High-Dimensional Statistical Learning: Introduction

Jean Feng & Ali Shojaie

November 15, 2020
Sixth Seattle Symposium in Biostatistics

Logistics

Schedule:

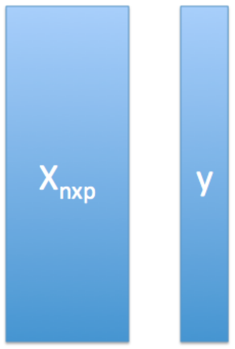
- 9-9:40 (A)
- 9:45-10:25 (A)
- 10:35-11:15 (J)
- 11:20-12:00 (J)
- BREAK (12:00-12:30)
- 12:30-1:20 (J)
- 1:25-2:00 (A)

Slides:
`http://faculty.washington.edu/ashojaie/teaching.html`

A Simple Example

- Suppose we have $n = 400$ people with diabetes for whom we have $p = 3$ serum-level measurements (LDL, HDL, GLU).
- We wish to predict these peoples' disease progression after 1 year.

A Simple Example



Notation:

- n is the number of observations.
- p the number of variables/features/predictors.
- y is a n-vector containing response/outcome for each of n observations.
- X is a $n \times p$ data matrix.

Linear Regression on a Simple Example

- You can perform linear regression to develop a model to predict progression using LDL, HDL, and GLU:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

where y is our continuous measure of disease progression, X_1, X_2, X_3 are our serum-level measurements, and ϵ is a noise term.

Linear Regression on a Simple Example

- You can perform linear regression to develop a model to predict progression using LDL, HDL, and GLU:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

where y is our continuous measure of disease progression, X_1, X_2, X_3 are our serum-level measurements, and ϵ is a noise term.

- You can look at the coefficients, p-values, and t-statistics for your linear regression model in order to interpret your results.

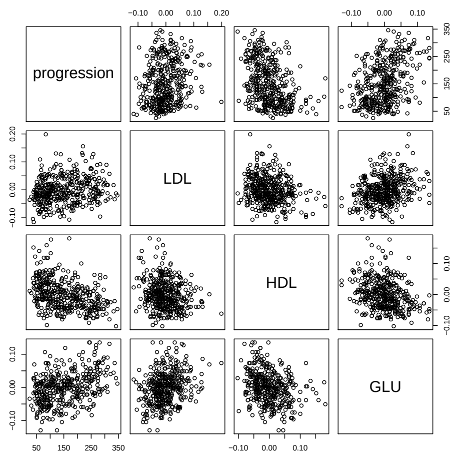
Linear Regression on a Simple Example

- You can perform linear regression to develop a model to predict progression using LDL, HDL, and GLU:

$$y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3 + \epsilon$$

  where y is our continuous measure of disease progression, X_1, X_2, X_3 are our serum-level measurements, and ϵ is a noise term.
- You can look at the coefficients, p-values, and t-statistics for your linear regression model in order to interpret your results.
- You learned everything (or most of what) you need to analyze this data set in AP Statistics!

A Relationship Between the Variables?



Linear Model Output

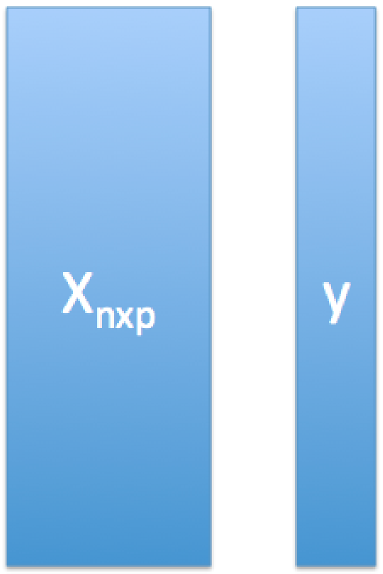
	Estimate	Std. Error	T-Stat	P-Value
Intercept	152.928	3.385	45.178	< 2e-16 ***
LDL	77.057	75.701	1.018	0.309
HDL	-487.574	75.605	-6.449	3.28e-10 ***
GLU	477.604	76.643	6.232	1.18e-09 ***

$$\text{progression\_measure} \approx 152.9 + 77.1 \times \text{LDL} - 487.6 \times \text{HDL} + 477.6 \times \text{GLU}.$$

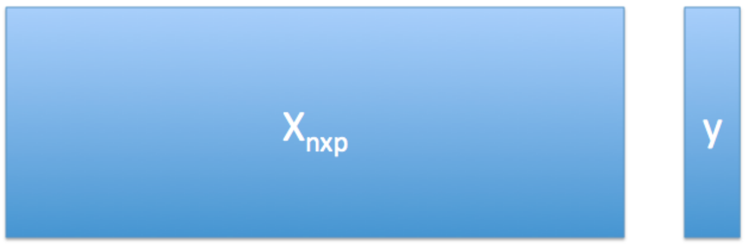
Low-Dimensional Versus High-Dimensional

- The data set that we just saw is low-dimensional: $n \gg p$.
- Lots of the data sets coming out of modern biological techniques are high-dimensional: $n \approx p$ or $n \ll p$.
- This poses statistical challenges! AP Statistics no longer applies.

Low Dimensional



High Dimensional



What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2

What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.

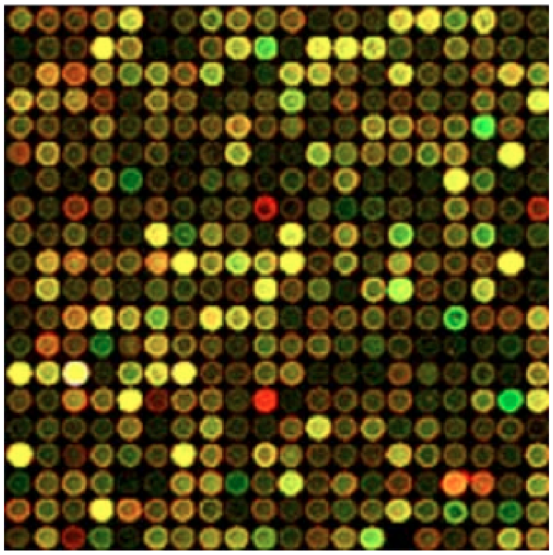
What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.
- If we include too many predictors, we will overfit the data.
- Overfitting: Model looks great on the data used to develop it, but will perform very poorly on future observations.

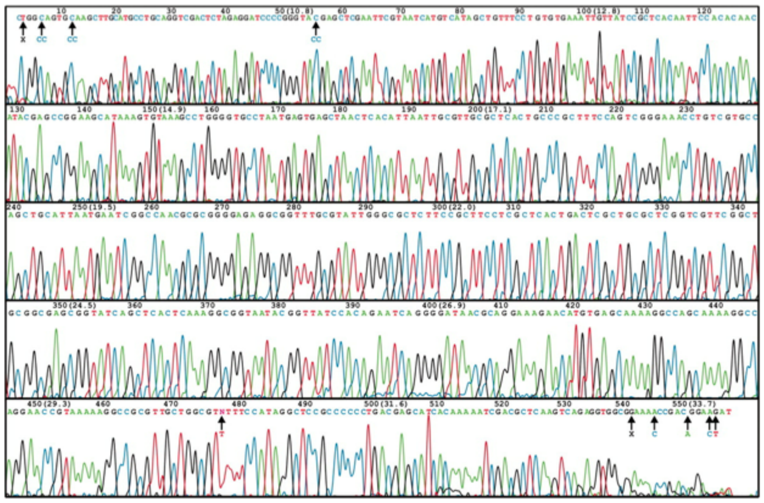
What Goes Wrong in High Dimensions?

- Suppose that we included many additional predictors in our model, such as
  - Age
  - Zodiac symbol
  - Favorite color
  - Mother's birthday, in base 2
- Some of these predictors are useful, others aren't.
- If we include too many predictors, we will overfit the data.
- Overfitting: Model looks great on the data used to develop it, but will perform very poorly on future observations.
- When $p \approx n$ or $p > n$, overfitting is guaranteed unless we are very careful.

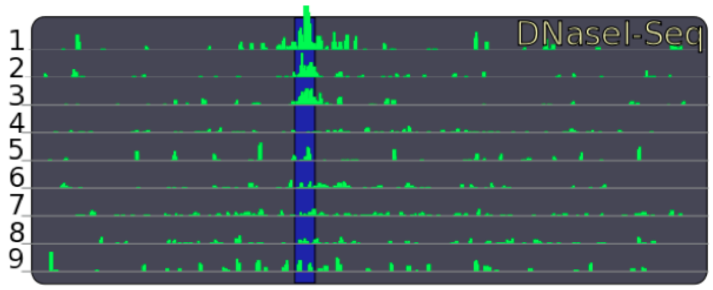
Gene Expression Data



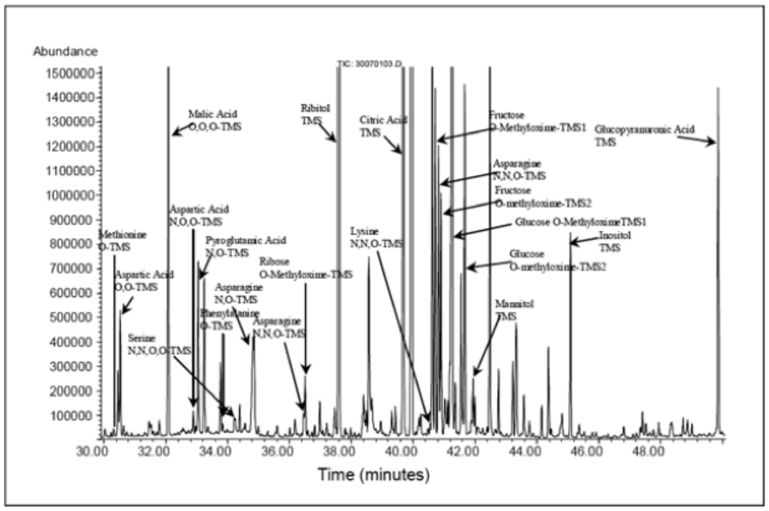
DNA Sequence Data



DNAse Hypersensitivity Data



Metabolomic Data



High-Dimensional Omics Analyses

For most omics analyses, we have many more variables than observations.... i.e. $p \gg n$.

Classical Statistics
Biological Big Data
Supervised and Unsupervised Learning

High-Dimensional Omics Analyses

For most omics analyses, we have many more variables than observations.... i.e. $p \gg n$.

- Predict risk of diabetes on the basis of DNA sequence data.... using $n = 1000$ patients and $p = 3000000$ variables.

High-Dimensional Omics Analyses

For most omics analyses, we have many more variables than observations.... i.e. $p \gg n$.

- Predict risk of diabetes on the basis of DNA sequence data.... using $n = 1000$ patients and $p = 3000000$ variables.
- Cluster tissue samples on the basis of DNase hypersensitivity... using $n = 200$ cell types and $p = 1000000000$ variables.

High-Dimensional Omics Analyses

For most omics analyses, we have many more variables than observations.... i.e. $p \gg n$.

- Predict risk of diabetes on the basis of DNA sequence data.... using $n = 1000$ patients and $p = 3000000$ variables.
- Cluster tissue samples on the basis of DNase hypersensitivity... using $n = 200$ cell types and $p = 1000000000$ variables.
- Identify genes whose expression is associated with survival time... using $n = 250$ cancer patients and $p = 20000$ variables.

Why Does Dimensionality Matter?

- Classical statistical techniques, such as linear regression, *cannot* be applied.
- Even very simple tasks, like identifying variables that are associated with a response, must be done with care.
- High risks of overfitting, false positives, and more.

Classical Statistics
Biological Big Data
Supervised and Unsupervised Learning

Why Does Dimensionality Matter?

- Classical statistical techniques, such as linear regression, *cannot* be applied.
- Even very simple tasks, like identifying variables that are associated with a response, must be done with care.
- High risks of overfitting, false positives, and more.

This course: Statistical machine learning tools for big – mostly high-dimensional – data.

Supervised and Unsupervised Learning

- Statistical machine learning can be divided into two main areas: supervised and unsupervised.

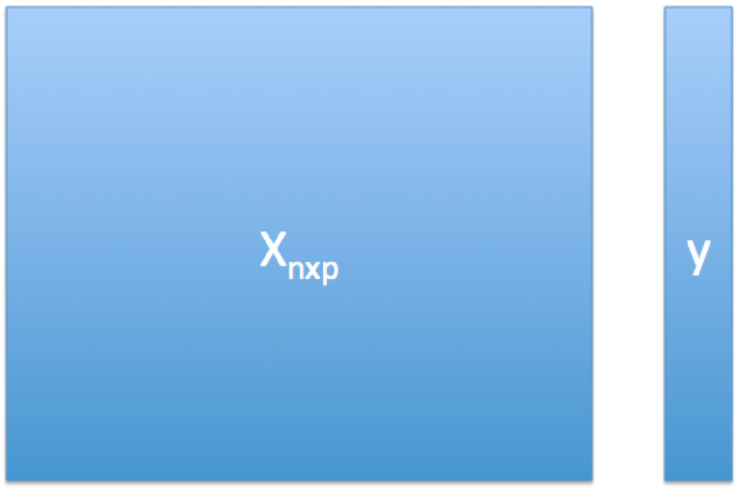
Supervised and Unsupervised Learning

- Statistical machine learning can be divided into two main areas: supervised and unsupervised.
- Supervised Learning: Use a data set X to predict or detect association with a response y.
  - Regression
  - Classification
  - Hypothesis Testing

Supervised and Unsupervised Learning

- Statistical machine learning can be divided into two main areas: supervised and unsupervised.
- Supervised Learning: Use a data set X to predict or detect association with a response y.
  - Regression
  - Classification
  - Hypothesis Testing
- Unsupervised Learning: Discover the signal in X, or detect associations within X.
  - Dimension Reduction
  - Clustering

Supervised Learning

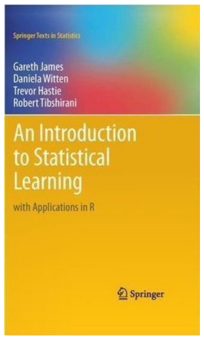


Unsupervised Learning

This Course

- We will cover the big ideas in supervised learning for big data.
- We will focus on applications in biomarker development.

"Reference Textbook" . . . with applications in R



- Available for (free!) download from www.statlearning.com.
- An accessible introduction to statistical machine learning, with an R lab at the end of each chapter!!
- To learn more, go through R labs on your own!

Where this fits in the Biomarker world

Goal:

To develop prescriptive/predictive biomarkers — indicate who will benefit from new treatment over SOC.

If target of therapy is well understood, use it!

likely do not need high-dimensional techniques

Where this fits in the Biomarker world

However, sometimes

- the therapy is not targeted
- the target is complicated (e.g. a large genomic pathway)
- the target turns out to be a poor predictor of effectiveness

In these cases, statistical machine learning can be effective for developing prescriptive biomarkers.

Examples from the Biomarker world

Oncotype DX

Exact Sciences

Prolaris, myRisk, etc...

Myriad Genetics

Decipher

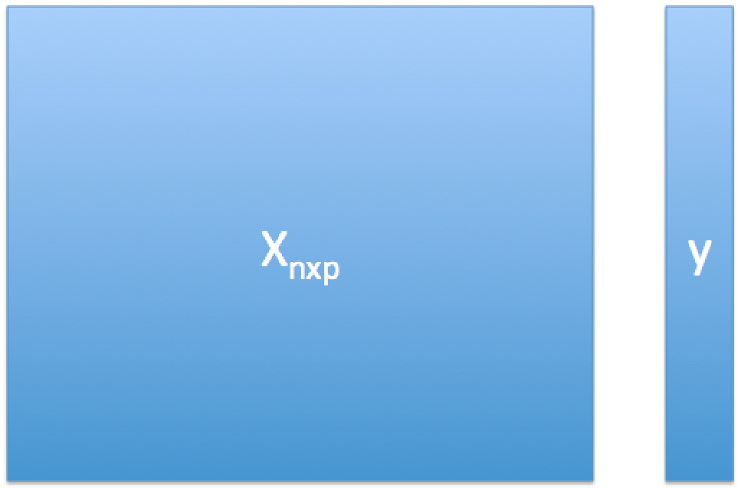
GenomeDx (Decipher) Biosciences

High-Dimensional Statistical Learning: Bias Variance Tradeoff and the Test Error

Jean Feng & Ali Shojaie

November 15, 2020
Sixth Seattle Symposium in Biostatistics

Supervised Learning



Regression Versus Classification

Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size

Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size
- Classification: Predict a categorical response, such as
  - tumor versus normal tissue
  - heart disease versus no heart disease
  - subtype of glioblastoma

Regression Versus Classification

- Regression: Predict a quantitative response, such as
  - blood pressure
  - cholesterol level
  - tumor size
- Classification: Predict a categorical response, such as
  - tumor versus normal tissue
  - heart disease versus no heart disease
  - subtype of glioblastoma
- This lecture: Regression.

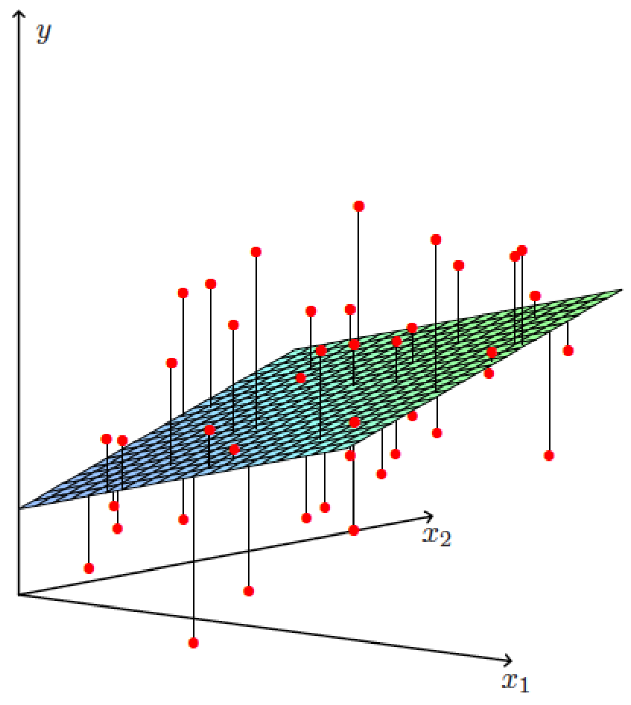
Linear Models

- We have n observations, for each of which we have p predictor measurements and a response measurement.
- Want to develop a model of the form

$$y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \epsilon_i.$$

- Here ϵ_i is a noise term associated with the ith observation.
- Must estimate $\beta_0, \beta_1, \ldots, \beta_p$ — i.e. we must fit the model.

Linear Model With $p = 2$ Predictors



Least Squares Regression

- There are many ways we could fit the model
$$y_i = \beta_0 + \beta_1 X_{i1} + \cdots + \beta_p X_{ip} + \epsilon_i.$$
- Most common approach in classical statistics is least squares:
$$\underset{\boldsymbol{\beta}}{\text{minimize}} \sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2$$
- We are looking for $\beta_1, \ldots, \beta_p$ such that
$$\sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2$$
is as small as possible, or in other words, such that
$$\sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$
is as small as possible, where $\hat{y}_i$ is the ith predicted value.

Least Squares Regression

- When we fit a model, we use a training set of observations.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}.$$

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \dots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \dots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.

- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}))^2.$$

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.
- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}))^2.$$

- The training error is closely related to the R^2 for a linear model — that is, the proportion of variance explained.

Least Squares Regression

- When we fit a model, we use a training set of observations.
- We get coefficient estimates $\hat{\beta}_1, \ldots, \hat{\beta}_p$.
- We also get predictions using our model, of the form

$$\hat{y}_i = \hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}.$$

- We can evaluate the training error, i.e. the extent to which the model fits the observations used to train it.
- One way to quantify the training error is using the mean squared error (MSE):

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (\hat{\beta}_1 X_{i1} + \cdots + \hat{\beta}_p X_{ip}))^2.$$

- The training error is closely related to the R^2 for a linear model — that is, the proportion of variance explained.
- Big $R^2 \Leftrightarrow$ Small Training Error.

Least Squares as More Variables are Included in the Model

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.

Least Squares as More Variables are Included in the Model

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.

Least Squares as More Variables are Included in the Model

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features to use in developing a model, then I could surely come up with a regression model that fits the training data perfectly! Unfortunately, this model wouldn't capture the true signal in the data.

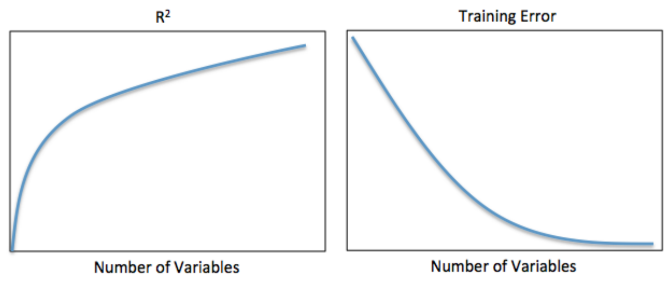
Least Squares as More Variables are Included in the Model

- Training error and R^2 are not good ways to evaluate a model's performance, because they will always improve as more variables are added into the model.
- The problem? Training error and R^2 evaluate the model's performance on the training observations.
- If I had an unlimited number of features to use in developing a model, then I could surely come up with a regression model that fits the training data perfectly! Unfortunately, this model wouldn't capture the true signal in the data.

- We really care about the model's performance on test observations — observations not used to fit the model.

The Problem

As we add more variables into the model...



... the training error decreases and the R^2 increases!

Why is this a Problem?

- We really care about the model's performance on observations not used to fit the model!
  - We want a model that will predict the survival time of a new patient who walks into the clinic!
  - We want a model that can be used to diagnose cancer for a patient not used in model training!
  - We want to predict risk of diabetes for a patient who wasn't used to fit the model!

Why is this a Problem?

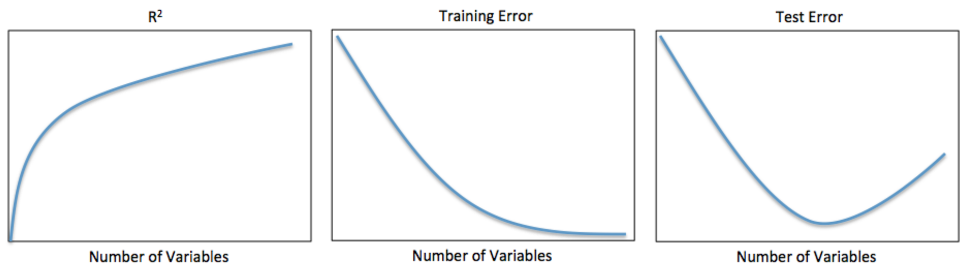
- We really care about the model's performance on observations not used to fit the model!
  - We want a model that will predict the survival time of a new patient who walks into the clinic!
  - We want a model that can be used to diagnose cancer for a patient not used in model training!
  - We want to predict risk of diabetes for a patient who wasn't used to fit the model!
- What we really care about:

$$(y_{test} - \hat{y}_{test})^2,$$

where
$$\hat{y}_{test} = \hat{\beta}_1 X_{test,1} + \cdots + \hat{\beta}_p X_{test,p},$$
and (X_{test}, y_{test}) was not used to train the model.
- The test error is the average of $(y_{test} - \hat{y}_{test})^2$ over a bunch of test observations.

Training Error versus Test Error

As we add more variables into the model...

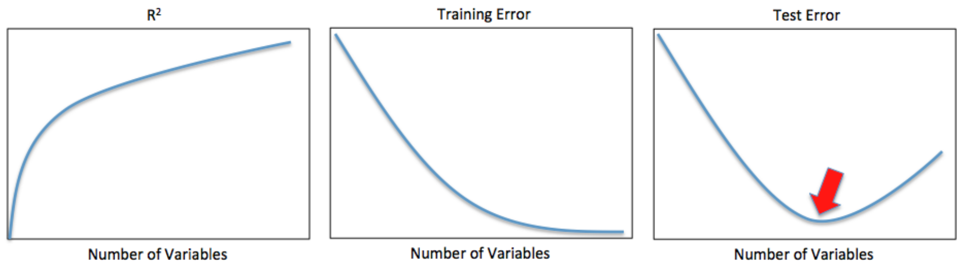


... the training error decreases and the R^2 increases!

But the test error might not!

Training Error versus Test Error

As we add more variables into the model...

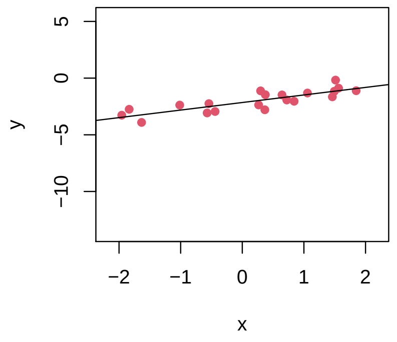


... the training error decreases and the R^2 increases!

But the test error might not!

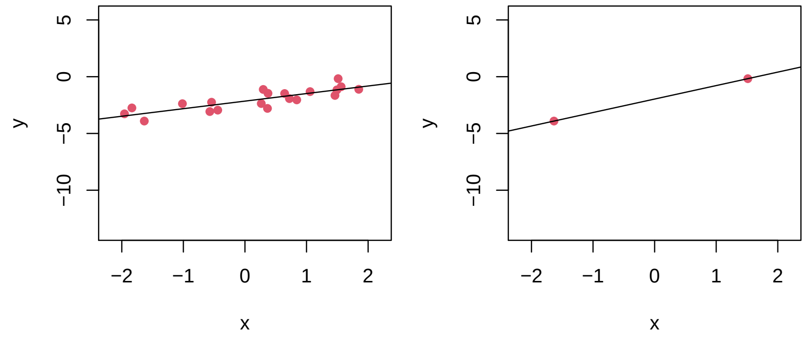
Why the Number of Variables Matters

- Linear regression will have a very low training error if p is large relative to n.
- A simple example:



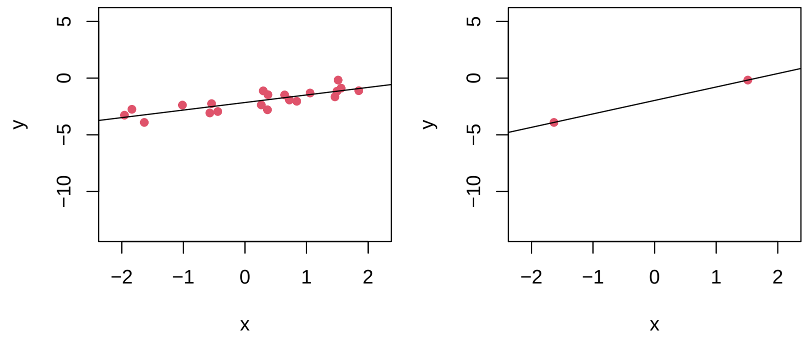
Why the Number of Variables Matters

- Linear regression will have a very low training error if p is large relative to n.
- A simple example:



Why the Number of Variables Matters

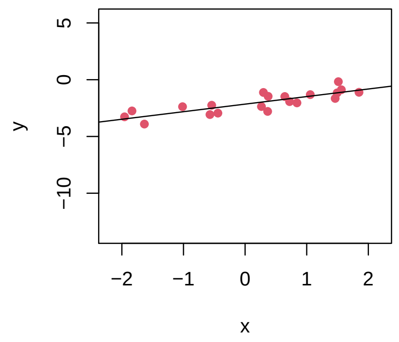
- Linear regression will have a very low training error if p is large relative to n.
- A simple example:



- When $n \leq p$, you can always get a perfect model fit to the training data!
- But the test error will be awful.

Why the Number of Variables Matters

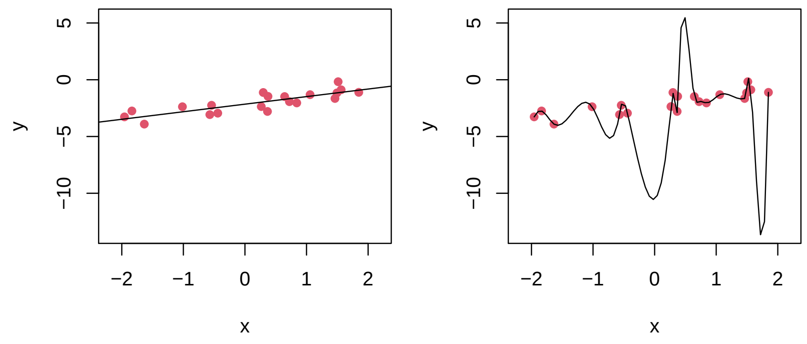
- Linear regression will have a very low training error if p is large relative to n.
- A simple example:



- When $n \leq p$, you can always get a perfect model fit to the training data!
- But the test error will be awful.

Why the Number of Variables Matters

- Linear regression will have a very low training error if p is large relative to n.
- A simple example:

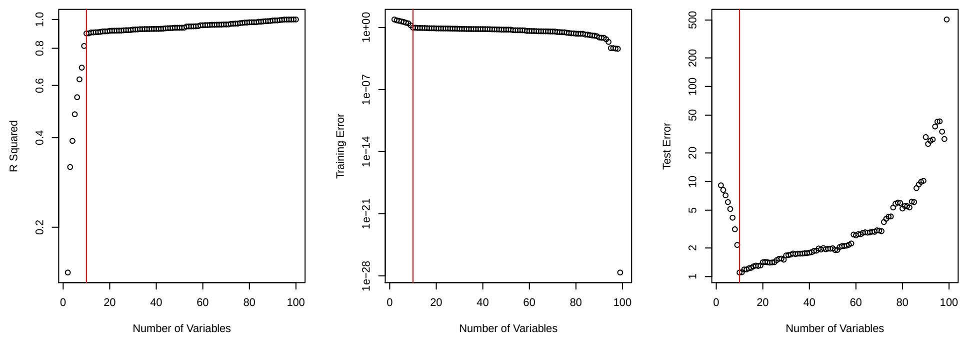


- When $n \leq p$, you can always get a perfect model fit to the training data!
- But the test error will be awful.

Model Complexity, Training Error, and Test Error

- In this course, we will consider various types of models.
- We will be very concerned with model complexity: e.g. the number of variables used to fit a model.
- As we fit more complex models — e.g. models with more variables — the training error will always decrease.
- But the test error might not.
- As we will see, the number of variables in the model is not the only — or even the best — way to quantify model complexity.

A Simulated Example



- 1st 10 variables are related to response; remaining 90 are not.
- R^2 increases and training error decreases as more variables are added to the model.
- Test error is lowest when only signal variables in model.

Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ — the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times — will decrease.

Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ — the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times — will decrease.

- But as complexity increases, the variance of $\hat{\beta}$ — the amount by which the $\hat{\beta}$'s will differ across experiments — will increase.

Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ — the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times — will decrease.
- But as complexity increases, the variance of $\hat{\beta}$ — the amount by which the $\hat{\beta}$'s will differ across experiments — will increase.
- The test error depends on both the bias and variance:

$$\text{Test Error} = \text{Bias}^2 + \text{Variance}.$$

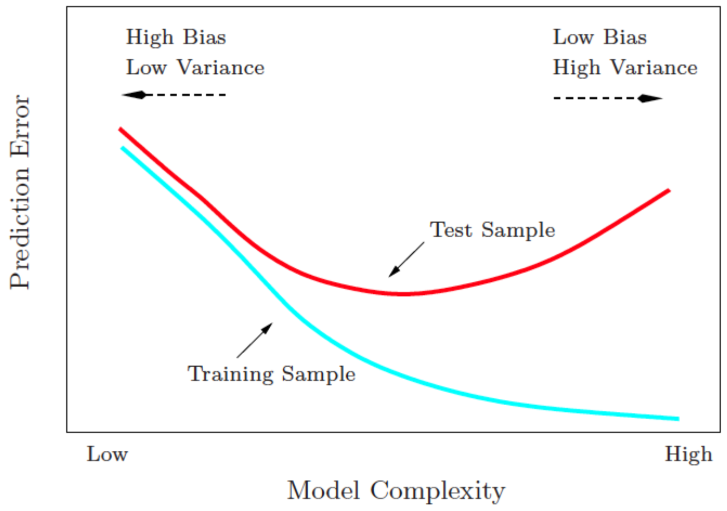
Bias and Variance

- As model complexity increases, the bias of $\hat{\beta}$ — the average difference between β and $\hat{\beta}$, if we were to repeat the experiment a huge number of times — will decrease.
- But as complexity increases, the variance of $\hat{\beta}$ — the amount by which the $\hat{\beta}$'s will differ across experiments — will increase.
- The test error depends on both the bias and variance:

$$\text{Test Error} = \text{Bias}^2 + \text{Variance}.$$

- There is a bias-variance trade-off. We want a model that is sufficiently complex as to have not too much bias, but not so complex that it has too much variance.

A Really Fundamental Picture



Overfitting

Overfitting

- Fitting an overly complex model — a model that has too much variance — is known as overfitting.

Overfitting

- Fitting an overly complex model — a model that has too much variance — is known as overfitting.
- In the omics setting, when $p \gg n$, we must work hard not to overfit the data.

Overfitting

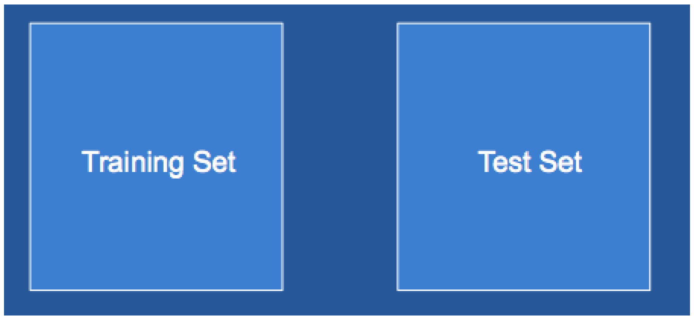
- Fitting an overly complex model — a model that has too much variance — is known as overfitting.
- In the omics setting, when $p \gg n$, we must work hard not to overfit the data.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.

Overfitting

- Fitting an overly complex model — a model that has too much variance — is known as overfitting.
- In the omics setting, when $p \gg n$, we must work hard not to overfit the data.
- In particular, we must rely not on training error, but on test error, as a measure of model performance.
- How can we estimate the test error?

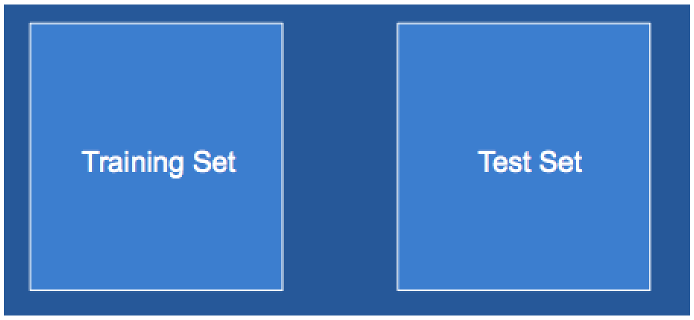
Training Set Versus Test Set

- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.



Training Set Versus Test Set

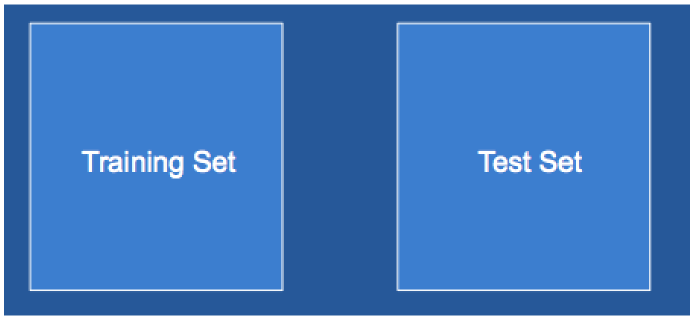
- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.



Q: Can there ever, under any circumstance, be sample overlap between the training and test sets?

Training Set Versus Test Set

- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.



Q: Can there ever, under any circumstance, be sample overlap between the training and test sets?
A: No no no no no no.

Training Set Versus Test Set

- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.

Training Set Versus Test Set

- Split samples into training set and test set.
- Fit model on training set, and evaluate on test set.

You can't peek at the test set until you are completely done all aspects of model-fitting on the training set!

Training Set And Test Set

To get an estimate of the test error of a particular model on a future observation:

1. Split the samples into a training set and a test set.
2. Fit the model on the training set.
3. Evaluate its performance on the test set.
4. The test set error rate is an estimate of the model's performance on a future observation.

Training Set And Test Set

To get an estimate of the test error of a particular model on a future observation:

1. Split the samples into a training set and a test set.
2. Fit the model on the training set.
3. Evaluate its performance on the test set.
4. The test set error rate is an estimate of the model's performance on a future observation.

But remember: no peeking!

Choosing Between Several Models

- In general, we will consider a lot of possible models — e.g. models with different levels of complexity. We must decide which model is best.

Choosing Between Several Models

- In general, we will consider a lot of possible models — e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!

Choosing Between Several Models

- In general, we will consider a lot of possible models — e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!

Choosing Between Several Models

- In general, we will consider a lot of possible models — e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?

Choosing Between Several Models

- In general, we will consider a lot of possible models — e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?

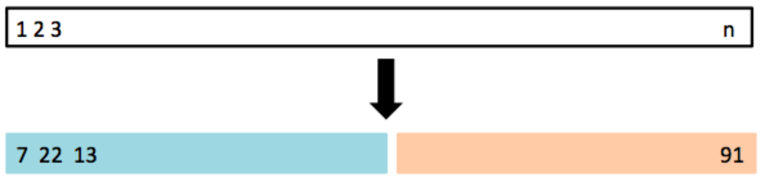
  1. The validation set approach.
  2. Leave-one-out cross-validation.
  3. K-fold cross-validation.

Choosing Between Several Models

- In general, we will consider a lot of possible models — e.g. models with different levels of complexity. We must decide which model is best.
- We have split our samples into a training set and a test set. But remember: we can't peek at the test set until we have completely finalized our choice of model!
- We must pick a best model based on the training set, but we want a model that will have low test error!
- How can we estimate test error using only the training set?
  1. The validation set approach.
  2. Leave-one-out cross-validation.
  3. K-fold cross-validation.
- In what follows, assume we have split the data into a training set and a test set, and the training set contains n observations.

Validation Set Approach

Split the n observations into two sets of approximately equal size. Train on one set, and evaluate performance on the other.



Validation Set Approach

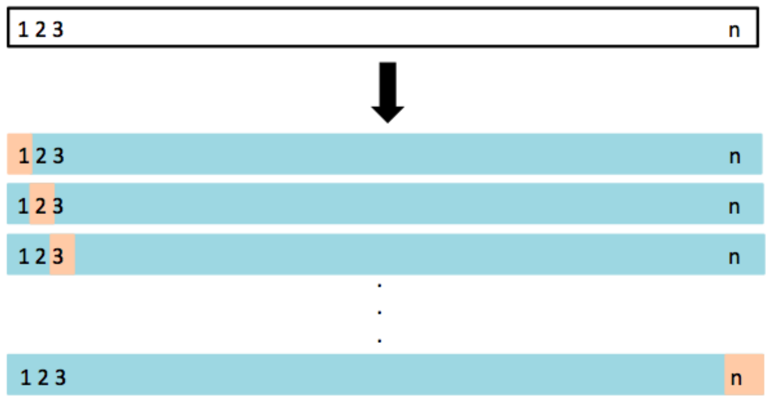
For a given model, we perform the following procedure:

1. Split the observations into two sets of approximately equal size, a training set and a validation set.
   a. Fit the model using the training observations. Let $\hat{\boldsymbol{\beta}}_{(train)}$ denote the regression coefficient estimates.
   b. For each observation in the validation set, compute the test error, $e_i = (y_i - \mathrm{x}_i^T \hat{\boldsymbol{\beta}}_{(train)})^2$.
2. Calculate the total validation set error by summing the e_i's over all of the validation set observations.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

Leave-One-Out Cross-Validation

Fit n models, each on $n-1$ of the observations. Evaluate each model on the left-out observation.



Leave-One-Out Cross-Validation

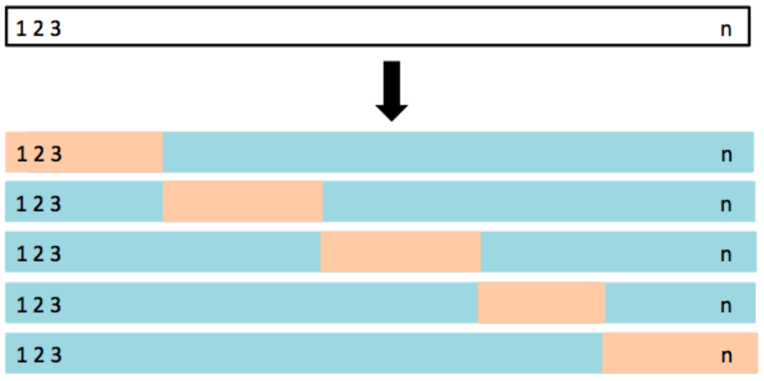
For a given model, we perform the following procedure:

1. For $i = 1, \ldots, n$:
   a. Fit the model using observations $1, \ldots, i-1, i+1, \ldots, n$. Let $\hat{\beta}_{(i)}$ denote the regression coefficient estimates.
   b. Compute the test error, $e_i = (y_i - x_i^T \hat{\beta}_{(i)})^2$.
2. Calculate $\sum_{i=1}^{n} e_i$, the total CV error.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

5-Fold Cross-Validation

Split the observations into 5 sets. Repeatedly train the model on 4 sets and evaluate its performance on the 5th.



K-fold cross-validation

A generalization of leave-one-out cross-validation. For a given model, we perform the following procedure:

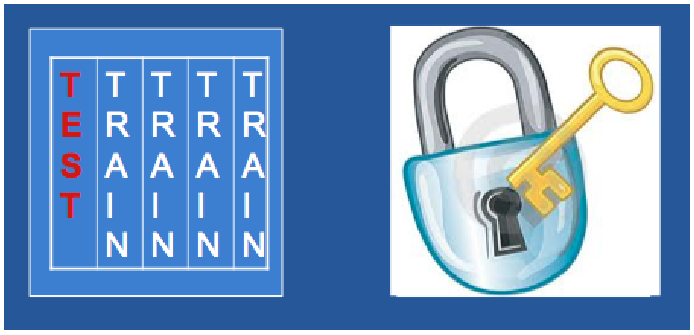
1. Split the n observations into K equally-sized folds.
2. For $k = 1, \ldots, K$:
   a. Fit the model using the observations not in the kth fold.
   b. Let e_k denote the test error for the observations in the kth fold.
3. Calculate $\sum_{k=1}^{K} e_k$, the total CV error.

Out of a set of candidate models, the "best" one is the one for which the total error is smallest.

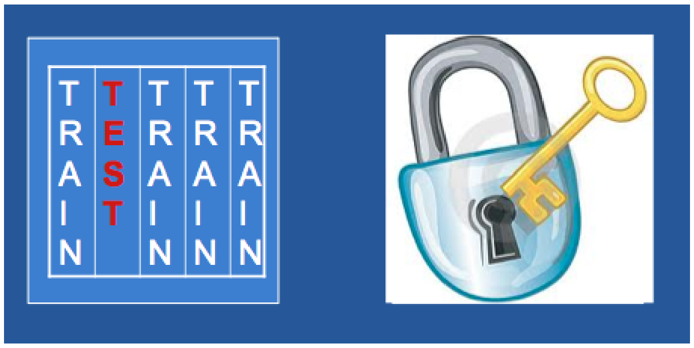
After Estimating the Test Error on the Training Set...

After we estimate the test error using the training set, we refit the "best" model on all of the available training observations. We then evaluate this model on the test set.

Big Picture



Big Picture



Big Picture

TRAIN
TRAIN
TEST
TRAIN
TRAIN

Big Picture

TRAIN
TRAIN
TRAIN
TEST
TRAIN

Big Picture

Summary: Four-Step Procedure

1. Split observations into training set and test set.
2. Fit a bunch of models on training set, and estimate the test error, using cross-validation or validation set approach.
3. Refit the best model on the full training set.
4. Evaluate the model's performance on the test set.

Why All the Bother?

Q: Why do I need to have a separate test set? Why can't I just estimate test error using cross-validation or a validation set approach using all the observations, and then be done with it?

Why All the Bother?

Q: Why do I need to have a separate test set? Why can't I just estimate test error using cross-validation or a validation set approach using all the observations, and then be done with it?

A: In general, we are choosing between a whole bunch of models, and we will use the cross-validation or validation set approach to pick between these models. If we use the resulting estimate as a final estimate of test error, then this could be an extreme underestimate, because one model might give a lower estimated test error than others by chance. To avoid having an extreme underestimate of test error, we need to evaluate the "best" model obtained on an independent test set. *This is particularly important in high dimensions!!*

Regression in High Dimensions

- We usually cannot perform least squares regression to fit a model in the omics setting, because we will get zero training error but a terrible test error.

Regression in High Dimensions

- We usually cannot perform least squares regression to fit a model in the omics setting, because we will get zero training error but a terrible test error.

- Instead, we must fit a less complex model, e.g. a model with fewer variables.

Regression in High Dimensions

- We usually cannot perform least squares regression to fit a model in the omics setting, because we will get zero training error but a terrible test error.
- Instead, we must fit a less complex model, e.g. a model with fewer variables.

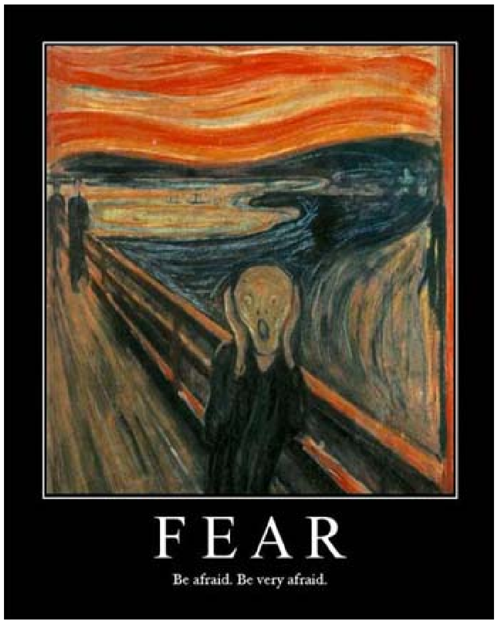
If you

- fit your model carelessly;
- do not properly estimate the test error;
- or select a model based on training set rather than test set performance;

then you will woefully overfit your training data, leading to a model that looks good on training data but will perform atrociously on future observations.

Our intuition breaks down in high dimensions, and so rigorous model-fitting is crucial.

The Curse of Dimensionality



The Curse of Dimensionality

Q: A data set with more variables is better than a data set with fewer variables, right?

The Curse of Dimensionality

Q: A data set with more variables is better than a data set with fewer variables, right?

A: Not necessarily!

Noise variables – such as genes whose expression levels are not truly associated with the response being studied – will simply increase the risk of overfitting, and the difficulty of developing an effective model that will perform well on future observations.

On the other hand, more signal variables – variables that are truly associated with the response being studied – are always useful!

Wise Words

In high-dimensional data analysis, common mistakes are simple, and simple mistakes are common.

– Keith Baggerly

Before You're Done Your Analysis

- Estimate the test error.
- Do a "sanity check" whenever possible.
  - "Spot-check" the variables that have the largest coefficients in the model.
  - Rewrite your code from scratch. Do you get the same answer again?

Fitting models in high-dimensions: one mistake away from disaster!

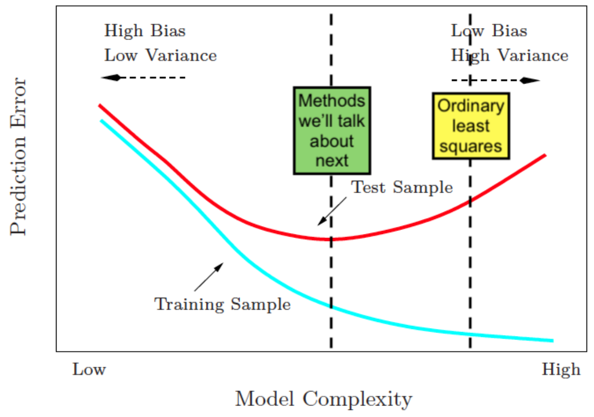
High-Dimensional Statistical Learning: Regression Methods

Jean Feng & Ali Shojaie

November 15, 2020
Sixth Seattle Symposium in Biostatistics

Linear Models in High Dimensions

- When $p \gg n$, least squares regression will lead to very low training error but terrible test error.
- Next, we'll talk about approaches for fitting linear models for high-dimensional settings.



Motivating example

- We would like to build a model to predict survival time for breast cancer patients using a number of clinical measurements (tumor stage, tumor grade, tumor size, patient age, etc.) as well as some high-dimensional biomarkers.
- For instance, these biomarkers could be:
  - the expression levels of genes
  - protein levels.
  - mutations in genes potentially implicated in breast cancer.

What are our options for analyzing high-dimensional data?

- Variable Pre-Selection
- Subset Selection
- Ridge Regression
- Lasso Regression
- ...

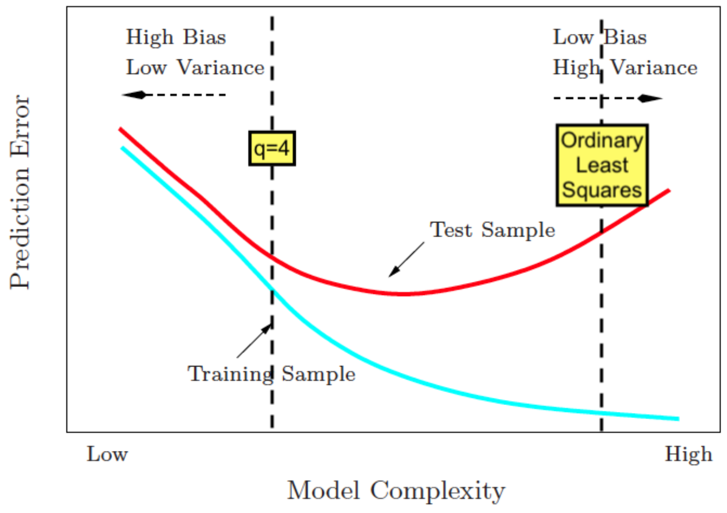
Variable Pre-Selection

A simple approach:

1. Choose a small set of variables, say the q variables that are most correlated with the response, where $q < n$.
2. Use least squares to fit a model predicting y using only these q variables.

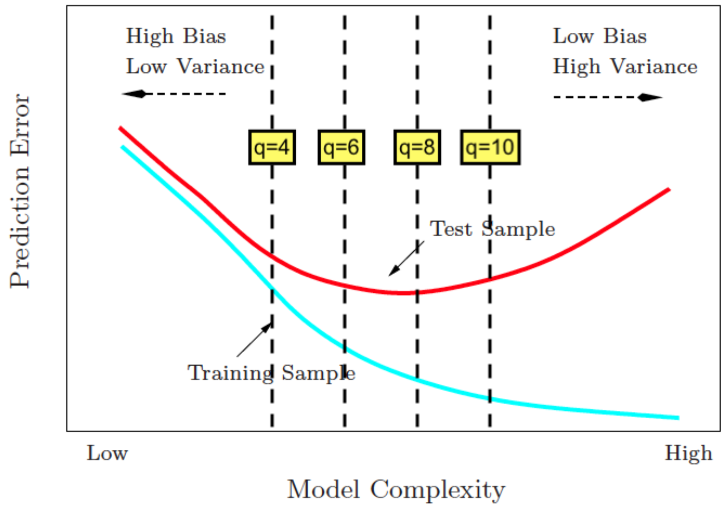
Variable Pre-Selection: Bias-Variance Trade-off

Variable pre-selection tries to find the right trade-off between the bias and variance.



Variable Pre-Selection: Bias-Variance Trade-off

Different values of q correspond to different model complexities.



How Many Variable to Use?

- **Q**: We need to choose q, the number of variables to select. How do we find the value of q that minimizes the test error?
- **A**: For a range of values of q, estimate the test error using:
  - validation set approach,
  - leave-one-out cross-validation,
  - or K-fold cross-validation.

  Choose the value of q whose estimated test error is the smallest.

Estimating the Test Error For a Given q

This is the wrong way to estimate the test error using the validation set approach:

1. Identify the q variables most associated with the response on the full data set.
2. Split the observations into a training set and a validation set.
3. Using the training set only:
    a. Use least squares to fit a model predicting y using those q variables.
    b. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
4. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Estimating the Test Error For a Given q

This is the right way to estimate the test error using the validation set approach:

1. Split the observations into a training set and a validation set.
2. Using the training set only:
    a. Identify the q variables most associated with the response.
    b. Use least squares to fit a model predicting y using those q variables.
    c. Let $\hat{\beta}_1, \ldots, \hat{\beta}_q$ denote the resulting coefficient estimates.
3. Use $\hat{\beta}_1, \ldots, \hat{\beta}_q$ obtained on training set to predict response on validation set, and compute the validation set MSE.

Finding the best subset of variables

- The variable pre-selection approach is simple and easy to implement, but it might not work well.
- Variable pre-selection finds the variables that are most correlated with the response.
- We might not have found the combination of variables that are most predictive of the response.

Subset Selection

Several Approaches:

- Best Subset Selection: Consider all subsets of predictors
  Computational intractable
- Stepwise Regression: Greedily add/remove predictors
  Heuristic and potentially inefficient
- Modern Penalized Methods

Ridge Regression and the Lasso

- Variable pre-selection and forward selection control model complexity by selecting subsets of the predictors.
- Ridge regression and the Lasso control model complexity by shrinking the regression coefficients.
- Ridge regression and the Lasso are examples of "regularization" or "penalization" methods.

Crazy Coefficients

- When $p > n$, some of the variables will be highly correlated in the training data just by chance.
- Why does correlation matter?
  - Suppose that $X_1 = X_2$ on the training data and this model is a good fit for the data:

$$\hat{y} = X_1 - 2X_2 + 3X_3.$$

  - But this model will be have the same training error!

$$\hat{y} = 100000001X_1 - 100000002X_2 + 3X_3.$$

- When there are too many variables, the least squares coefficients can get crazy!
- This craziness is directly responsible for poor test error.

A Solution: Don't Let the Coefficients Get Too Crazy

- Recall that least squares solves

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2$$

- Ridge regression solves

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2 + \lambda \underbrace{\sum_{j=1}^{p} \beta_j^2}_{\text{Ridge penalty}}$$

- The ridge penalty can be written as a squared norm:

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

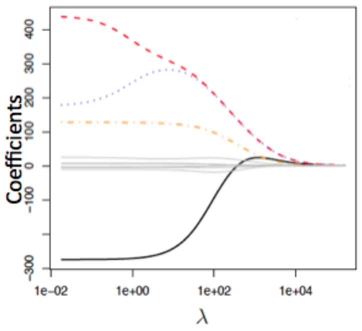
Ridge penalty parameter λ

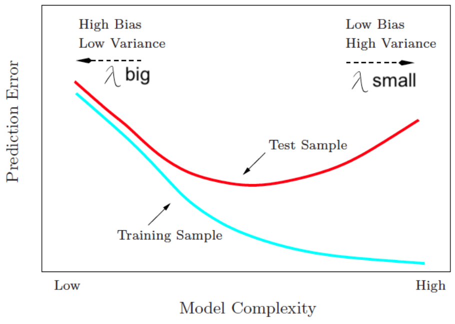
$$\hat{\beta}_\lambda^R = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2 + \lambda \|\beta\|_2^2.$$

Here λ is a nonnegative tuning parameter that shrinks the coefficient estimates.

- When $\lambda = 0$, then ridge regression is just the same as least squares.
- As λ increases, coefficients shrink towards zero.
- When $\lambda = \infty$, $\hat{\beta}_\lambda^R = 0$.

Ridge Regression As λ Varies





Selecting λ

- Perform cross-validation or the validation set approach to search over a fine grid of λ values and find the one with the smallest test error.
- Use the selected λ to perform ridge regression on the full data set.

Ridge Summary

- Using ridge regression, the final model will contain all p variables, no matter what. This works well when most of your predictors have some non-zero effect on the outcome.
- But what if you believe the outcome depends on only a small number of predictors? Or what if you want a simpler model that depends on a small number of features?

The Lasso

- The lasso involves performing a little tweak to ridge regression so that the resulting model contains mostly zeros.
- In other words, the resulting model is sparse. We say that the lasso performs feature selection.

The Lasso

- The lasso solves

$$\hat{\beta}_\lambda^L = \underset{\beta}{\text{argmin}} \sum_{i=1}^{n} (y_i - (\beta_1 X_{i1} + \cdots + \beta_p X_{ip}))^2 + \lambda \underbrace{\sum_{j=1}^{p} |\beta_j|}_{\text{Lasso penalty}}$$

- So lasso is just like ridge, except that β_j^2 has been replaced with $|\beta_j|$.
- The lasso can be written using the L1-norm:

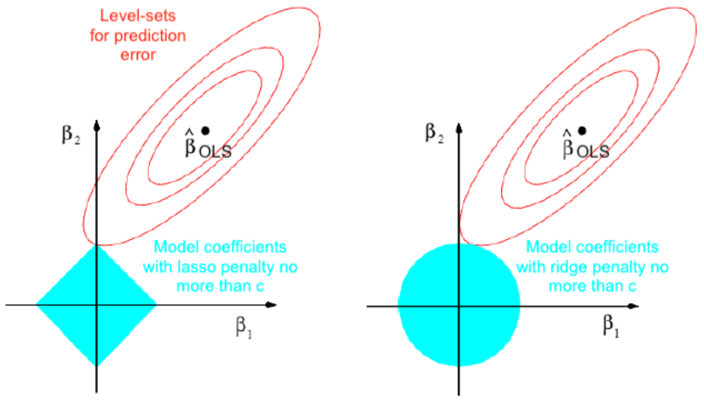
$$\|\boldsymbol{\beta}\|_1 = \sum_{j=1}^{p} |\beta_j|$$

The Lasso

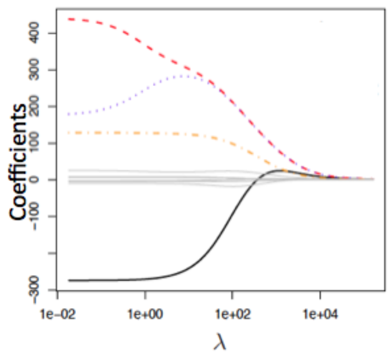
- Lasso is a lot like ridge:
  - λ is a nonnegative tuning parameter that controls model complexity.
  - When $\lambda = 0$, we get least squares.
  - When λ is very large, we get $\hat{\beta}_\lambda^L = 0$.
- But unlike ridge, lasso will give some coefficients exactly equal to zero for intermediate values of λ!

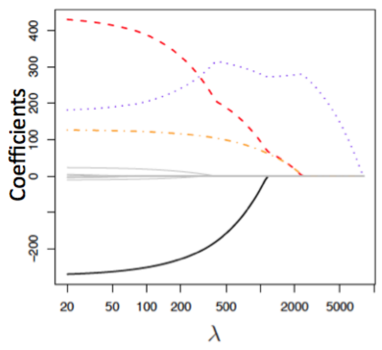
Lasso vs Ridge

- Lasso finds the best linear model such that the lasso penalty is no larger than $c > 0$.
- Ridge regression finds the best linear model such that the ridge penalty is no larger than $c > 0$.



Lasso As λ Varies


(a) Ridge


(b) Lasso

Selecting λ

- Perform cross-validation or the validation set approach to search over a fine grid of λ values and find the one with the smallest test error.
- Use the selected λ to perform the lasso on the full data set.

Pros/Cons of Each Approach

Approach	Sparsity?*	Predictions?**
Pre-Selection	Yes	Ok
Forward Stepwise	Yes	Ok
Ridge	No	Better
Lasso	Yes	Better

* Does this approach perform feature selection?
** How good are the predictions resulting from this model?

No "Best" Approach

- There is no "best" approach to regression in high dimensions.
- Some approaches will work better than others. For instance:
  - Lasso will work well if it's really true that just a few features are associated with the response.
  - Ridge will do better if all of the features are associated with the response.
- For a given dataset, how do you decide which approach to use?

Modeling non-linear relationships

What if the relationship isn't linear?

$$y = 3\sin(x) + \epsilon$$
$$y = 2e^x + \epsilon$$
$$y = 3x^2 + 2x + 1 + \epsilon$$

If we know the functional form we can still use "linear regression"

Linear regression for non-linear functions

$y = 3\sin(x) + \epsilon$:

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \begin{pmatrix} sin(x) \end{pmatrix}$$

$y = 3x^2 + 2x + 1 + \epsilon$:

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \left(\begin{array}{c|c} x & x^2 \end{array}\right)$$

Linear regression for non-linear functions

What if we don't know the right functional form?

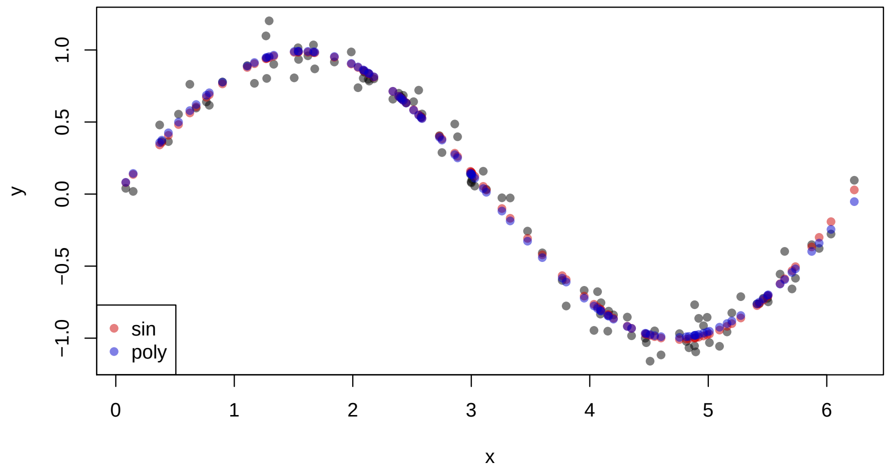
Use a flexible basis expansion $h_1, h_2, \ldots, h_k$:

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \left(\begin{array}{c|c|c|c} h_1(x) & h_2(x) & \cdots & h_k(x) \end{array}\right)$$

For example, the polynomial basis:

$$\begin{pmatrix} x \end{pmatrix} \rightarrow \left(\begin{array}{c|c|c|c} x & x^2 & \cdots & x^k \end{array}\right)$$

Example



Non-linear modeling for multiple variables

Expand each variable and use the Lasso:

$$\left(x_1 \;\middle|\; x_2 \;\middle|\; \cdots \;\middle|\; x_p \right) \rightarrow \left(x_1 \;\cdots\; x_1^k \;\middle|\; x_2 \;\cdots\; x_2^k \;\middle|\; \cdots \;\middle|\; x_p \;\cdots\; x_p^k \right)$$

Summary

- You can use variable pre-selection or subset selection to limit the number of variables you feed into ordinary least squares.
- Ridge regression protects against overfitting by shrinking model coefficients to zero.
- Lasso regression protects against overfitting by setting some model coefficients to exactly zero.
- To model non-linear relationships, expand the features using a basis expansion.

High-Dimensional Statistical Learning: Classification

Jean Feng & Ali Shojaie

November 15, 2020
Sixth Seattle Symposium in Biostatistics

Classification

- Regression involves predicting a continuous-valued response, like tumor size.
- Classification involves predicting a categorical response:
  - Cancer versus Normal
  - Tumor Type 1 versus Tumor Type 2 versus Tumor Type 3
- Just like regression,
  - Classification cannot be blindly performed in high-dimensions because you will get zero training error but awful test error;
  - Properly estimating the test error is crucial; and
  - There are a few tricks to extend classical classification approaches to high-dimensions, which we have already seen in the regression context!

Classification

- There are many approaches out there for performing classification.
- We will discuss two, logistic regression and support vector machines.

Logistic Regression

- Logistic regression is the straightforward extension of linear regression to the classification setting.
- For simplicity, suppose $y \in \{-1, 1\}$: a two-class classification problem.
- The logistic regression model is defined as

$$\Pr(Y = 1|X; \beta) = \frac{\exp(X^\top \beta)}{1 + \exp(X^\top \beta)}.$$

- Logistic regression solves the following problem:

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \log(1 + \exp(-y_i x_i^\top \beta))$$

Extending Logistic Regression to High Dimensions

1. Variable Pre-Selection
2. Forward Stepwise Logistic Regression
3. Ridge Logistic Regression

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \log(1 + \exp(-y_i x_i^\top \beta)) + \lambda \underbrace{\|\beta\|_2^2}_{\text{Ridge}}$$

4. Lasso Logistic Regression

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} \log(1 + \exp(-y_i x_i^\top \beta)) + \lambda \underbrace{\|\beta\|_1}_{\text{Lasso}}$$

Extending Logistic Regression to High Dimensions

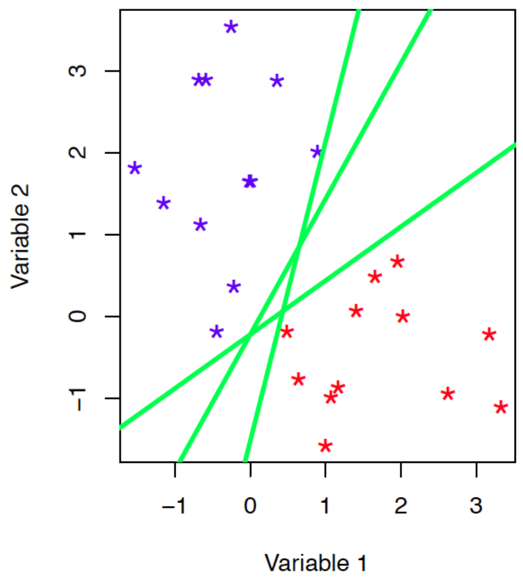
How to decide which approach is best, and which tuning parameter value to use for each approach?

- Cross-validation or validation set approach

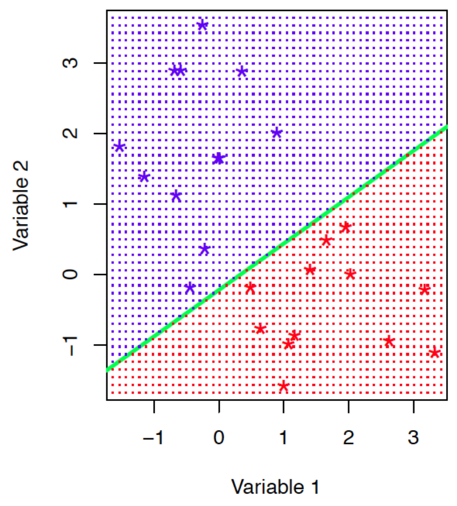
Support Vector Machines

- Suitable for binary classification.
- Unlike logistic regression, there is no probability model for the data. Instead, SVMs are geometrically motivated.
- Shares many similarities with logistic regression.

Separating Hyperplane

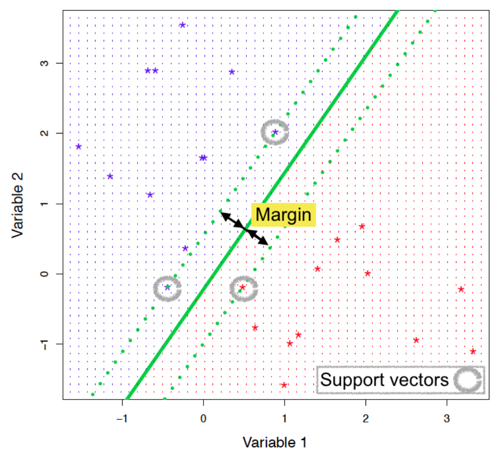


Classification Via a Separating Hyperplane



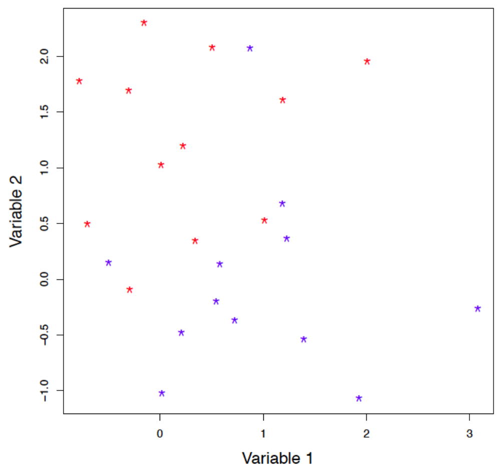
Blue class if $\beta_0 + \beta_1 X_1 + \beta_2 X_2 > c$; red class otherwise.

Maximal Separating Hyperplane

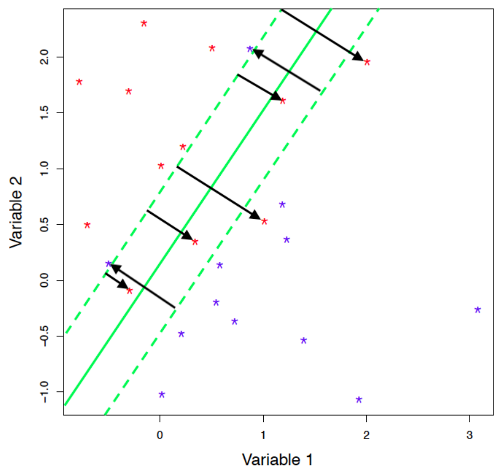


Support vector classifiers try to find a separating hyperplane that maximizes the margin. Points used to define the hyperplane are called support vectors.

What if There is No Separating Hyperplane?



Support Vector Classifier: Allow for Violations



Instead, find a hyperplane that maximizes the margin such that the total violation is below some tuning parameter $C \geq 0$.

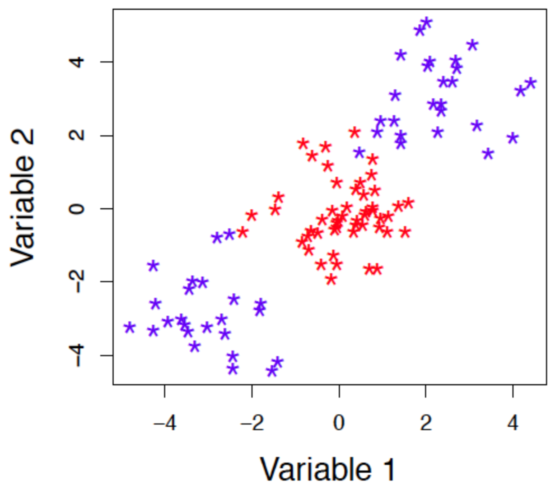
Support Vector Machine

- Support vector machines extend support vector classifiers by fitting non-linear decision boundaries using a kernel with some basis expansion $\{h_1, h_2, \ldots, h_k\}$:

$$\left(x \right) \rightarrow \left(\begin{array}{c|c|c|c} h_1(x) & h_2(x) & \cdots & h_k(x) \end{array} \right)$$

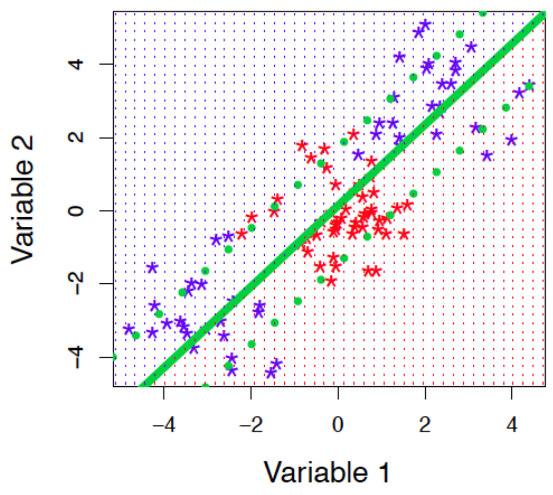
- Example kernels: Radial Basis Function (RBF) and polynomial
- Note that this "kernel trick" can also be used for linear and logistic regression as well.

Non-Linear Class Structure



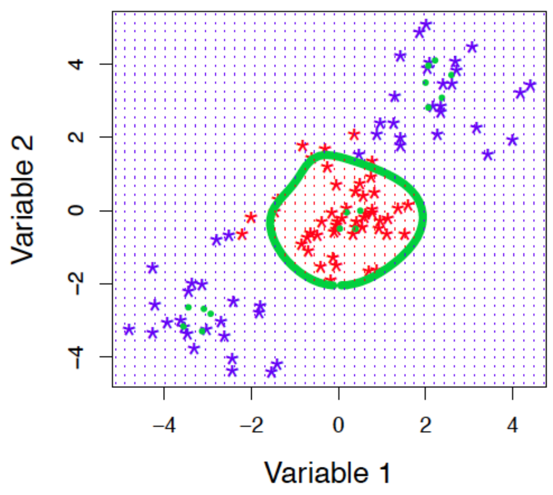
This will be hard for a linear classifier!

Try a Support Vector Classifier



Uh-oh!!

Support Vector Machine



Much Better.

Is A Non-Linear Kernel Better?

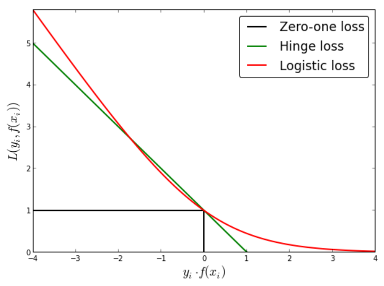
- Yes, if the true decision boundary between the classes is non-linear, and you have enough observations to accurately estimate the decision boundary.
- No, if you are in a very high-dimensional setting such that estimating a non-linear decision boundary is hopeless.

Another view of SVMs

It turns out that SVMs can be written in a similar form as logistic regression with a ridge penalty:

$$\underset{\beta}{\text{minimize}} \sum_{i=1}^{n} L(f_\beta(x_i), y_i) + \lambda \|\beta\|^2$$

- Hinge loss: $\max(0, 1 - f(x_i)y_i)$
- Logistic loss: $\log(1 + \exp(-f(x_i)y_i))$



In High Dimensions...

- In SVMs, the tuning parameter C controls the amount of flexibility of the classifier.
- Because SVMs can be written as ridge regression but with a hinge loss, tuning C is similar to tuning the penalty parameter in ridge regression. The SVM decision rule involves all p variables.
- Can get a sparse SVM using a lasso penalty; this yields a decision rule involving only a subset of the features.
- People used to claim that SVMs overcome the "curse of dimensionality". This is not true!

Assessing the Performance of Classifiers

- A binary classifier can have two types of errors: false positives and false negatives

		Predicted Class	
		$-$	$+$
True	$-$	TN	FP
Class	$+$	FN	TP

- False Positive Rate (FPR) = FP/(TN + FP), also known as 1− Specificity
- True Positive Rate (TPR) = TP/(TP + FN), also known as Sensitivity

Assessing the Performance of Classifiers

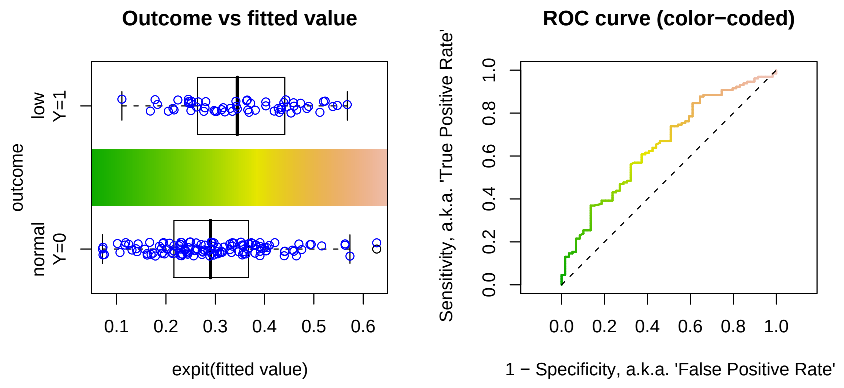
- By default, all classification methods assume that all errors have the same "cost"; i.e., FP and FN are equally costly
- However, in many applications (e.g. in cancer diagnostics) it may be more costly to have FN than FP
- We can obtain different classifiers by changing the "cutoff"...

Cutoffs for binary classifiers

- In logistic regression, we apply the cutoff to...
  - the estimated probabilities.
- In support vector classifiers, we apply the cutoff to...
  - how much above the observation is relative to the separating hyperplane.

ROC

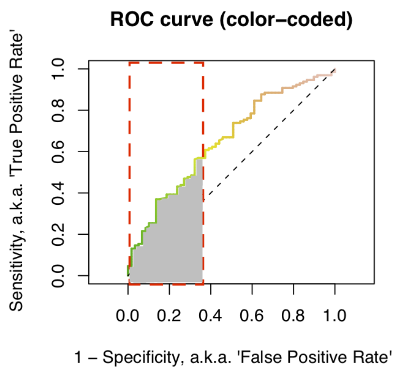
- The ROC (Receiver Operating Characteristic) curve summarizes the performance of a binary classifier over a range of decisions



- Each point on the curve represents a single decision (cutoff)
- The area under the ROC curve (AUC) measures the overall performance of a classifier (over the range of cutoffs)

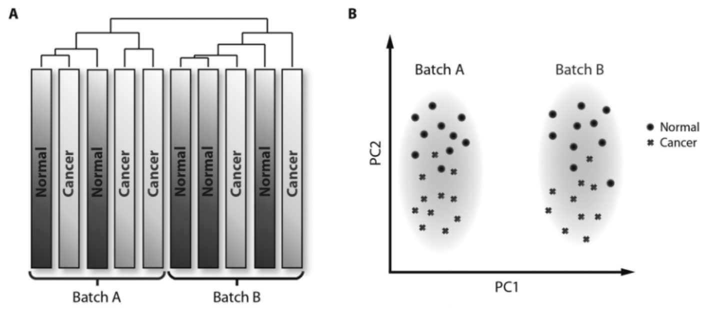
The partial AUC

- In many cases, the AUC summarizes the performance of a classifier over regions of no practical interest.
  - In diagnostic testing, we only care about the performance in the region with high TPRs.
  - In population screening, we only care about the performance in the region with low FPRs.
- The partial AUC calculates the AUC only in the region of interest.



Batch Effects

- In omics experiments, batch effects are non-biological factors, such as inter-machine or inter-lab or inter-operator variability, time of day, and day of week, that confound the relationship between the predictors and outcome of interest.
- It has been shown many times that batch effects can be much stronger than biological effects of interest!



Batch Effects: A cautionary tale

- Petricoin et. al. (2002) reported that their predictive model could use proteomic patterns to discriminate between healthy patients and those with ovarian cancer with 100% sensitivity and 95% specificity!
- However, when independent researchers looked at the publicly released data, they found:
  - inadvertent changes in protocol mid-experiment
  - difference in processing between tumor and normal samples.

Steps to Reduce Batch Effects

Good experimental design is the best strategy for reducing batch effects and avoiding faulty conclusions.

- Randomize sample run times: e.g. don't run cases first and controls second.
- Document how the data was collected (date, machine, cell culture medium, ...)
- Avoid any extraneous sources of variation, e.g. due to change in person running the experiment.

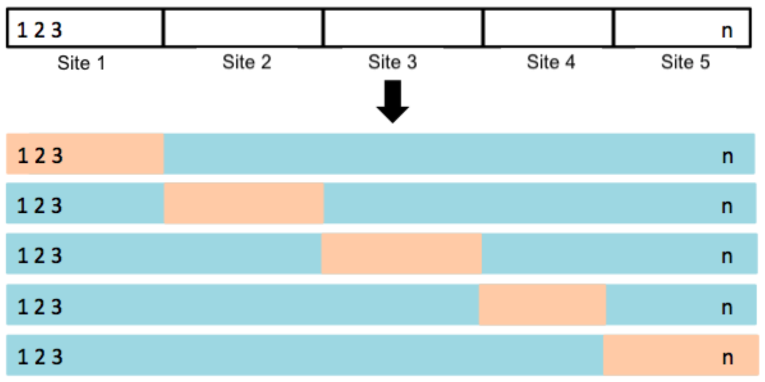
Steps to Reduce Batch Effects

When analyzing the data...

- Apply methods that adjust for potential batch effects
- Train on data from different institutions, rather than using a single data set.
- Validate results on independent data sets from a different institution.
- Make sure your data analysis is reproducible. Clearly document your code that contains *every* step, from data pre-processing to the final analysis.

Cross-Validation when combining datasets

When training models on datasets across different institutions, remember to do *site-wise* cross-validation!



Summary

- Logistic regression can be extended to high-dimensional settings using variable subset selection techniques and penalization methods.
- Support vector machines try to find a decision boundary that best separates the data.
- Be wary of batch effects! They can make you think that your classifier is much better than it really is.

High-Dimensional Statistical Learning: Tree-Based Methods

Jean Feng & Ali Shojaie

November 15, 2020
Sixth Seattle Symposium in Biostatistics

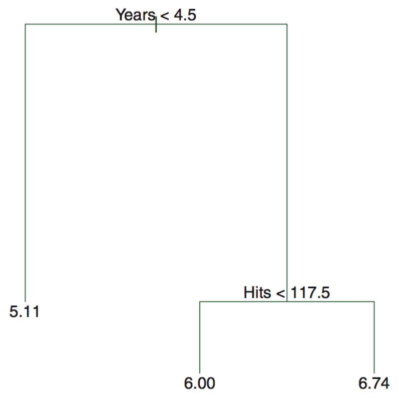
Tree-Based Methods

Tree-based methods can be used in both regression and classification tasks.

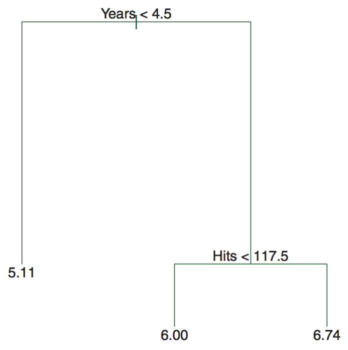
- Trees are able to model non-linearities and interactions between variables very naturally.
- The basic steps are:
  1) Stratify or segment the predictor space into small and simple regions
  2) Predict the mean (regression) or mode (classification) of observed outcomes in that segment

Toy Example: predicting salaries of baseball players I

- Outcome: Salary of baseball players (in millions)
- Two predictors: years of experience in MLB (`Years`) and number of hits made in the previous year (`Hits`)



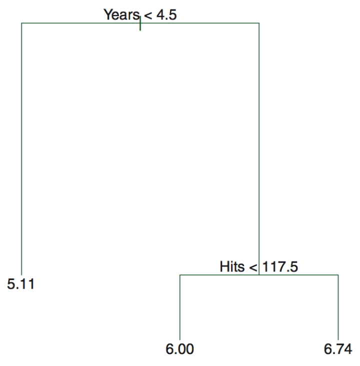
Toy Example: predicting salaries of baseball players II

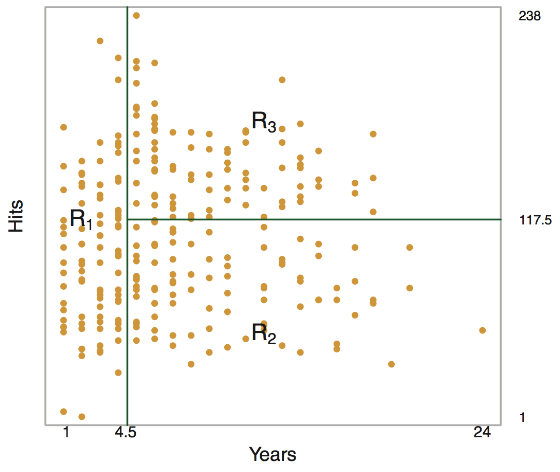


- The top split predicts that baseball players having `Years` < 4.5 earn \$5.11 million.
- Among players with > 4.5 `Years` of experience...
  - For those with < 117.5 `Hits`, the predicted salary is \$6.00 million.
  - For those with > 117.5 `Hits`, the predicted salary is \$6.74 million.

Toy Example: predicting salaries of baseball players III

The tree divides the predictor space into 3 regions





$$
\begin{aligned}
R_1 &= \{X \mid Years < 4.5\} \\
R_2 &= \{X \mid Years \geq 4.5, Hits < 117.5\} \\
R_3 &= \{X \mid Years \geq 4.5, Hits \geq 117.5\}
\end{aligned}
$$

Decision Trees: the general idea

- The *regression* tree in the above example is likely an over-simplification of the true relationship between `Hits`, `Years`, and `Salary`.
- However, it is very easy to interpret and have a nice graphical representation.

Decision Trees: the general idea

Steps for building a regression (or classification) tree:

1) Stratify/Segment: Partition the predictor space into J disjoint regions $R_1, R_2, \dots, R_J$.
   - How should we construct the regions $R_1, \dots, R_J$?
   - How many regions should there be? (How big should J be?)
2) Prediction: For observation X in region R_j, output the mean (regression) or mode (classification) of the observed outcomes for the training observations in region R_j

Partitioning the Predictor Space

For now assume J is known.

- Unfortunately, it is not possible to consider every possible partition of the space into J regions.
- We introduce two simplifications:
  1. We will only split the region into rectangles/boxes.
  2. We'll make splits in a greedy manner. We'll do this using recursive binary splitting.

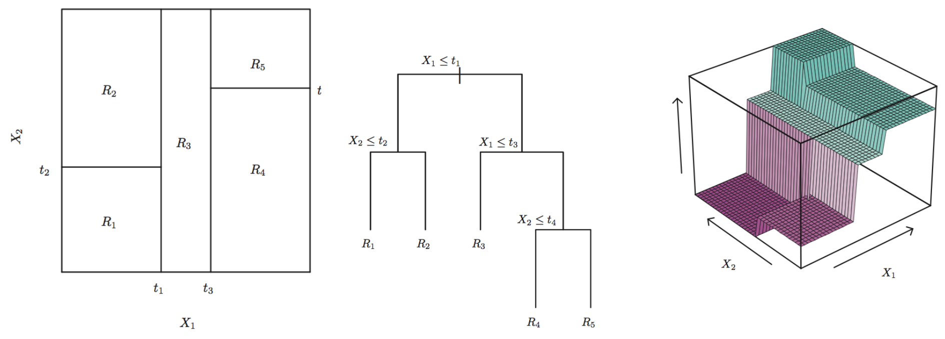
Recursive Binary Splitting

- Start with all the data.
- For all predictors X_j and cut points t:
    - Consider splitting the space into:

$$R_1(j,t) = \{X \mid X_j < t\}, \qquad R_2(j,t) = \{X \mid X_j \geq t\}$$

    - Evaluate the quality of this candidate split according to some metric (e.g. drop in mean squared error).
- Choose the best split according to this metric.
- Rinse and repeat the above process for each new region until there are J regions.

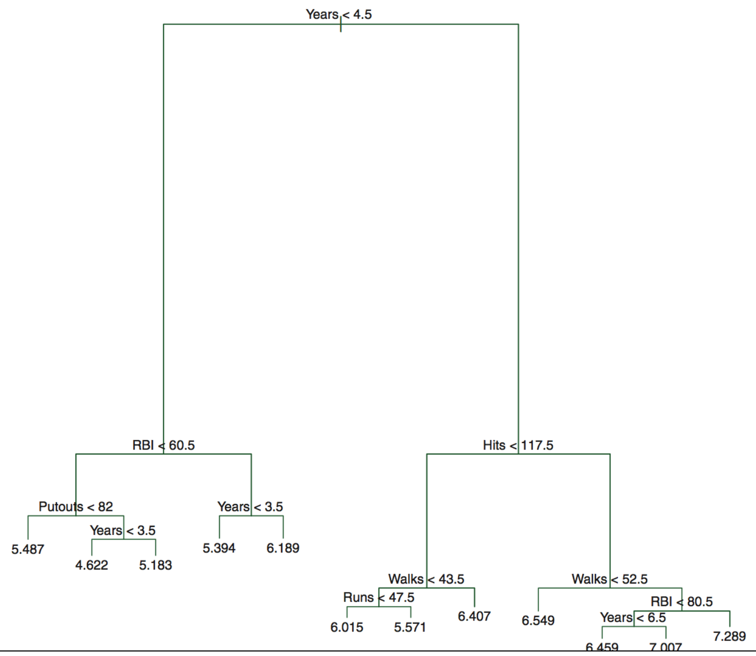
Recursive Binary Splitting



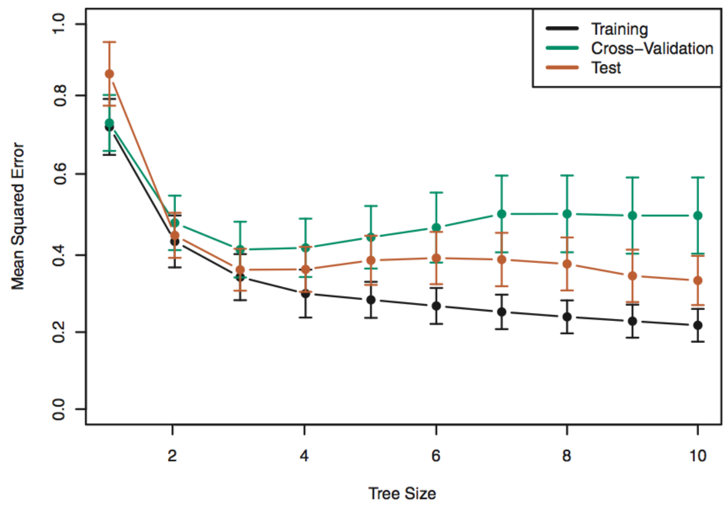
Tuning the number of regions

- The complexity of the regression tree is determined by the number of regions J.
- A tree with large J might overfit to the training data!
- A smaller tree with fewer splits might have lower variance (and better interpretability), at the cost of a little bias.
- To find a good small tree, a common approach is to:
  1. Grow a large tree, e.g. until no region has > 5 observations.
  2. Prune the tree to obtain a subtree.
  3. Use cross-validation to select J.

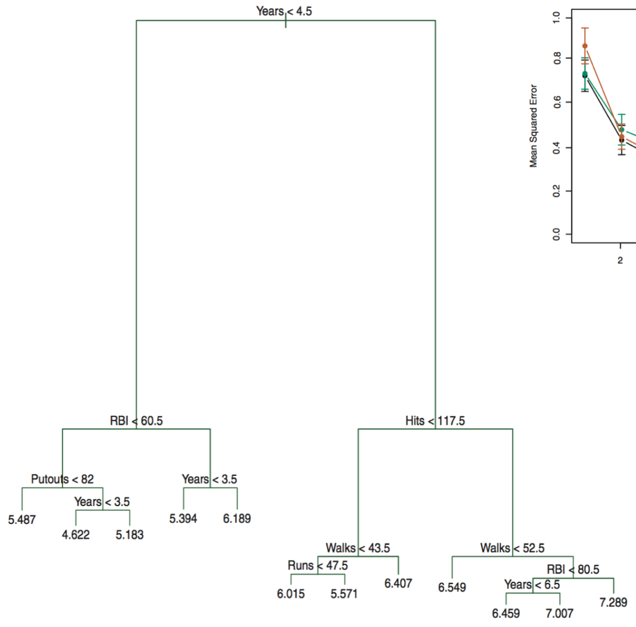
Full Tree for the `Hitters` Data Set

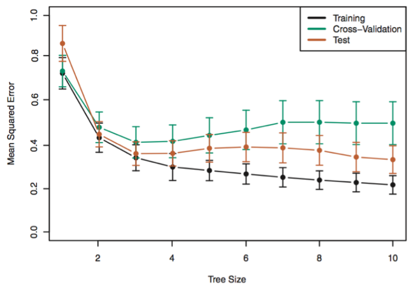


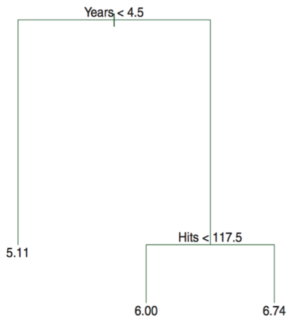
Cross-validation for the Hitters Data Set



Final tree for the Hitters Data Set



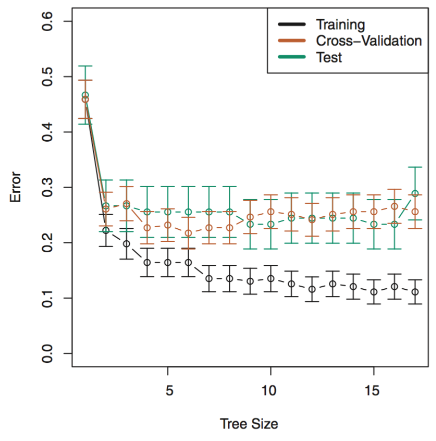


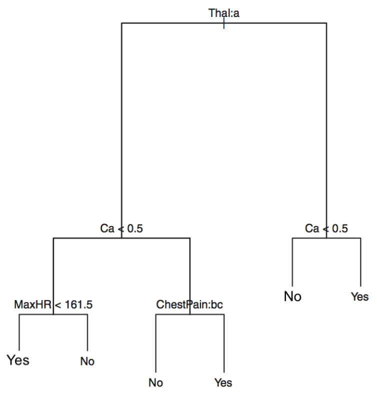


Regression vs Classification Trees

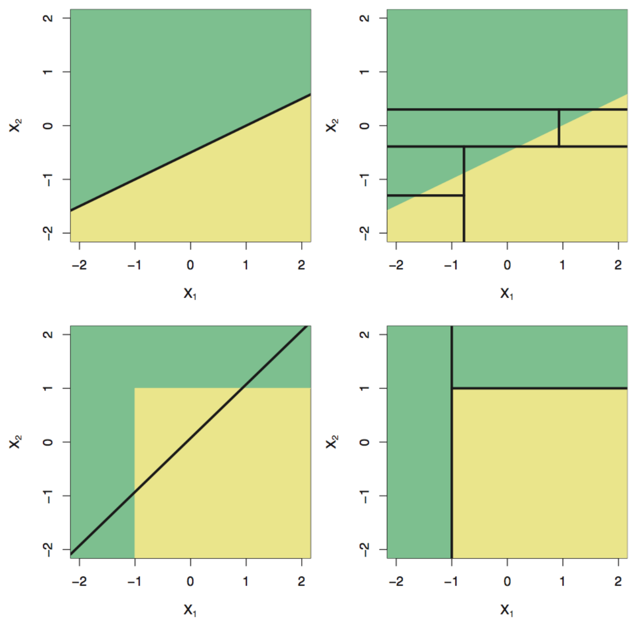
Regression	Classification
▶ Predict mean ▶ Evaluate split by improvement in mean squared error: how different are the outcomes from the mean for that split	▶ Predict mode ▶ Evaluate split by decrease in impurity: how often do outcomes differ from the mode for its split (e.g. Gini Index and Cross-Entropy)

Classification Tree for the `Heart` Data





Trees vs Linear Models



Trees: Pros and Cons

- Pros:
  - very easy to interpret and explain to others
  - can easily handle both categorical and continuous predictors
- Cons:
  - Decision trees have high variance but low bias. If we split the training data into two parts, we get very different trees.

Bagging

- Bootstrap aggregation, or bagging, is a general-purpose procedure for reducing the variance of a statistical learning method by averaging.
- We know that averaging reduces the variance:
  - Specifically, if we take the average of n independent observations $Z_1, \dots, Z_n$, each with variance σ^2, $\mathrm{Var}(\bar{Z}) = \sigma^2/n$
- Idea: What if we can average models estimated on the B datasets?

$$\hat{f}_{avg}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{b}(x)$$

Bagging: The Main Idea

- How do we make B datasets when we only have one dataset?
- We create bootstrap samples:

  For $b = 1, \dots B$: Randomly draw n of them with replacement.
- Bagging averages the models trained on B bootstrap samples:

$$\hat{f}_{bag}(x) = \frac{1}{B}\sum_{b=1}^{B} \hat{f}^{*b}(x)$$

Bagging for Trees

- For classification trees, output the average as a probability or do majority voting.
- The value of B is not critical:
  - As B increases, the bagged model will converge.
  - In practice $B = 100$ to $B = 1000$ works pretty well
- Note: Because bagging outputs the average of many trees, the bagged model is harder to interpret.

Random Forests: The Motivation

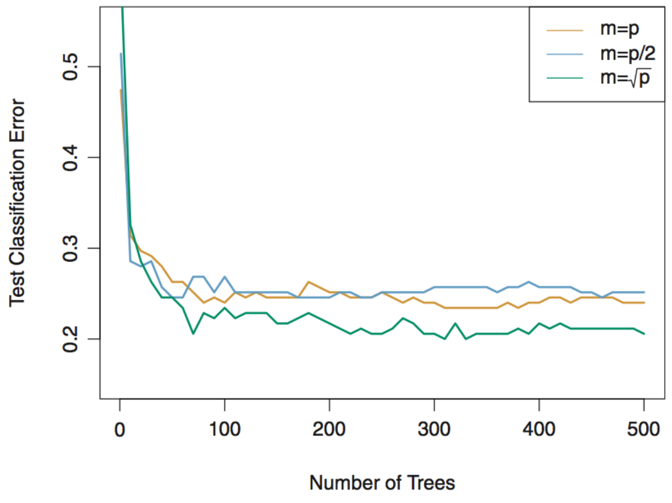
- Suppose that there is one very strong predictor in the data set, along with a number of moderately strong ones
- Then in the collection of bagged trees, most or all of the trees will use this strong predictor in the top split, and they all look somewhat similar
- This means that predictions from the bagged trees can be highly correlated.
- In this setting, bagging will only reduce the model variance slightly.

Random Forests: The Idea

Idea: Make the trees less similar.

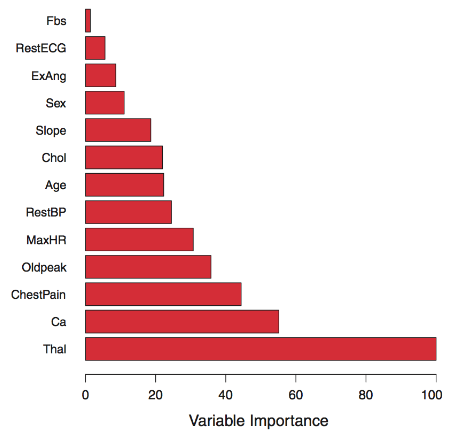
- When fitting a tree for each bootstrap sample, randomly pick *m* variables for each split.
- By randomly selecting variables to use in each tree, we "decorrelate" the trees and achieve a larger reduction in variance.
- How big should *m* be?
  - Typical choice is $m = \sqrt{p}$. Still, you should think about the bias-variance tradeoff.
  - As *m* increases, the trees become more similar.
  - If *m* is very small, the randomly selected variables will have very little predictive power and the trees will have higher bias.

Application to gene expression microarrays



Interpreting The Results

- Unfortunately, bagging and random forest result in models that are not easily interpretable.
- A very useful tool for gaining insight about individual variables is the variable importance plot



Boosting

- Boosting is another approach for improving the performance of tree-based methods
- Like bagging and random forests, boosting aggregates trees to construct a final model.
- Boosting is also a general-purpose method that can be combined with other machine learning algorithms, not just trees.

Boosting: Main Idea

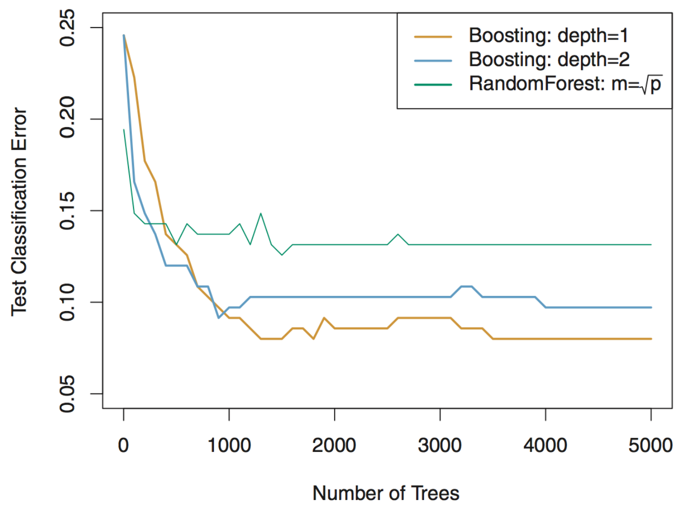
- In boosting, the trees are grown sequentially: each tree is grown using information from previously grown trees
  - Each tree is small, with just a few terminal nodes
  - Given the current model, we fit a decision tree to the residuals from the previous model as the response
  - We then add this new decision tree into the fitted function and update the residuals
- Because we are fitting *small* trees at each iteration, the training error will decrease *slowly* over time.

Tuning Parameters

Boosting has three tuning parameters that give us very fine control over the model complexity:

- The number of splits in each tree d: Controls the complexity of each tree. In practice, a single split $d = 1$ can actually work quite well.
- The number of trees B: As B increases, the complexity of our boosted model increases (unlike in bagging and random forests!)
- The shrinkage parameter λ: λ controls the rate of learning, where a smaller value means the model changes slowly.

Comparison on Gene Expression Data



Summary

- Tree-based models are easy to interpret and naturally equipped to learn interactions between predictors.
- Because single classification and regression trees have high variance, we can aggregate many trees to reduce the variance.
- Bagging and random forests reduce the variance by averaging.
- Boosting fits trees sequentially by modifying the datasets repeatedly. We tune its hyperparameters to obtain the optimal bias-variance tradeoff.

High-Dimensional Statistical Learning: Biomarkers

Jean Feng & Ali Shojaie

November 15, 2020
Sixth Seattle Symposium in Biostatistics

The Data

On each of n patients measure

y_i – outcome
(eg. tumor growth, treatment response, survival time)

x_i – p-vector of features
(eg. SNPs, gene expression values)

Biomarkers

Prognostic

Gives information on outcome independent of treatment
Generally not informative for treatment decisions
(*main effect term*)

Predictive

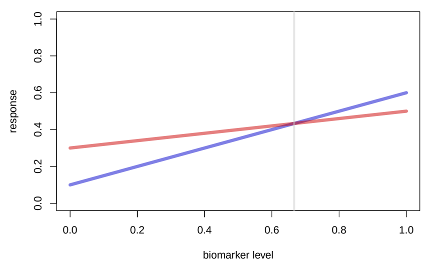
Gives information on relative effectiveness of treatments
Informative for treatment decisions
(*interaction term*)

Actionable Predictive Biomarkers

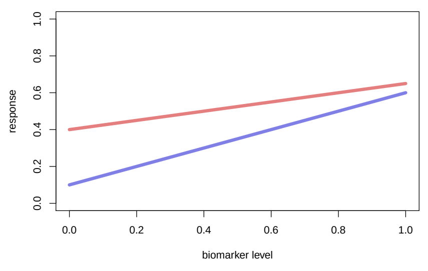
We only care about actionable (qualitative) interactions:

These are interactions which result in a crossing of response curves

Actionable



non-Actionable



Biomarkers

Prognostic

Interested in understanding

$$\mathrm{E}\left[y|x\right]$$

Predictive/Prescriptive

Interested in finding all x with

$$\mathrm{E}_T\left[y|x\right] > \mathrm{E}_C\left[y|x\right]$$

Prognostic Biomarkers

How do we characterize $\mathrm{E}\left[y|x\right]$?

Statistical Machine Learning!!

Everything developed so far aims at this problem.

Predictive Biomarkers

How do we find all x with $\mathrm{E}_T[y|x] > \mathrm{E}_C[y|x]$?

Pretty simple:

$$\text{Statistical Machine Learning} \longrightarrow \mathrm{E}_T[y|x]$$
$$\text{Statistical Machine Learning} \longrightarrow \mathrm{E}_C[y|x]$$

and finally

Machine Learning for Biomarker Development

Predictive Biomarkers

How do we find all x with $\mathrm{E}_T[y|x] > \mathrm{E}_C[y|x]$?

Pretty simple:

$$\text{Statistical Machine Learning} \longrightarrow \mathrm{E}_T[y|x]$$
$$\text{Statistical Machine Learning} \longrightarrow \mathrm{E}_C[y|x]$$

and finally

Statistical Machine. . . nevermind

$$\text{Subtraction} \longrightarrow \mathrm{E}_T[y|x] - \mathrm{E}_C[y|x] \overset{?}{>} 0$$

Predictive Biomarkers — Cox

For survival data, using Cox Model, need to be a bit more careful.

- Assumes hazard factors as

$$H(time, x) = h(time)f(x)$$

- Decisions are generally made based on $f(x)$
- shared baseline hazard, $h(time)$, often considered a nuisance

Predictive Biomarkers — Cox

However if we completely separately estimate

$$H_T(time, x) \longrightarrow \hat{h}_T(time)\hat{f}_T(x)$$
$$H_C(time, x) \longrightarrow \hat{h}_C(time)\hat{f}_C(x)$$

then

- no longer *shared* baseline hazard: $\hat{h}_T(time) \neq \hat{h}_C(time)$
- $h(time)$ is no longer a nuisance
- $\hat{f}_T(x)$ and $\hat{f}_C(x)$ are no longer comparable

Predictive Biomarkers — Cox

Instead estimate a single $h_{shared}(time)$:

$$\begin{bmatrix} H_T(time, x) \\ H_C(time, x) \end{bmatrix} \longrightarrow \begin{bmatrix} \hat{h}_{shared}(time)\hat{f}_T(x) \\ \hat{h}_{shared}(time)\hat{f}_C(x) \end{bmatrix}$$

and make inference on

$$\hat{f}_T(x) \text{ vs } \hat{f}_C(x)$$

Evaluating Predictive Biomarkers

What are we not particularly interested in?

- How well we estimate $\mathrm{E}_T[y|x]$ and $\mathrm{E}_C[y|x]$
- Whether or not there is a general *shape* difference between $\mathrm{E}_T[y|x]$ and $\mathrm{E}_C[y|x]$.

Evaluating Predictive Biomarkers

What are we particularly interested in?

How well does treatment perform in the identified subgroup?

Specifically, we care about

$$\int_{\text{identified subgroup}} \left(\mathrm{E}_T\left[y|x\right] - \mathrm{E}_C\left[y|x\right]\right)$$

the average treatment effect of the indicated population.

Finding something is better than finding nothing!

In an ideal world, we would find a perfect characterization of who does/doesn't benefit from treatment.

In the real world, with a complex set of covariates, this is unreasonable.

If treatment is not effective on average in the entire population, or the effect is too small to find, then finding any subset for which there is benefit, is pretty darn useful!

Evaluating Predictive Biomarkers

As in all high-dimensional settings,

we cannot build and evaluate models on the same data.

How do we evaluate average treatment effect then?

Two options:

- Sample splitting
- Cross validation

Sample Splitting

1. Split data in training and validation set
2. Build models for $\mathrm{E}_T[y|x]$ and $\mathrm{E}_C[y|x]$ on training data
3. For each observation in validation data, calculate a biomarker score:
$$Z_i = \hat{\mathrm{E}}_T[y|x_i] - \hat{\mathrm{E}}_C[y|x_i]$$
4. For the subset predicted to benefit from treatment ($Z_i > 0$), run a test comparing treatment to control in the validation data.

In clinical trials, known as “adaptive signature design”

Cross Validation

1. Split data into folds: $fold_1, \ldots, fold_K$.
2. Cycle through the folds; for each k train and evaluate:

$$folds_{-k} \longrightarrow \begin{pmatrix} \hat{\mathrm{E}}_T\left[y|x\right] \\ \hat{\mathrm{E}}_C\left[y|x\right] \end{pmatrix}$$

$$\text{apply} \begin{pmatrix} \hat{\mathrm{E}}_T\left[y|x\right] \\ \hat{\mathrm{E}}_C\left[y|x\right] \end{pmatrix} \text{ to } folds_k \longrightarrow Z_{1,k}, \ldots, Z_{n_k,k}$$

3. Have a cross-validated Z_i for each observation
4. For obs with $Z_i > 0$, compare treatment to control.

Testing in Cross Validation

- Cannot directly run a t-test on obs with $Z_i > 0$ when comparing treatment to control.
- Because of cross-validation, empirical variance is wrong!
- Use permutation test instead.

Permutation Test

1. Run CV-procedure and calculate a t-statistic, T, for obs with $Z_i > 0$
2. For $b = 1, \ldots, B$:
   - 2.1 Permute the treatment labels
   - 2.2 Run the CV procedure on permuted data to get permuted scores, $Z_i^{(b)}$
   - 2.3 Calculate a t-statistic, T_b, comparing permuted treatment to control for those with $Z_i^{(b)} > 0$
3. Compare T to empirical distribution of $\{T_b\}_{b=1}^{B}$

In clinical trials, known as “cross-validated adaptive signature design”