Model Selection & Regularization

Lecture 7

Termeh Shafie

1

Recall: Linear Models and Least Squares

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon \qquad \text{RSS} = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j-1}^{p} \hat{\beta}_{j_1} x_{ij})^2$$

Model with all available predictor variables is commonly referred to as **the full model**

Issues:

- predictive accuracy
- model interpretability

Solutions:

- select subset of predictors
- consider extension to the least squares solution of full model

2

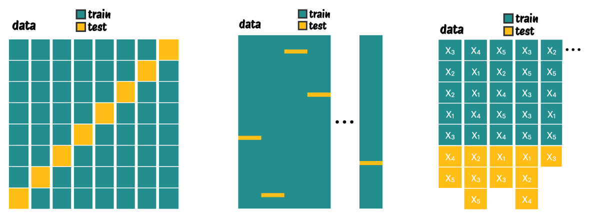
Part I - Variable Subset Selection

3

Model Selection Criteria: Validation by Prediction Error

Last week: how to use cross validation to choose a set of predictors
by directly estimate prediction error using cross-validation techniques

$$\text{e.g. MSE} = \frac{\text{RSS}}{n} \qquad \text{RMSE} = \sqrt{\frac{\text{RSS}}{n}} \qquad R^2 = 1 - \frac{\text{RSS}}{\text{TSS}}$$



Now: indirectly estimating test performance using an approximation

4

Model Selection Criteria

Four ways to estimate test performance using an approximation

Full model has p predictors

RSS is the residual sum of squares for model with d predictors

$\hat{\sigma}^2 = \text{RSS}_p/(n-p-1)$ is an estimate of the error variance for full model

1. Mallow's C_p criterion:

For a given model with d (out of the p available) predictors

$$C_p = \frac{1}{n}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

we are penalizing models of higher dimensionality (larger d, greater penalty)

$\Longrightarrow$ choose the model which has **minimum** C_p

Model Selection Criteria

Four ways to estimate test performance using an approximation

Full model has p predictors

RSS is the residual sum of squares for model with d predictors

$\hat{\sigma}^2 = \text{RSS}_p/(n-p-1)$ is an estimate of the error variance for full model

2. Akaike Information Criterion (AIC)

For linear models: equivalent to Mallow's C_p (proportional to)

$$AIC = \frac{1}{n\hat{\sigma}^2}\left(\text{RSS} + 2d\hat{\sigma}^2\right)$$

we are penalizing models of higher dimensionality (larger d, greater penalty)

$\Longrightarrow$ choose the model which has **minimum** AIC

Model Selection Criteria

Four ways to estimate test performance using an approximation

Full model has p predictors

RSS is the residual sum of squares for model with d predictors

$\hat{\sigma}^2 = \text{RSS}_p/(n-p-1)$ is an estimate of the error variance for full model

3. Bayesian Information Criterion (BIC)

$$BIC = \frac{1}{n\hat{\sigma}^2}\left(\text{RSS} + \underbrace{\log(n)}_{\text{heavier penalty}}d\hat{\sigma}^2\right)$$

we are penalizing models of higher dimensionality (larger d, greater penalty)

$\Longrightarrow$ choose the model which has **minimum** BIC

Model Selection Criteria

Four ways to estimate test performance using an approximation

Full model has p predictors

RSS is the residual sum of squares for model with d predictors

$\hat{\sigma}^2 = \text{RSS}_p/(n-p-1)$ is an estimate of the error variance for full model

4. Adjusted R-squared value

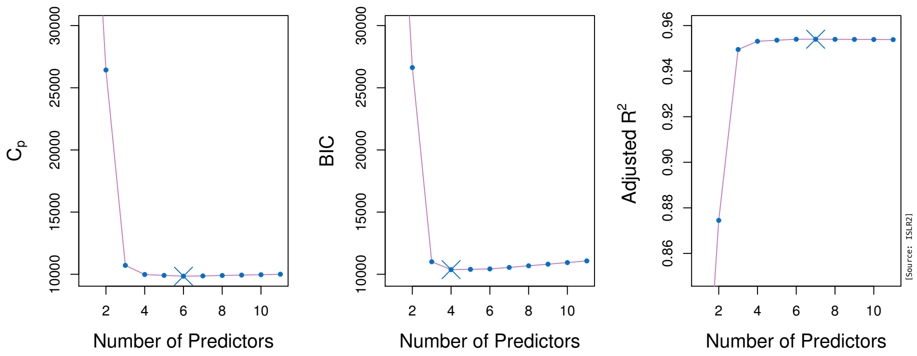
Adjust the regular R^2 by taking into account number of predictors

$$\text{Adjusted-}R^2 = 1 - \frac{\text{RSS}/(n-d-1)}{\text{TSS}/(n-1)}$$

$\Longrightarrow$ choose the model which has **maximum** Adjusted-R^2

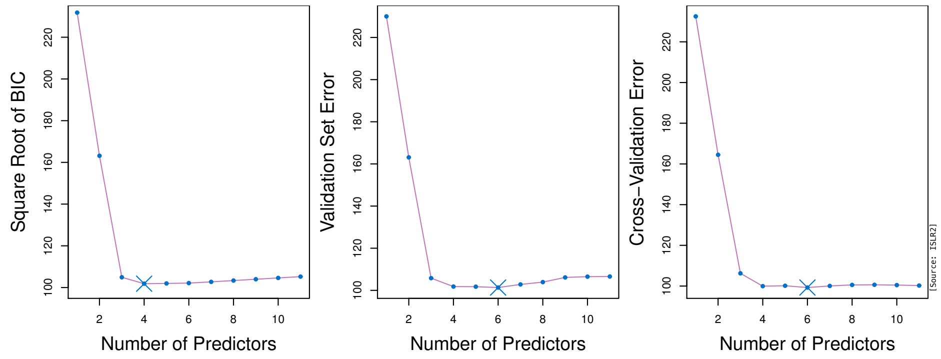
Model Selection Criteria

Four ways to estimate test performance using an approximation



Model Selection Criteria

...and compared to cross validation



Model Search Methods

Best Subset Selection

1. Let M_0 denote null model which contains no predictors. This model simply predicts the response for each observation.
2. For $k = 1,2,\ldots,p$
   - Fit all $\binom{p}{k}$ models that contain exactly p predictors
   - Pick the best among these $\binom{p}{k}$ models and call it M_k.

   Here, *best* is defined as having the smallest RSS or largest R^2
3. Select a single best model from among $M_0, M_1, \ldots, M_p$ using cross validated prediction error, C_p (AIC), BIC, or Adjusted-R^2

requires training 2^p models

Example

$p = 3$

M_0 : intercept only (null)

C_1: X_1 X_2 X_3

lowest training RSS within C_1 $\Longrightarrow M_1$

C_2: X_1, X_2 X_1, X_3 X_2, X_3

lowest training RSS within C_2 $\Longrightarrow M_2$

M_3 : full model with X_1 X_2 X_3

Model Search Methods

Forward Stepwise Selection

1. Let M_0 denote null model which contains no predictors.
2. For $k = 1,2,\ldots,p-1$
   - Consider all $p - k$ models that augment the predictors in M_k with one additional predictor
   - Choose the best among these $p - k$ models and call it M_{k+1}.

   Here, *best* is defined as having the smallest RSS or largest R^2
3. Select a single best model from among $M_0, M_1, \ldots, M_p$ using cross validated prediction error, C_p (AIC), BIC, or Adjusted-R^2

requires training $1 + \dfrac{p(p+1)}{2}$ models

Example

$p = 3$

M_0 : intercept only (null)

C_1: X_1 X_2 X_3

lowest training RSS within C_1 $\Longrightarrow M_1$

C_2: X_1, X_2 X_2, X_3

lowest training RSS within C_2 $\Longrightarrow M_2$

M_3 : full model with X_1 X_2 X_3

Model Search Methods

Backward Stepwise Selection

1. Let M_p denote full model which all predictors.
2. For $k = p, p-1, p-2, \ldots, 1$
   - Consider all k models that contain all but one of the predictors in M_k, for a total of $k-1$ predictors
   - Choose the best among these k models and call it M_{k-1}. Here, *best* is defined as having the smallest RSS or largest R^2
3. Select a single best model from among $M_0, M_1, \ldots, M_p$ using cross validated prediction error, C_p (AIC), BIC, or Adjusted-R^2

requires training $1 + \dfrac{p(p+1)}{2}$ models

Example

$p = 3$

M_3 : full model X_1 X_2 X_3

C_2: X_1, X_2 X_1, X_3 X_2, X_3

$\downarrow$

lowest training RSS within C_1

$\Longrightarrow M_2$

C_1: X_1 X_2

$\downarrow$

lowest training RSS within C_2

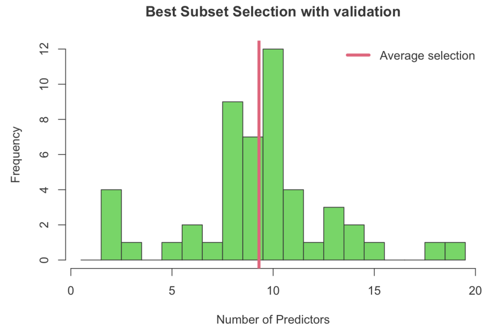
$\Longrightarrow M_1$

M_0 : intercept only (null)

Model Search Methods

Best Subset Selection

validation approach based on 50 different seeds and storing number of predictors in selected model each time



[plot is made based on the 'hitters' data se used in ISLR2]

Part II - Shrinkage

Shrinkage Methods

Before: **Discrete model search methods**

$$\underbrace{\text{model fit}}_{\text{RSS}} + \text{penalty on model dimensionality}$$

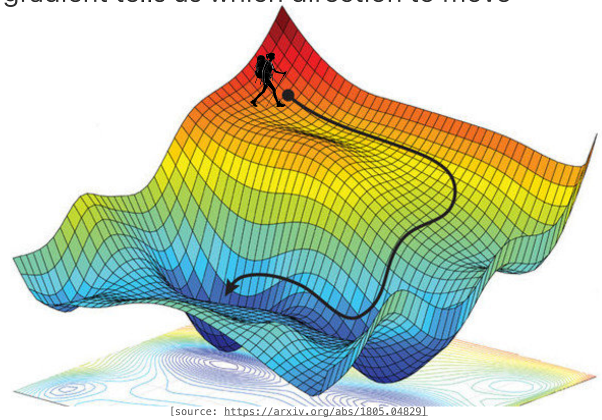
Now: **Continuous model search method** (also faster)

$$\underbrace{\text{model fit}}_{\text{RSS}} + \text{penalty on size of coefficients}$$

this is called **penalized** or **regularized regression**

Review: Gradient Descent

- The goal is to minimize the loss function
- The gradient tells us which direction to move



[source: https://arxiv.org/abs/1805.04829]

17

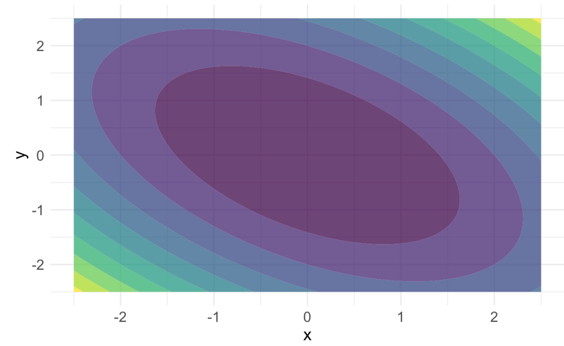
Gradient Descent

$f(x, y) = x^2 + xy + y^2$

Partial Derivatives:

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial x} \end{bmatrix} = \begin{bmatrix} 2x + 1 \\ x + 2y \end{bmatrix}$$



18

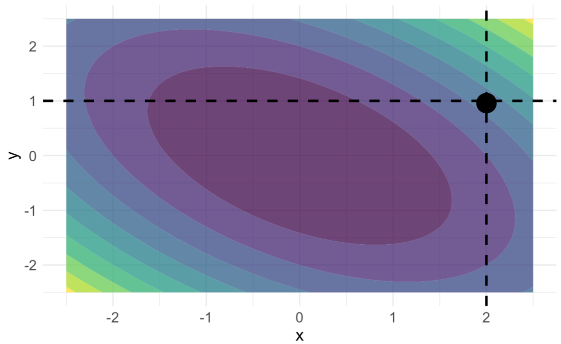
Gradient Descent

$f(x, y) = x^2 + xy + y^2$

Partial Derivatives:

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial x} \end{bmatrix} = \begin{bmatrix} 2x + 1 \\ x + 2y \end{bmatrix}$$



19

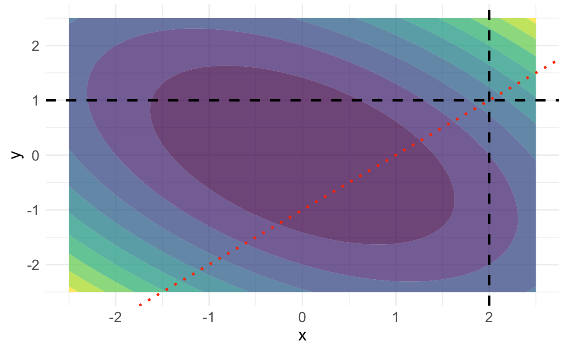
Gradient Descent

$f(x, y) = x^2 + xy + y^2$

Partial Derivatives:

$$\frac{\partial f}{\partial x} = 2x + y \qquad \frac{\partial f}{\partial y} = x + 2y$$

$$\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial x} \end{bmatrix} = \begin{bmatrix} 2x + 1 \\ x + 2y \end{bmatrix}$$



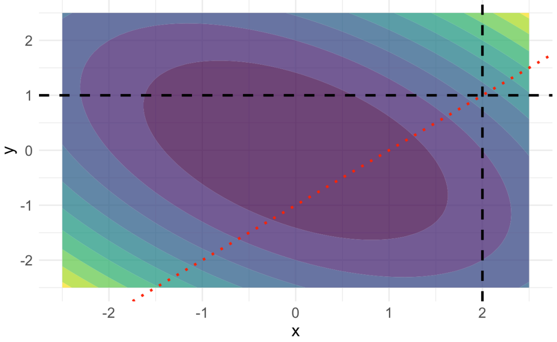
20

Gradient Descent

$f(x, y) = x^2 + xy + y^2$

Step size determined by Learning Rate ρ:

$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \rho \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

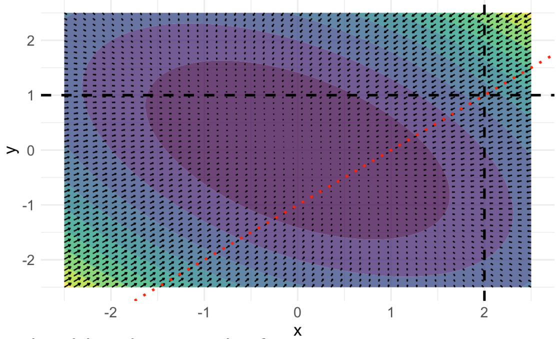


Gradient Descent

$f(x, y) = x^2 + xy + y^2$

Step size determined by Learning Rate ρ:

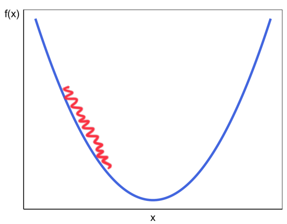
$$\begin{bmatrix} x_{new} \\ y_{new} \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} - \rho \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

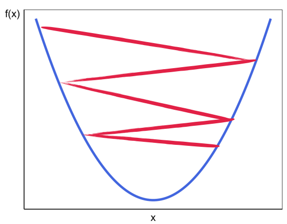


gradient tells us what adjustments we should make to each of our parameters

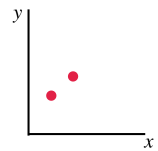
Estimation: Gradient Descent

- The goal is to minimize the loss function
- The gradient tells us which direction to move
- The **Learning Rate** controls how big of a step we take
  - Small steps mean slower convergence
  - Large steps mean we might step over minima





Example: Very Simple Linear Regression



$\hat{y} = b_0 + b_1 x$

Loss function:

$$RSS = \sum_{i}^{N} (\text{actual - predicted})^2 = \sum_{i}^{N} (y_i - \hat{y}_i)^2$$

$$= \sum_{i}^{N} (y_i - (b_0 + b_1 x))^2$$

$$= \sum_{i}^{N} (y_i - b_0 - b_1 x)^2$$

Assume only 2 data points: $(x_1, y_1) = (1,2), (x_2, y_2) = (2,3)$

Example: Very Simple Linear Regression

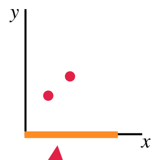
Gradient:

$$\begin{bmatrix} \frac{\partial RSS}{\partial b_0} \\ \frac{\partial RSS}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i(y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

Initialize the gradient algorithm at **(0,0)**

$$\implies \begin{bmatrix} -2\sum_i^N (y_i - (0 + 0x_i)) \\ -2\sum_i^N x_i(y_i - (0 + 0x_i)) \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i) \\ -2\sum_i^N x_i(y_i) \end{bmatrix}$$

plug in our actual values



Example: Very Simple Linear Regression

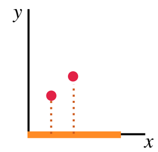
Gradient:

$$\begin{bmatrix} \frac{\partial RSS}{\partial b_0} \\ \frac{\partial RSS}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i(y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

Initialize the gradient algorithm at (0,0)

$$\implies \begin{bmatrix} -2\sum_i^N (y_i) \\ -2\sum_i^N x_i(y_i) \end{bmatrix} = \begin{bmatrix} -2(2+3) \\ -2(1 \cdot 2 + 2 \cdot 3 \end{bmatrix} = \begin{bmatrix} -10 \\ -16 \end{bmatrix}$$

These are the changes we need to make to intercept and slope in order to reduce our loss function.



Example: Very Simple Linear Regression

Gradient:

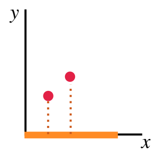
$$\begin{bmatrix} \frac{\partial RSS}{\partial b_0} \\ \frac{\partial RSS}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i(y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

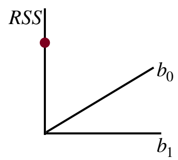
Initialize the gradient algorithm at (0,0)

$$\implies \begin{bmatrix} -2\sum_i^N (y_i) \\ -2\sum_i^N x_i(y_i) \end{bmatrix} = \begin{bmatrix} -2(2+3) \\ -2(1 \cdot 2 + 2 \cdot 3 \end{bmatrix} = \begin{bmatrix} -10 \\ -16 \end{bmatrix}$$

Compute loss function value:

$$RSS = \sum_i^N (y_i - b_0 - b_1 x_i)^2 = (2 - 0 - 0 \cdot 1)^2 + (3 - 0 - 0 \cdot 3)^2 = 13$$





Example: Very Simple Linear Regression

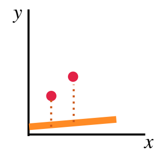
Gradient:

$$\begin{bmatrix} \frac{\partial RSS}{\partial b_0} \\ \frac{\partial RSS}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i(y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

Apply the changes (learning rate 0.01):

$$\begin{bmatrix} b_{0_{new}} \\ b_{1_{new}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} - 0.01 \begin{bmatrix} -10 \\ -16 \end{bmatrix}$$

$$\implies \begin{bmatrix} b_{0_{new}} \\ b_{1_{new}} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix} + 0.01 \begin{bmatrix} 10 \\ 16 \end{bmatrix} = \begin{bmatrix} 0.10 \\ 0.16 \end{bmatrix}$$



Example: Very Simple Linear Regression

Gradient:

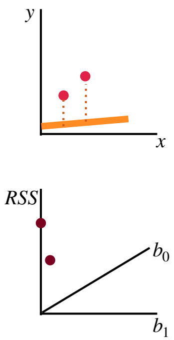
$$\begin{bmatrix} \frac{\partial RSS}{\partial b_0} \\ \frac{\partial RSS}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i(y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

Apply the changes:

$$\begin{bmatrix} b_{0_{new}} \\ b_{1_{new}} \end{bmatrix} = \begin{bmatrix} 0.10 \\ 0.16 \end{bmatrix}$$

Compute loss function value:

$$RSS = \sum_i^N (y_i - b_0 - b_1 x_i)^2 = (2 - 0.10 - 0.16 \cdot 1)^2 + (3 - 0.10 - 0.16 \cdot 3)^2 = 8.88$$



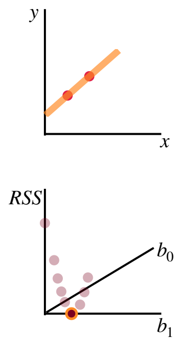
Example: Very Simple Linear Regression

Gradient:

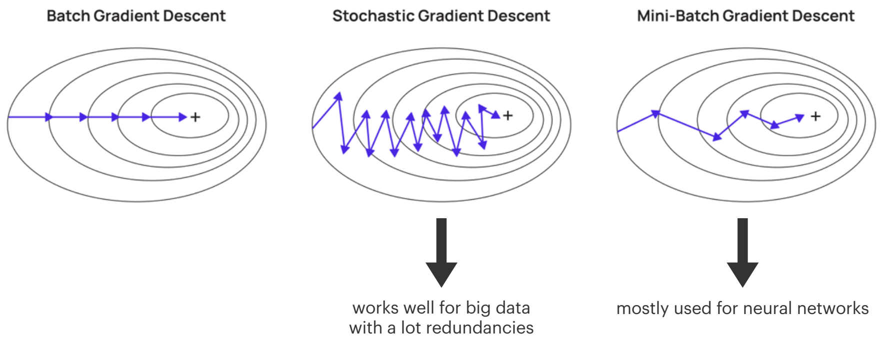
$$\begin{bmatrix} \frac{\partial RSS}{\partial b_0} \\ \frac{\partial RSS}{\partial b_1} \end{bmatrix} = \begin{bmatrix} -2\sum_i^N (y_i - (b_0 + b_1 x_i)) \\ -2\sum_i^N x_i(y_i - (b_0 + b_1 x_i)) \end{bmatrix}$$

Repeat until RSS is doesn't reduce significantly anymore in this toy example, it happens at

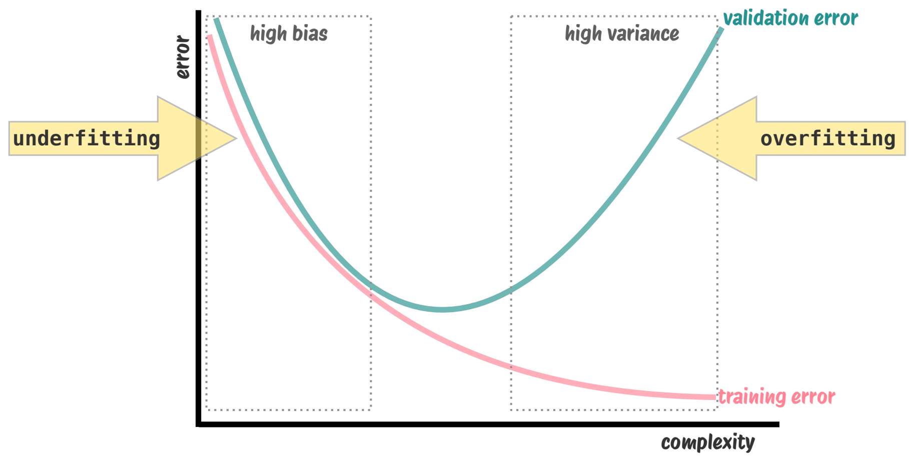
$$b_0 = 1,\ b_1 = 1 \implies RSS = 0$$



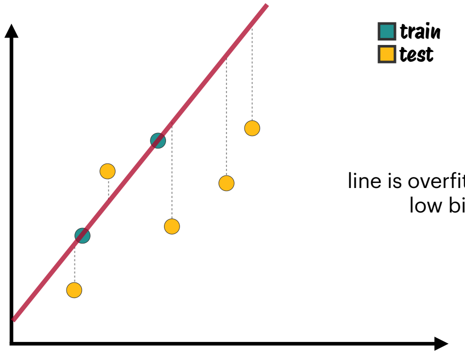
Versions of Gradient Descent



Bias Variance Trade-Off

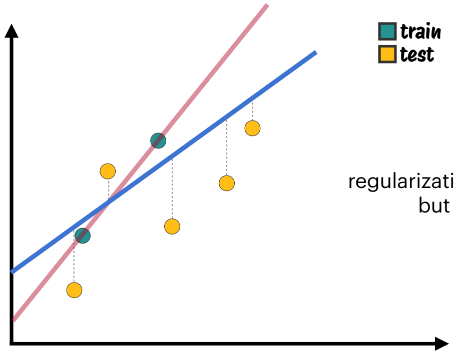


Bias Variance Trade-Off: Regularization



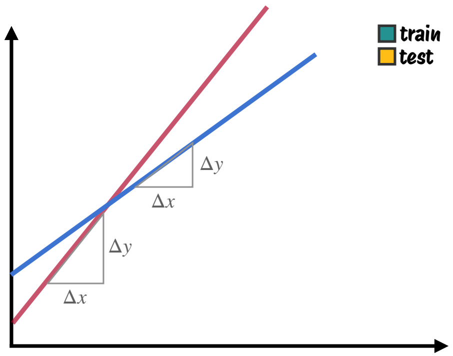
line is overfitted to the training data
low bias, high variance

Bias Variance Trade-Off: Regularization



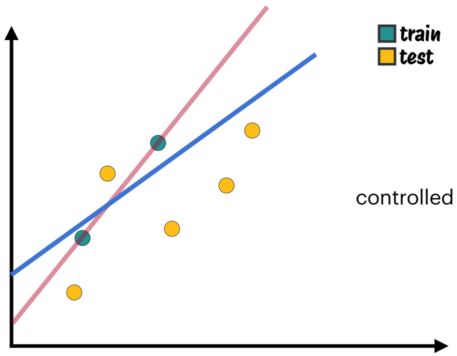
regularization introduces some bias
but reduces variance

Bias Variance Trade-Off: Regularization



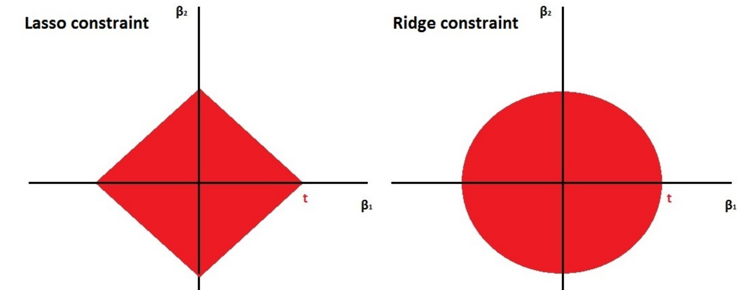
when the slope of the line is small
predictions for y are much less sensitive to changes in x

Bias Variance Trade-Off: Regularization



controlled by penalty parameter λ

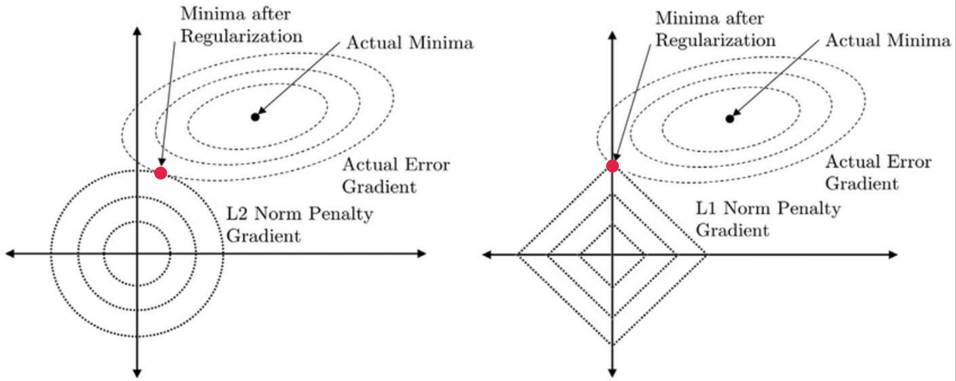
Ridge and Lasso Regression: The Constraints



$|\beta_1| + |\beta_2| \leq t$

$\beta_1^2 + \beta_2^2 \leq t$

Ridge and Lasso Regression: The Constraints



Ridge Regression

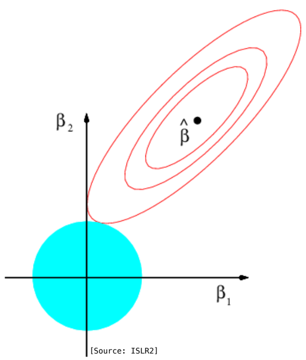
ridge uses ℓ_2 penalty

Least Squares produces estimates by minimizing

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_{j_1} x_{ij})^2$$

Ridge regression instead minimizes

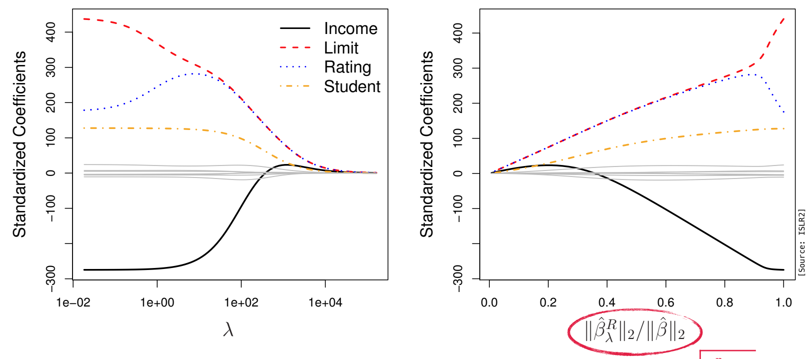
$$\underbrace{\sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j=1}^{p} \hat{\beta}_{j_1} x_{ij})^2}_{\text{model fit}} + \lambda \underbrace{\sum_{j=1}^{p} \beta_j^2}_{\text{penalty}} = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$



where $\lambda \geq 0$ is the tuning parameter controlling trade off between model fit and size of coefficients ($\lambda \to \infty, \hat{\beta}_j \to 0$)

Ridge Regression

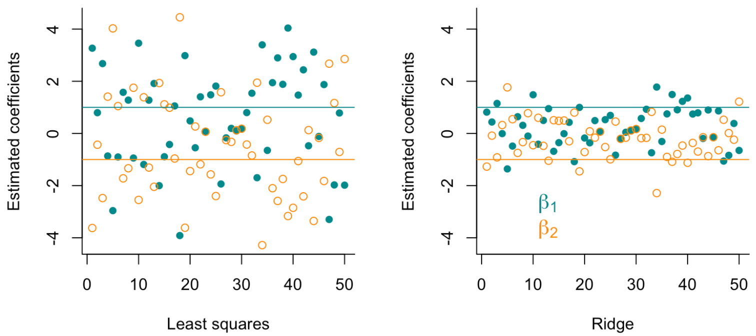
Regularization Paths



ℓ_2 norm $= \|\beta\|_2 = \sqrt{\sum_{j=1}^{p} \beta_j^2}$

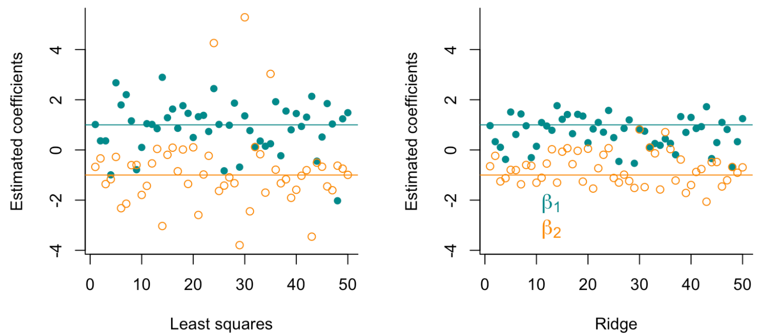
Ridge Regression

Advantage 1: Multicollinearity (a simulation study)



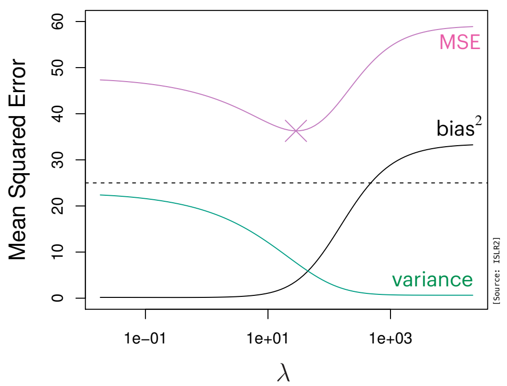
Ridge Regression

Advantage 2: When p is close to n (a simulation study)



Ridge Regression

Bias-Variance Trade Off



Lasso Regression

Least Absolute Shrinkage and Selection Operator

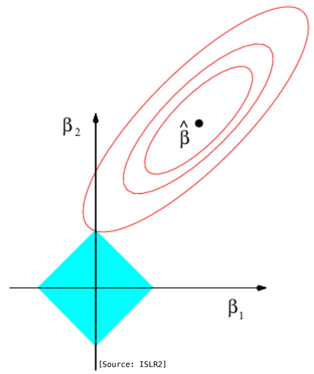
lasso uses ℓ_1 penalty

Least Squares produces estimates by minimizing

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j-1}^{p} \hat{\beta}_{j_1} x_{ij})^2$$

Lasso regression instead minimizes

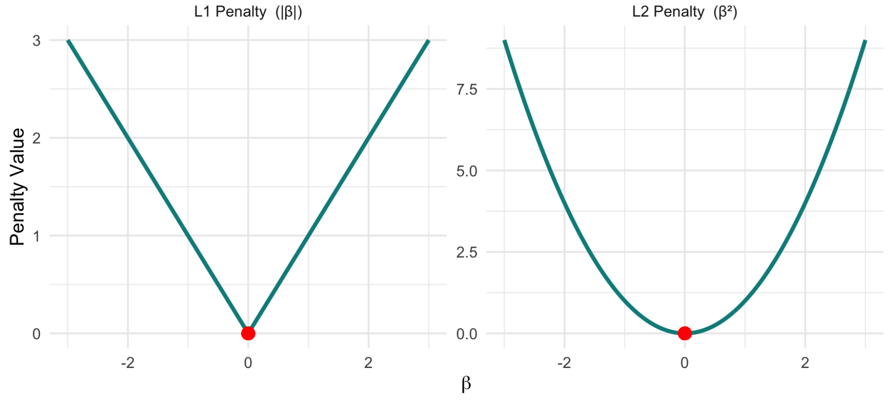
$$\underbrace{\sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \sum_{j-1}^{p} \hat{\beta}_{j_1} x_{ij})^2}_{\text{model fit}} + \lambda \underbrace{\sum_{j=1}^{p} |\beta_j|}_{\text{penalty}} = \text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$



where $\lambda \geq 0$ is the tuning parameter controlling trade off between model fit and size of coefficients ($\lambda \to \infty$, $\hat{\beta}_j = 0$)

Lasso Regression

L1 vs L2 Penalty Functions



45

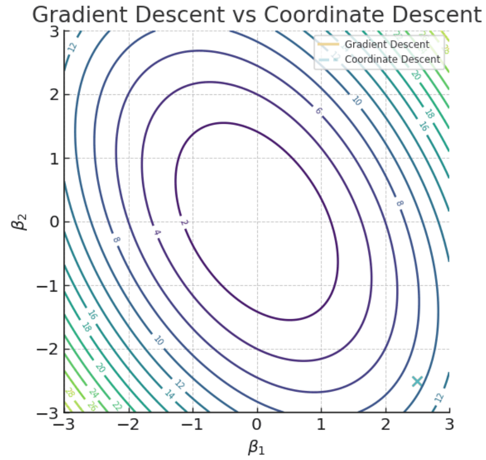
Lasso Regression

For Lasso, when updating coefficient β_j, coordinate descent solves:

$$\min_{\beta_j} \left[\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{k \neq j} \beta_k x_{ik} - \beta_j x_{ij} \right)^2 + \lambda |\beta_j| \right]$$

holding all other coefficients fixed.

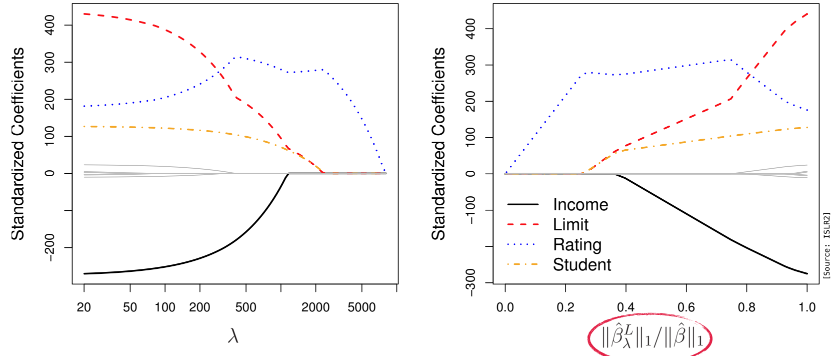
This reduces to a 1D optimization problem for each coordinate



46

Lasso Regression

Regularization Paths

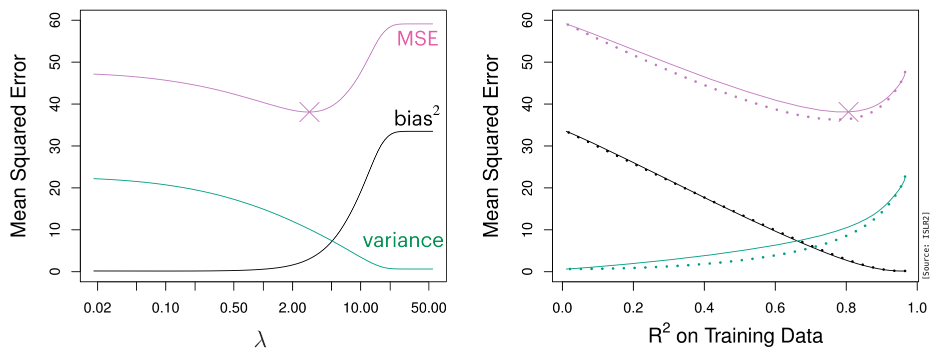


ℓ_1 norm $= \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$

47

Lasso Regression

Bias-Variance Trade Off



48

Ridge vs. Lasso Regression

- Both ridge and lasso are convex optimization
- The ridge solution exists in closed form
- Lasso does not have closed form solution, but very efficient optimization algorithms exist

When to choose which?

- When the actual data-generating mechanism is sparse lasso has the advantage
- When the actual data-generating mechanism is dense ridge has the advantage

Sparse mechanisms: Few predictors are relevant to the response → good setting for lasso regression
Dense mechanisms: A lot of predictors are relevant to the response → good setting for ridge regression

- Also depends on:
  - Signal strength (the magnitude of the effects of the relevant variables)
  - The correlation structure among predictors
  - Sample size n vs. number of predictors p

Ridge vs. Lasso Regression

Ridge

+ Reduces Multicollinearity
+ Continuous Shrinking
+ Stable Solutions
+ Computationally Efficient

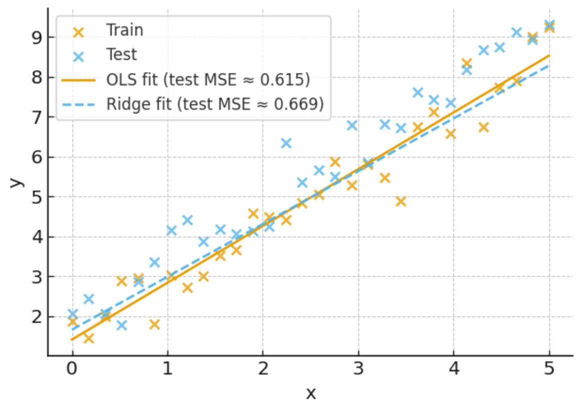
- No variable selection
- Interpretability
- Sensitive to scale

Lasso

+ Variable selection
+ Sparse models
+ Improves interpretability
+ Particularly useful for when $p > n$

- Collinearity issues
- Bias in coefficients (ℓ_1 penalty is harsher)
- Computationally intensive

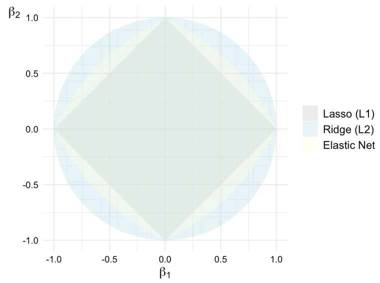
Note: Regularization under Dataset Shift



λ Tuning

- K-fold Cross Validation
  1. Choose the number of folds K
  2. Split the data accordingly into training and testing sets.
  3. Define a grid of values for λ
  4. For each λ, calculate the validation MSE within each fold
  5. For each λ, calculate the overall cross-validation MSE
  6. Locate under which λ cross-validation MSE is minimized, i.e. minimum_cv λ
- Packages such as will `glmnet` do this automatically

Hybrid Approach: Elastic Nets



$$\text{RSS} + \lambda_1 \underbrace{\sum_{j=1}^{p} \beta_j^2}_{\text{“ridge”}} + \lambda_2 \underbrace{\sum_{j=1}^{p} |\beta_j|}_{\text{“lasso”}}$$

λ_1 and λ_2 are regularization parameters controlling the strength of the penalties

- Helps stabilize the solution when predictors are correlated
- Shrinks some coefficients to zero, enabling feature selection
- Particularly useful for high-dimensional datasets with correlated predictors

Part III- Dimensionality Reduction

another strategy which aims to reduce dimensionality **before** applying LS
create q transformed variables which are linear combinations of the original predictors ($q < p$)
we return to this during our PCA lecture...

Part IV- Transformations: next week!

extensions to the regression model when the best straight line doesn't quite work!

This Week's Practical

Hands on discrete and continuous model search!