

STAT540

Lecture 19: March 16th 2016

Supervised Learning III: Feature Selection & Regularization

Sara Mostafavi

Department of Statistics

Department of Medical Genetics

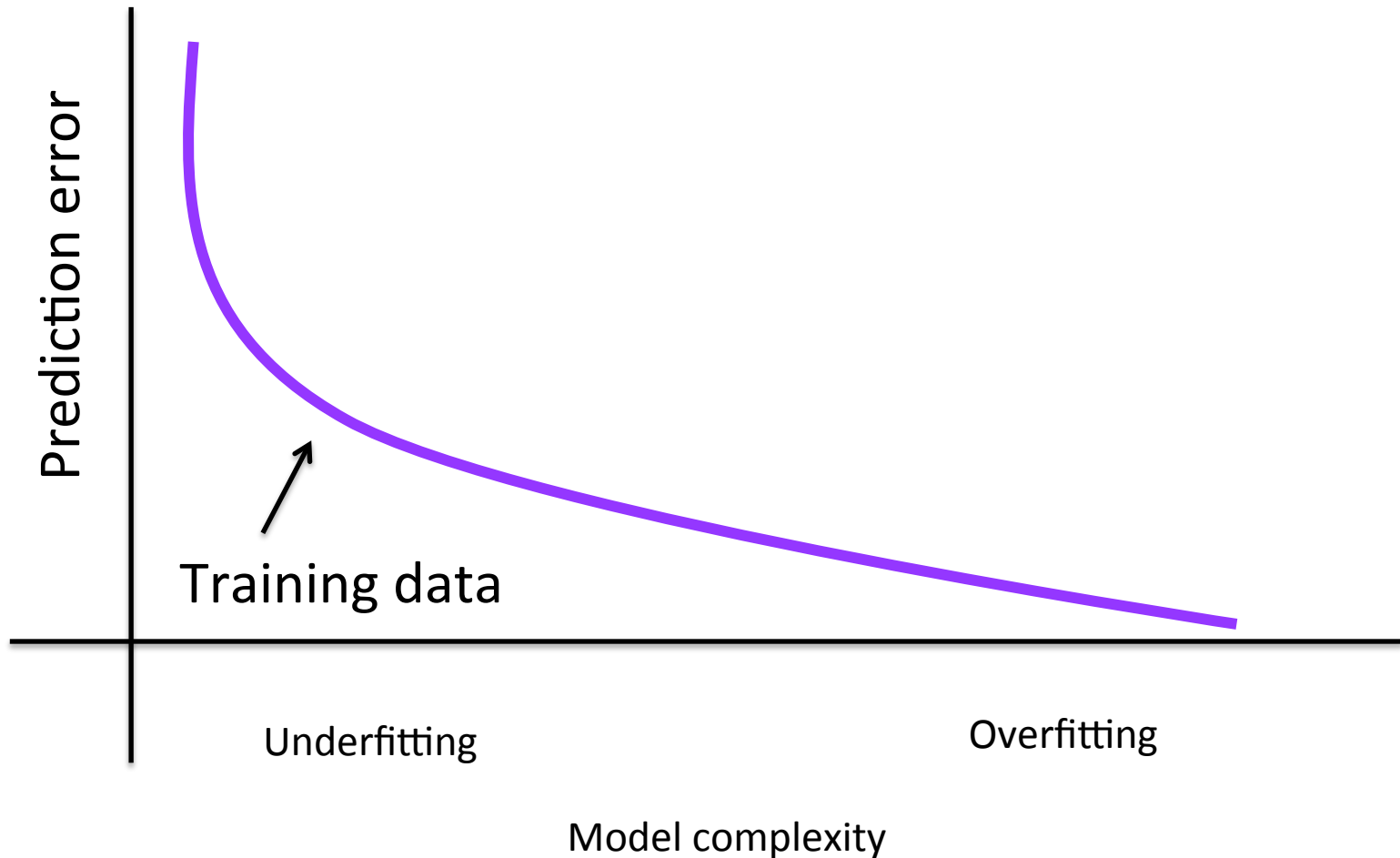
Center for Molecular Medicine and Therapeutics

**** Many thanks to Drs. Gabriela Cohen-Freue and Ryan Tibshirani for lecture slides****

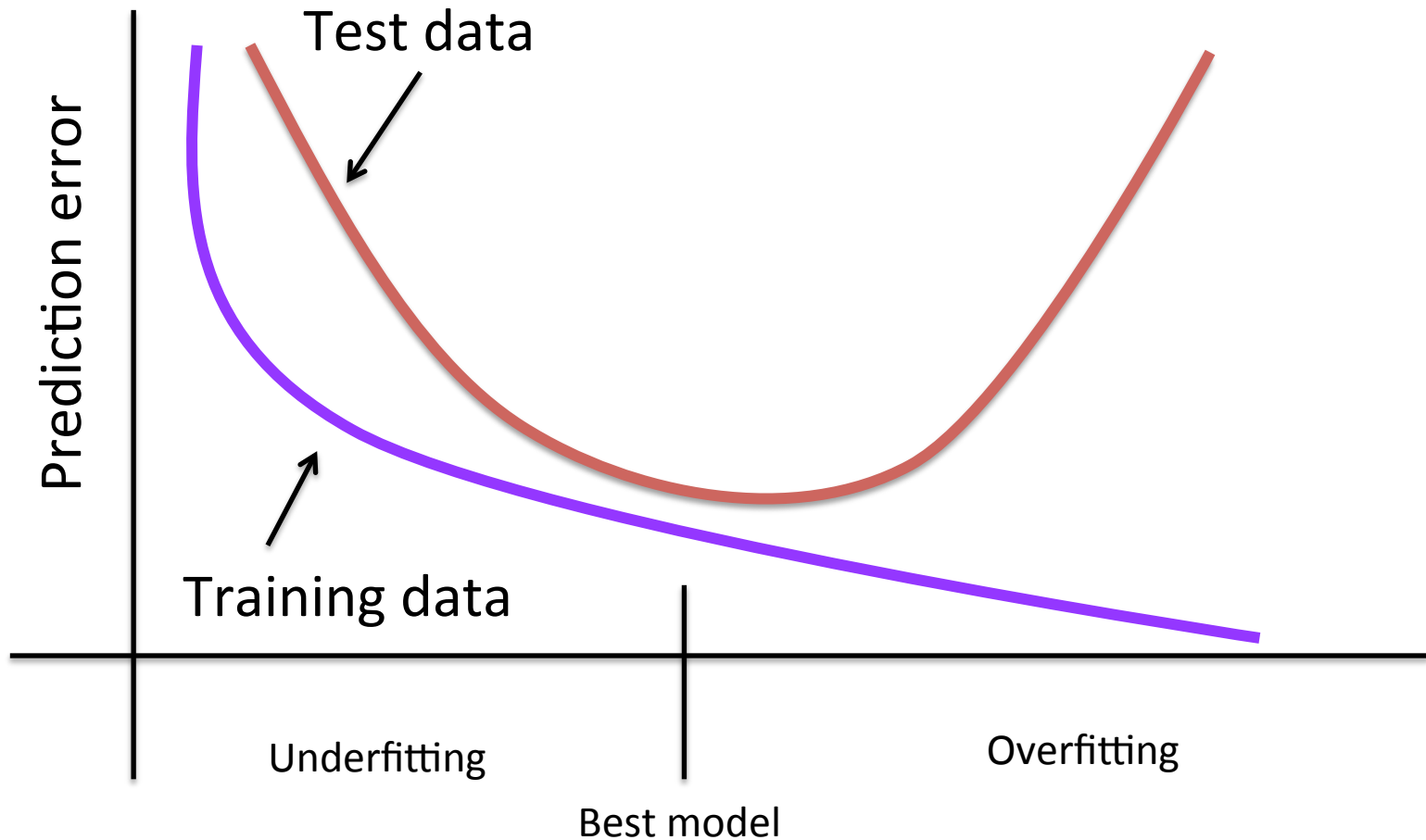
Model selection

- How do we choose the right model?
 - How do we set the parameters of that model?
- Many supervised learning approaches – ranging in complexity.
- Almost all models have some parameters that we need to set:
 - k in KNN classifier
 - Kernel parameter in SVM
 - Regularization parameter in penalized regression
 - Degree of the polynomial in polynomial regression
- **How do we select among a very large number of features?** (genomics application: selecting biomarkers among all measured genes)

Error and model complexity



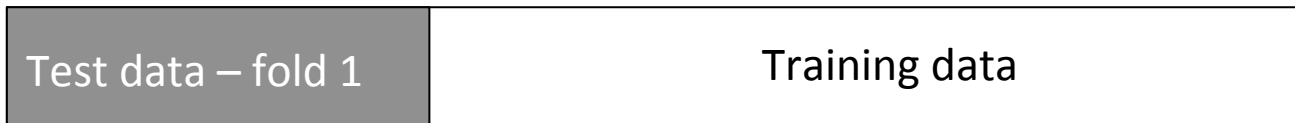
Error and model complexity



Model selection

- Cross-validation
- Complexity regularization
 - Variable selection / feature selection:
 - Only consider a limited set of features
 - Variable penalization
 - Reduce the magnitude of the coefficients so that the model doesn't become “overconfident”

3-fold CV



k-fold cross-validation (CV)

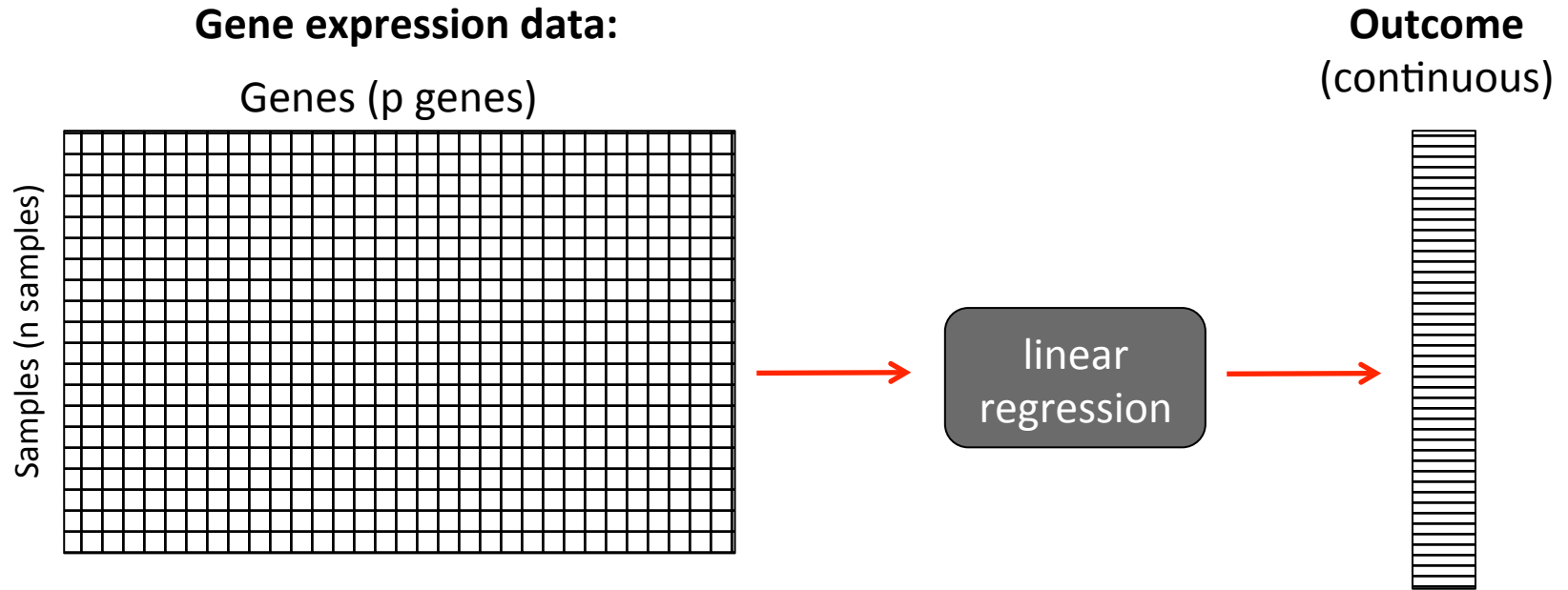
- k can range from 2 to n:
 - Larger k:
 - Variance of true error will be high
 - Computational time will be large
 - + More data to fit model parameters
 - Smaller k:
 - + reduced computation time
 - + variance of error estimate is small
 - Less data to fit model parameters
- n-fold CV is called “Leave-One-Out” CV
- In practice:
 - Lower k is more reliable, eg., 3-fold CV is standard.

- In high dimensional biology (“omics studies”) we typically have thousands to millions of features/covariates ($p \gg n$ problem):
 - Gene expression datasets: >20,000 genes
 - Genotyping datasets: > 1M SNPs
 - DNA Methylation datasets: > 200K methylation sites
- The main analysis goal is to identify features/covariates that important in predicting a trait/disease/outcome
- **Complexity regularization through feature selection and penalization**


Case study: why feature selection can help with overfitting

- Problem setup:
 - Gene expression data for n individuals & p genes
 - Response data is continuous
- Goals:
 - Build predictive model for inferring outcome
 - Identify ‘important’ genes

Case study: feature selection can help with overfitting




1. Generate some random data:
 - $X \sim \text{randn}(100,1000)$: 100 samples and 1000 genes, normally-distributed
 - $Y \sim \text{randn}(100,1)$: 100 samples, continuous + normally-distributed
2. Predict Y from X using linear regression:


$$\underset{\alpha, b}{\operatorname{argmin}} \sum_{i=1} (y_i - (b + \alpha^T \vec{x}_i))^2 \quad \text{Scalar notation}$$

(append a column of 1s to X, or mean-center X & y)

$$\underset{\alpha}{\operatorname{argmin}} \|\vec{y} - X\vec{\alpha}\|_2^2 \quad \text{Matrix notation}$$


$$= \underset{\alpha}{\operatorname{argmin}} (\vec{y} - X\vec{\alpha})^T (\vec{y} - X\vec{\alpha})$$

$$\vec{\alpha} = (X^T X)^{-1} X^T \vec{y} \quad \text{Solution for } \alpha$$

Toy example: how well can we predict y from X?

1. Generate some random data:

- $X \sim \text{randn}(100,1000)$: 100 samples and 1000 genes, normally-distributed
- $Y \sim \text{randn}(100,1)$: 100 samples, continuous + normally-distributed

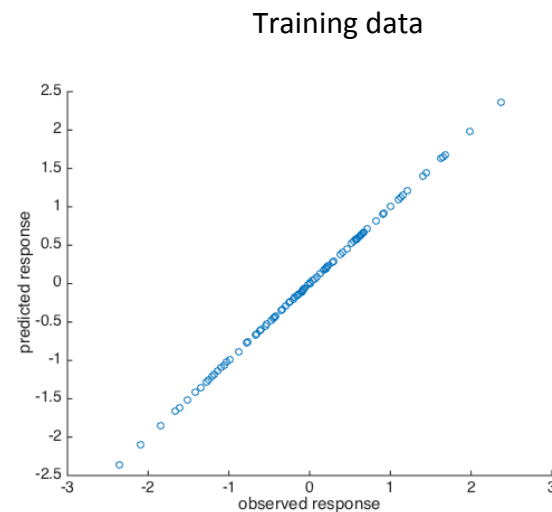
“Pseudo-code”

```
X = randn(100,1000);  
y = randn(100,1);  
[a,b] = corr(X,y);  
figure,hist(b)  
sum(b<.05)
```

```
Xo = [ones(100,1) X];  
c = (Xo'*Xo)\(Xo'*y);
```

```
pr = Xo*c;
```

```
figure,plot(y,pr,'o');
```

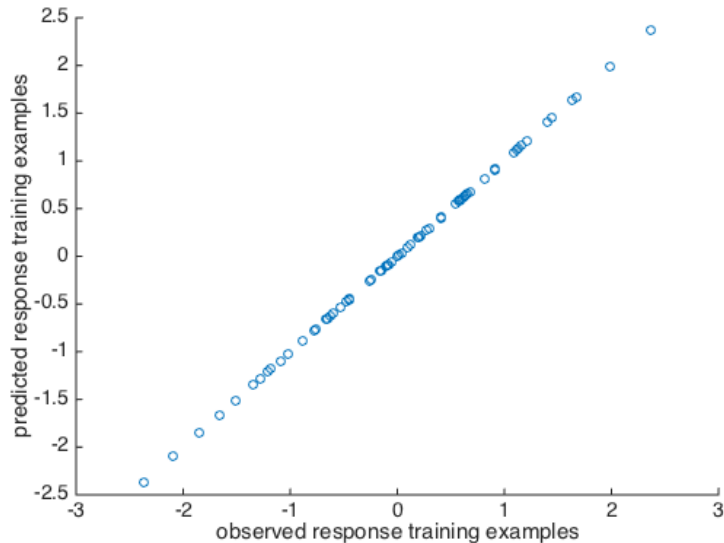


NOTE: matrix inversion not stable/reliable when $p > n$

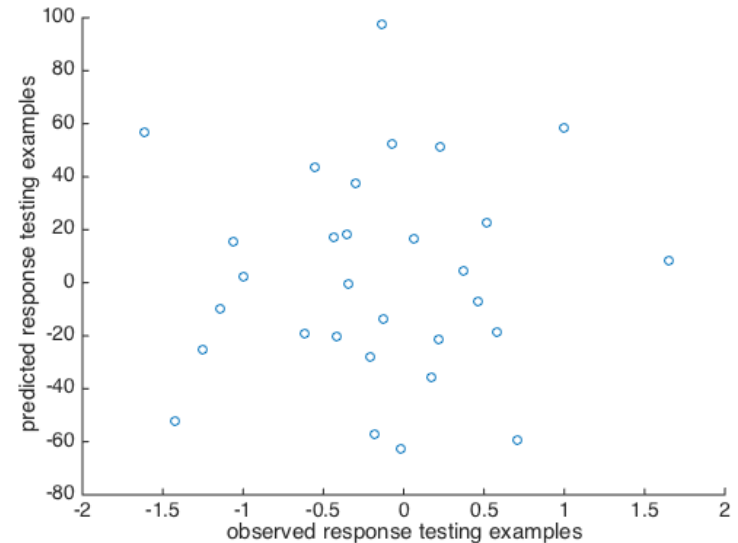
Demo: how well can we predict y from X?

```
%% split data to test and training sets %%  
  
r = randperm(100);  
idx1 = r(1:70);    %% index of the training examples  
idx2 = r(71:100);  %% index of the test examples  
  
a_training = (Xo(idx1,:)'\*Xo(idx1,:))\ (Xo(idx1,:)'*y(idx1));  
  
pr_training = Xo(idx1,:)*a_training;  
  
pr_test = Xo(idx2,:)*a_training;
```

Prediction on training data

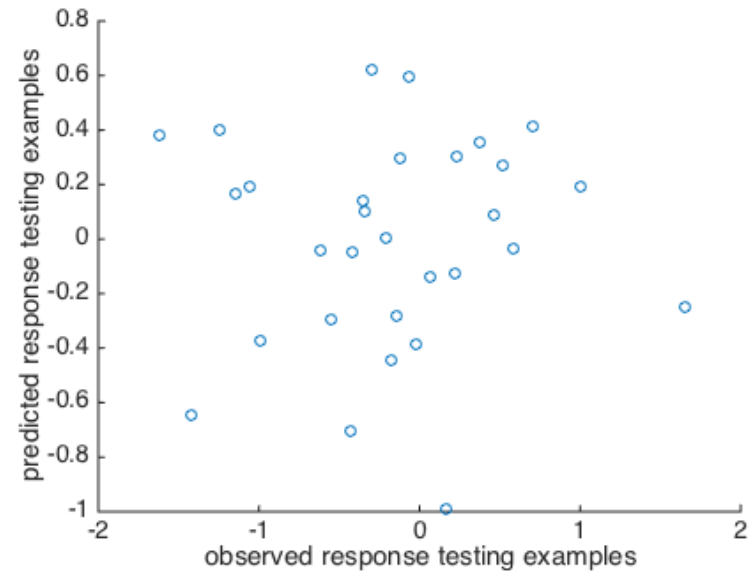
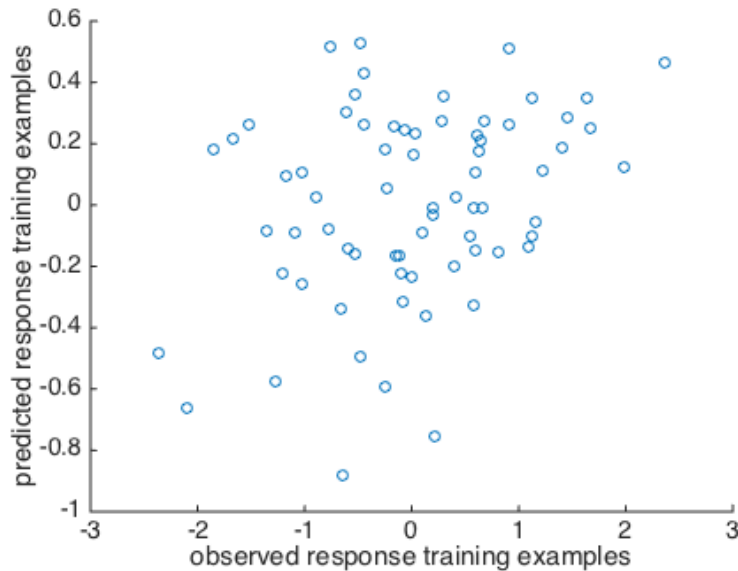


Prediction on testing data



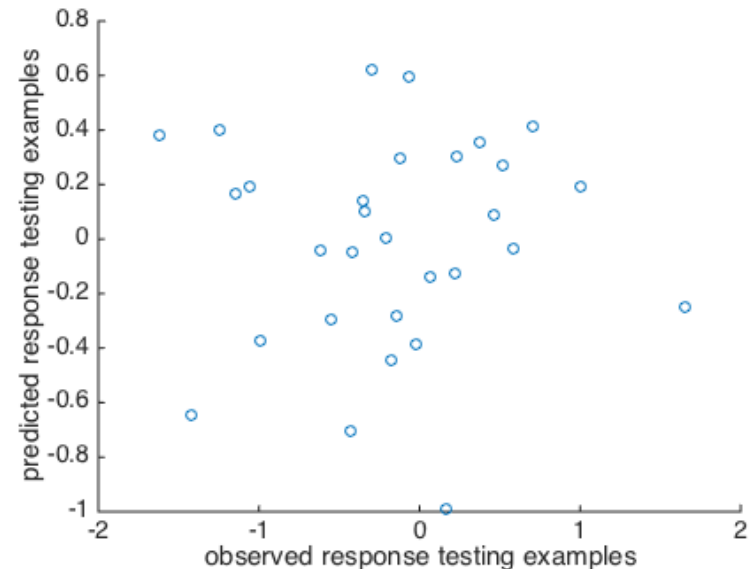
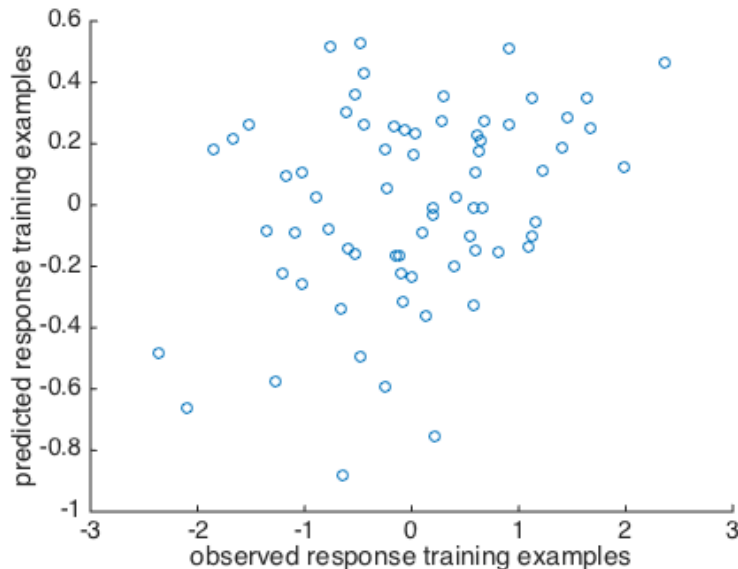
What's the main cause of this severe overfitting? Linear model should be simple enough ??

- Reduce the number of features: randomly choose 10 of the 1000 features



What's the main cause of this severe overfitting? Linear model should be simple enough ??

- Reduce the number of features: randomly choose 10 of the 1000 features



Aren't there better ways of filtering the genes?

- Correlation between expression of gene and outcome
Common mistake – don't look at test data when filtering
- Filter genes based on mean/std/prior knowledge

Feature selection

- The “filter” approach:
 - Don’t look at the labels/response: e.g., only keep genes with STD/mean > threshold
 - Consider the labels/response: e.g., only keep genes significantly correlated with response (note: must be careful to do this in a proper, nested, CV framework)
- The “wrapper” approach:
 - Identifying features that lead to good performance by the specific classifier
 - Cross-validation
- The “embedded” approach:
 - Modify objective function to identify a reduced feature set intrinsically
 - Regularization

Feature selection in linear regression

- **Best subset regression:**
 - Finds subset of size k with smallest error (e.g., mean squared error).
 - Unfeasible for omics studies, because $p\text{-choose-}k$ is huge!! (e.g., $>10^{10}$)
- **Forward/backward stepwise regression**
 - Sequentially add the best covariate that most improves the fit (e.g., minimizes prediction error)
- **Regularization methods (shrinkage/penalization)**
 - Modify the objective function, to explicitly penalize “magnitude” of coefficients

Regularized regression

$$\operatorname{argmin}_{\boldsymbol{\beta}} \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\text{Squared error}} + \underbrace{\lambda P(\boldsymbol{\beta})}_{\text{Penalization function}}$$

Squared error

Penalization function

Penalization (tuning) parameter

** Note change in notation: use bold lower case letters for vectors, upper case letters for matrices, lower case letters for scalars **

Regularized regression: ridge

$$\operatorname{argmin}_{\boldsymbol{\beta}} \underbrace{(\mathbf{y}-\mathbf{X}\boldsymbol{\beta})^T (\mathbf{y}-\mathbf{X}\boldsymbol{\beta})}_{\text{Squared error}} + \underbrace{\lambda P(\boldsymbol{\beta})}_{\text{Penalization function}}$$

Squared error

Penalization function

$$P(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^p \beta_i^2$$

Ridge regression: L2 norm
("squared") penalty function

(Hoerl and Kennard, *Technometrics*, 1970)

Regularized regression: ridge

$$\operatorname{argmin}_{\boldsymbol{\beta}} \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\text{Squared error}} + \underbrace{\lambda P(\boldsymbol{\beta})}_{\text{Penalization function}}$$

Squared error

Penalization function

$$P(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^p \beta_i^2$$

Ridge regression: L2 norm
("squared") penalty function

$\lambda \geq 0$ is the penalization parameter that we can "tune"

- When $\lambda = 0$ we get the linear regression estimate
- When $\lambda = \infty$ what happens?
- For λ in between 0 and infinity we are balancing error term vs penalty

Regularized regression: ridge

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\text{Squared error}} + \underbrace{\lambda \|\boldsymbol{\beta}\|_2^2}_{\text{Penalization function}}$$

$$\boldsymbol{\beta}^{\text{ridge}} = (\mathbf{X}^T \mathbf{X} + \lambda \mathbf{I})^{-1} \mathbf{X}^T \mathbf{y}$$

Regularized regression: ridge

$$\operatorname{argmin}_{\boldsymbol{\beta}} \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\text{Squared error}} + \underbrace{\lambda P(\boldsymbol{\beta})}_{\text{Penalization function}}$$

Squared error

Penalization function

$$P(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^p \beta_i^2$$

Ridge regression: L2 norm
("squared") penalty function

Important details:

- How do you choose λ ? Need nested CV to tune parameter and estimate test error.
- Requires standardization of the covariates (columns of X) :
 - the covariates need to be on the same "scale"
- Since columns standardized no intercept required

Regularized regression: ridge

$$\operatorname{argmin}_{\boldsymbol{\beta}} \underbrace{(\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})}_{\text{Squared error}} + \underbrace{\lambda P(\boldsymbol{\beta})}_{\text{Penalization function}}$$

Squared error

Penalization function

$$P(\boldsymbol{\beta}) = \|\boldsymbol{\beta}\|_2^2 = \sum_{i=1}^p \beta_i^2$$

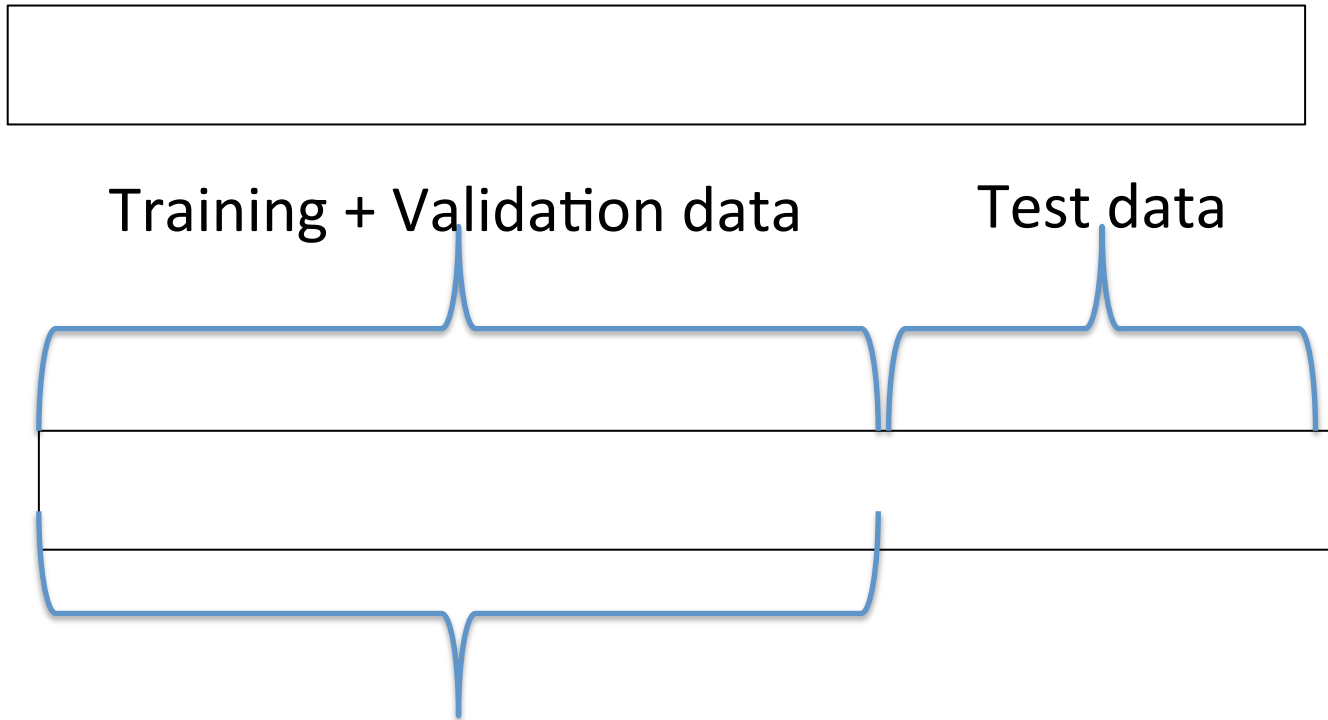
Ridge regression: L2 norm
("squared") penalty function

Choosing the penalization parameter:

To choose the parameter and assess testing error we need to do nested cross-validation. Why?

Nested cross-validation

Full data



Do k-fold CV here to find tuning parameter, report results on test data

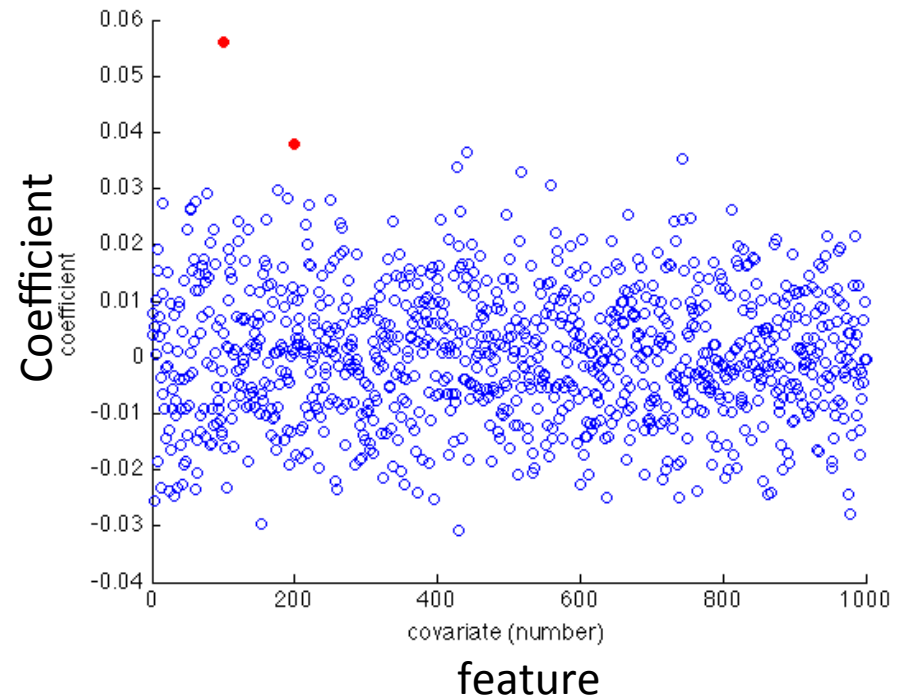
Does ridge allow us to identify “important” features?

Simulation:

$X \sim \text{randn}(100, 1000)$

$y \sim \text{randn}(100, 1) + X(:, 100) * .5 + X(:, 200) * .5$

Two “important” features:
feature 100 and feature 200

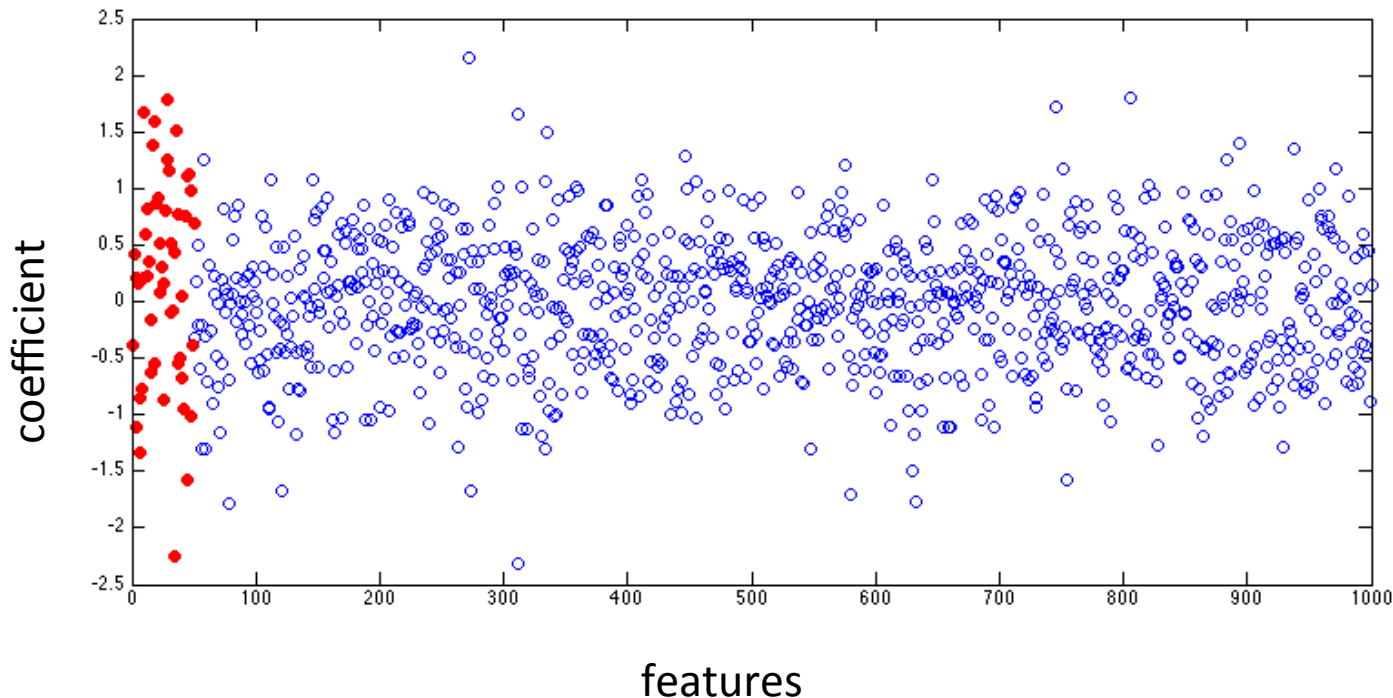


Does ridge allow us to identify “important” features?

Simulation: let's try having 50 important features

$X \sim \text{randn}(100, 1000)$

$y \sim \text{randn}(100, 1) + \text{sum}(X(:, 1:50), 2)$



Does ridge allow us to identify “important” features?

Simulation:

$X \sim \text{randn}(100, 1000)$

$y \sim \text{randn}(100, 1) + X(:, 100) * .5 + X(:, 200) * .5$

Observation:

- Many coefficients are “small” but not exactly zero (i.e., does not give a sparse solution)
- So ridge regression doesn’t perform variable selection!
- Performance is reasonable when only a few features are important but it breaks down when larger number of features are important.

feature

Regularized regression: ridge

$$\hat{\beta} = \underset{\beta}{\operatorname{argmin}} \underbrace{(y - X\beta)^T (y - X\beta)}_{\text{Squared error}} + \underbrace{\lambda \|\beta\|_2^2}_{\text{Penalization function}}$$

Squared error

Penalization function

Probabilistic interpretation:

$$L(\theta \mid \text{Data}) = L(\theta \mid X, y)$$

$$= p(y \mid X, \theta) p(\theta) = \underbrace{N(y \mid X\beta, \sigma)}_{\text{Likelihood}} \underbrace{N(\beta \mid 0, \nu)}_{\text{Prior}}$$

Regularized regression: lasso

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \quad (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda \|\boldsymbol{\beta}\|_1$$

$$\|\boldsymbol{\beta}\|_1 = \sum_{i=1}^p |\beta_i| \quad \longrightarrow \quad \text{L1 norm penalty}$$

- Proposed by Tibshirani (1996)
- Lasso is an acronym: Least Absolute Selection and Shrinkage Operator
- The only difference between ridge and lasso regularization is that ridge uses L2 norm vs lasso uses L1 norm. But turns out that the solutions to lasso and ridge behaves very differently.

$$\hat{\beta}_{lasso} = \underset{\beta}{\operatorname{argmin}} \quad (\mathbf{y} - \mathbf{X}\beta)^T (\mathbf{y} - \mathbf{X}\beta) + \lambda \|\beta\|_1$$

As before:

- The tuning parameter $\lambda \geq 0$ controls the strength of the penalty. What happens at the 0 and infinity?

But now:

- For $\lambda \geq 0$ between the two extreme setting, we are balancing two ideas: minimizing the squared error, and penalizing the magnitude of the coefficient. But the L1 penalty causes some coefficients to be set to **zero exactly**.
- Because some coefficients are set to zero exactly, lasso performs variable selection.
- No closed form solution: requires numerical optimization.
 - LARS algorithm
 - ADMM algorithm **

Comparing lasso and ridge

- Predictive performance: lasso and ridge are typically comparable
- Interpretation: lasso has a big advantage because it leads to variable selection

Comparing ridge and lasso: toy example

```
library(MASS)

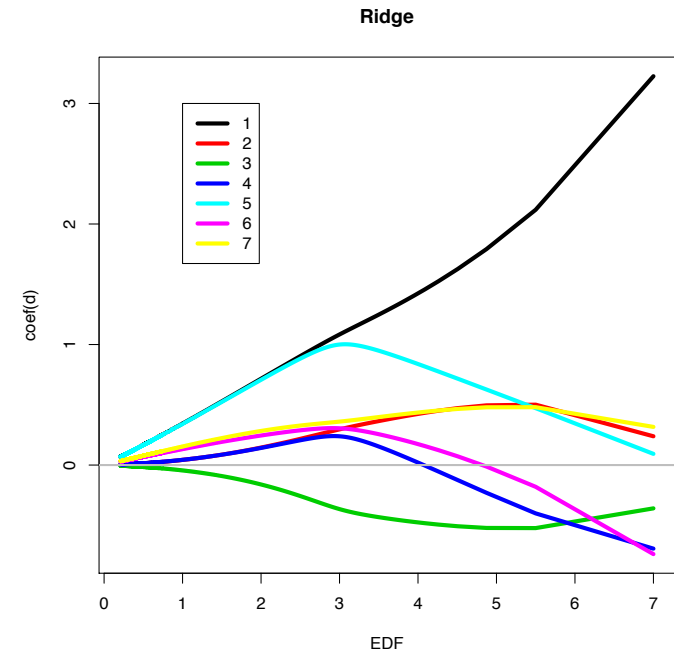
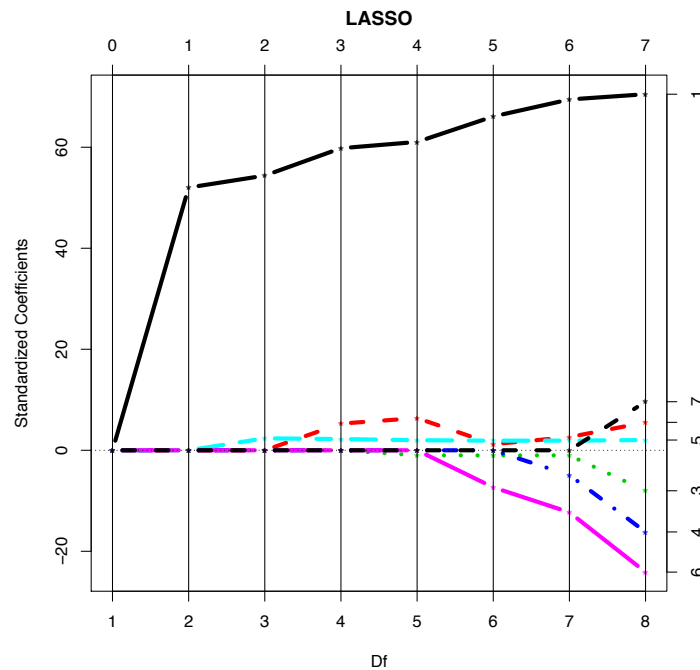
set.seed(123)
x1 <- rnorm(506)
x2 <- rnorm(506, mean = 2, sd = 1)

x3 <- rexp(506, rate = 1)
x4 <- x2 + rnorm(506, sd = 0.1)
x5 <- x1 + rnorm(506, sd = 0.1)
x6 <- x1 - x2 + rnorm(506, sd = 0.1)
x7 <- x1 + x3 + rnorm(506, sd = 0.1)

# Let's make x1 and x2 important covariates

y <- x1 * 3 + x2/3 + rnorm(506, sd = 2.2)
```


Comparing lasso and ridge



Intuitively df (degrees of freedom) of an estimate describes its effective number of parameters.

$$\text{df}(\hat{y}) = E[\text{number of nonzero coefficients in } \hat{\beta}^{\text{lasso}}]$$

Some limitations and considerations

- If $p \gg n$, the lasso can select at most n features (rest of the coefficients will be set to 0).
- If there is a group of highly correlated features, the lasso will choose only one among them (arbitrarily).
- Sensitivity to the choice of penalization parameter.

The elastic net: combined L2 and L1 penalty

$$\hat{\boldsymbol{\beta}} = \underset{\boldsymbol{\beta}}{\operatorname{argmin}} \quad (\mathbf{y} - \mathbf{X}\boldsymbol{\beta})^T (\mathbf{y} - \mathbf{X}\boldsymbol{\beta}) + \lambda((1 - \alpha)\|\boldsymbol{\beta}\|_1 + \alpha\|\boldsymbol{\beta}\|_2^2)$$

- Convex combination of the L1 and L2 penalty
- Addresses the shortcoming of the lasso in the presence of correlated features.
- But now we have to set 2 parameters.

Discussion

- Regularization methods can be extended to other models beyond linear regression (e.g., logistic regression)
- Designing variable penalization functions is an active area of research (e.g., fused lasso, group lasso, etc)
- How do we choose the regularization/penalty parameter?
 - Cross-validation
 - But note that even good performance in CV doesn't mean you get good performance on new test data ...