment methods and cross validation are needed to avoid overfit – i.e. to keep you from using too many parameters.

In this case, overfit happens when you use too many features.

If you're choosing from among a set of models that all have the same number of features (step two), you don't need to adjust.

Is the perfect the enemy of the good?

The "best" subset selection procedure is effective – it guarantees you'll find *the best* model.

But it's not always practical to implement. Why?

Is the perfect the enemy of the good?

The "best" subset selection procedure is effective – it guarantees you'll find *the best* model.

But it's not always practical to implement. Why?

First, suppose $p = 40$ and we're looking at models with $k = 10$ features. Then there are

Is the perfect the enemy of the good?

The "best" subset selection procedure is effective – it guarantees you'll find *the best* model.

But it's not always practical to implement. Why?

First, suppose $p = 40$ and we're looking at models with $k = 10$ features. Then there are

$$\binom{p}{k} = \frac{p!}{k!(p-k)!} = \frac{40!}{10!30!} = 8.5 \times 10^9 \tag{2}$$

possible models with 10 features.

That's a lot to evaluate on a single level.

Perfect v. good, ctd...

But it gets worse! Best subset evaluates models with *every* possible number of features.

Imagine a binary number where each place represents whether or not a feature is included in the model.

For example 00101 would be a model for which only feature 3 and 5 are included from among 5 possible features.

How many *possible* models are there if $p = 5$?

If $p = 40$?, this number explodes to

Perfect v. good, ctd...

But it gets worse! Best subset evaluates models with *every* possible number of features.

Imagine a binary number where each place represents whether or not a feature is included in the model.

For example 00101 would be a model for which only feature 3 and 5 are included from among 5 possible features.

How many *possible* models are there if $p = 5$?

$$2^5 = 32$$

If $p = 40$?, this number explodes to

$$2^{40} \approx 10^{12}$$

Perfect v. good, conclusion

Why is best subset selection impractical?

- Computing time! You'd need to test 1 trillion models if you had $p = 40$.

The alternatives: "hueristic" algorithms that give you a strategy to search the space of models but don't guarantee that you'll get the best model.

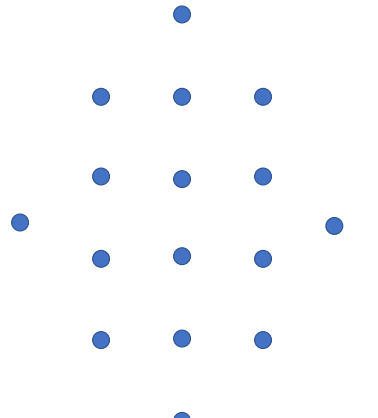
Stepwise selection (forward selection and backward selection) can help here.

Stepwise selection: Forward Selection

Forward selection: Start with $\mathcal{M}_0$. Then to choose the best model from each higher "level":

First, within each level, add one predictor at a time to the best model from the lower level. Use R^2 or other to find the best model from this set of $\mathcal{M}_{k-1}$ "$+1$"

Second, choose from your list $\{\mathcal{M}_0, \mathcal{M}_1, ...\mathcal{M}_p\}$ via cross validation or adjusted error metrics.

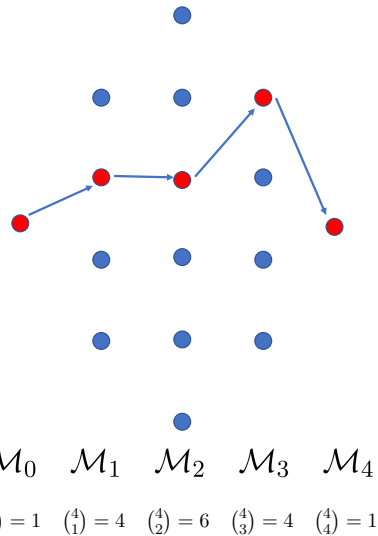


Stepwise selection: Forward Selection

Forward selection: Start with $\mathcal{M}_0$. Then to choose the best model from each higher "level":

First, within each level, add one predictor at a time to the best model from the lower level. Use R^2 or other to find the best model from this set of $\mathcal{M}_{k-1}$ "$+1$"

Second, choose from your list $\{\mathcal{M}_0, \mathcal{M}_1, ...\mathcal{M}_p\}$ via cross validation or adjusted error metrics.



Why is Forward Selection better than Best Subset Selection?

Why is Forward Selection better than Best Subset Selection?

...because we build on the best model from the lower level ($\mathcal{M}_{k-1}$)

Suppose $p = 10$.

- How many models will we evaluate on level 1?

Why is Forward Selection better than Best Subset Selection?

...because we build on the best model from the lower level ($\mathcal{M}_{k-1}$)

Suppose $p = 10$.

- How many models will we evaluate on level 1? 10.

Why is Forward Selection better than Best Subset Selection?

...because we build on the best model from the lower level ($\mathcal{M}_{k-1}$)

Suppose $p = 10$.

- How many models will we evaluate on level 1? 10.
- How many models will we evaluate on level 2?

Why is Forward Selection better than Best Subset Selection?

...because we build on the best model from the lower level ($\mathcal{M}_{k-1}$)

Suppose $p = 10$.

- How many models will we evaluate on level 1? 10.
- How many models will we evaluate on level 2? 9.

...For each level we are only choosing from $p + 1 - k$ possible models.

So, for $p = 10$:

Why is Forward Selection better than Best Subset Selection?

...because we build on the best model from the lower level ($\mathcal{M}_{k-1}$)

Suppose $p = 10$.

- How many models will we evaluate on level 1? 10.
- How many models will we evaluate on level 2? 9.

...For each level we are only choosing from $p + 1 - k$ possible models.

So, for $p = 10$:

- Best subset: evaluate $2^p = 1024$ models
- Forward: evaluate $1 + 10 + 9 + 8 + 7 + ... + 1 = 56$ models

Stepwise selection: Backward Selection

Stepwise selection: Backward Selection

Backward selection: like forward selection, except that you start with $\mathcal{M}_p$ (the model with all predictors) and *remove* one predictor at a time.

Pop quiz

Challenge question: which stepwise selection approach works in situations where $n < p$?

Pop quiz

Challenge question: which stepwise selection approach works in situations where $n < p$?

Answer: Forward selection

Pop quiz

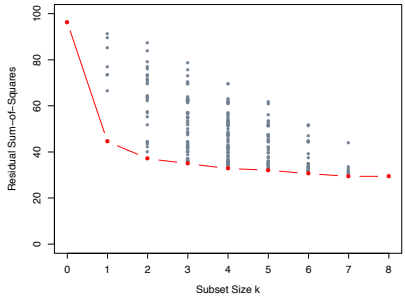
Challenge question: which stepwise selection approach works in situations where $n < p$?

Answer: Forward selection

...because it naturally stops at models at level $k = n - 1$.

OLS can't solve models with $k > n$ features.

Tracing the best model



From Elements of Statistical Learning

Dots are performance for each possible model

Red line is the performance for the best model in each subset size group.

Stepwise selection would either ride on the red line or above.

Shrinkage methods (regularization)

Let's begin by remembering that the processes of choosing parameters *and* choosing models are rooted in ideas of optimization.

When we run OLS, we are solving a formal optimization problem

$$Y = \beta_0 + \beta_1 X_1 + \cdots + \beta_p X_p + \epsilon \qquad Y_i = \sum_{k=1}^{K} X_{ik} \cdot \beta_k + \epsilon_i$$

$$\Rightarrow \hat{\beta}_{\text{ols}} = \arg \min_{\beta} \sum_{i=1}^{N} \left(Y_i - X_i \beta\right)^2$$

To solve, set the derivatives wrt βs to zero and

$$\hat{\beta}_{\text{ols}} = \left(\mathbf{X}^T \mathbf{X}\right)^{-1} \left(\mathbf{X}^T \mathbf{Y}\right)$$

Subset selection as optimization

In subset selection, you can show that you're running this optimization model:

Here, $\|\cdot\|_0$ is the "zero norm," where $I(\cdot)$ is the 'indicator function':

Subset selection as optimization

In subset selection, you can show that you're running this optimization model:

$$\hat{\beta}_{SS} = \arg\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_0 \text{ , where } \|\beta\|_0 = \sum_{k=1}^{K} I(\beta_k)$$

Here, $\|\cdot\|_0$ is the "zero norm," where $I(\cdot)$ is the 'indicator function':

$$I(\beta) = \begin{cases} 1 & \text{for } \beta \neq 0 \\ 0 & \text{for } \beta = 0 \end{cases}$$

Note that in this framing, λ becomes a new parameter to choose. High λ favors models with fewer features and vice versa.

We can use cross-validation to choose λ. Stay tuned.

Subset selection as optimization, ctd

For those of you with experience with integer programming

- The formulation on the last slide is a mixed integer quadratic program
- There are solvers for this type of problem, so you don't have to take the heuristic approaches (forward selection, etc) if you don't want to.
- That will likely give better performance than forward or backward subset selection, but it will still be slower.

Ridge regression

In ridge regression, we use a simple modification to the subset selection objective:

Ridge regression

In ridge regression, we use a simple modification to the subset selection objective:

$$\begin{aligned}\hat{\beta}_{\text{ridge}} &= \arg\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \sum_{k=1}^{K} \beta_k^2 \\ &= \arg\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot ||\beta||_2^2\end{aligned}$$

This leads to:

$$\hat{\beta}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_k\right)^{-1} \left(\mathbf{X}^T\mathbf{Y}\right)$$

Here

- λ is a tuning parameter – it is not unique.
- $\mathbf{I}_k$ is the $k \times k$ identity matrix

Ridge regression advantages

Ridge regression advantages

First. Suppose some of your features are linear combinations of the others. That means you can write $x_{i,j} = Ax_{i,-j}$ for at least 1 value of j.

Then $\mathbf{X}^T\mathbf{X}$ is not "full rank" and you can't invert it. I.e., $(\mathbf{X}^T\mathbf{X})^{-1}$ doesn't exist.

e.g., $\mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix}$

Ridge regression advantages

First. Suppose some of your features are linear combinations of the others. That means you can write $x_{i,j} = Ax_{i,-j}$ for at least 1 value of j.

Then $\mathbf{X}^T\mathbf{X}$ is not "full rank" and you can't invert it. I.e., $(\mathbf{X}^T\mathbf{X})^{-1}$ doesn't exist.

$$\text{e.g., } \mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} \quad \text{vs.} \quad \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I} = \begin{bmatrix} 1+\lambda & 2 & 3 \\ 2 & 4+\lambda & 6 \\ 3 & 6 & 9+\lambda \end{bmatrix}$$

But you *can* invert $\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_k\right)$. (Since you've added a little shift to the diagonals of the matrix, which restores linear independence)

Ridge regression advantages

First. Suppose some of your features are linear combinations of the others. That means you can write $x_{i,j} = Ax_{i,-j}$ for at least 1 value of j.

Then $\mathbf{X}^T\mathbf{X}$ is not "full rank" and you can't invert it. I.e., $(\mathbf{X}^T\mathbf{X})^{-1}$ doesn't exist.

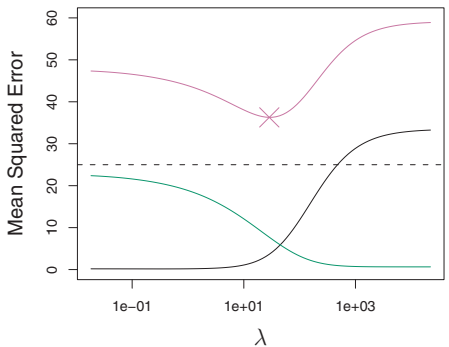
$$\text{e.g., } \mathbf{X}^T\mathbf{X} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 4 & 6 \\ 3 & 6 & 9 \end{bmatrix} \quad \text{vs.} \quad \mathbf{X}^T\mathbf{X} + \lambda\mathbf{I} = \begin{bmatrix} 1+\lambda & 2 & 3 \\ 2 & 4+\lambda & 6 \\ 3 & 6 & 9+\lambda \end{bmatrix}$$

But you *can* invert $\left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_k\right)$. (Since you've added a little shift to the diagonals of the matrix, which restores linear independence)

Second. Computation! It's faster than subset selection (but solves a different problem if your objective is parameter interpretation).

Ridge regression advantages, ctd

Third. Bias-variance tradeoff! Figure from ISLR



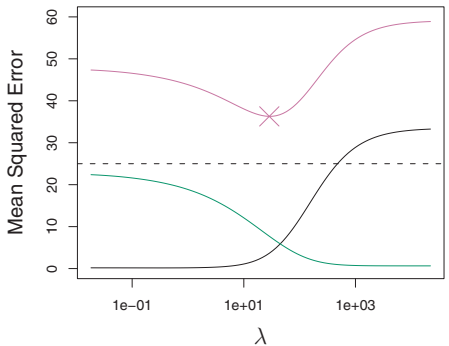
green: variance
black: bias
red: total error

Tuning λ ...

- squashes down parameters associated with variables that increase model variance.
- but doesn't set parameters to zero, as subset selection does.

Ridge regression advantages, ctd

Third. Bias-variance tradeoff! Figure from ISLR



green: variance
black: bias
red: total error

Tuning λ ...

- squashes down parameters associated with variables that increase model variance.
- but doesn't set parameters to zero, as subset selection does.

...but how can we choose λ? We'll discuss next time.

Ridge regression disadvantage

$$\hat{\beta}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_k\right)^{-1}\left(\mathbf{X}^T\mathbf{Y}\right)$$

Ridge regression disadvantage

$$\hat{\beta}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_k\right)^{-1}\left(\mathbf{X}^T\mathbf{Y}\right)$$

Ridge regression produces different β estimates for different choices of λ.

Why is that important?

Ridge regression disadvantage

$$\hat{\beta}_{\text{ridge}} = \left(\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I}_k\right)^{-1}\left(\mathbf{X}^T\mathbf{Y}\right)$$

Ridge regression produces different β estimates for different choices of λ.

Why is that important?

It means interpretation of the *size* of the coefficients (i.e. inference) is problematic.

Side note: "Regularization"

Regularization Refers to the process of adding a term to the objective function of a problem that

- Makes the problem "well behaved" (easier to solve)
- Solves a different problem from the one you originally wanted.

In our case, the sum of squared coefficients in Ridge makes the problem very simple to solve, but we get coefficients that are biased.

In forecasting, the reduction in output variance is more important than the growth in coefficient bias.

But it means we need to throw inference out the window.

The Lasso

Least Absolute Selection and Shrinkage Operator (Tibshirani, 1996)

Lasso is a simple modification of Ridge:

The Lasso

Least Absolute Selection and Shrinkage Operator (Tibshirani, 1996)

Lasso is a simple modification of Ridge:

$$\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_1 \text{ , where } \|\beta\|_1 = \sum_{k=1}^{K} |\beta_k|$$

Note, we won't find a closed-form expression for the coefficients (unlike ridge and OLS). Why?

The Lasso

Least Absolute Selection and Shrinkage Operator (Tibshirani, 1996)

Lasso is a simple modification of Ridge:

$$\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_1 \text{ , where } \|\beta\|_1 = \sum_{k=1}^{K} |\beta_k|$$

Note, we won't find a closed-form expression for the coefficients (unlike ridge and OLS). Why? ...Because the ℓ_1 norm is not differentiable everywhere.

But you can still use your computer to find the $\hat{\beta}_{\text{lasso}}$ in a much more efficient way than subset selection.

Lasso tradeoffs vs OLS, subset selection and ridge regression

Lasso tradeoffs vs OLS, subset selection and ridge regression

- $+$ Lasso is faster to solve than subset selection
- $-$ Lasso is slower to solve than ridge (no analytic solution)
- $+$ Lasso drives parameters to zero, like subset selection and unlike ridge
  - $\Rightarrow$ interpretability
  - But if there really are a lot of modest sized β, this is undesirable!
- $-$ Lasso yields biased parameters (unlike OLS / subset selection)
- $\pm$ Lasso similar to ridge's prediction bias-variance properties
  - $+$ less prediction variance than OLS, especially with many predictors
  - $-$ more prediction bias than OLS
- $-$ With highly correlated predictors, Lasso is unstable: indifferent between
  - $\hat{\beta}_1 = 0$ and $\hat{\beta}_2 = \beta_1 + \beta_2$
  - $\hat{\beta}_1 = \beta_1 + \beta_2$ and $\hat{\beta}_2 = 0$

A bit of intuition about what Lasso and ridge are doing

(We will go into this in much more detail next time)

Each line is a "contour" for either the RSS or the regularizing penalties

Solutions happen when the contours are tangent to each other

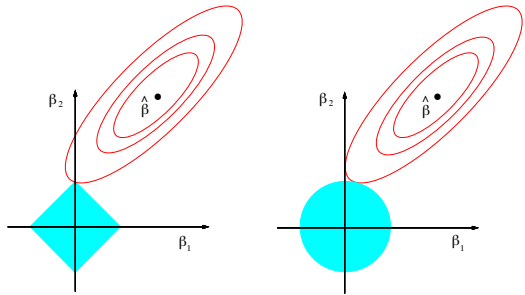


Figure taken from Imbens 2015 NBER lecture

For lasso, you can see this is much more likely to happen at a corner (one parameter zero) than it is for ridge

Supplemental slides

C_p, AIC, BIC or "Adjusted R^2"

Metric	Formula
C_p	$\frac{1}{n}\left(RSS + 2d\hat{\sigma}^2\right)$
AIC	$\frac{1}{n\hat{\sigma}^2}\left(RSS + 2d\hat{\sigma}^2\right)$
BIC	$\frac{1}{n\hat{\sigma}^2}\left(RSS + \log(n)d\hat{\sigma}^2\right)$
Adjusted R^2	$1 - \frac{RSS}{TSS}\frac{(n-1)}{(n-d-1)}$

...where d is the number of predictors and $\hat{\sigma}^2$ is your best estimate of the variance of the model error

Note that Adjusted R^2 is related to the F-statistic but not the same, and that we don't *always* have $0 <$ Adjusted $R^2 < 1$

Which error adjustment to use?

The answer is somewhat a matter of personal taste.

- C_p and AIC are effectively the same.
- BIC penalizes added features more than C_p and AIC.
- C_p, AIC and BIC have rigorous derivations.
- R^2 does not have a derivation but it has an intuitive interpretation.

My two cents:

1. Look at both AIC and BIC and make a judgment call if they differ
2. Use Adjusted R^2 if you want readers to understand the numeric value
3. But if you can afford the computing time, use cross validation instead.