Galvanize DSI
Review Session

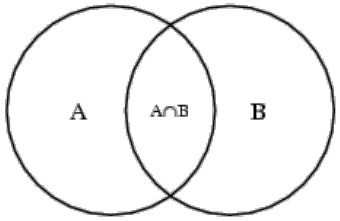
Winter 2015

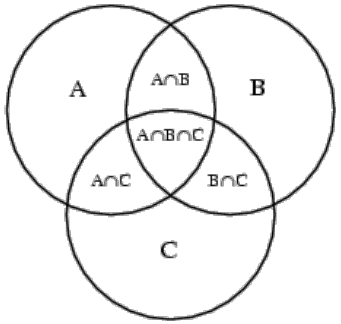
Overview – Part I

- Probability/Statistics
  - Bayes Rule, Permutations, Combinations
  - Expectation, Variance, Covariance, Correlation
  - Hypothesis Testing
  - Statistical Distributions
- Regression
  - Linear, Logistic, Lasso, Ridge
- Modeling Framework
  - Cross-validation, Feature Reduction, Bias-Variance Tradeoff
- Trees
  - Decision Trees, Bagging, Random Forests, Boosting

Probability

- Conditional Probability

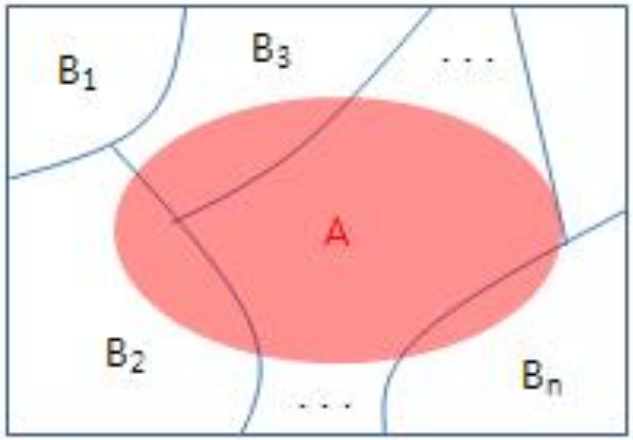




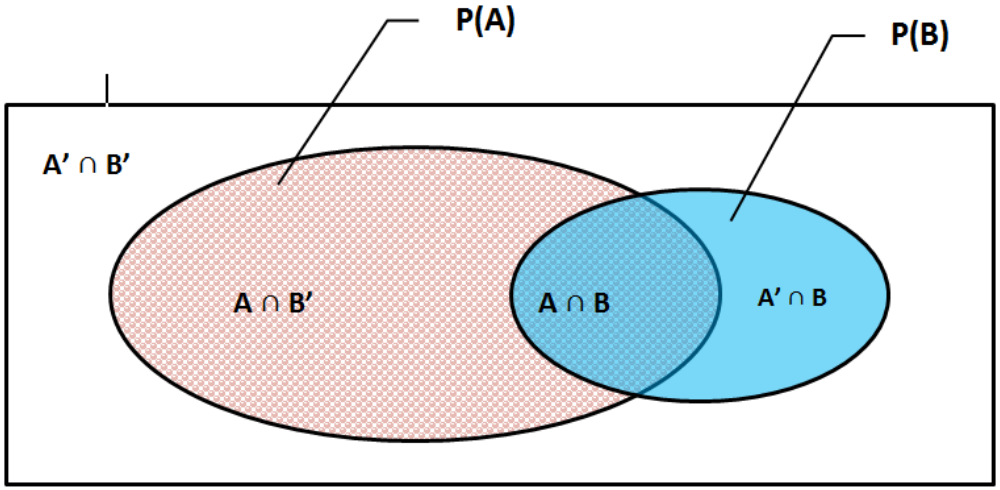
- Bayes Rule & Law of Total Probability

$$P[A] = \sum_{i=1}^{n} P[A \mid B_i] P[B_i]$$

$$P(A \mid B) = \frac{P(B \mid A) P(A)}{P(B)}$$



Bayes Theorem



$$P(B|A) = \frac{P(A|B) * P(B)}{P(A)}$$

$$P(A|B) = \frac{P(B|A) * P(A)}{P(B)}$$

$$Recall : P(A \cap B) = P(B) * P(A|B) = P(A) * P(B|A)$$

- Out of the students in a class, 60% are geniuses , 70% love chocolate, and 40% fall into both categories. Probability that student is neither genius nor chocolate lover?

- You have a 0.1% chance of picking up a coin with both heads and a 99.9% chance that you pick up a fair coin.

- You flip the coin and it comes up 10 heads in a row! What's the chance that you picked up the fair coin, given what you observed?

Permutations & Combinations

- Permutations
  - # of **unique** ways of choose k out of n = n!/(n-k)!
  - Ex. 3 out of 10? 10!/7! = 10*9*8

- Combinations
  - # of ways of choose k out of n = n!/[(n-k)!*k!]
  - Ex. 3 out of 10? 10!/(7!*3!)
    - the 3! "removes" the duplicates

- 2 Uber and 3 Lyft cars are called at the same time. Arrival times are iid.

- What's the probability that the 2 Ubers arrive before the 3 Lyfts?

Q: 2 Uber and 3 Lyft cars are called at the same time. Arrival times are iid. What's the probability that the 2 Ubers arrive before the 3 Lyfts?

(i) $\underline{U}\ \underline{U}\ \_\ \_\ \_ = (2/5)*(1/4) = 1/10$

(ii) Permutations: $(2!*3!)/5! = 1/10$

- Think of each car as unique (U_1 U_2 L_1 L_2 L_3)
- Then there are 5! unique orderings
- Of those 5! unique orderings, 2!*3! result in 2 Uber before 3 Lyft

(iii) Combinations: $1/({}_5C_2) = 1/10$

- Don't think of cars as unique among types (U U L L L)
- Then there are 5 choose 2 total orderings, ${}_5C_2$
- Of those ${}_5C_2$ orderings, only 1 results in 2 Uber before 3 Lyft

Expectation

- Discrete $\mathrm{E}[X] = x_1 p_1 + x_2 p_2 + \cdots + x_k p_k \ .$

- Continuous: Same idea, except replace Σ with integral, and replace probabilities with probability densities

$$\mathrm{E}[X] = \int_{-\infty}^{\infty} x f(x)\,\mathrm{d}x.$$

Variance

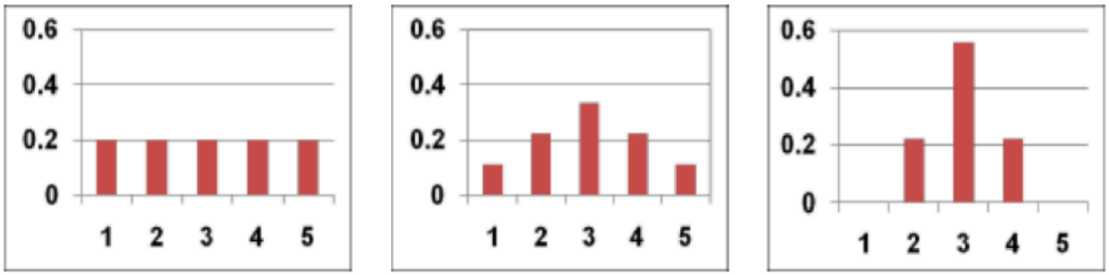
Measures how much "spread" there is in a set of numbers

$$\operatorname{Var}(X) = \mathrm{E}\left[(X - \mu)^2\right]$$

Discrete: $\operatorname{Var}(X) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu)^2$ ← for set of *n* equally likely outcomes

Discrete: $\operatorname{Var}(X) = \sum_{i=1}^{n} p_i \cdot (x_i - \mu)^2 = \sum_{i=1}^{n}(p_i \cdot x_i^2) - \mu^2$

Continuous: $\operatorname{Var}(X) = \sigma^2 = \int (x-\mu)^2 f(x)\,dx = \int x^2 f(x)\,dx - \mu^2$



Covariance

- Covariance measures how much two random variables change together

$$Cov(X,Y) = E[(X - E(X))(Y - E(Y))]$$

- Correlation is the normalized version of the correlation coefficient
  - Measures strength of linear relationship between two variables.

$$\rho_{X,Y} = \operatorname{corr}(X,Y) = \frac{\operatorname{cov}(X,Y)}{\sigma_X \sigma_Y} = \frac{E[(X - \mu_X)(Y - \mu_Y)]}{\sigma_X \sigma_Y},$$

Frequentist vs. Bayesian?

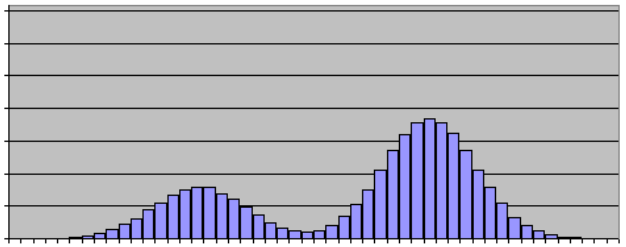
- Frequentist – Interpret a probability as a statistical average across many independent realizations (law of large numbers)

- Bayesian – Interpret probability as degree of belief
  - Takes into account our prior knowledge.
  - Prior probability distribution updated into posterior distribution as we get more and more data.

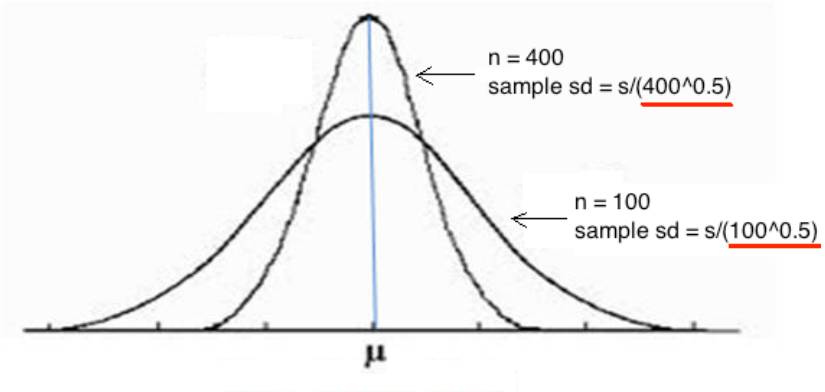
Central Limit Theorem

- Given certain conditions, the **mean** of a sufficiently large number of i.i.d. random variables, will be approximately normal, ***regardless*** of the underlying distribution.

Underlying distribution



draw i.i.d. samples and average them



Central Limit Theorem

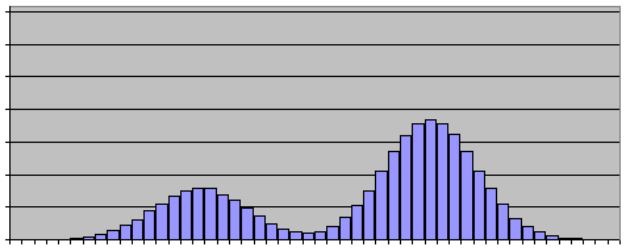
- Not only is the sample mean normally distributed, we have….

$$\bar{X} \sim Normal(\mu, \frac{\sigma^2}{n})$$

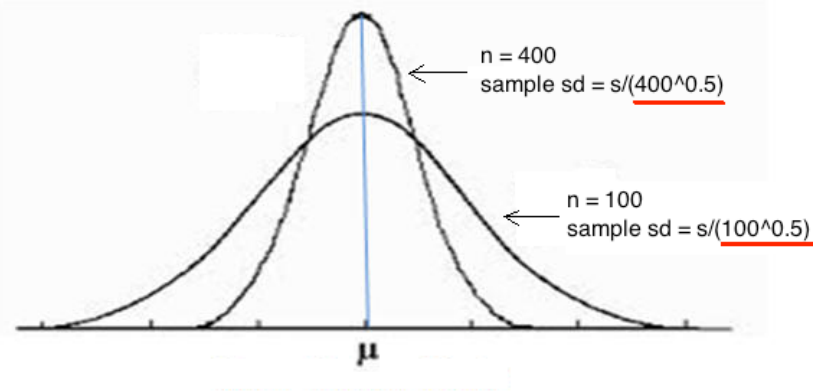
- And as usual, from any normally distributed random variable, we can derive a standard normal variable. In this case…

$$Z = \frac{\bar{X} - \mu}{\sigma / \sqrt{n}}$$

Underlying distribution



draw i.i.d. samples and average them



Hypothesis Testing – Framework

1. State null hypothesis (H_0) and alternative hypothesis (H_A)

   H_0: $\mu = 100$

   H_A: $\mu \neq 100$

2. Choose significance level, α

   $\alpha = 0.05$

3. Compute appropriate test statistic using collected data.

$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

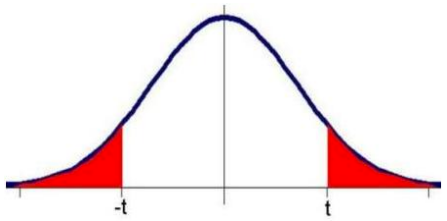
where $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}.$

⟹

n=30
Sample mean ($\bar{x}$)= 102
Sample standard deviation (s) = 7

t = (102-100)/(7/(30^.5)) = **1.565**

4. Compute p-value based on test statistic



Since we are doing a two-sided test, we take area to left of t = -1.565 and right of t = +1.565

p-value ≈ 0.1284 > 0.05 ⟹ Fail to reject null H_0

Hypothesis Testing – Framework

- Notice we "fail to reject the null", H_0: μ = 100
  - We don't conclude that μ = 100!
- What if all sample stats were the same, except instead, n = 100??

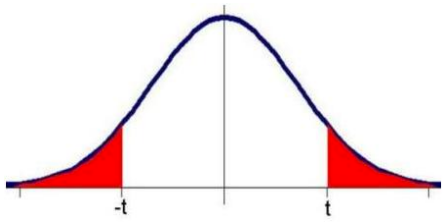
$$t = \frac{\overline{x} - \mu_0}{s/\sqrt{n}}$$

where $s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2}.$

⇒

n=**100**
Sample mean ($\overline{\mathbf{x}}$)= 102
Sample standard deviation (*s*) = 7

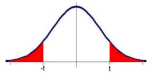
t = (102-100)/(7/(**100**^.5)) = **2.857**



Since we are doing a two-sided test, we take area to left of t = -2.857 and right of t = +2.857

p-value ≈ 0.0052 < 0.05 ⇒ Reject null H_0 in favor of H_A

Type I and Type II Errors

- p-value is **"the probability of observing the data we observed, or more extreme, given the null hypothesis is true"**
- Conceptually, we simply look at the tail end(s) of what we might expect for the distribution of the sample mean, under the null hypothesis. But we could always in earnest, be wrong, due to random variation. We're willing to accept this α of the time. 

	H_0 is true	H_0 is false
Accept H_0	Correct Decision $(1-\alpha)$	Type II Error (β)
Reject H_0	Type I Error (α)	Correction Decision $(1-\beta)$

Two-sample Comparison of Means

1. State null hypothesis (H_0) and alternative hypothesis (H_A), let α=0.05

   H_0: $\mu_1 = \mu_2$

   H_A: $\mu_1 > \mu_2$

2. Compute appropriate test statistic using collected data.

$$t = \frac{\overline{X}_1 - \overline{X}_2}{s_{\overline{X}_1 - \overline{X}_2}}$$

where $s_{\overline{X}_1 - \overline{X}_2} = \sqrt{\frac{s_1^2}{n_1} + \frac{s_2^2}{n_2}}$.

$\Longrightarrow$

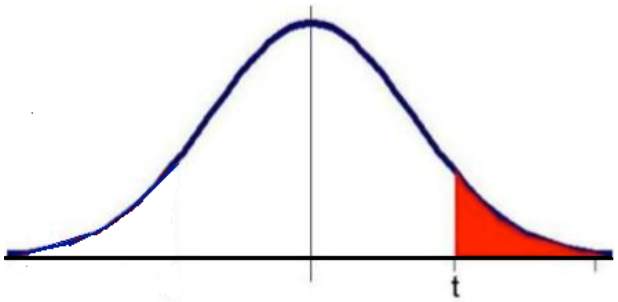
n_1=20, n_2=30

$\bar{\mathbf{x}}_1$= 101, $\bar{\mathbf{x}}_2$= 95

s_1=7, s_2=5

t = (101-95)/((49/20+25/30)^.5) = **3.311**

4. Compute p-value based on test statistic



Since we are doing a **one**-sided test, we take area to right of t = +3.311

p-value ≈ 0.00116 < 0.05 $\Longrightarrow$ Reject null H_0 in favor of H_A

Two-sample Comparison of Proportions

1. State null hypothesis (H_0) and alternative hypothesis (H_A), let α=0.05

   H_0: $p_1 = p_2$

   H_A: $p_1 > p_2$

2. Compute appropriate test statistic using collected data.

$$Z = \frac{(\hat{p}_1 - \hat{p}_2) - 0}{\sqrt{\frac{\hat{p}_1(1-\hat{p}_1)}{n_1} + \frac{\hat{p}_2(1-\hat{p}_2)}{n_2}}}$$

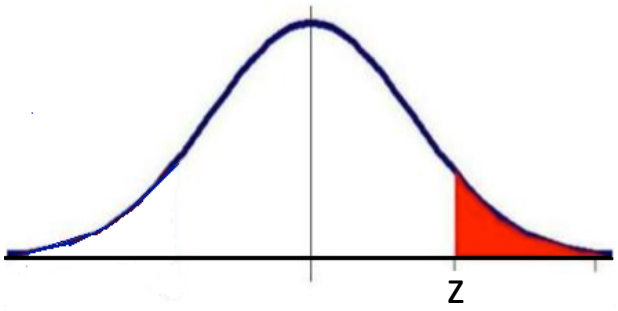
$\Longrightarrow$

n_1=300, n_2=1000

$\hat{p}_1$ = 0.05, $\hat{p}_2$ = 0.03

z = (0.05-0.03)/
((0.05*0.95)/300+(0.03*0.97)/1000)^0.5
= 1.46

3. Compute p-value based on test statistic

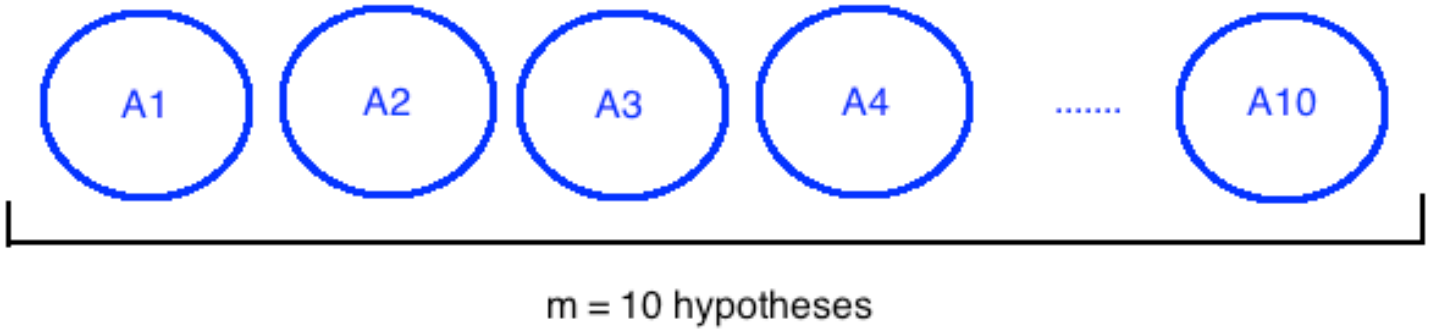


Since we are doing a **one**-sided test, we take area to right of z = +1.46

p-value ≈ 0.072 > 0.05 $\Longrightarrow$ Fail to reject null H_0

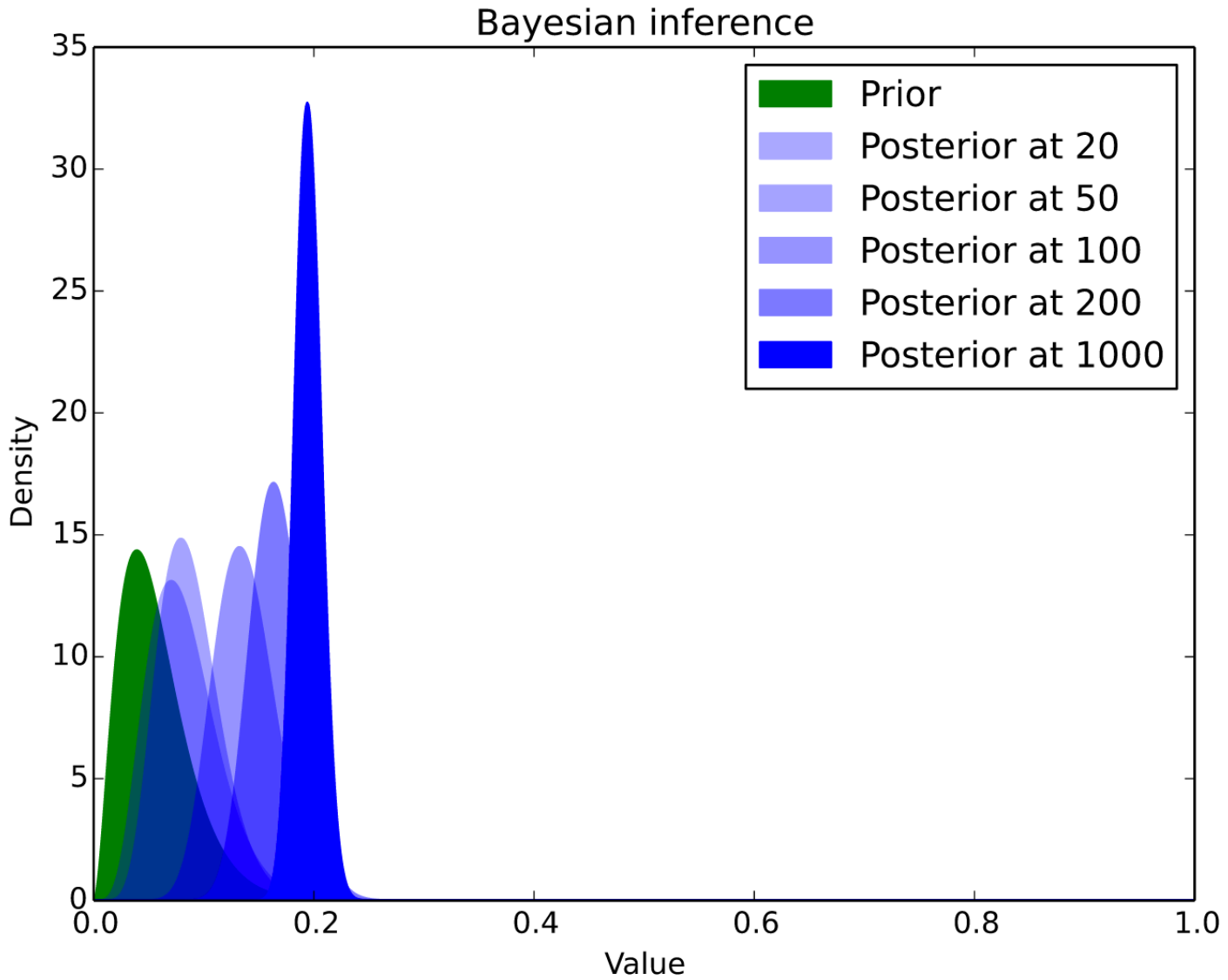
Multiple Comparisons Problem – Bonferroni Correction

- In pictures…suppose we test **10** hypotheses, so that m=10

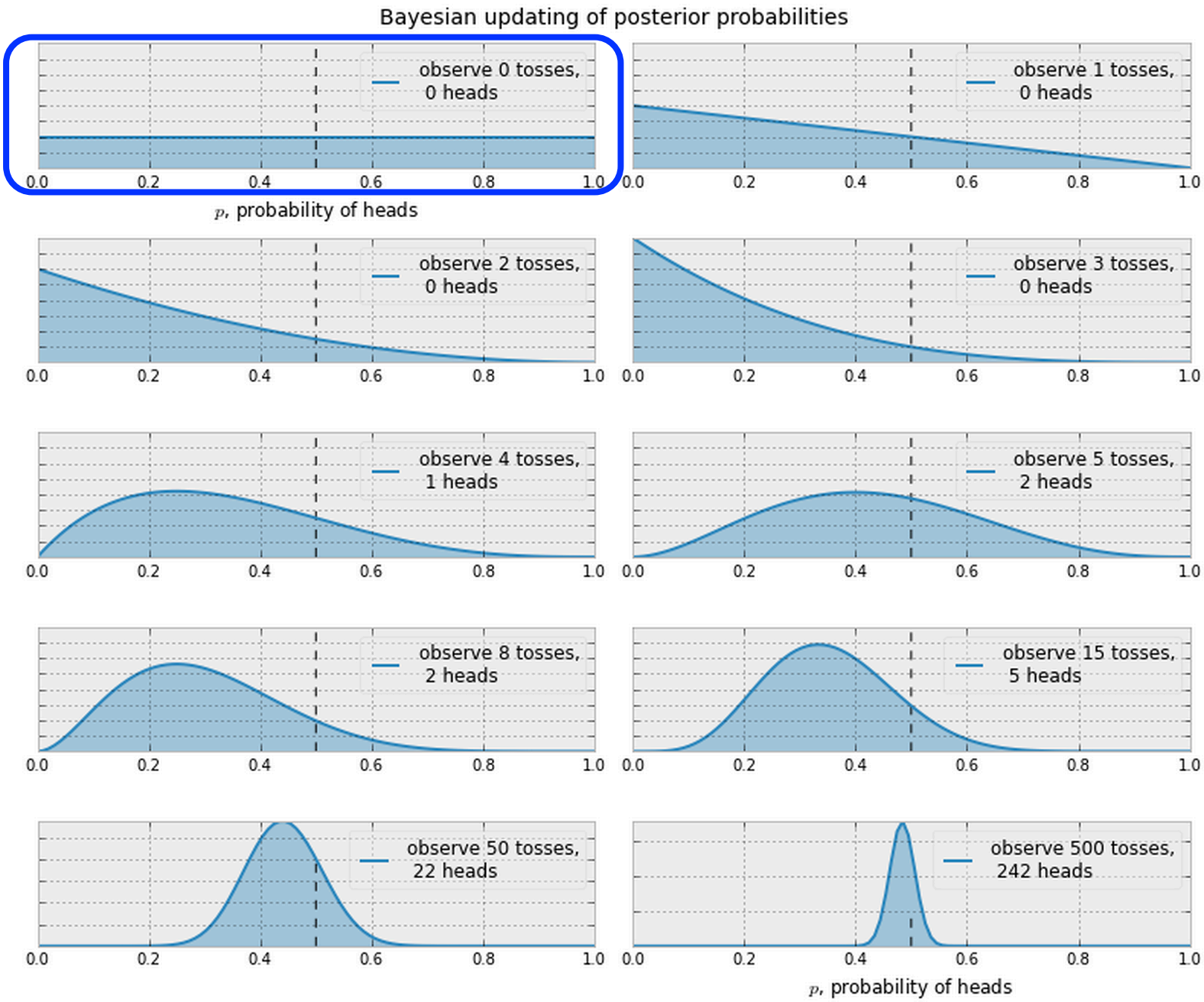


- If $\alpha = 0.05$, we want overall Type I error to be bounded by 5%
- In the worst case scenario, it's conservative to measure each hypothesis against an adjusted significance level, **0.05/10 = 0.005.**

Hypothesis Testing - Bayesian



Prior


Bayesian updating of posterior probabilities
observe 0 tosses, 0 heads
observe 1 tosses, 0 heads
observe 2 tosses, 0 heads
observe 3 tosses, 0 heads
observe 4 tosses, 1 heads
observe 5 tosses, 2 heads
observe 8 tosses, 2 heads
observe 15 tosses, 5 heads
observe 50 tosses, 22 heads
observe 500 tosses, 242 heads
0.0
0.2
0.4
0.6
0.8
1.0
p, probability of heads

Statistical Distributions

Distribution	PDF or PMF	Mean	Variance
$Bernoulli(p)$	$\begin{cases} p, & \text{if } x = 1 \\ 1-p, & \text{if } x = 0. \end{cases}$	p	$p(1-p)$
$Binomial(n,p)$	$\binom{n}{k} p^k (1-p)^{n-k}$ for $0 \leq k \leq n$	np	npq
$Geometric(p)$	$p(1-p)^{k-1}$ for $k = 1, 2, \ldots$	$\frac{1}{p}$	$\frac{1-p}{p^2}$
$Poisson(\lambda)$	$e^{-\lambda}\lambda^x / x!$ for $k = 1, 2, \ldots$	λ	λ
$Uniform(a,b)$	$\frac{1}{b-a}$ $\forall x \in (a,b)$	$\frac{a+b}{2}$	$\frac{(b-a)^2}{12}$
$Gaussian(\mu, \sigma^2)$	$\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$	μ	σ^2
$Exponential(\lambda)$	$\lambda e^{-\lambda x}$ $x \geq 0, \lambda > 0$	$\frac{1}{\lambda}$	$\frac{1}{\lambda^2}$

- X ~ Bernoulli(p) = Single coin flip turns out to be Heads
- X ~ Binomial(100, p) = # of coin flips out of 100 that turn out to be Heads
- X ~ Geometric(p) = # of Trials until coin flip turns out to be Heads

- X ~ Poisson(λ=10) = # of taxis passing a street corner in a given hour (on avg 10/hr)
- X ~ Exponential(λ=10) = Time until taxi will pass street corner

- X ~ Uniform(0,360) = Degrees between hour hand and minute hand
- X ~ Gaussian(100, 10) = IQ Score

Bernoulli(p), Binomial(n,p), Geometric(p)

- Will the **1** coin flip turn out to be heads? 

$$\begin{aligned} P(X=1) &= 1 - P(X=0) \\ &= 1 - q \\ &= p \end{aligned}$$

- How many of the **n** coin flips will turn out to be heads?

$$P(X=k) = \binom{n}{k} p^k (1-p)^{n-k}, \qquad k = 0, 1, \ldots, n$$

- How many trials until coin flip turns out to be heads?

$$\Pr(X=k) = (1-p)^{k-1} p \qquad k = 1, 2, 3, \ldots.$$

<u>Link between Bernoulli and Binomial</u>

If $X_1, \ldots, X_n$ are independent Bernoulli(p) trials…

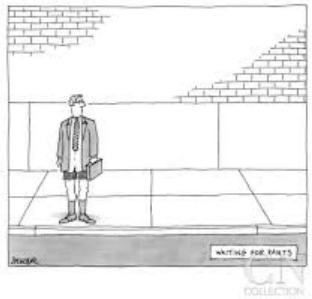
$$Y = \sum_{i=1}^{n} X_i \sim Binomial(n, p)$$

Poisson(λ)

- Number of events occurring in fixed interval of time or space.
  - Events occur at some average rate, λ, and independently of time since last event.

- Ex. Number of taxis passing a street corner in a given hour. On average there are 10/hour.

$$P(X = k) = \frac{\lambda^k * e^{-\lambda}}{k!} \qquad E(X) = \lambda$$



- Website gets 2 users per minute on average.

  What is the probability of no users in 5 minutes?

Exponential(λ)

- Time between Poisson events
  - Note because Poisson events occur continuously and independently, Exponential is ”memoryless”

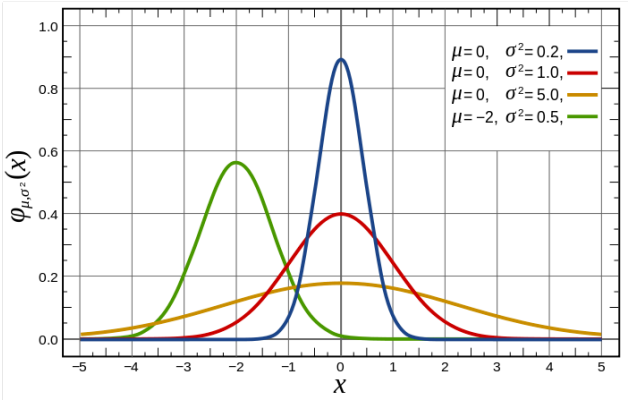
- Ex. Time until taxi arrives at street corner for pickup. On average there are 10/hour.

$$f(x) = \lambda e^{-\lambda x}, \quad x \geq 0 \qquad E(X) = \frac{1}{\lambda}$$

  - Why memoryless? Well even if a taxi just went by 2 minutes ago, assuming taxis are arriving continuously and independently, that doesn’t affect your expected wait time still.

Gaussian(μ,σ), or Normal(μ,σ)

- Very commonly occurring distribution shaped like a bell curve.



- Related to **Central Limit Theorem**,

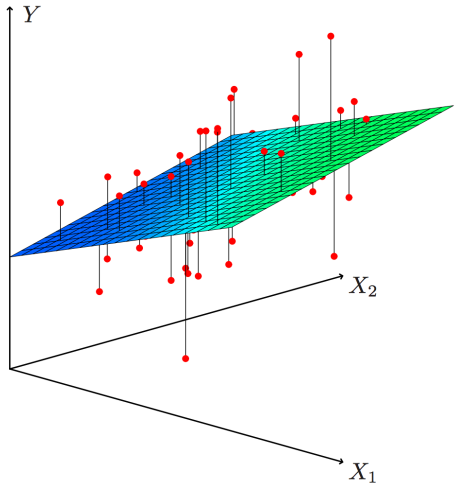
- Normal(μ,σ) is just a stretched and shifted Standard Normal(0,1).
  - If X~Normal(μ,σ) and Z~Normal(0,1),

    $X = Z*\sigma + \mu$ and $Z = (X - \mu)/\sigma$

Linear Regression

Model

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$



Fitted Value

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x_1 + \hat{\beta}_2 x_2 + \cdots + \hat{\beta}_p x_p$$

Residual Sum of Squares

$$\begin{aligned} \text{RSS} &= \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 \\ &= \sum_{i=1}^{n} (y_i - \hat{\beta}_0 - \hat{\beta}_1 x_{i1} - \hat{\beta}_2 x_{i2} - \cdots - \hat{\beta}_p x_{ip})^2 \end{aligned}$$

Coefficient Estimates

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}$$

Assessing Accuracy

Residual Sum of Squares

$$RSS = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Not great…

Residual Standard Error

$$RSE = \sqrt{\frac{1}{n-p-1}RSS} = \sqrt{\frac{(y_i - \hat{y}_i)^2}{n-p-1}}$$

Better…can roughly think of as average amount that response will deviate from regression line

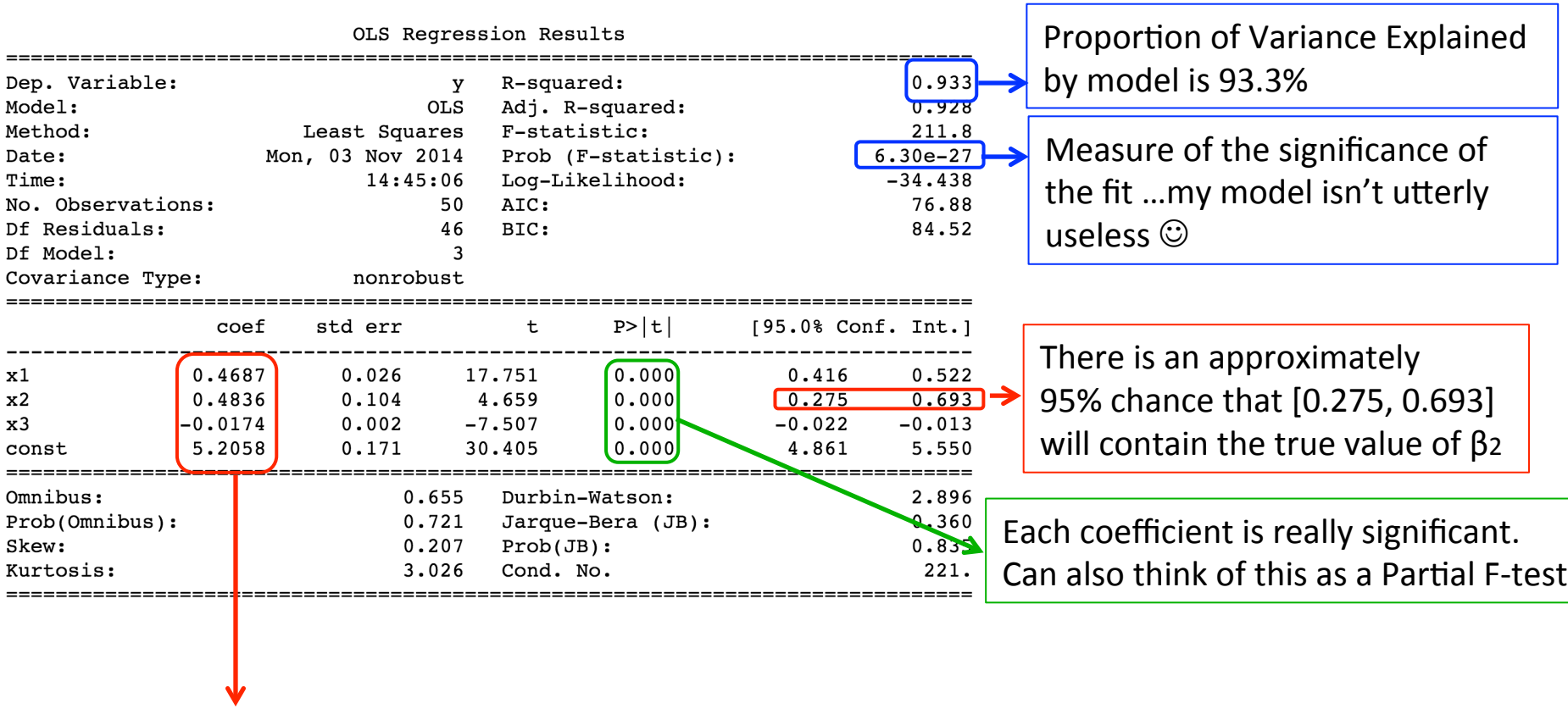
R-Squared, or "Proportion of Variance Explained"

$$R^2 = \frac{\text{TSS} - \text{RSS}}{\text{TSS}} = 1 - \frac{\text{RSS}}{\text{TSS}}$$

where $\text{TSS} = \sum_{i=1}^{n}(y_i - \bar{y})^2$

☺ Nice interpretation
Independent of scale of y

Interpretation



OLS Regression Results

Dep. Variable:	y	R-squared:	0.933
Model:	OLS	Adj. R-squared:	0.928
Method:	Least Squares	F-statistic:	211.8
Date:	Mon, 03 Nov 2014	Prob (F-statistic):	6.30e-27
Time:	14:45:06	Log-Likelihood:	-34.438
No. Observations:	50	AIC:	76.88
Df Residuals:	46	BIC:	84.52
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf.	Int.]
x1	0.4687	0.026	17.751	0.000	0.416	0.522
x2	0.4836	0.104	4.659	0.000	0.275	0.693
x3	-0.0174	0.002	-7.507	0.000	-0.022	-0.013
const	5.2058	0.171	30.405	0.000	4.861	5.550

Omnibus:	0.655	Durbin-Watson:	2.896
Prob(Omnibus):	0.721	Jarque-Bera (JB):	0.360
Skew:	0.207	Prob(JB):	0.835
Kurtosis:	3.026	Cond. No.	221.

Proportion of Variance Explained by model is 93.3%

Measure of the significance of the fit …my model isn't utterly useless ☺

There is an approximately 95% chance that [0.275, 0.693] will contain the true value of β_2

Each coefficient is really significant. Can also think of this as a Partial F-test.

"The average effect on Y of a one unit increase in X_2, holding all other predictors (X_1 & X_3) fixed, is 0.4836"

- However, interpretations are generally pretty hazardous due to correlations among predictors.
- p-values for each coefficient ≈ 0, so might be okay here

Note: Magnitude of the Beta coefficients is NOT how to determine whether predictor contributes. Why?

Linear Regression - Woes of Interpretation

- Don't use the magnitude of coefficient to determine *how significant* the variable is.
  - You can get a sense of contribution in the context of other predictors, but that's about it.
  - Feet vs. Inches

- If p-value of the coefficient is not significant, don't interpret coefficient.

```
Residuals:
    Min      1Q  Median      3Q     Max
-149.95  -34.42  -14.74   11.58  560.38

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  53.3483    11.6908   4.563 1.17e-05 ***
Cr            1.8577     0.2324   7.994 6.66e-13 ***
Co            2.1808     1.7530   1.244    0.216
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 74.76 on 128 degrees of freedom
Multiple R-squared: 0.544,      Adjusted R-squared: 0.5369
F-statistic: 76.36 on 2 and 128 DF,  p-value: < 2.2e-16
```

Interactions

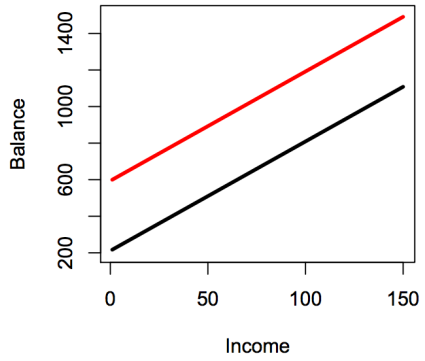
Interacting ***student*** (qualitative) and ***income*** (quantitative)

No Interaction

$$balance_i = \beta_0 + \beta_1 * income_i + \beta_2 * student_i$$

$$\texttt{balance}_i \approx \beta_0 + \beta_1 \times \texttt{income}_i + \begin{cases} \beta_2 & \text{if } i\text{th person is a student} \\ 0 & \text{if } i\text{th person is not a student} \end{cases}$$

$$= \beta_1 \times \texttt{income}_i + \begin{cases} \beta_0 + \beta_2 & \text{if } i\text{th person is a student} \\ \beta_0 & \text{if } i\text{th person is not a student} \end{cases}$$

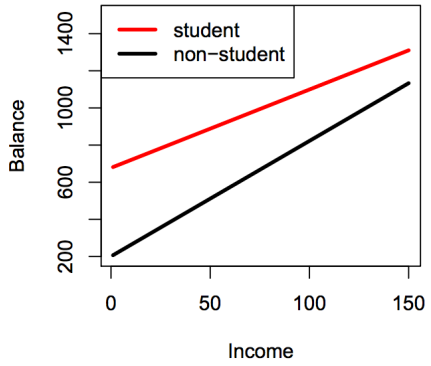


With Interaction

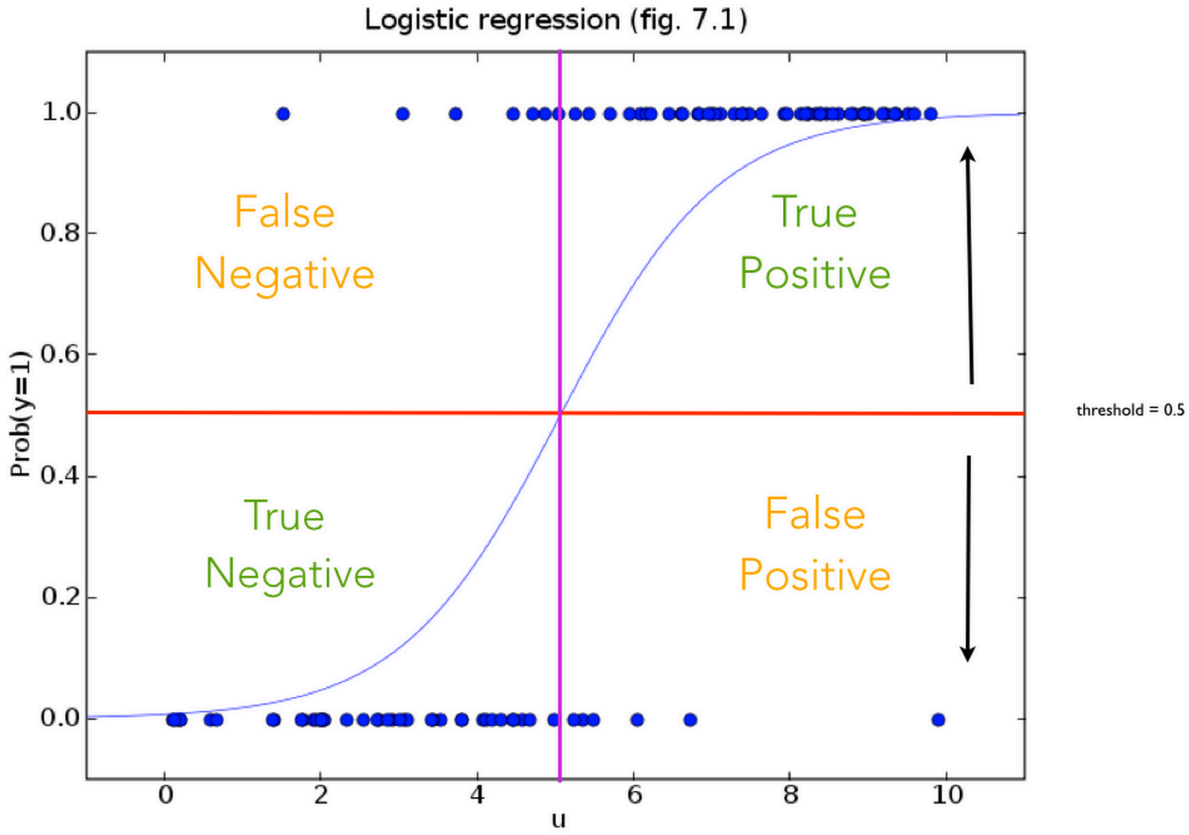
$$balance_i = \beta_0 + \beta_1 * income_i + \beta_2 * student_i + \beta_3 * income_i * student_i$$

$$\texttt{balance}_i \approx \beta_0 + \beta_1 \times \texttt{income}_i + \begin{cases} \beta_2 + \beta_3 \times \texttt{income}_i & \text{if student} \\ 0 & \text{if not student} \end{cases}$$

$$= \begin{cases} (\beta_0 + \beta_2) + (\beta_1 + \beta_3) \times \texttt{income}_i & \text{if student} \\ \beta_0 + \beta_1 \times \texttt{income}_i & \text{if not student} \end{cases}$$

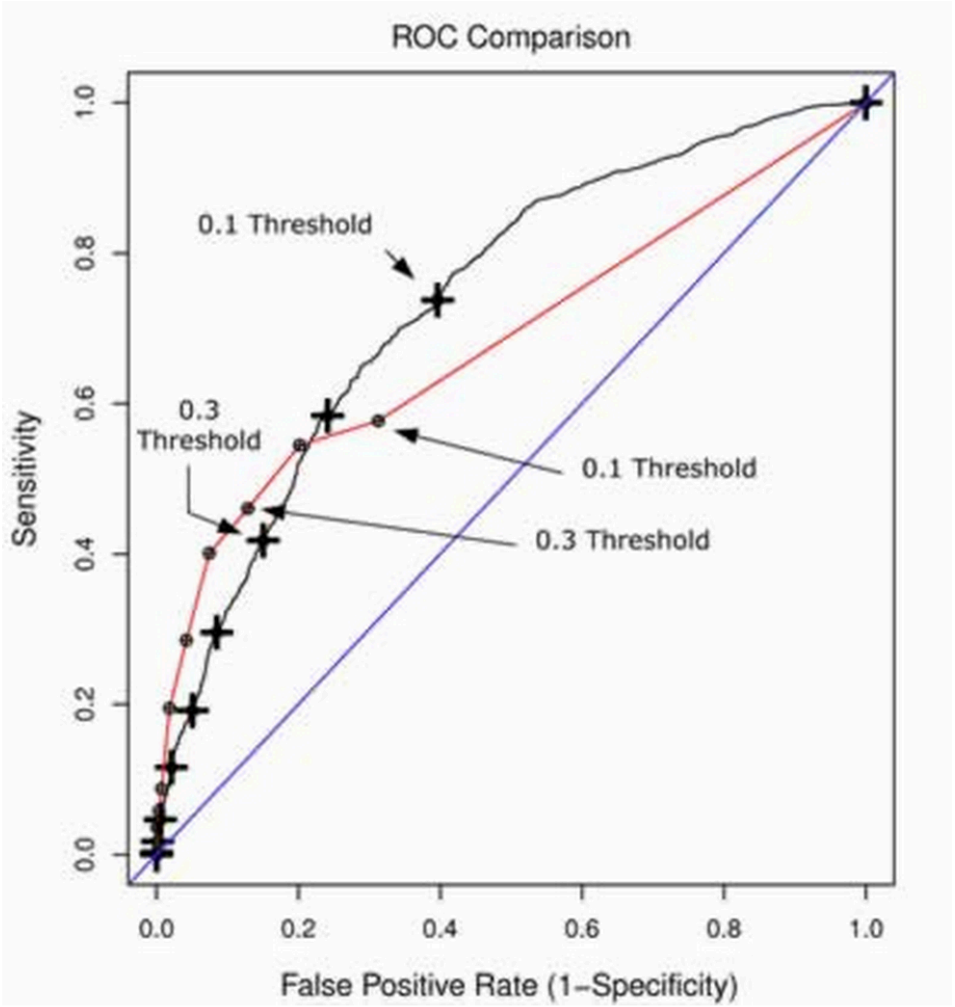


Logistic Regression



	Predicted Yes	**Predicted No**
Actual Yes	True positive	False negative
Actual No	False positive	True negative

Logistic Regression - Evaluation



- Area under the Curve (AUC)
- F1 Score
- Precision / Recall
- Sensitivity / Specificity

Logistic Regression - Evaluation

true positive (TP)

eqv. with hit

true negative (TN)

eqv. with correct rejection

false positive (FP)

eqv. with false alarm, Type I error

false negative (FN)

eqv. with miss, Type II error

accuracy (ACC)

$$ACC = (TP + TN)/(P + N)$$

F1 score

is the harmonic mean of precision and sensitivity

$$F1 = 2TP/(2TP + FP + FN)$$

sensitivity or true positive rate (TPR)

eqv. with hit rate, recall

$$TPR = TP/P = TP/(TP + FN)$$

specificity (SPC) or true negative rate (TNR)

$$SPC = TN/N = TN/(FP + TN)$$

precision or positive predictive value (PPV)

$$PPV = TP/(TP + FP)$$

negative predictive value (NPV)

$$NPV = TN/(TN + FN)$$

fall-out or false positive rate (FPR)

$$FPR = FP/N = FP/(FP + TN)$$

false discovery rate (FDR)

$$FDR = FP/(FP + TP) = 1 - PPV$$

false negative rate (FNR)

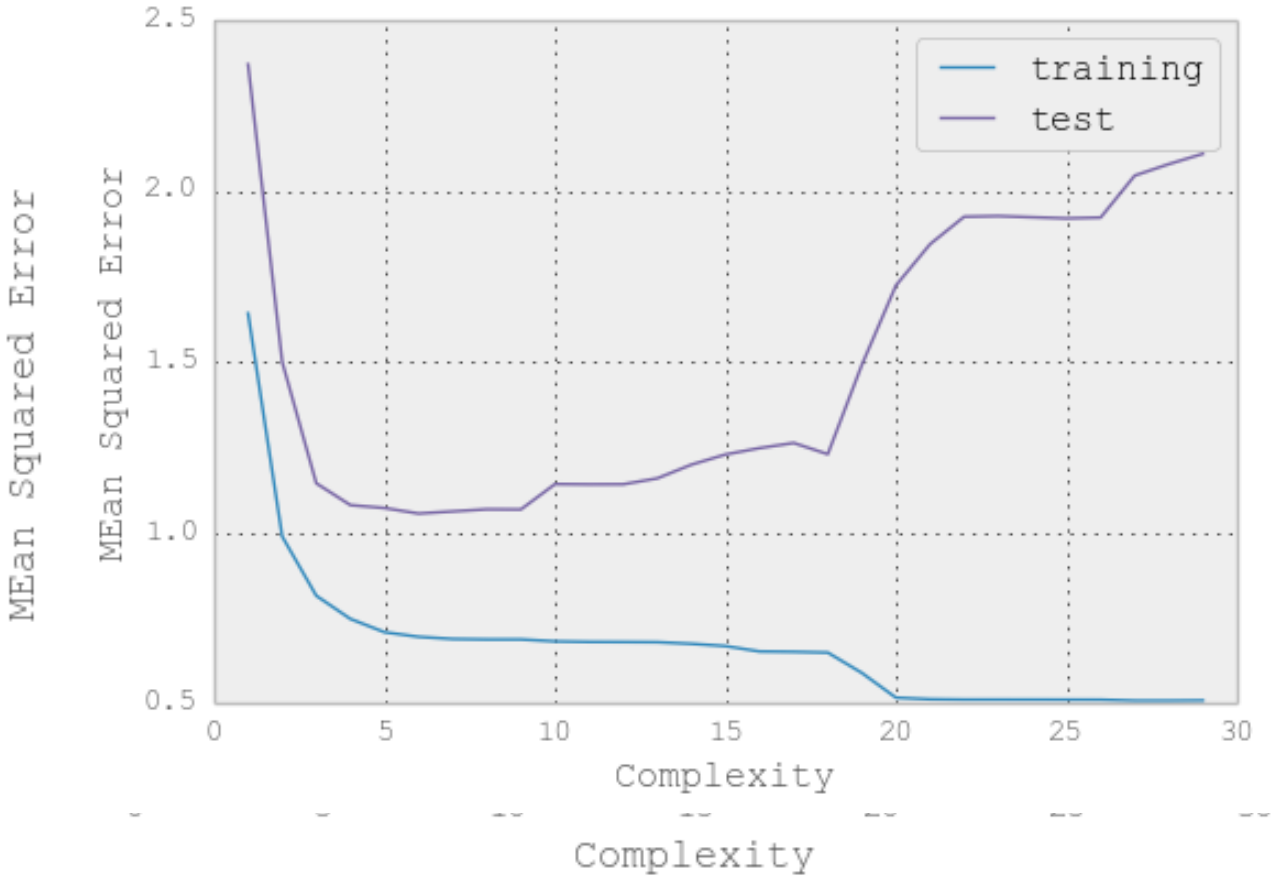
$$FNR = FN/(FN + TP) = 1 - TPR$$

Logistic Regression - Interpretation

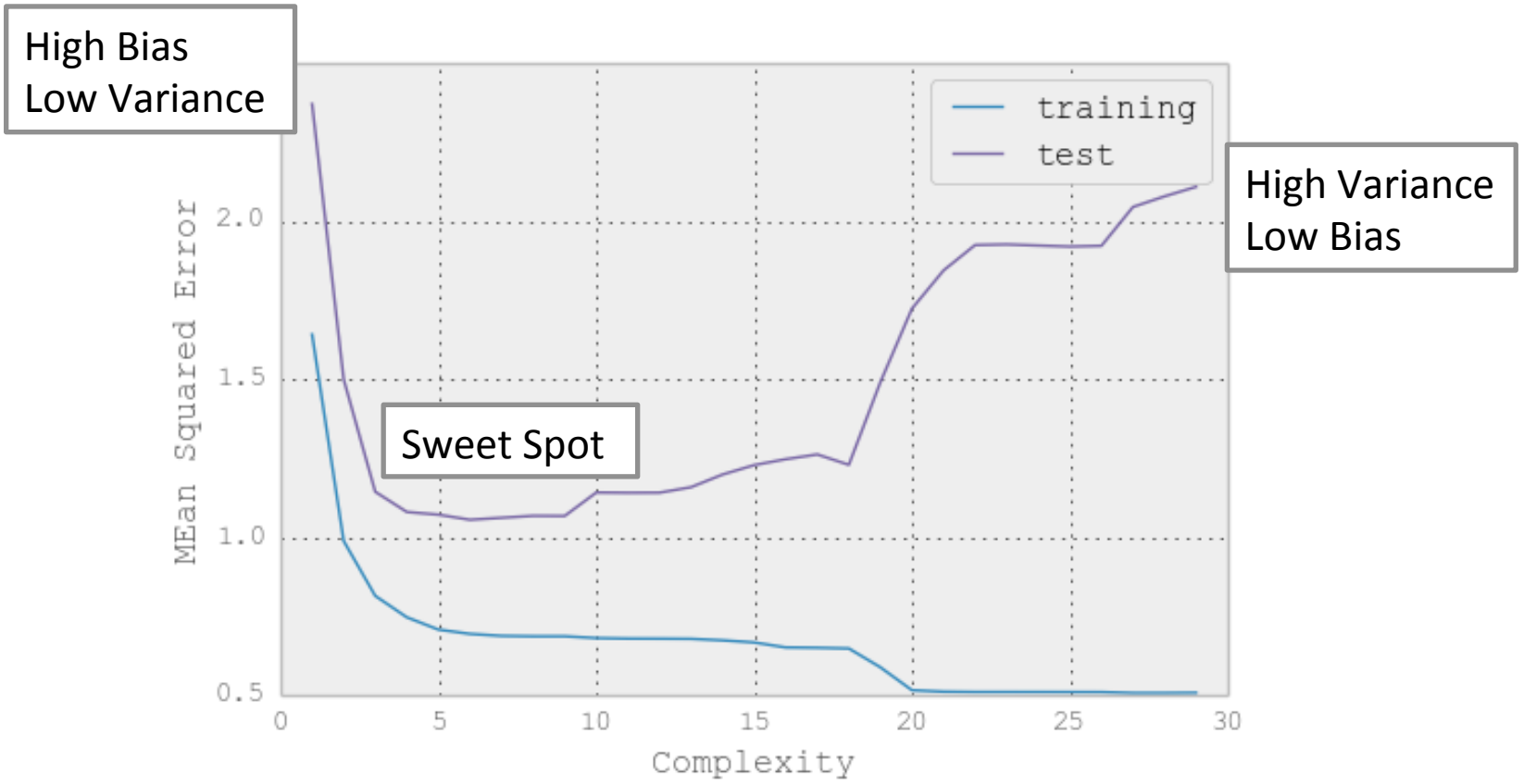
$$\frac{p}{1-p} = e^{\beta_0+\beta_1 X_1+\beta_2 X_2+\ldots+\beta_n X_n} = e^{\beta_0} e^{\beta_1 X_1} e^{\beta_2 X_2} \ldots e^{\beta_n X_n}$$

It tells you how much 1-unit increase of a feature increases the odds of being classified in the positive class. In this way, the coefficients of the logistic regression can be interpreted similarly to that of linear regression

Model Framework - Evaluation

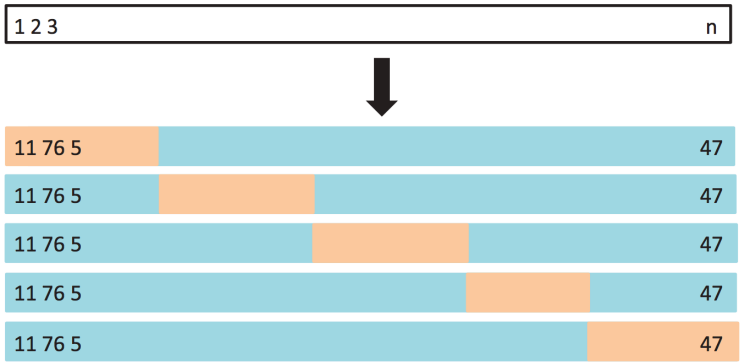


Model Framework - Evaluation



- Can break this complexity tradeoff into what we call "bias" and "variance"

K-Fold Cross-Validation



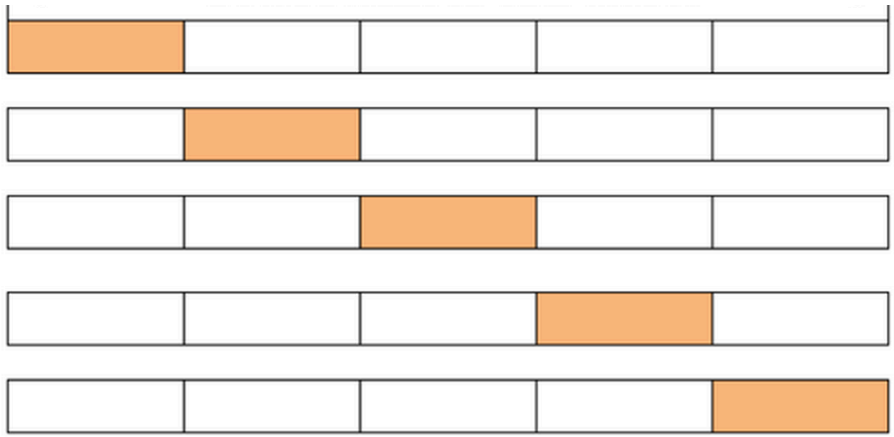
Randomly divide data into K=5 folds. Typically choose K=5 or 10.

Run K times

1. Fit model on training set, using (K-1) folds
2. Use fitted model in 1. to predict responses for validation set, 1 of the folds
3. Compute validation-set error
   - Quantitative Response: Typically MSE
   - Qualitative Response: Typically Misclassification Rate

$$\mathrm{CV}_{(k)} = \frac{1}{k}\sum_{i=1}^{k}\mathrm{MSE}_i$$

Model Framework – Cross-Validate & Test



Training

Validation

TRY STUFF
- tune parameter 1
- tune parameter 2
- feature engineering
- feature selection
- scale factors 1 way
- scale factors another way
- etc. etc.

Test Set

Don't touch until end for final evaluation

Bias-Variance Tradeoff

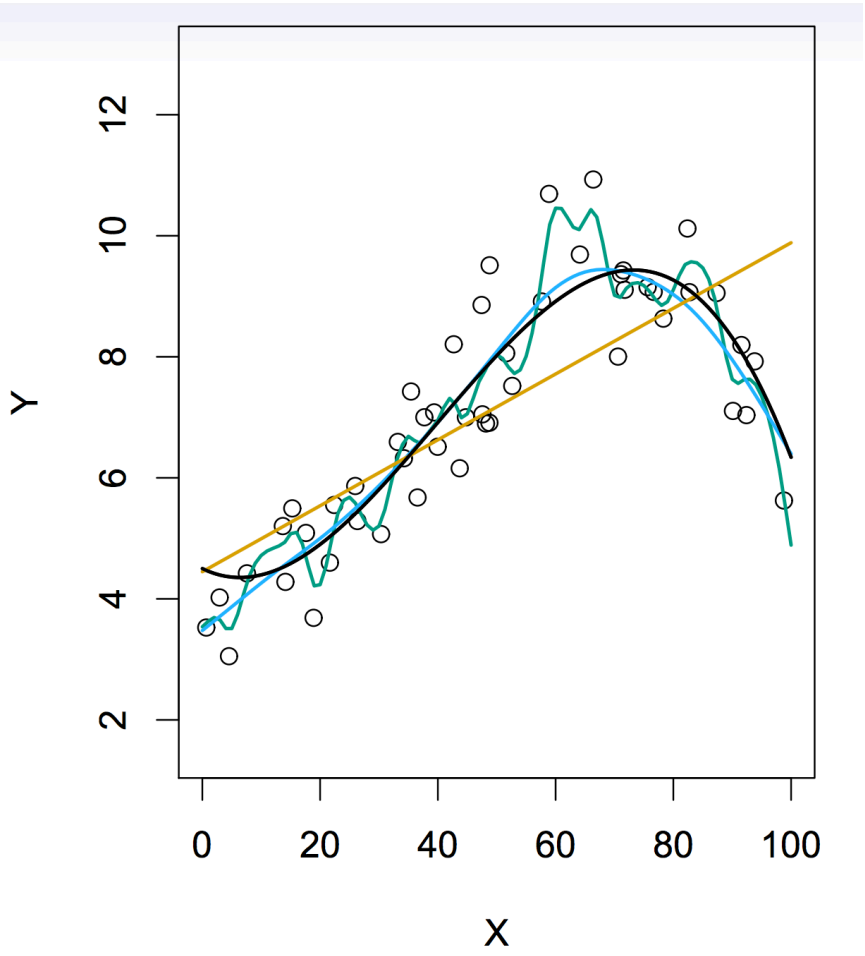
Suppose we have fit a model $\hat{f}(x)$ to some training data Tr, and let (x_0, y_0) be a test observation drawn from the population. If the true model is $Y = f(X) + \epsilon$ (with $f(x) = E(Y|X = x)$), then

$$E\left(y_0 - \hat{f}(x_0)\right)^2 = \text{Var}(\hat{f}(x_0)) + [\text{Bias}(\hat{f}(x_0))]^2 + \text{Var}(\epsilon).$$

Ok….what is going on here?

- Applies to modeling in general, beyond Linear Regression
- Want your model to minimize the expected test MSE on LHS. But how?
  - Var(ε), or "Irreducible Error". Can't do anything about that!
  - **Can** reduce Variance
  - **Can** reduce Bias

Bias-Variance Tradeoff



$$\mathrm{Var}(\hat{f}(x_0))$$

Amount by which $\hat{f}$ would change if estimated it using a different training dataset

$$\mathrm{Bias}(\hat{f}(x_0))] = E[\hat{f}(x_0)] - f(x_0)$$

Difference between expected prediction of our model and correct value we are trying to predict

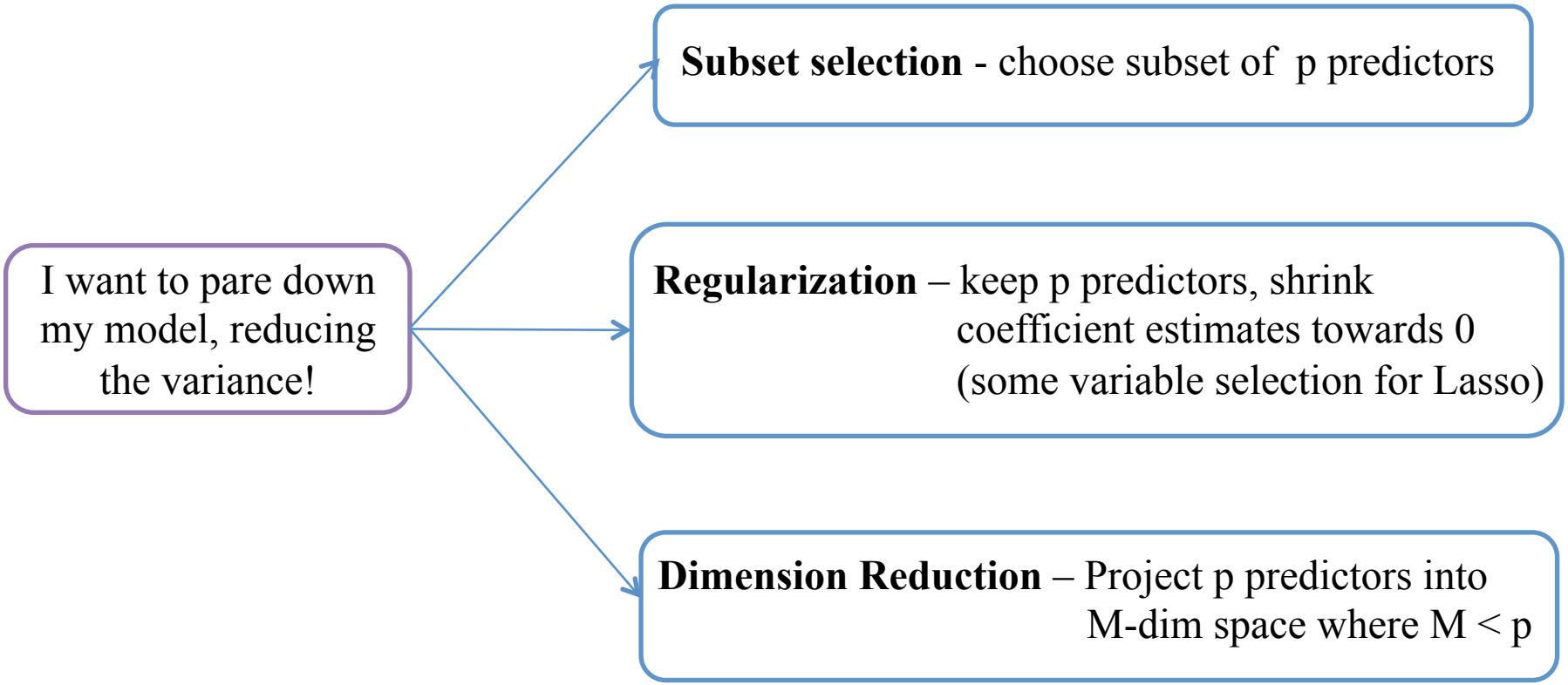
$$\mathrm{Var}(\epsilon)$$

Simply because $Y = f(X) + \epsilon$

Generally speaking, the *more flexible* the model, the *greater the variance*.

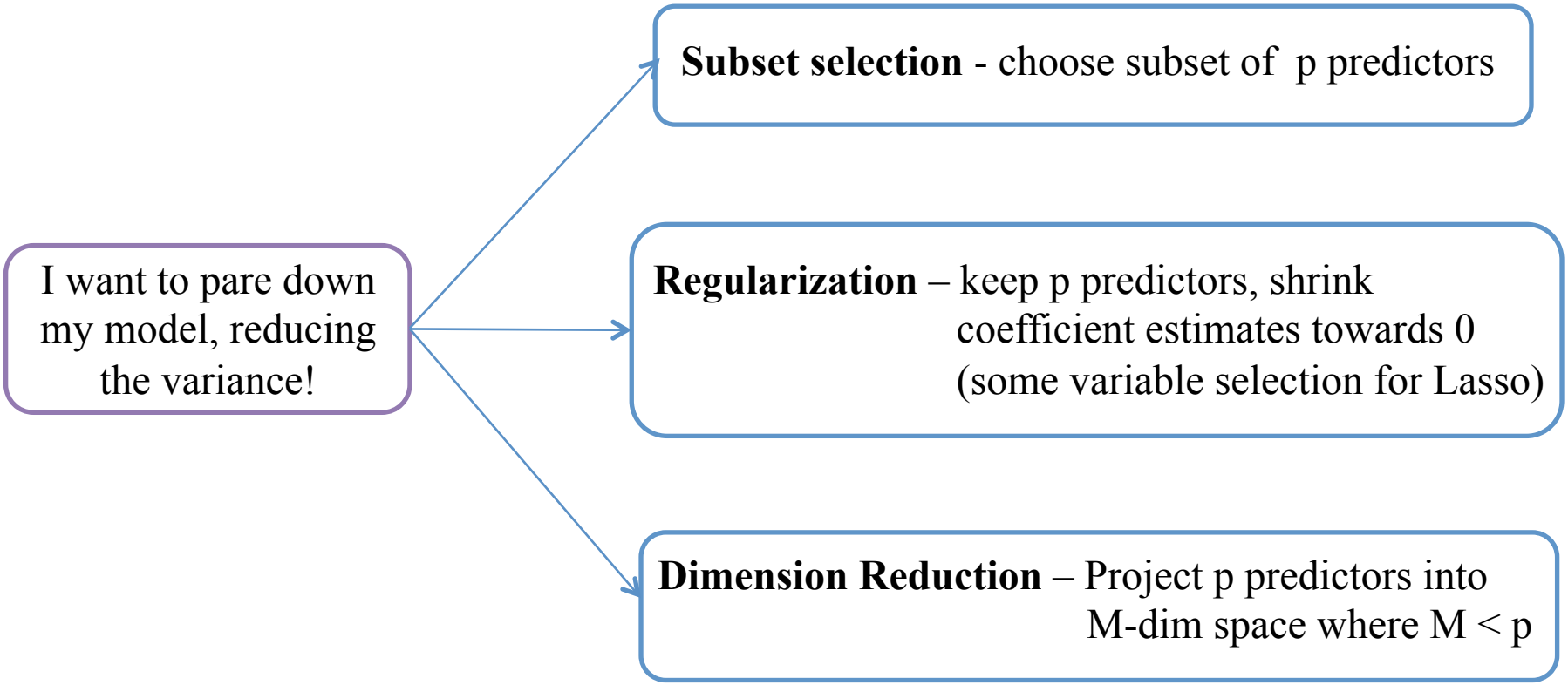
Managing the Bias-Variance Tradeoff with Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$



Managing the Bias-Variance Tradeoff with Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

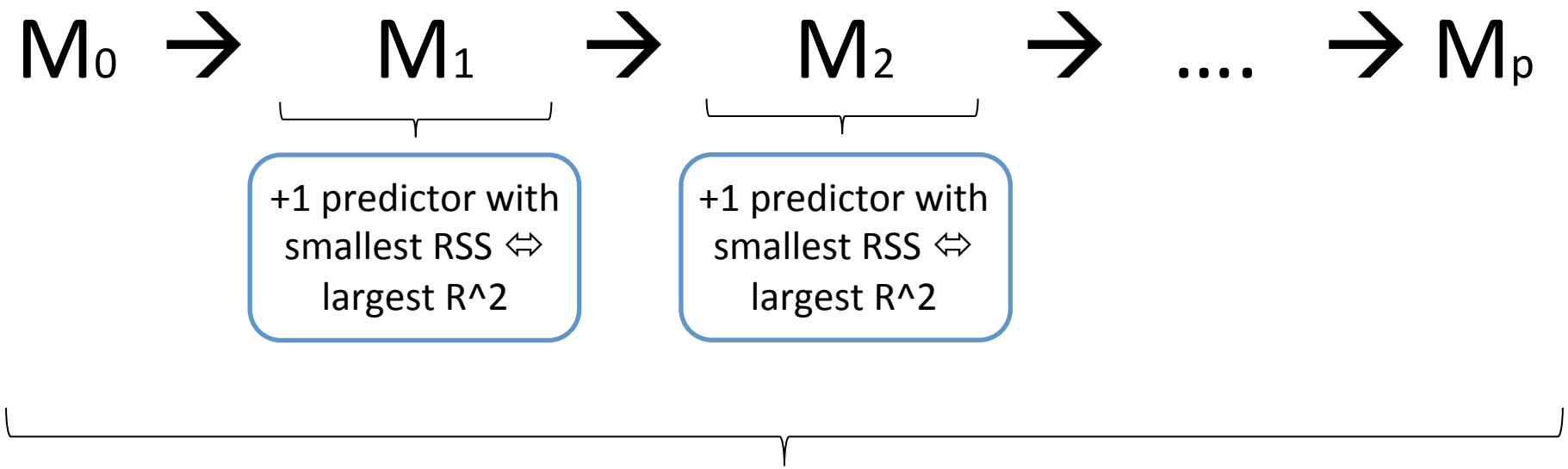


Subset Selection

- Best subset: Try every model. Every possible combination of *p* predictors
  - Computationally intensive, especially for *p* large
  - Also, huge search space. Higher chance of finding models that look good on training data but have little predictive power on future data

- Stepwise
  - In practice, commonly done
  - Forward, Backward, Forward + Backward

Subset Selection - Forward Stepwise



Now we have *p* candidate models

Are RSS and R^2 good ways to decide amongst the *p* candidates?

Subset selection

Choosing among *p* candidate models…

- Cross-validation - always a great standby
- Mallow's C_p
- AIC
- BIC
- Adjusted R^2

OLS Regression Results

Dep. Variable:	y	R-squared:	0.933
Model:	OLS	Adj. R-squared:	0.928
Method:	Least Squares	F-statistic:	211.8
Date:	Mon, 03 Nov 2014	Prob (F-statistic):	6.30e-27
Time:	14:45:06	Log-Likelihood:	-34.438
No. Observations:	50	AIC:	76.88
Df Residuals:	46	BIC:	84.52
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[95.0% Conf.	Int.]
x1	0.4687	0.026	17.751	0.000	0.416	0.522
x2	0.4836	0.104	4.659	0.000	0.275	0.693
x3	-0.0174	0.002	-7.507	0.000	-0.022	-0.013
const	5.2058	0.171	30.405	0.000	4.861	5.550

Omnibus:	0.655	Durbin-Watson:	2.896
Prob(Omnibus):	0.721	Jarque-Bera (JB):	0.360
Skew:	0.207	Prob(JB):	0.835
Kurtosis:	3.026	Cond. No.	221.

Subset selection

Mallow's C_p:

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

where d is the total # of parameters used and $\hat{\sigma}^2$ is an estimate of the variance of the error ϵ associated with each response measurement.

The *AIC* criterion is defined for a large class of models fit by maximum likelihood:

$$\mathrm{AIC} = -2\log L + 2 \cdot d$$

where L is the maximized value of the likelihood function for the estimated model.

Can show AIC and Mallow's Cp are equivalent for linear case

Subset selection

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

Notice that BIC replaces the $2d\hat{\sigma}^2$ used by C_p with a $\log(n)d\hat{\sigma}^2$ term, where n is the number of observations.

Since $\log n > 2$ for any $n > 7$, the BIC statistic generally places a heavier penalty on models with many variables

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

Unlike the R^2 statistic, the adjusted R^2 statistic *pays a price* for the inclusion of unnecessary variables in the model.

Regularization – Ridge regression

- Recall that the least squares fitting procedure estimates $\beta_0, \beta_1, \ldots, \beta_p$ using the values that minimize

$$\text{RSS} = \sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 .$$

- In contrast, the ridge regression coefficient estimates $\hat{\beta}^R$ are the values that minimize

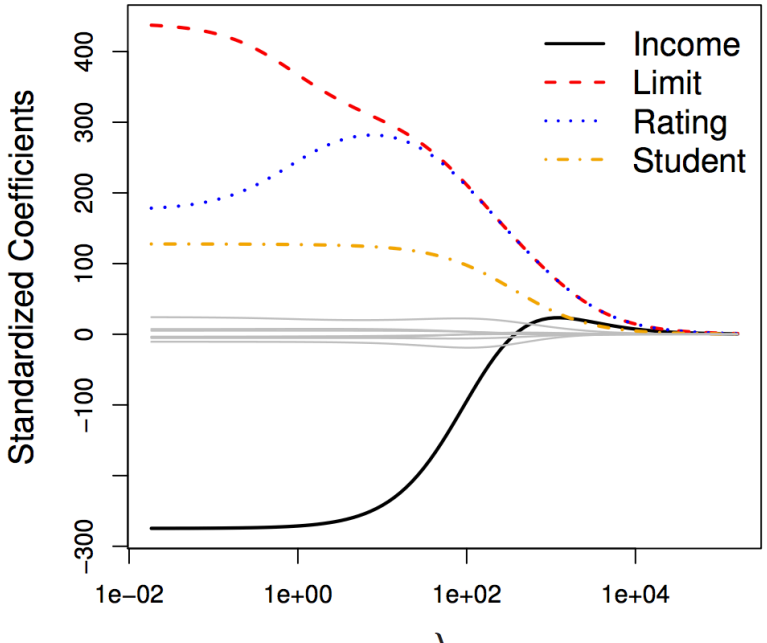
$$\sum_{i=1}^{n} \left(y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij} \right)^2 + \lambda \sum_{j=1}^{p} \beta_j^2 = \text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2,$$

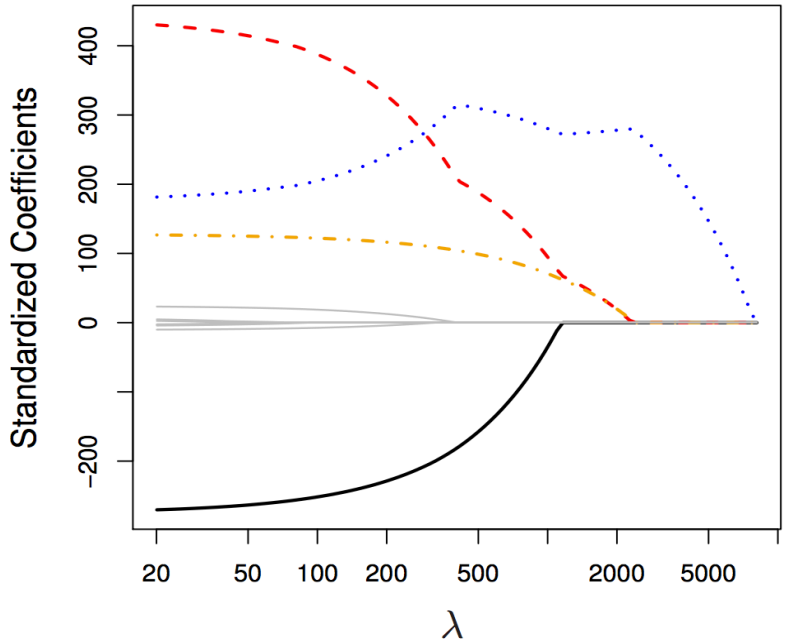
where $\lambda \geq 0$ is a *tuning parameter*, to be determined separately.

Regularization – Lasso regression

$$\sum_{i=1}^{n}\left(y_i-\beta_0-\sum_{j=1}^{p}\beta_j x_{ij}\right)^2+\lambda\sum_{j=1}^{p}|\beta_j| \;=\mathrm{RSS}+\lambda\sum_{j=1}^{p}|\beta_j|$$

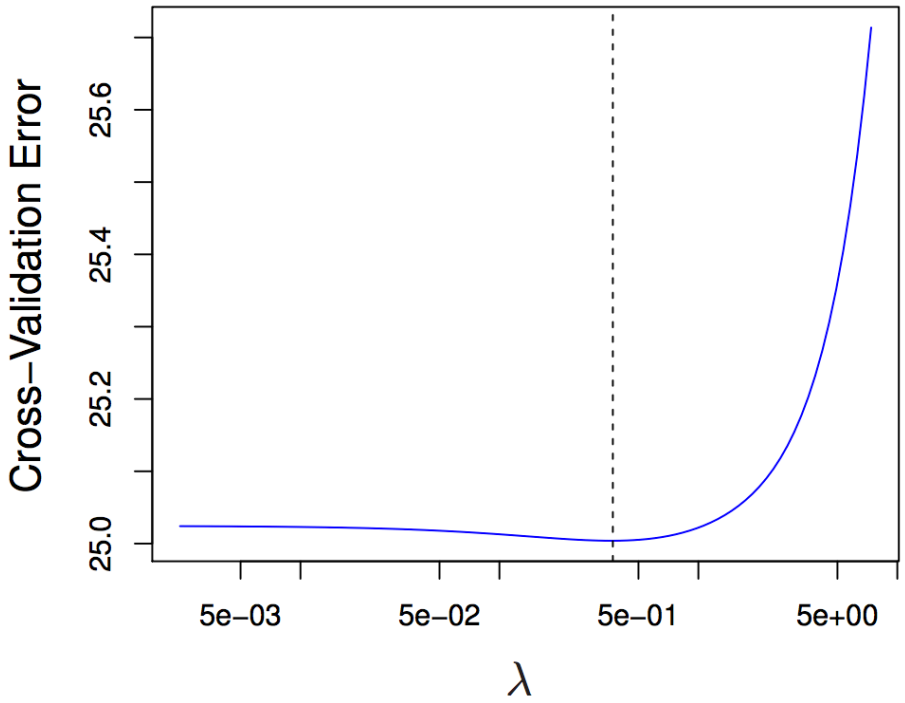
Ridge vs. Lasso

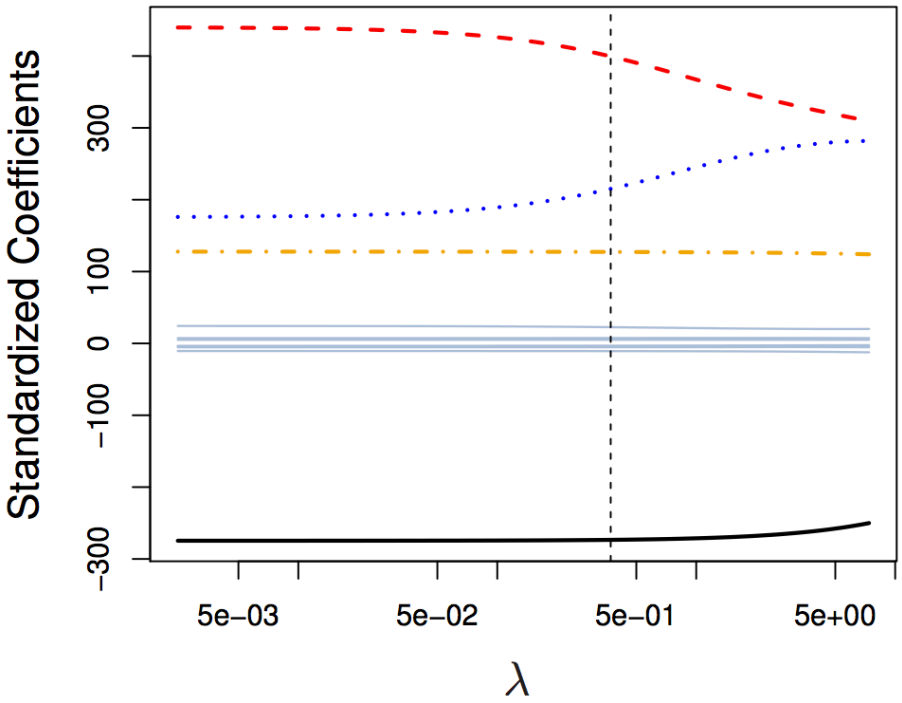




- When λ = 0, we simply have linear models.
- As λ increases, both models become less flexible, reducing variance, but increasing bias.
- Lasso has the advantage of variable selection as well (especially nice when *p* is large)
- Neither universally dominate, but in general one might expect Lasso to do better when response is function of relatively few predictors.
  - Of course you never actually know this, so use your friend, cross-validation!

Choosing λ





- Just increment λ along, fit a large number of models (1 per increment), and choose λ which minimizes cross-validated error, and voila! You have your corresponding optimized model for Ridge Regression.

Don't forget….

- The standard least squares coefficient estimates are *scale equivariant*: multiplying X_j by a constant c simply leads to a scaling of the least squares coefficient estimates by a factor of $1/c$. In other words, regardless of how the jth predictor is scaled, $X_j\hat{\beta}_j$ will remain the same.
- In contrast, the ridge regression coefficient estimates can change *substantially* when multiplying a given predictor by a constant, due to the sum of squared coefficients term in the penalty part of the ridge regression objective function.
- Therefore, it is best to apply ridge regression after *standardizing the predictors*, using the formula

$$\tilde{x}_{ij} = \frac{x_{ij}}{\sqrt{\frac{1}{n}\sum_{i=1}^{n}(x_{ij} - \overline{x}_j)^2}}$$

Review Session:
Part II

Overview – Part II

- Supervised vs. Unsupervised
- Supervised Learning
  - Decision Trees, Bagging/Random Forests, Boosting
  - SVM, kNN
- Unsupervised Learning
  - K-means clustering, Hierarchical clustering, PCA (vs. PCA Regression)

Supervised Learning

- The label is the supervisor!

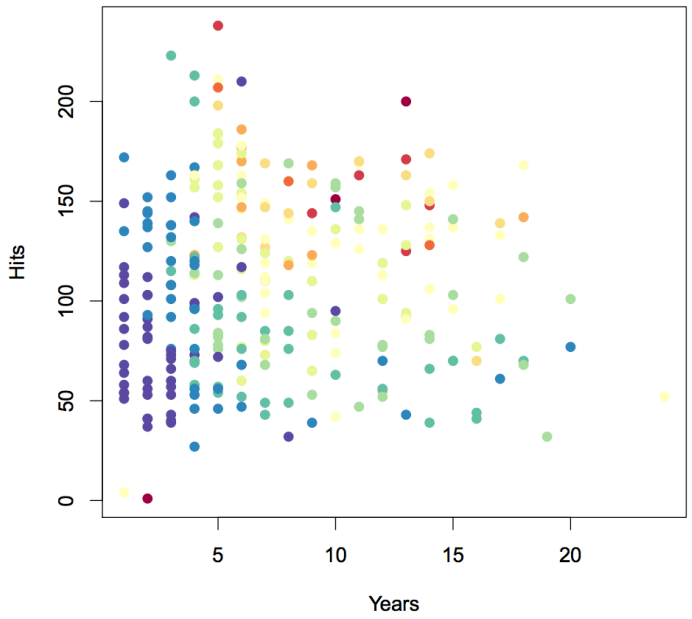
  No label ⇔ Not supervised
  - K-means clustering is not supervised learning, nor is hierarchical clustering
  - PCA is not supervised learning

  ⇒Though again, both can be used in supervised learning!

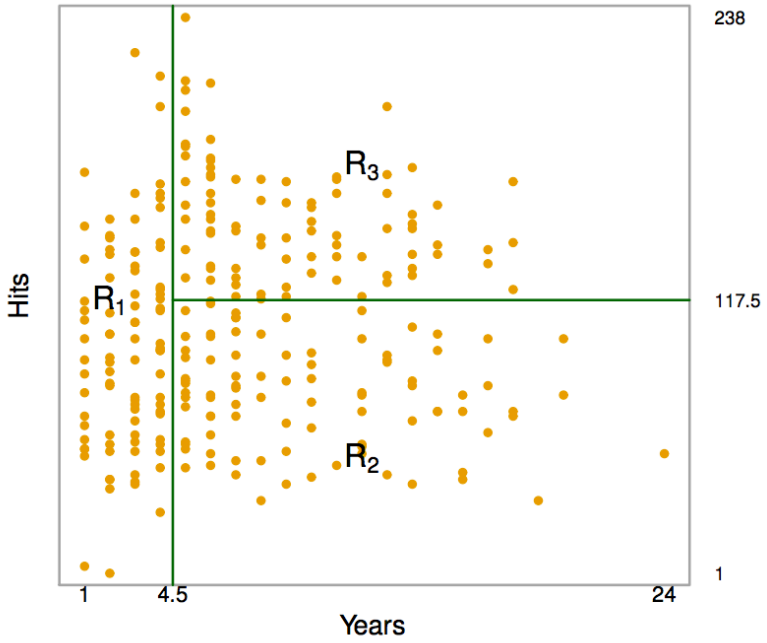
- Supervised Learning
  - Linear, Logistic, Lasso, Ridge
  - Decision Trees, Bagging, Random Forest, Boosting
  - SVM
  - kNN

Decision Trees – Regression

Salary is color-coded from low (blue, green) to high (yellow,red)



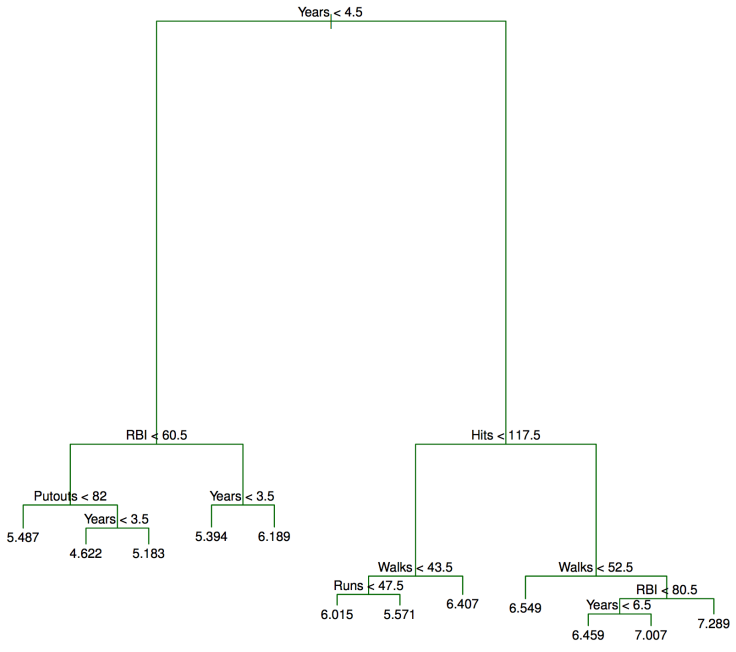
Decision Trees – Regression



At each split, we aim to minimize:

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2$$

Decision Trees – Regression



When to stop?

- You don't! Best practice to grow a very large "bushy" tree and prune backwards.

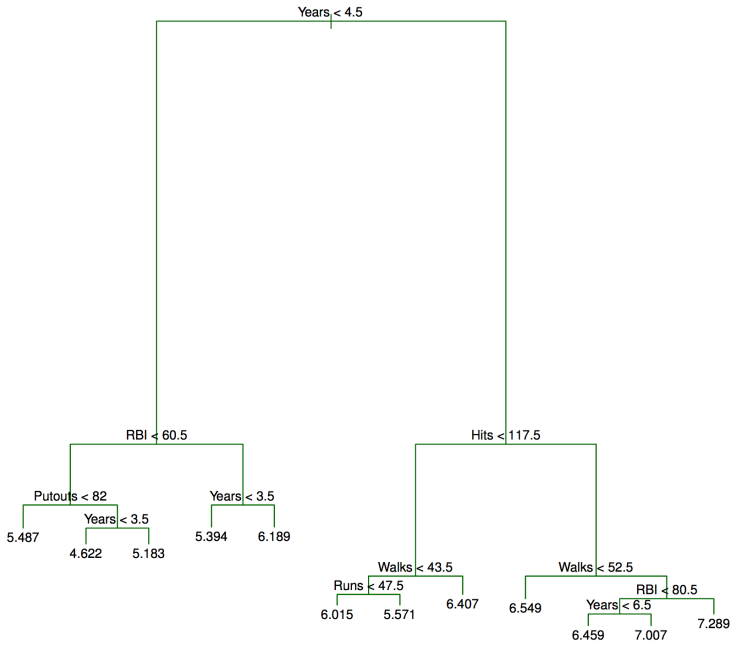
How?

- Let |T| = # of terminal nodes
- Then for any penalty term α, we have a subtree T which minimizes

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- Cross-validate as usual to choose α and it's corresponding tree.

Decision Trees – Regression



When to stop?

- You don't! Best practice to grow a very large "bushy" tree and prune backwards.

How?

- Let |T| = # of terminal nodes
- Then for any penalty term α, we have a subtree T which minimizes

$$\sum_{m=1}^{|T|} \sum_{i:\, x_i \in R_m} (y_i - \hat{y}_{R_m})^2 + \alpha |T|$$

- Cross-validate as usual to choose α and it's corresponding tree.

This should feel familiar!

In Lasso/Ridge, attack high variance of *linear regression* with cost penalty λ

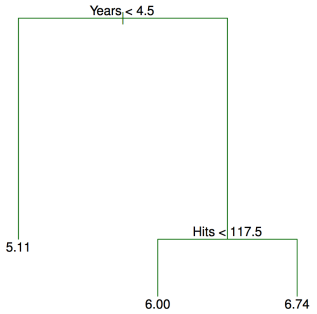
Here attack high variance of *decision tree* with cost penalty α

Decision Trees – Classification

Making Predictions

At each terminal node (or rectangular region), predict

- Regression: Average
- Classification: Most commonly occurring class



How to split?

At each potential splitting node, minimize (in terms in information gain

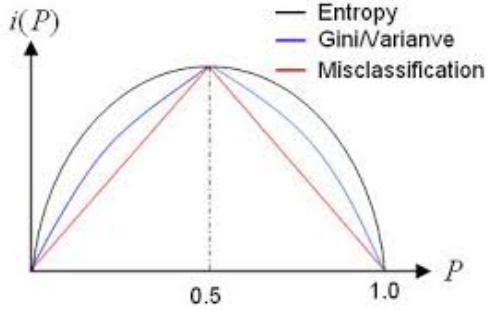
- Regression: RSS
- Classification:

Classification Error Rate $E = 1 - \max_k(\hat{p}_{mk})$

Gini index $G = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk})$

Cross-entropy $D = -\sum_{k=1}^{K} \hat{p}_{mk} \log \hat{p}_{mk}$

$\hat{p}_{mk}$ is proportion in m-th region in k-th class



Bagging

- Previously we looked at post-pruning our single decision tree to attack the variance
- Instead, we can just grow many large "bushy" trees and average away the variance (*central limit theorem*) by growing lots of trees (*bootstrapping*)!

Bagging

Making Predictions

Training set → ***B*** bootstrapped training datasets

- Regression: Average prediction of B trees

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

- Classification: Majority vote among B trees

Error Estimation

- Since bootstrapped, each tree only uses about 2/3 of observations → remaining 1/3 can be used to estimate OOB (out-of-bag) error. Like Test-error!

Random Forests

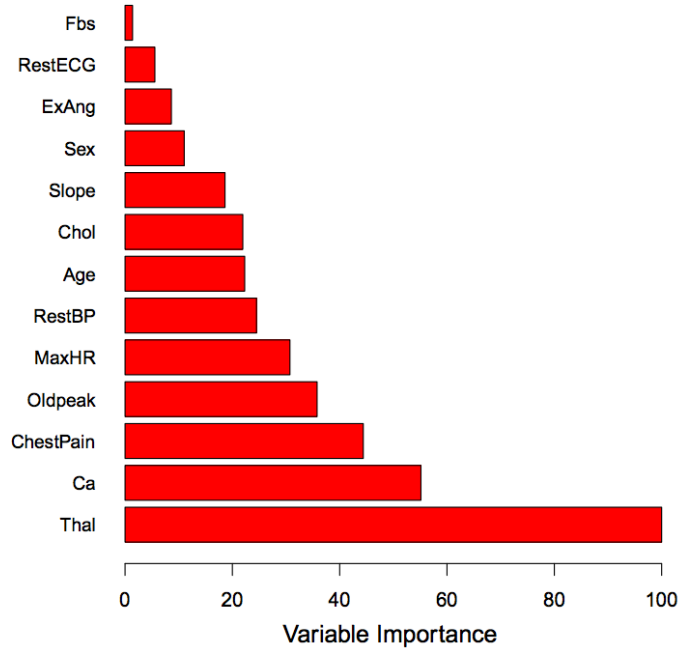
Same idea except at each split considered choose a random selection of m predictors

Typically $m \approx \sqrt{p}$ so that if you have 100 predictors, you randomly 10 candidate features at **each** split point.

This "decorrelation" of the trees leads to improved performance over bagging.

Bagging/RF – Variable Importance

- For bagged/RF regression trees, we record the total amount that the RSS is decreased due to splits over a given predictor, averaged over all B trees. A large value indicates an important predictor.
- Similarly, for bagged/RF classification trees, we add up the total amount that the Gini index is decreased by splits over a given predictor, averaged over all B trees.



Variable importance plot for the `Heart` data

Boosting Algorithm for Regression Trees

1. Set $\hat{f}(x) = 0$ and $r_i = y_i$ for all i in the training set.
2. For $b = 1, 2, \ldots, B$, repeat:
   - 2.1 Fit a tree $\hat{f}^b$ with d splits ($d + 1$ terminal nodes) to the training data (X, r).
   - 2.2 Update $\hat{f}$ by adding in a shrunken version of the new tree:

$$\hat{f}(x) \leftarrow \hat{f}(x) + \lambda \hat{f}^b(x).$$

   - 2.3 Update the residuals,

$$r_i \leftarrow r_i - \lambda \hat{f}^b(x_i).$$

3. Output the boosted model,

$$\hat{f}(x) = \sum_{b=1}^{B} \lambda \hat{f}^b(x).$$

Boosting

Bagging – Bootstrap many trees, each tree independently grown, in an effort to decrease variance through averaging

Random Forest – Similar idea, but take random subset of possible features at each split to "decorrelate the trees".

<u>What is Boosting? Not at all like Bagging or Random Forest!</u>

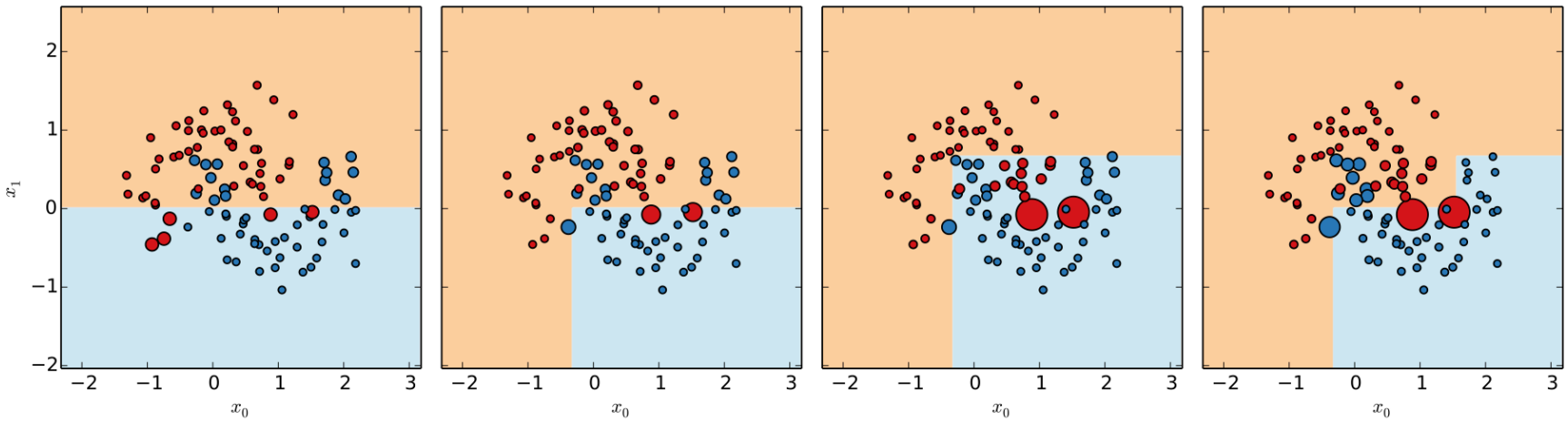
- Idea: Combine set of "weak" learners to form strong learner
  - "weak" in that error rate only slightly better than random guessing
- How: Sequentially apply weak classification algorithm to modified versions of the data → sequence of weak classifiers
  - Each tree is grown using information from last tree

Boosting – the heart of it

- **Learns slowly**
  - As opposed to fitting the data hard (like in Bagging and RF)
- **Each tree might be rather small**, with just a few terminal nodes
  - As opposed to growing the trees really large and averaging away variance
- **Focuses on areas of weakness**
  - By fitting small trees to residuals bit by bit, different shaped trees can be used to attack residuals
- **Lots of tuning**
  - Number of trees - can overfit if too large
  - Learning rate
  - Depth of tree
  - Minimum leaves per node
  - Stochastic Gradient Boosting (random subsample of features, random subset of training)

AdaBoost

- Each tree is expert on attacking errors of predecessor
- Iteratively re-weights observations based on errors



Discrete AdaBoost

One of the most popular boosting algorithms

Y: {-1, 1}

G(X): classifier producing predictions taking two values {-1, 1}

Error rate on the training set:

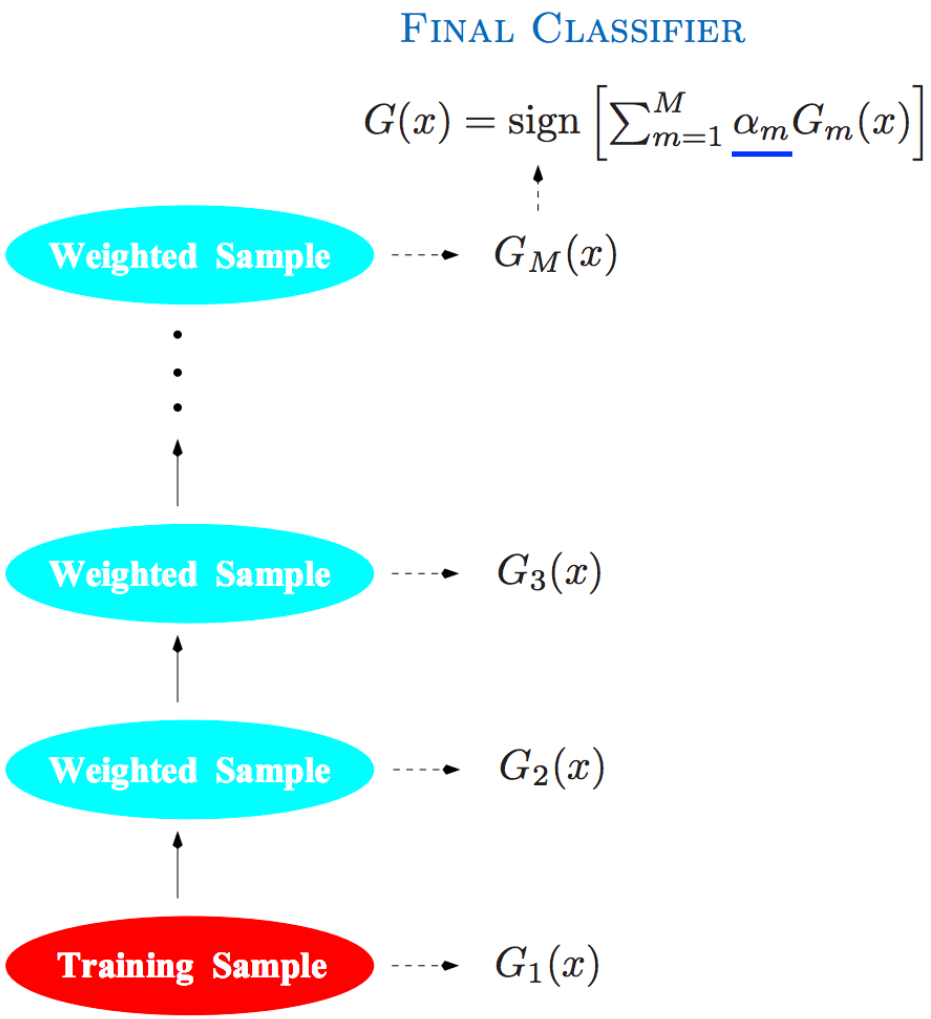
$$\overline{\text{err}} = \frac{1}{N}\sum_{i=1}^{N} I(y_i \neq G(x_i))$$

Discrete AdaBoost

$G_i(x)$ weak classifiers

$G(x)$ strong learner

Note only $G_1(x)$ fit on training



1. Initialize the observation weights $w_i = 1/N$, $i = 1, 2, \ldots, N$.

2. For $m = 1$ to M:

   (a) Fit a classifier $G_m(x)$ to the training data using weights w_i.

   (b) Compute

$$\text{err}_m = \frac{\sum_{i=1}^{N} w_i I(y_i \neq G_m(x_i))}{\sum_{i=1}^{N} w_i}.$$

   (c) Compute $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$.

   (d) Set $w_i \leftarrow w_i \cdot \exp[\alpha_m \cdot I(y_i \neq G_m(x_i))]$, $i = 1, 2, \ldots, N$.

3. Output $G(x) = \text{sign}\left[\sum_{m=1}^{M} \alpha_m G_m(x)\right]$.

- For first weak classifier, $G_1(x)$, just use training data
- For subsequent weak classifier, same classification algorithm but modify weights
  - If previously misclassified, scale by e^{α_m}
  - else, w_i same
- Final strong classifier G(x) determined by weighted majority votes
  - $\alpha_1, \ldots \alpha_M$ as weight of votes
  - The smaller the error of the weak classifier, the greater the weight

Observe that…

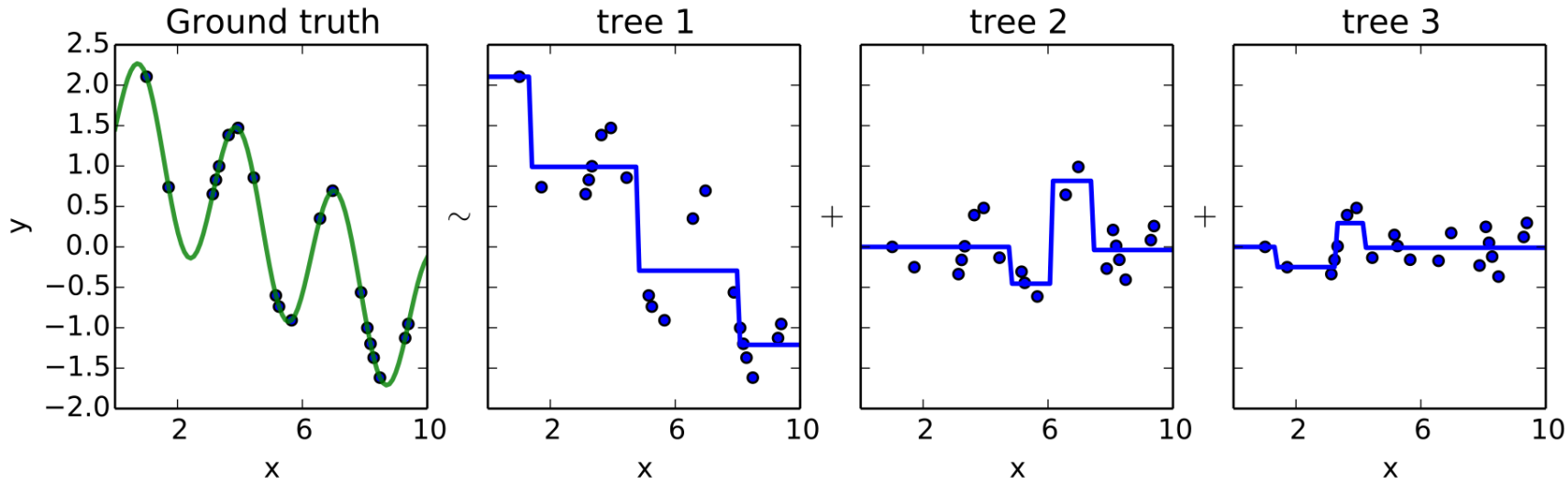
- $\alpha_1, \ldots, \alpha_M$ give higher influence to more accurate classifiers
- At each step m, observations previously misclassified by $G_{m-1}(x)$ have their weights increased

  → Each successive classifier forced to concentrate on training observations previously missed

Boosting

Gradient Boosted Regression Trees

- Instead of fitting to reweighted training observations, fit residuals to of previous tree



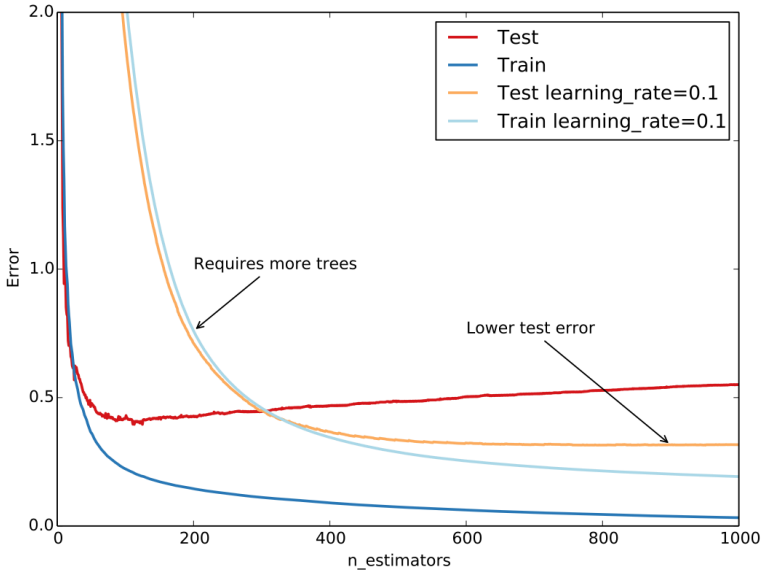
GBRT in sklearn

Tree Structure

- `max_depth`
  - controls degree of interactions
  - Ex. Latitude and Longitude
- `min_samples_per_leaf`
  - may not want terminal nodes with too few leaves

Shrinkage

- `n_estimators`
  - number of trees grown
- `learning_rate`
  - lower learning rate requires higher n_estimators



GBRT in sklearn

Stochastic Gradient Boosting

- `max_features`
  - random subsample of features
  - Especially good when you have lots of features

- `sub_sample`
  - random subset of training set

SVMs – the idea

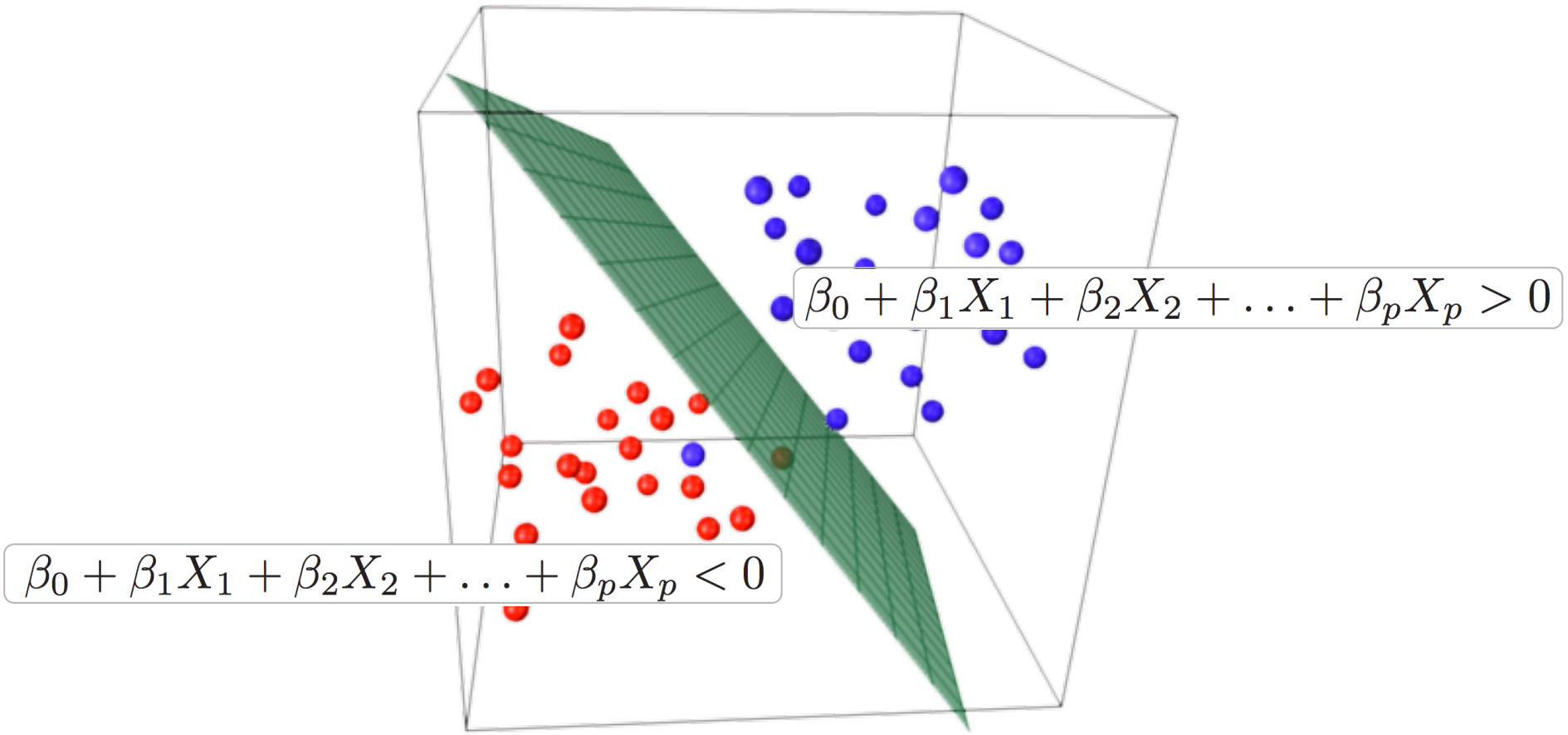
Here we approach the two-class classification problem in a direct way:

> *We try and find a plane that separates the classes in feature space.*

If we cannot, we get creative in two ways:

- We soften what we mean by "separates", and
- We enrich and enlarge the feature space so that separation is possible.

Hyperplanes



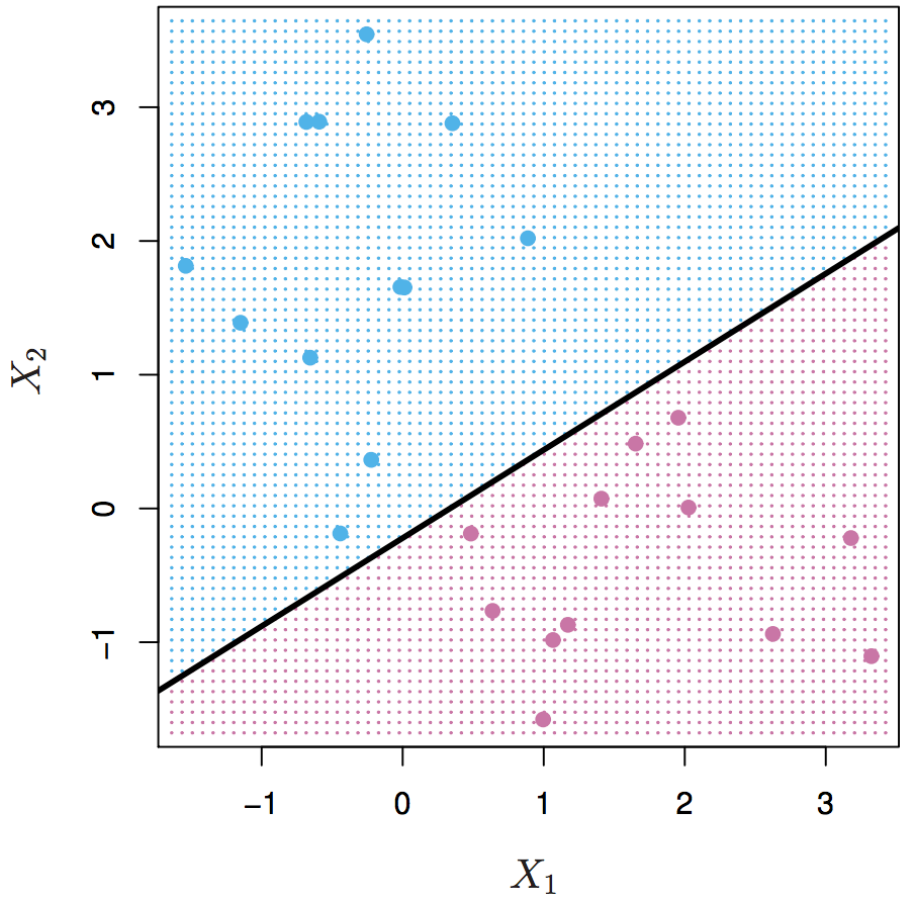
Can think of hyperplane as dividing p-dimensional space into two halves

Separating Hyperplane

Suppose we code…

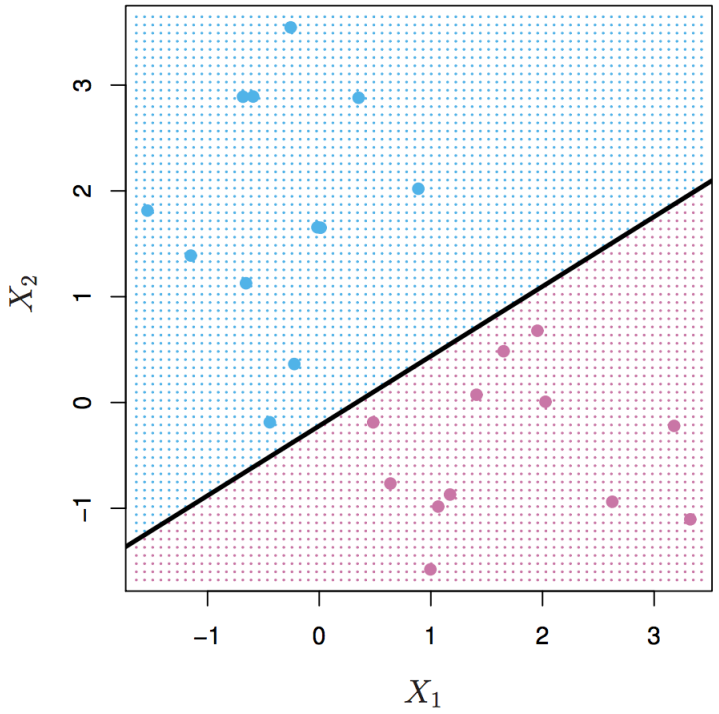
If y_i = Blue $\Longrightarrow$ y_i = +1

If y_i = Red $\Longrightarrow$ y_i = -1



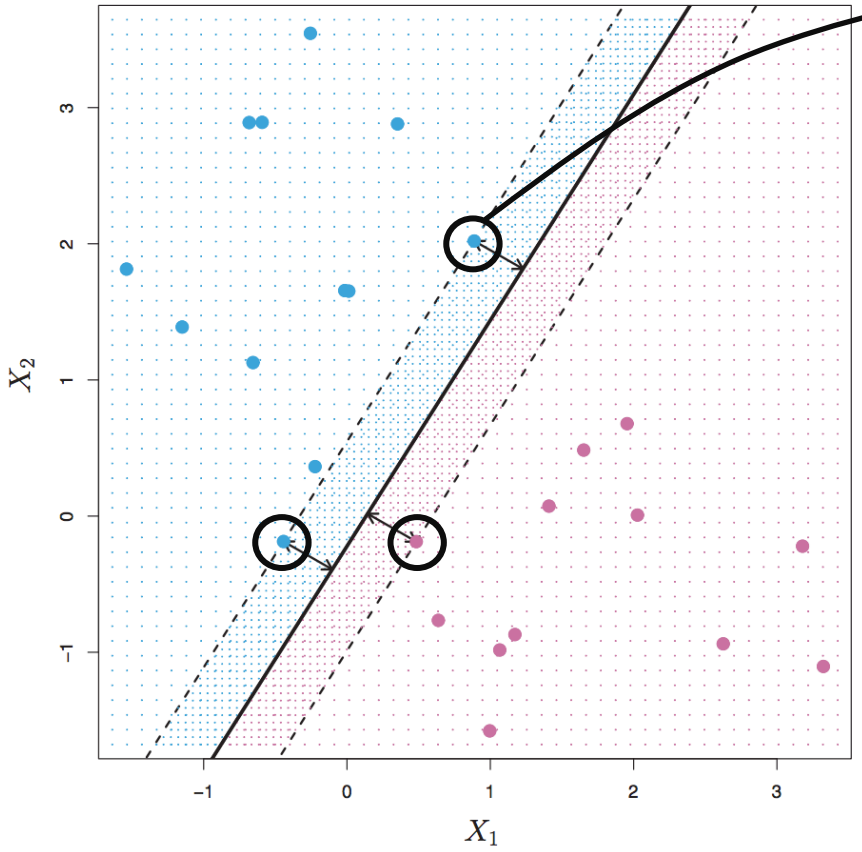
We have a *separating hyperplane*,
if for *all points*, we have…

$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} > 0 \;\; when \;\; y_i = +1$$



$$\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} < 0 \;\; when \;\; y_i = -1$$

In particular, we fit…

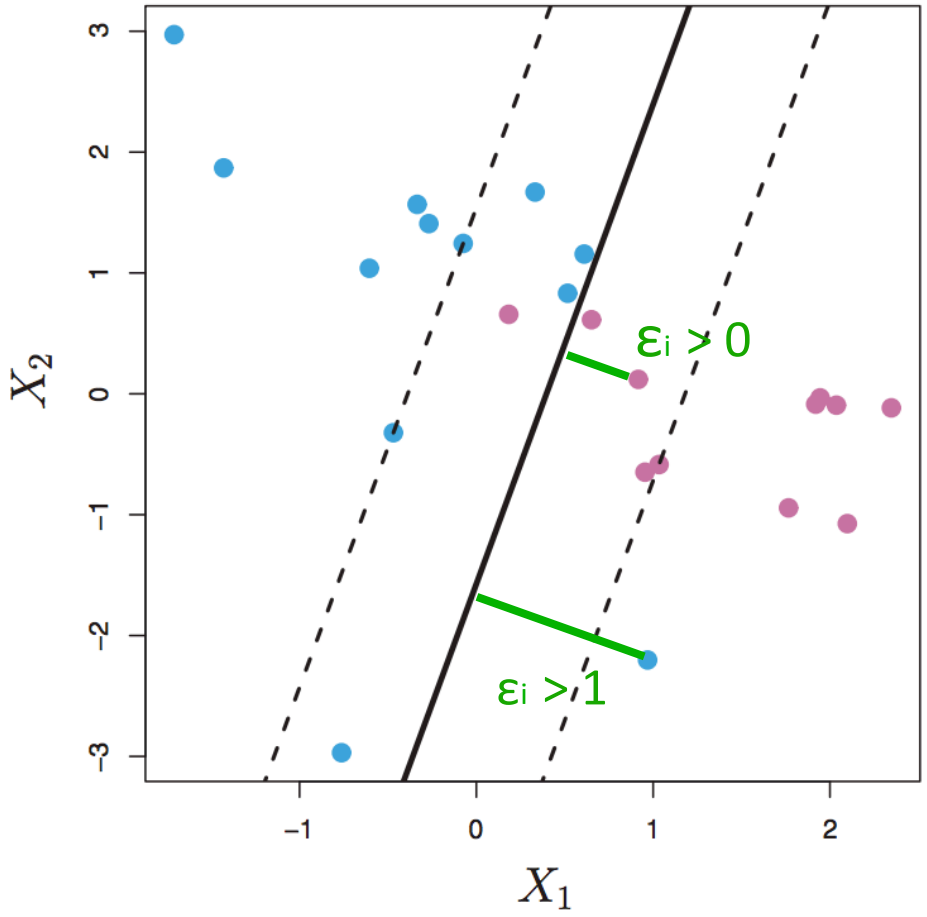


$$\underset{\beta_0,\beta_1,\ldots,\beta_p}{\text{maximize}}\, M$$

$$\text{subject to } \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M \;\; \forall\, i = 1, \ldots, n$$

need some sort of *budget*



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

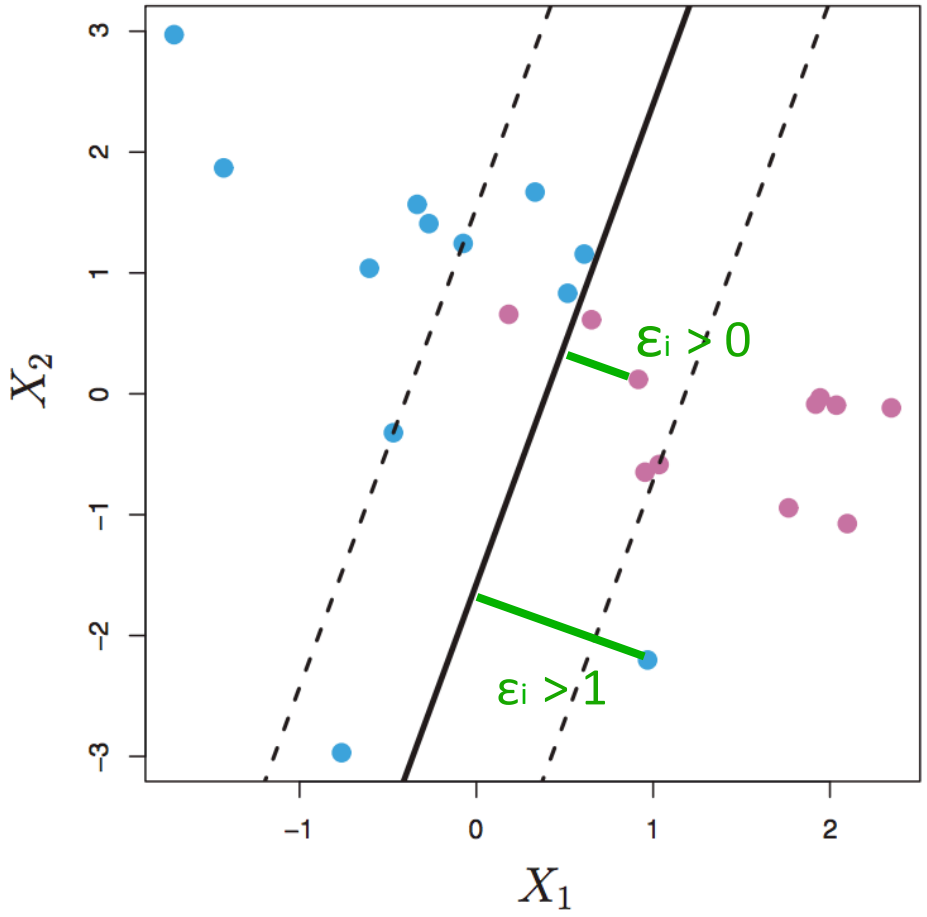
$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

Slack from each point

Budget that we can tune

εi = 0 for being on correct side of margin
εi > 0 for violating the margin
εi > 1 for being on wrong side of hyperplane

need some sort of *budget*



$$\underset{\beta_0,\beta_1,\ldots,\beta_p,\epsilon_1,\ldots,\epsilon_n}{\text{maximize}} \quad M$$

$$\text{subject to} \quad \sum_{j=1}^{p} \beta_j^2 = 1,$$

$$y_i(\beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \ldots + \beta_p x_{ip}) \geq M(1 - \epsilon_i)$$

$$\epsilon_i \geq 0, \quad \sum_{i=1}^{n} \epsilon_i \leq C,$$

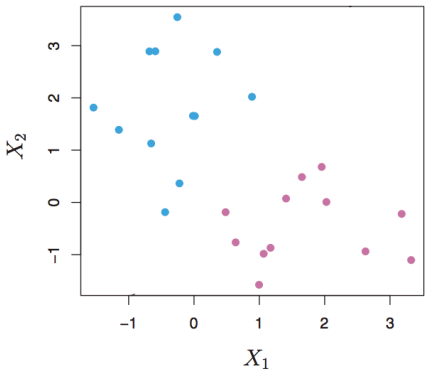
Slack from each point

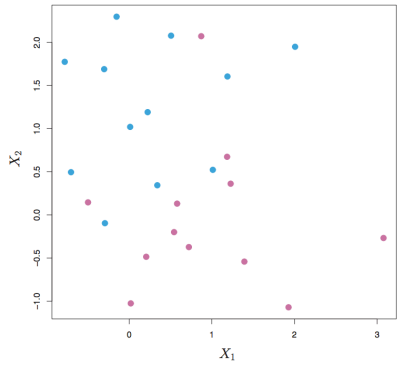
Budget that we can tune

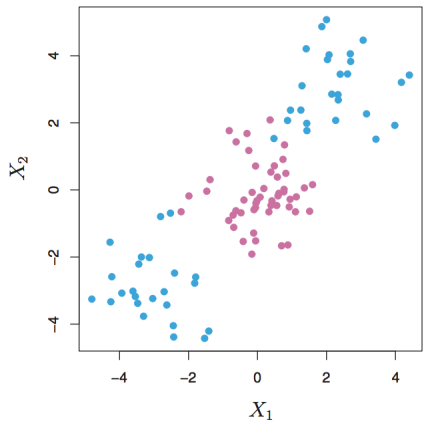
εi = 0 for being on correct side of margin
εi > 0 for violating the margin
εi > 1 for being on wrong side of hyperplane

Bias Variance Tradeoff

- C small ⇔ Low bias, High Variance
- C large ⇔ High bias, Low Variance
  (not quite as clear cut)







Maximal Margin Classifier

allow "soft margin"

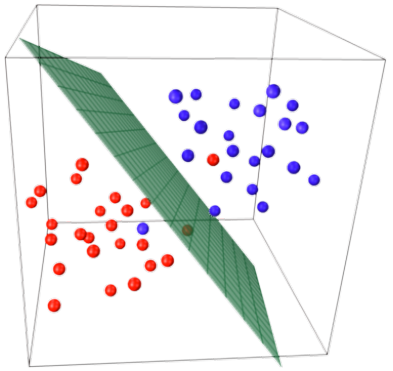
Support Vector Classifier

use "kernels"

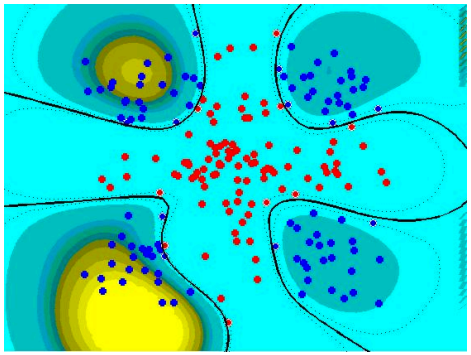
Support Vector Machine

So what are SVMs again?

- Hyperplane that separates data as well as possible, while allowing some room for error ("soft margin")



- Kernels are powerful way to accommodate non-linear class boundaries.



$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i)$$

Kernels

Solution to SVC only involves inner product of observations

$$\langle x_i, x_{i'} \rangle = \sum_{j=1}^{p} x_{ij} x_{i'j}$$

$$f(x) = \beta_0 + \sum_{i=1}^{n} \alpha_i \langle x, x_i \rangle \quad \leftarrow \text{SVC}$$

$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i \langle x, x_i \rangle \quad \leftarrow \text{Only requires support vectors}$$

More generally, instead of just taking inner product, we can use *Kernels*

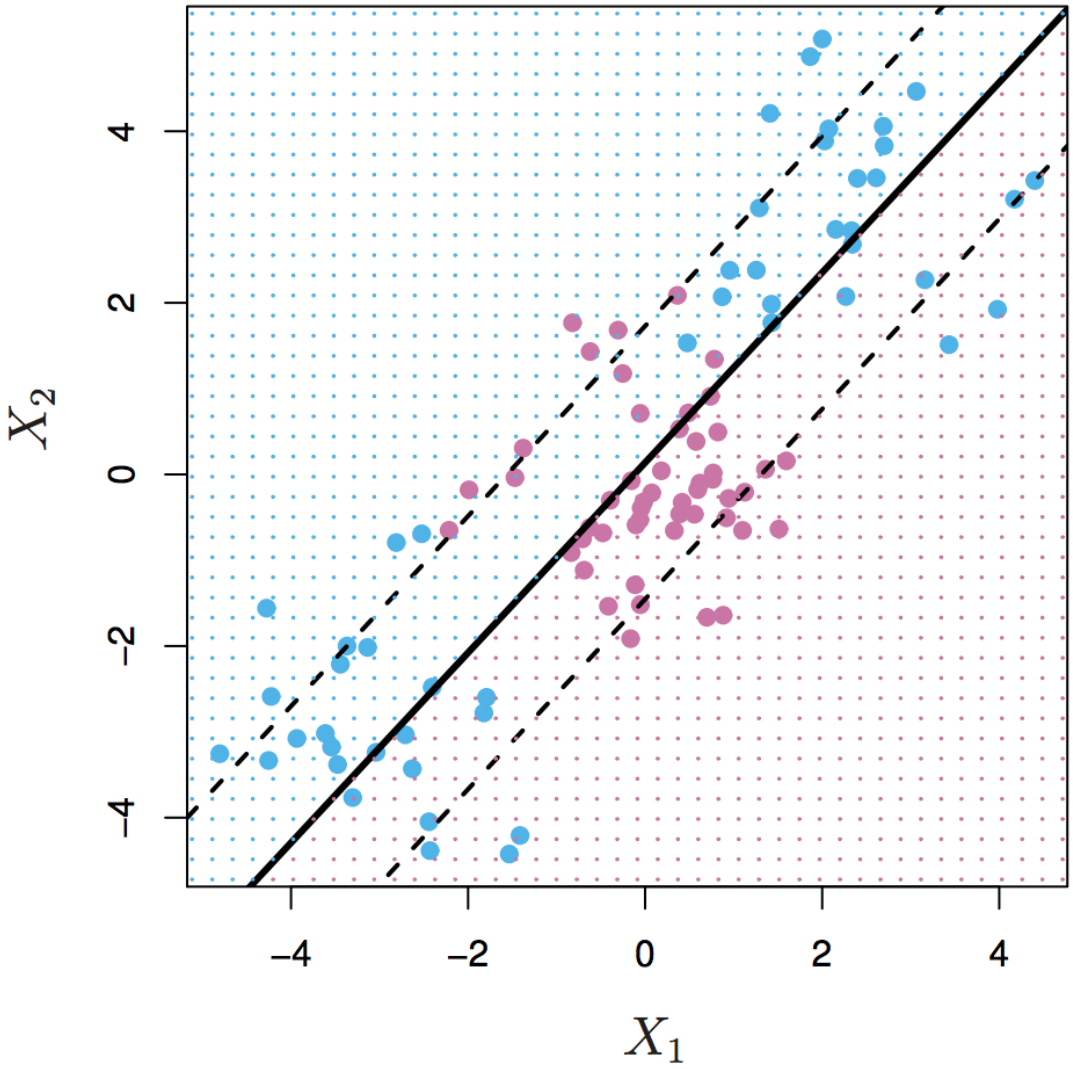
$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \alpha_i K(x, x_i) \quad \leftarrow \text{SVM, since using Kernels now}$$

$$K(x_i, x_{i'}) = \sum_{j=1}^{p} x_{ij} x_{i'j} \qquad \text{Linear Kernel}$$

$$K(x_i, x_{i'}) = (1 + \sum_{j=1}^{p} x_{ij} x_{i'j})^d \qquad \text{Polynomial Kernel}$$

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2) \qquad \text{Radial Basis Function Kernel ("Gaussian")}$$

SVMs – uh.....



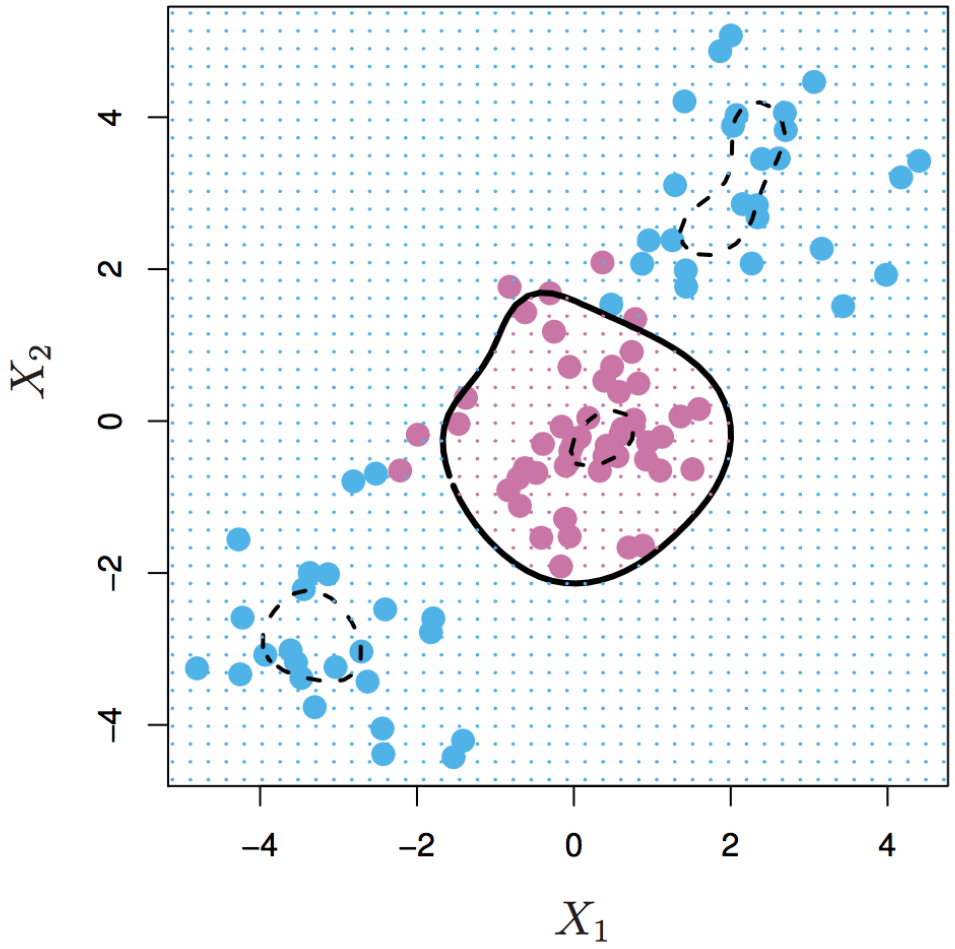
Sometime a linear boundary simply won't work, no matter what value of C.

The example on the left is such a case.

What to do?

SVMs

$$K(x_i, x_{i'}) = \exp(-\gamma \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2).$$



$$f(x) = \beta_0 + \sum_{i \in \mathcal{S}} \hat{\alpha}_i K(x, x_i)$$

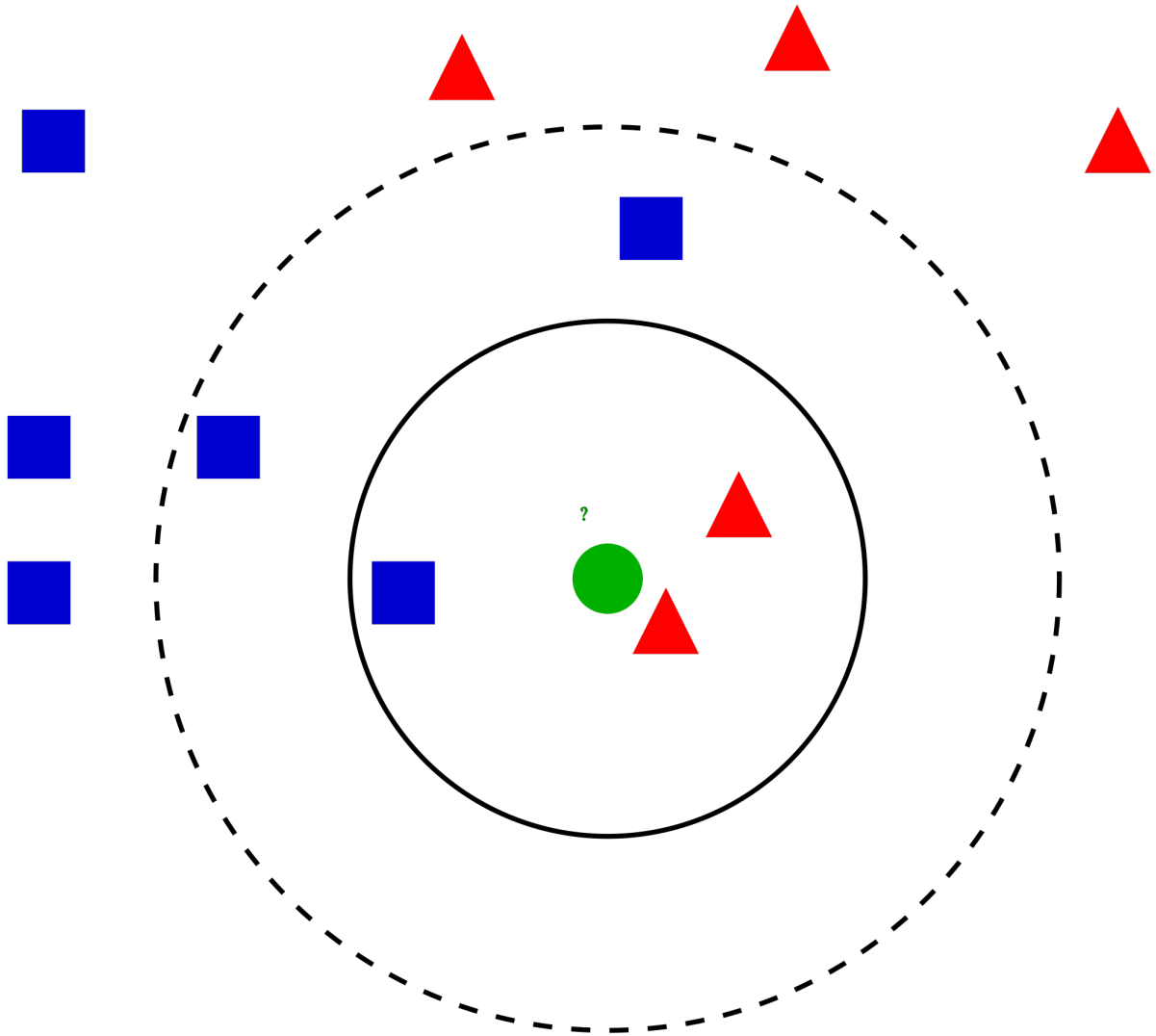
Implicit feature space; very high dimensional.

Controls variance by squashing down most dimensions severely

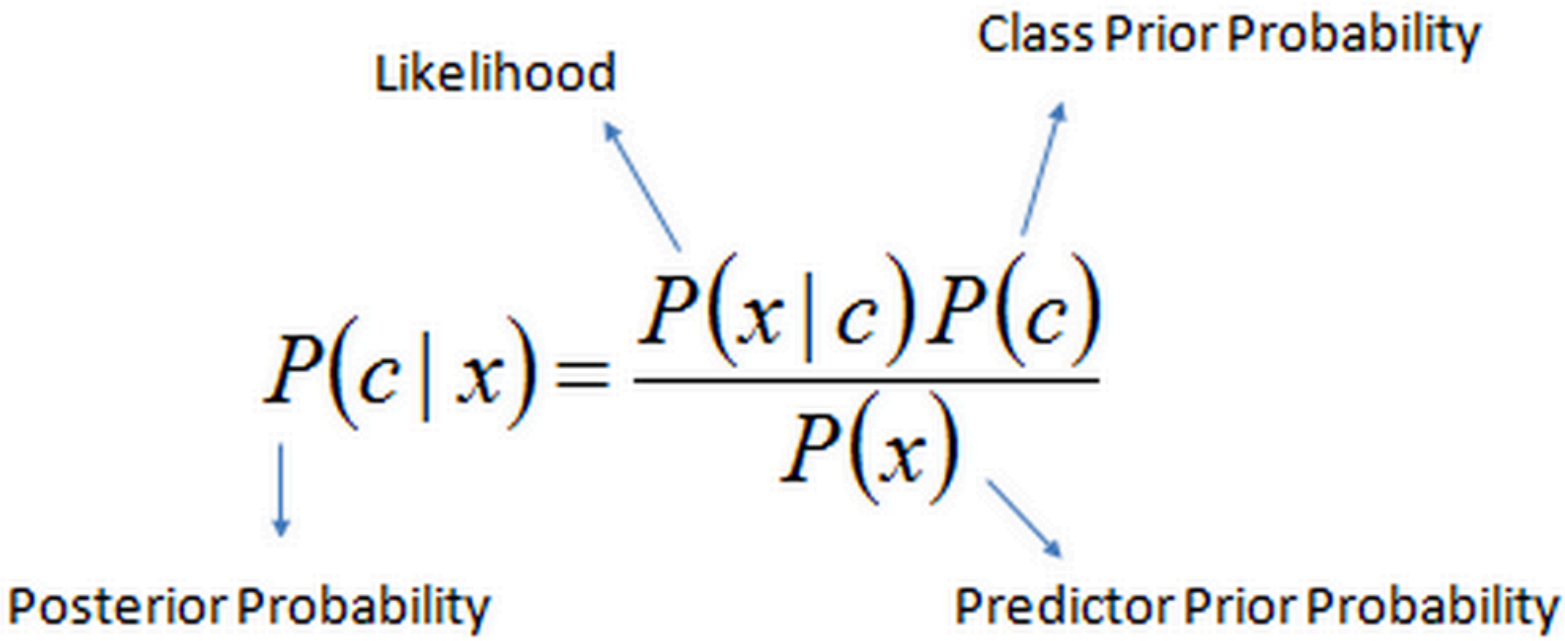
SVMs vs. Logistic Regression

- When classes are (nearly) separable, SVM does better than LR. So does LDA.
- When not, LR (with ridge penalty) and SVM very similar.
- If you wish to estimate probabilities, LR is the choice.
- For nonlinear boundaries, kernel SVMs are popular. Can use kernels with LR and LDA as well, but computations are more expensive.

kth Nearest Neighbor



Naïve Bayes



$$P(c \mid \mathrm{X}) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

how Naïve!

Naïve Bayes

Article	Occurrences of "ball"	Total # of words
Sports 1	5	101
Sports 2	7	93
Sports 3	0	122
Politics 1	0	39
Politics 2	0	81
Politics 3	0	142
Politics 4	0	77
Arts 1	2	198

$$P(\text{“ball”}|\text{sports}) = \frac{5+7+0}{101+93+122} = \frac{12}{316} = 0.038$$

$$P(\text{“ball”}|\text{politics}) = \frac{0+0+0+0}{39+81+142+77} = \frac{0}{339} = 0.0$$

$$P(\text{“ball”}|\text{arts}) = \frac{2}{198} = 0.010$$

Naïve Bayes

Which category for very short article “the giants beat the nationals”?

$$
\begin{aligned}
P(\text{sports}|X) = & P(\text{sports}) \\
& \times P(\text{“the”}|\text{sports}) \\
& \times P(\text{“giants”}|\text{sports}) \\
& \times P(\text{“beat”}|\text{sports}) \\
& \times P(\text{“the”}|\text{sports}) \\
& \times P(\text{“nationals”}|\text{sports})
\end{aligned}
$$

Don’t forget LaPlace smoothing….

$$
P(x|c) = \frac{(\#\text{ of times } x \text{ appears in articles of class } c) + \alpha}{(\text{total } \#\text{ of words in articles of class } c) + \alpha \cdot (\#\text{ of words in corpus})}
$$

Unsupervised Learning

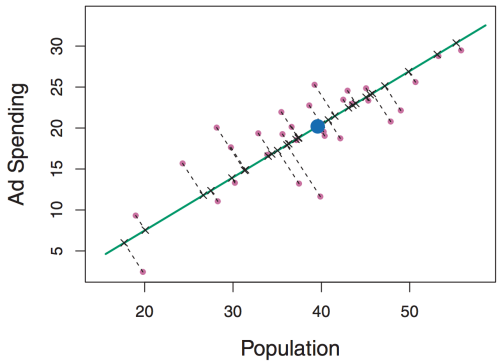
- No response variable, y
  - Just based on predictors, X_1, X_2, X_3, …, X_p
- A fuzzy endeavor...
  - Not cross-validating to choose best "model" in usual sense
  - Not cross-validating to know how well you're doing
- Can be useful as
  - ✓ preprocessing step for supervised learning
  - ✓ better understand features

Unsupervised Learning

Two most common and contrasting unsupervised techniques

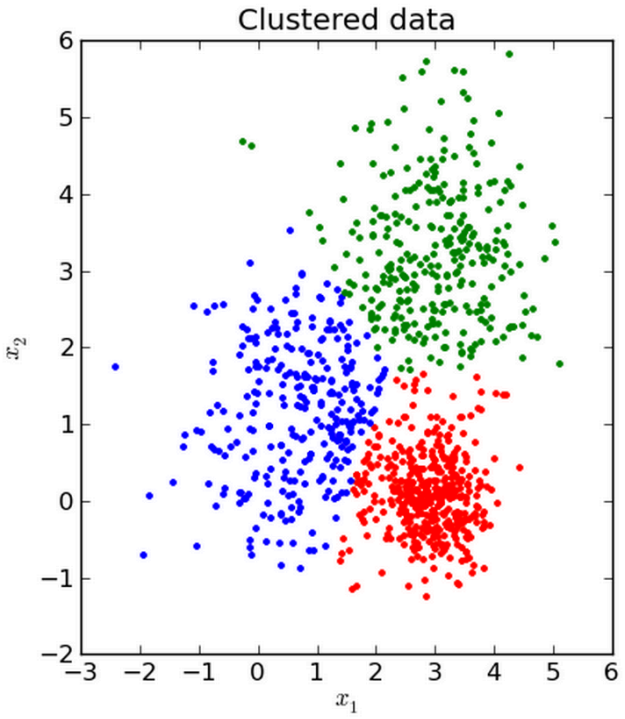
PCA

Low-dim representation of data that explains good fraction of variance



Clustering

Find homogenous subgroups among data



Curse of Dimensionality

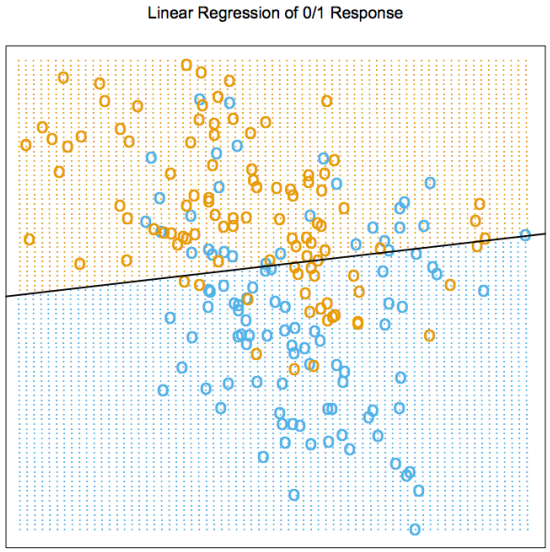
First let's take a detour and re-visit Linear Regression and k-th Nearest Neighbor

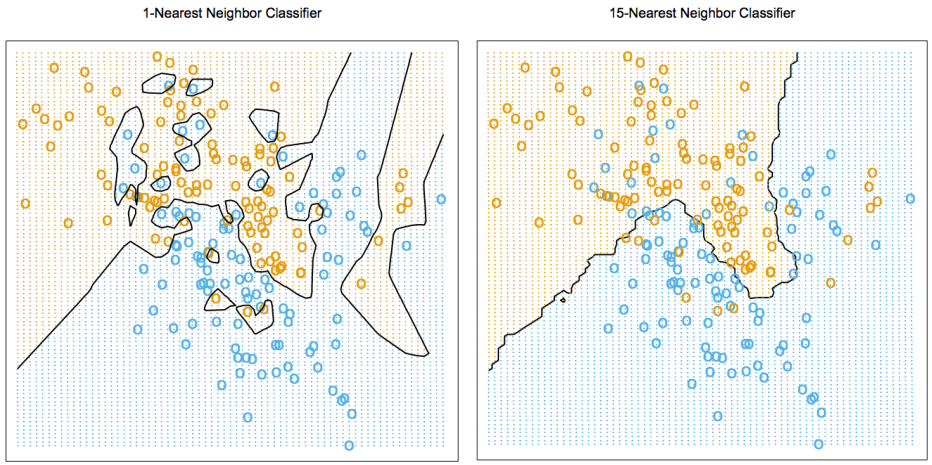
Linear Models

- Very structured
- Stable but possibly inaccurate
- Low Variance, High Bias

k-th Nearest Neighbor

- Very mildly structural
- Often accurate, but unstable
- High Variance, Low Bias





Curse of Dimensionality

- kNN is problematic in high-dim spaces
  - Though can be pretty good for p<=4 and N on the large side
- Nearest neighbors can be "far" in high dimensions
- Need to get a reasonable fraction of the N values of y_i to average to bring down the variance

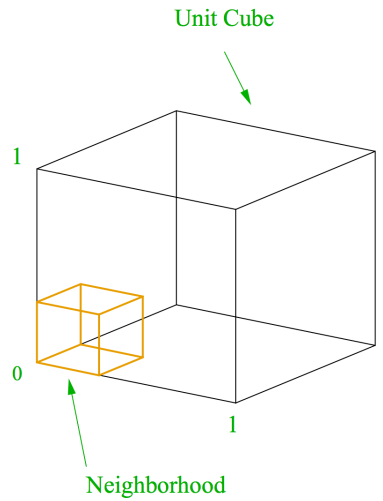
Okay….what's "far"?
To start, let's consider 10% to be a reasonable fraction

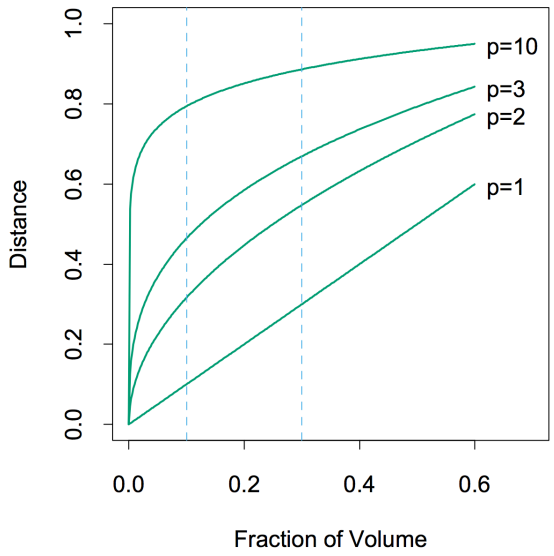
Curse of Dimensionality

Another way to think about dimensionality and its curse

- Hyper-cubical neighborhood about target point to capture fraction ***v*** of the the unit volume
- Expected edge length will be:

$$e_p(v) = v^{1/p}$$

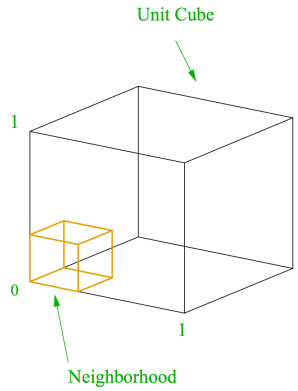


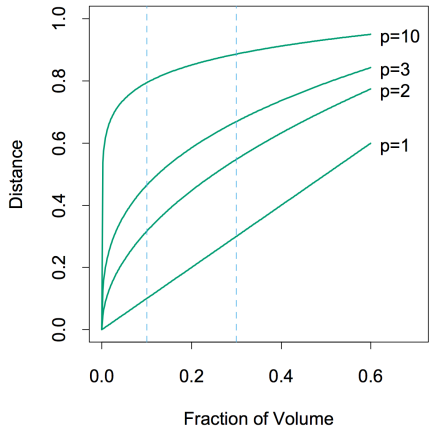


Can you work out out the 10% neighborhood for the unit cube case?

How much more data do we need to compensate for increasing dimensions (p)?

Curse of Dimensionality





Expected edge length

$$e_p(v) = v^{1/p}$$

Sampling density proportional to

$$N^{1/p}$$

p is dimensions of input space
N is number of points

Edge length example: Suppose interested in a v = 10% neighborhood

p = 1 → edge = (0.1)^1 = 0.1
p = 10 → edge = (0.1)^(1/10) = 0.794

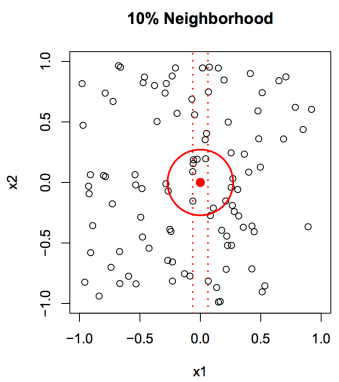
Sampling density example: How to achieve equivalent density in higher dimensions

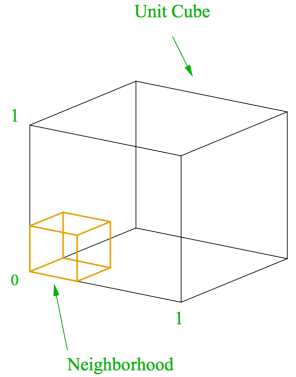
If $N_1 = 100$ represents dense sample for a single dim feature space
To achieve same density for 10 inputs, we need $N_{10} = 100^{10}$ points

Curse of Dimensionality - Takeaways

- kNN, or any method involving this sort of distancing, suffers majorly from curse of dimensionality
  - Nearest neighbors "far" in high dimensions (even for p = 10)
  - As we'll see, **k-means** and **hierarchical clustering** fall prey to curse.

- We can mathematically think of idea of "far" and sparsity of points in high dimensions using both radii approach and hypercube approaches





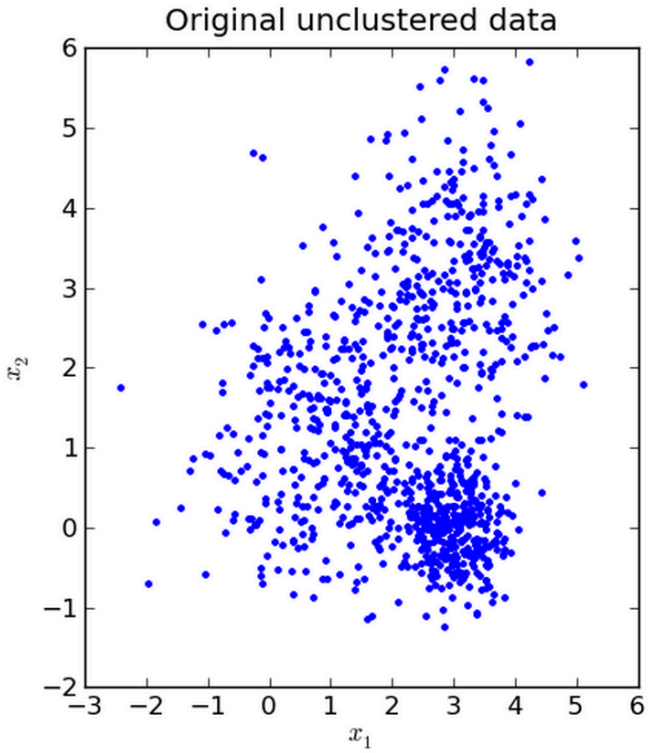
- It takes **a lot** of data to make up for increase in dimensions

Expected edge length $e_p(v) = v^{1/p}$

Sampling density $\propto$ to $N^{1/p}$

What is clustering?

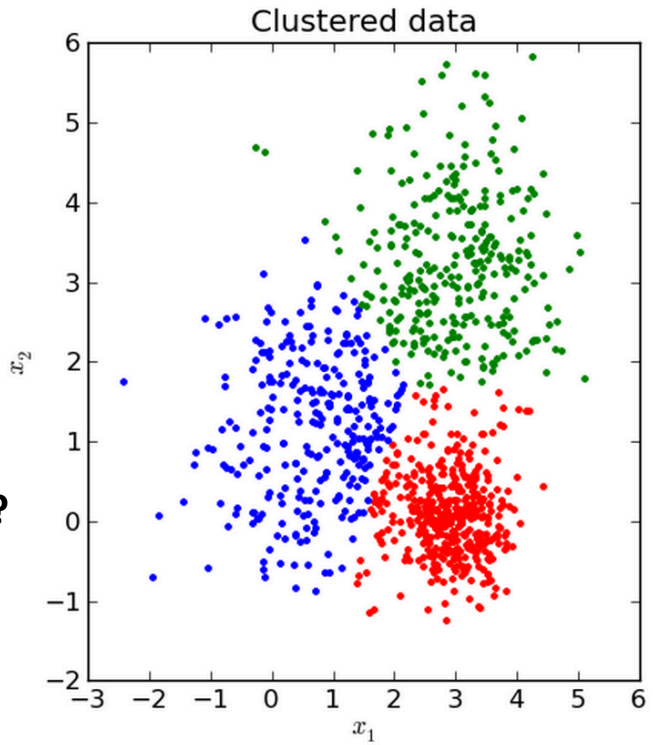
Divide data into distinct subgroups such that observations within each group are quite similar



How many subgroups?

What's considered similar?

How am I even doing this?



Two most popular approaches

- K-means
- Hierarchical clustering

Things to know

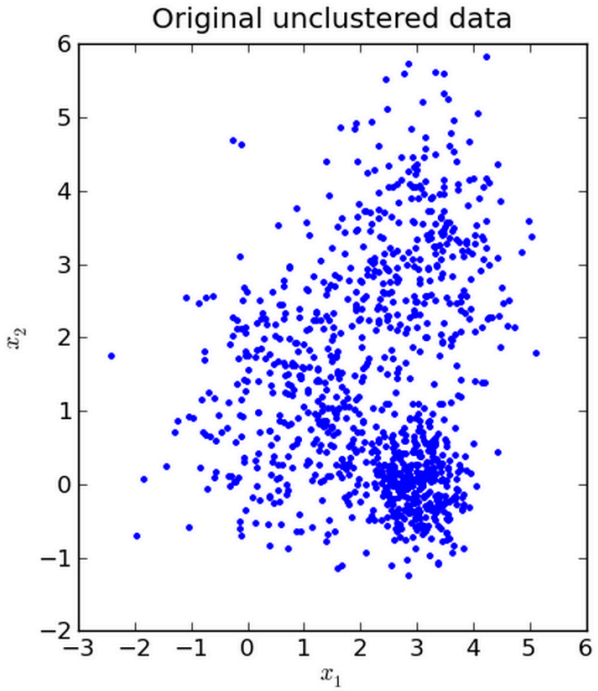
- Algorithm
- Choosing K

K-means

Idea: Want "within-cluster variation" to be small

Suppose: A fixed K, say K=3. Want to assign each of *n* data point to one of 3 clusters, such that "within-cluster variation" is smallest

- There are K^n possible choices! Pretty unwieldy



K-means

Altogether, we're picking C_1, … C_K such that

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

But again the problem is that there are K^n ways.
Too many!

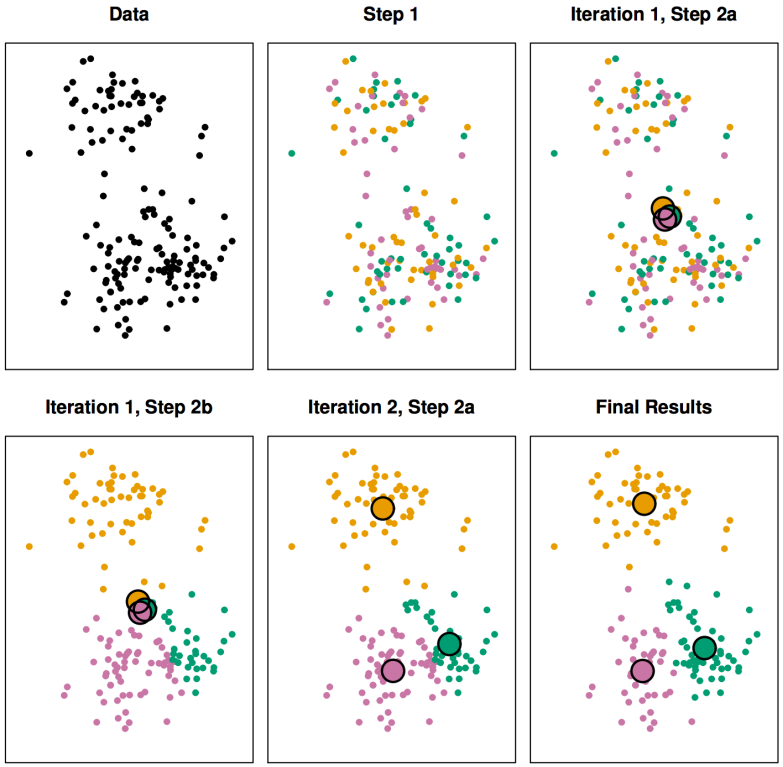
K-means algorithm

(1) Randomly assign number, from 1 to K, to each data point.

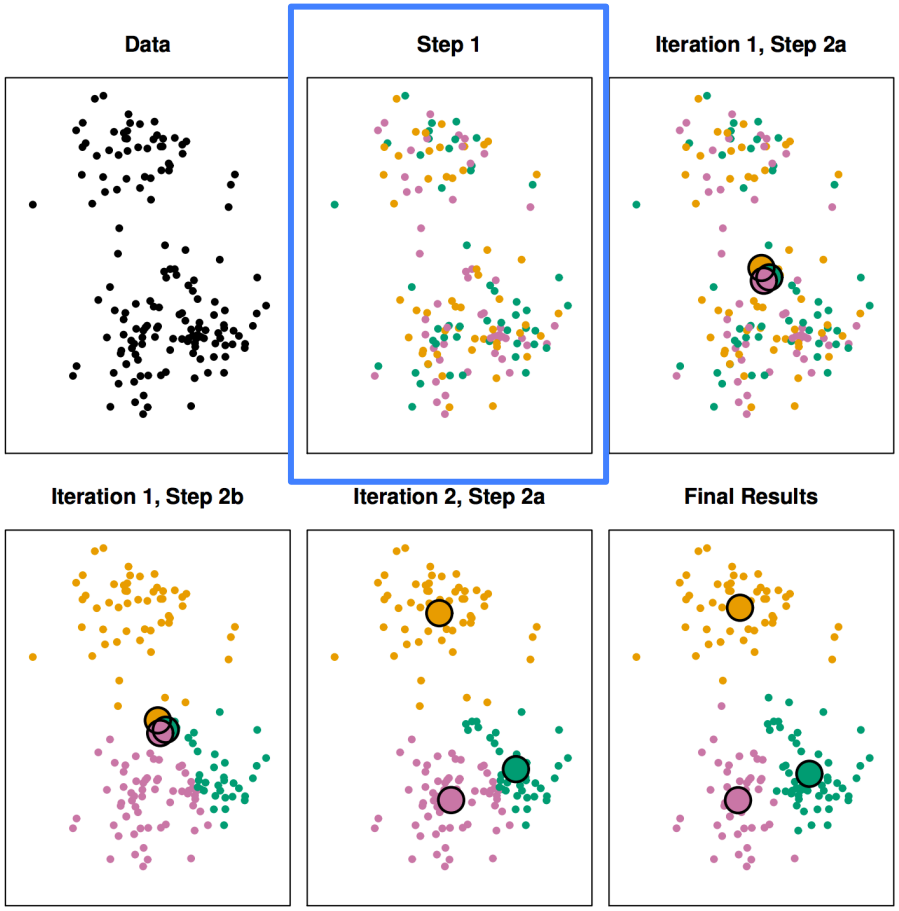
(2) Repeat until cluster assignments stop changing

a. For each of K clusters, compute cluster **centroid** by taking vector of p feature means

b. Assign data point to cluster for which centroid is closest (Euclidean)



K-means algorithm



Finds local optimum!

Results depend on random initialization

Solution

Try multiple initializations and pick one with lowest

$$\underset{C_1,\ldots,C_K}{\text{minimize}} \left\{ \sum_{k=1}^{K} \frac{1}{|C_k|} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2 \right\}$$

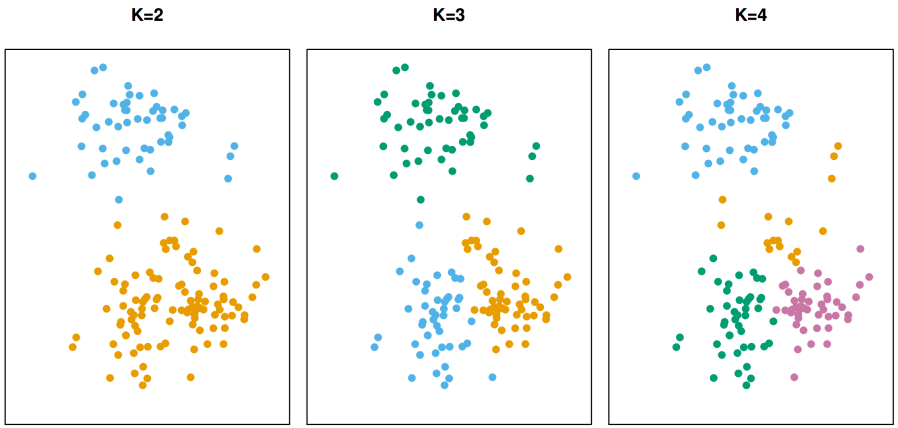
* Also could consider smarter initializations such as kmeans++ http://en.wikipedia. org/wiki/K-means%2B%2B

Choosing K

- No easy answer
- A fuzzy endeavor
  - May just want K similar groups
  - But more often, want something useful or interpretable that exposes some interesting aspect of data
    - Presence/absence of natural distinct groups
    - Descriptive statistics about groups
  - Ex. Are there certain segments of my market that tend to be alike?
    - Ex. middle-aged living in suburbs who log-in infrequently

Choosing K

- Fuzziness aside, there are many methods we can employ to choose K
- Three popular ones
  - "Elbow" method
  - GAP statistic
  - Silhouette Coefficient



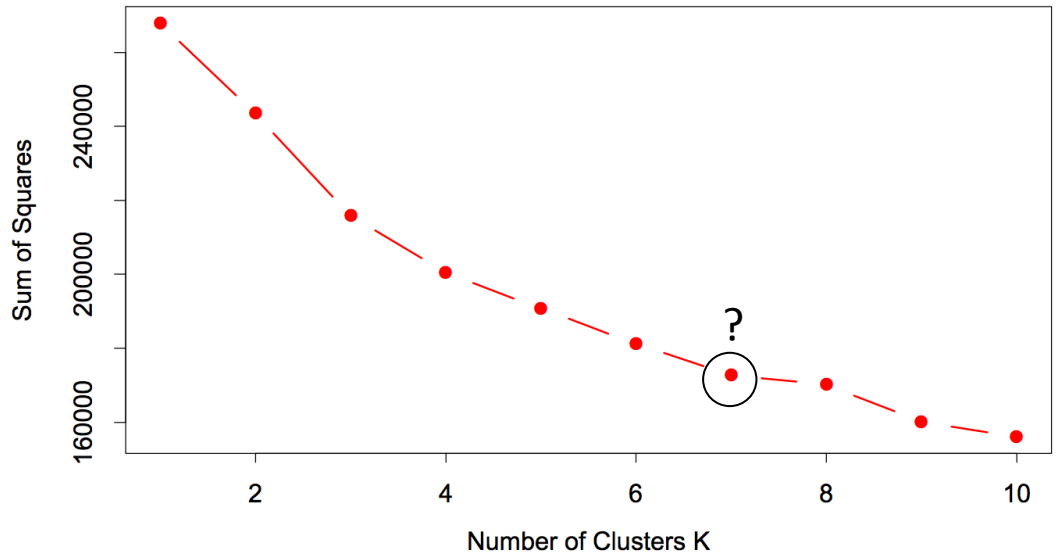
Choosing K – "Elbow" method

- Same Idea: Choose a number of clusters so that adding another cluster doesn't give us that much more

$$W(C) = \frac{1}{2}\sum_{k=1}^{K}\sum_{C(i)=k}\sum_{C(i')=k}||x_i - x_{i'}||^2$$

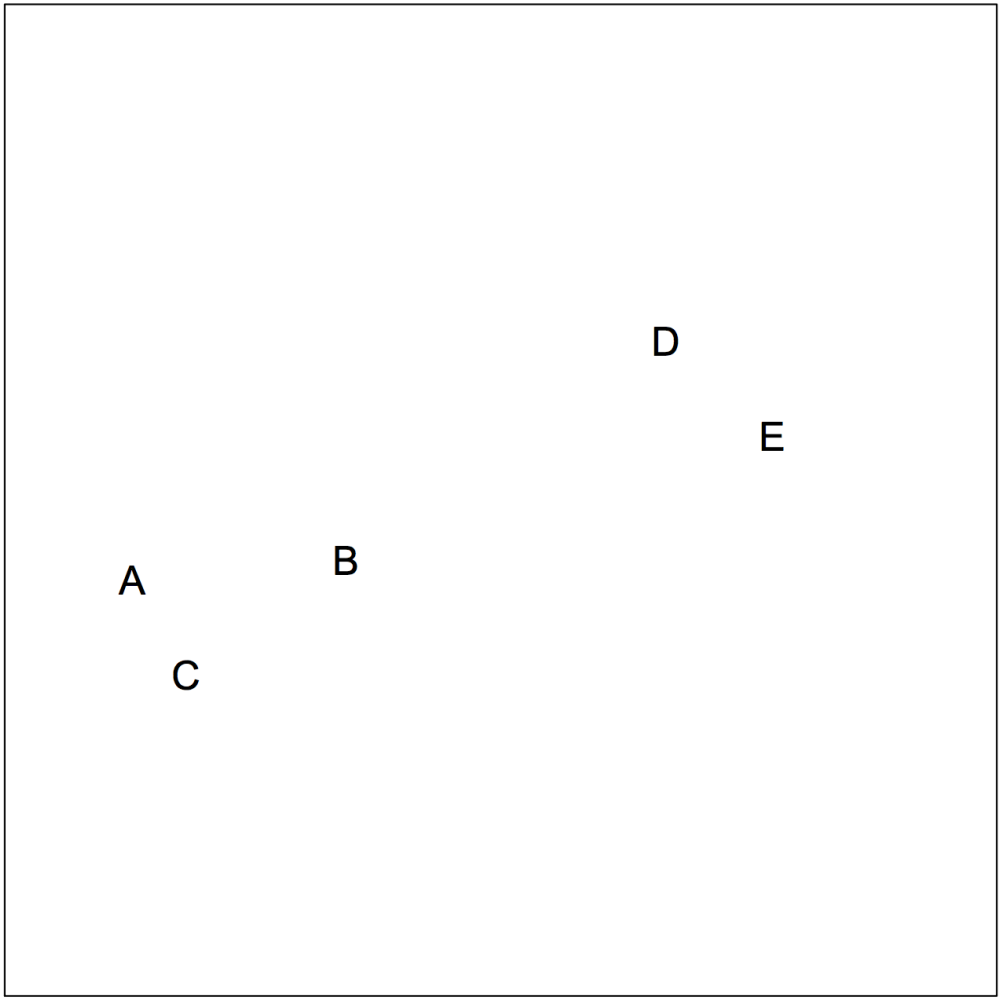
Within Cluster Point Scatter

A natural loss function is the sum pairwise distances of the points within each cluster, summed over all clusters.

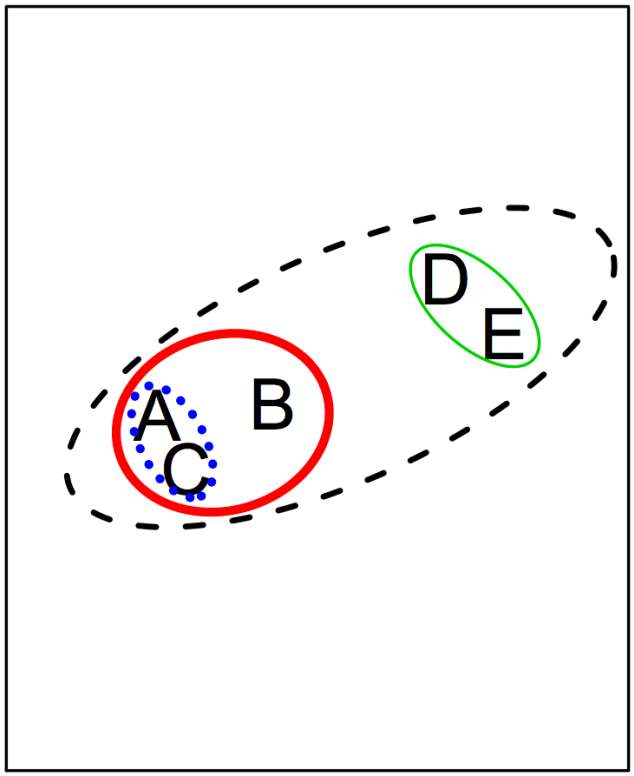


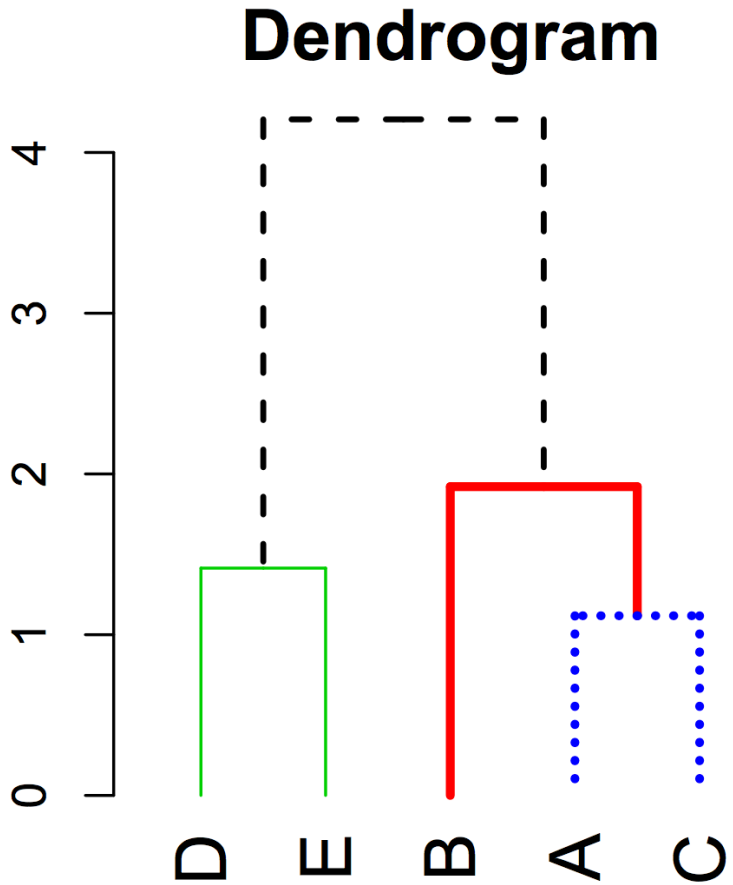
Hierarchical Clustering

Hierarchical Clustering



Hierarchical Clustering





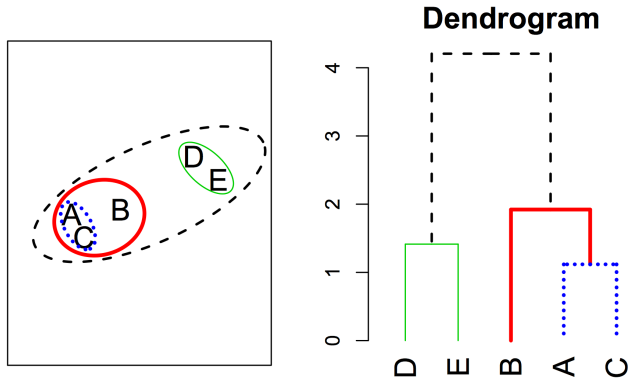
Hierarchical Clustering

Algorithm

(1) Each point as its own cluster

(2) Merge closest clusters

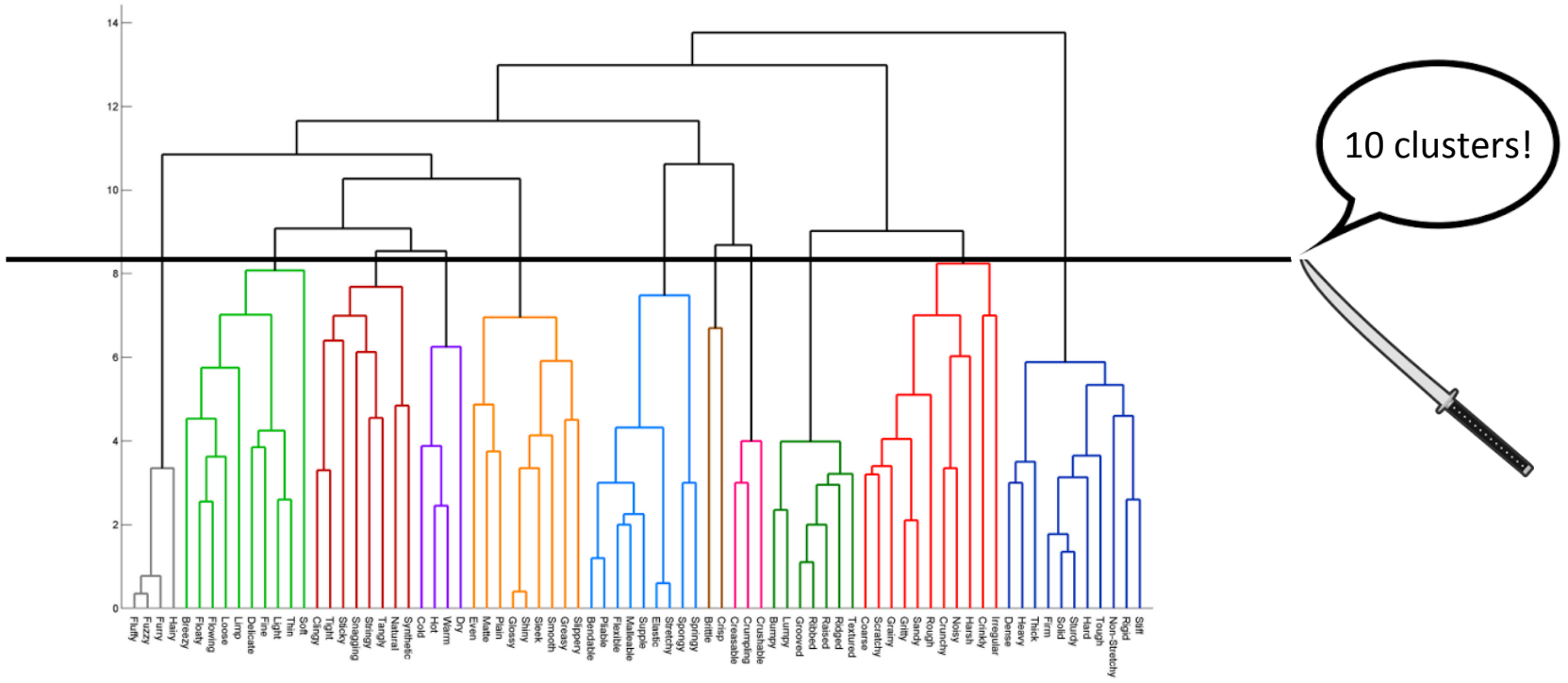
(3) End when all points in single cluster



Notice

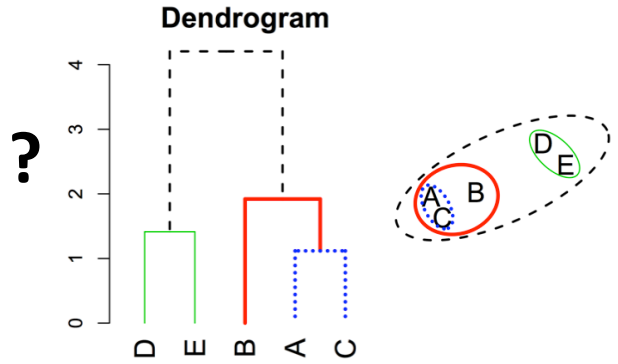
- Skipped over the notion of “distance” between clusters
- Height of fusion tells you how close clusters are!
  - A and C are pretty close, at around 1.2
  - Red and Green are not that close, fusing at around 4.1

Varying K



- In contrast to K-means, don't have to choose K from the start!
  - Depending on where precisely we cut, we have anywhere from 1 to n clusters
- Choosing K: Can again use Elbow Method, Gap Statistic, Silhouette
  - But notice the heights give you sense of separation of clusters depending on cut.

Distance between two clusters?



☺ Tends to be balanced → (more commonly used)

Extended trailing clusters → (less commonly used)

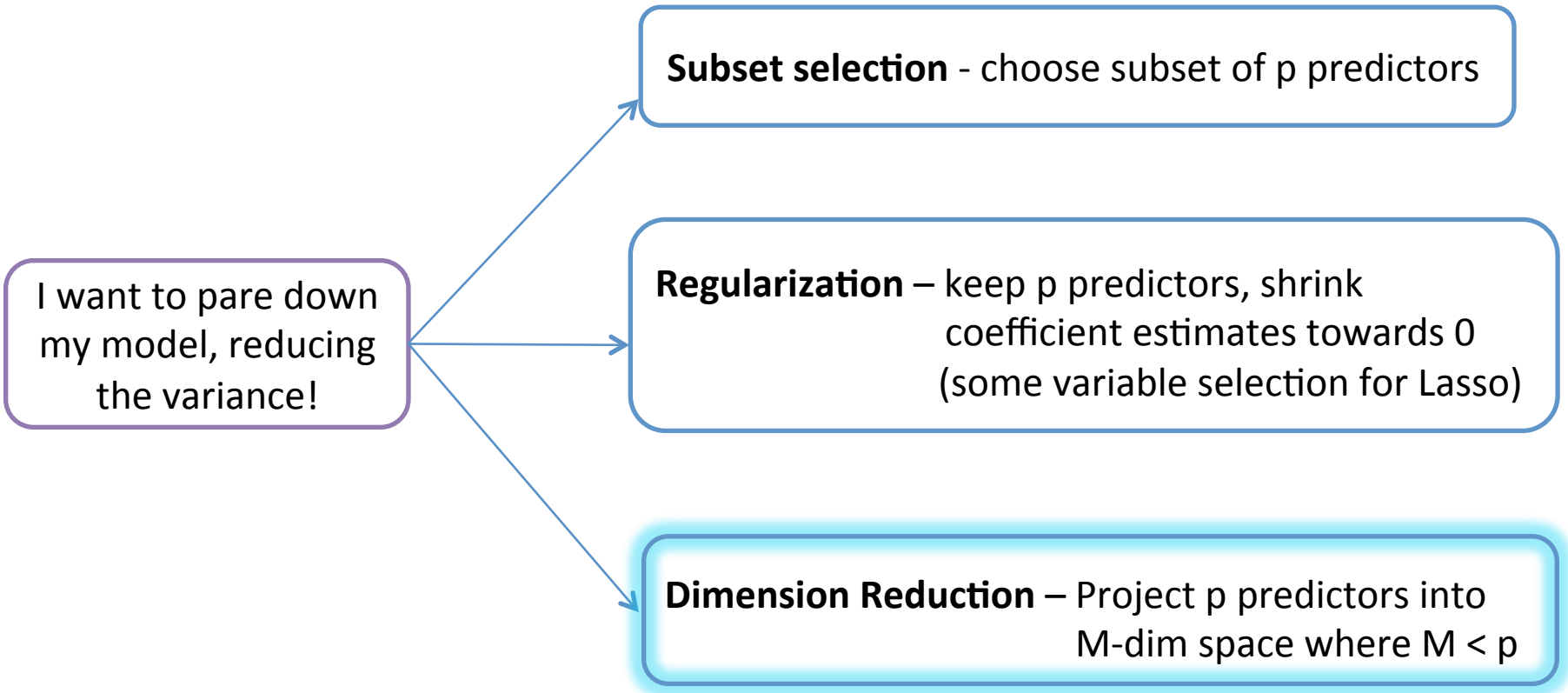
☺ Tends to be balanced → (more commonly used)

(not as commonly used, though popular in Genomics)

Linkage	*Description*
Complete	Maximal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *largest* of these dissimilarities.
Single	Minimal intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *smallest* of these dissimilarities. Single linkage can result in extended, trailing clusters in which single observations are fused one-at-a-time.
Average	Mean intercluster dissimilarity. Compute all pairwise dissimilarities between the observations in cluster A and the observations in cluster B, and record the *average* of these dissimilarities.
Centroid	Dissimilarity between the centroid for cluster A (a mean vector of length p) and the centroid for cluster B. Centroid linkage can result in undesirable *inversions*.

Managing the Bias-Variance Tradeoff with Linear Regression

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$



Steps to PCR

(1)

Y	X_1	X_2	X_3	X_4	…	X_p

(2) Define principal components Z_1, …, Z_m

1st principal component: $Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$

2nd principal component:

…

m-th principal component: $Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$

(3) Fit Linear Regression model with Z_1, …, Z_m

$$y_i = \theta_0 + \sum_{m=1}^{M} \theta_m z_{im} + \epsilon_i$$

Really, nothing unfamiliar yet! Except these ϕ vector things….

First, notice…

- That if Z_i is just a linear combination of all predictors $X_1, \ldots, X_p$ then a linear combination of Z_i is just a linear combination of $X_1, \ldots, X_p$

$$\sum_{m=1}^{M} \theta_m z_{im} = \sum_{m=1}^{M} \theta_m \sum_{j=1}^{p} \phi_{jm} x_{ij} = \sum_{j=1}^{p} \sum_{m=1}^{M} \theta_m \phi_{jm} x_{ij} = \sum_{j=1}^{p} \beta_j x_{ij}$$

- Dimensionality reduction: (p+1) → (M+1)

Calculating the 1st Principal Component… ϕ?

$$Z_1 = \phi_{11} X_1 + \phi_{21} X_2 + \ldots + \phi_{p1} X_p$$

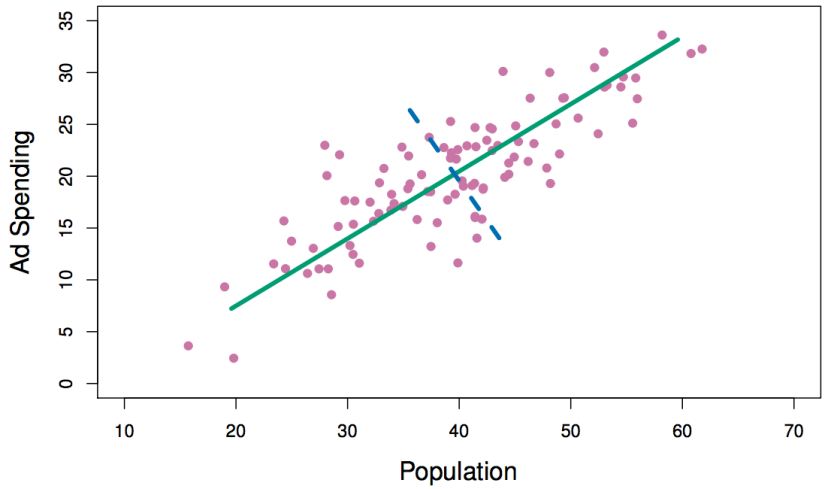
- PCA finds the **linear combinations of predictors that maximize the variance**
  - But for this to make sense we need to constrain the ϕs

$$\sum_{j=1}^{p} \phi_{j1}^2 = 1$$

  - And we need to standardize the X's by centering to mean 0 and scaling (typically each X to have standard deviation=1)

What happens if we don't constrain the weights?
What happens if a particular predictor X_k is measured in dollars? Millions of dollars?

Calculating the 2nd, 3rd, 4th…



- 2nd: Same exercise, except constrain Z_2 to be "uncorrelated" with Z_1 ⇔ constrain ϕ_1 orthogonal to the direction of ϕ_2
- 3rd: orthogonal to 1st and 2nd
- 4th, 5th, … same idea…

PCR: Troubles of multi-collinearity, begone! ☺

Proportion of Variance Explained

- Assuming the X's have been centered to mean 0 and scaled, we have total variance in the dataset

$$\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$$

- And total variance explained by the m-th component

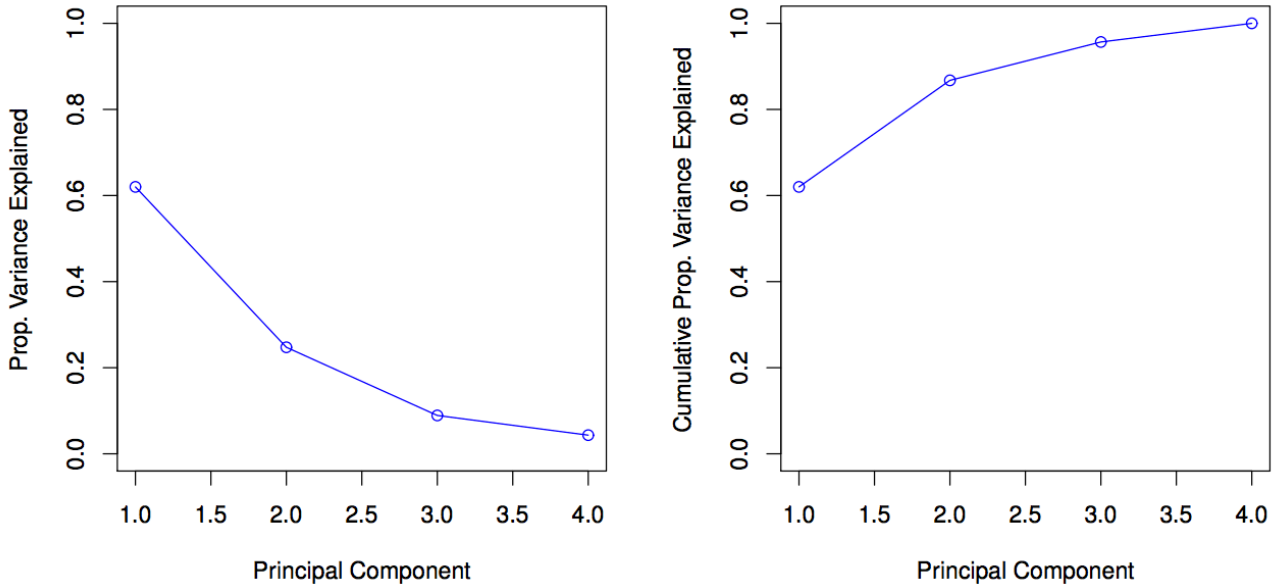
$$\mathrm{Var}(Z_m) = \frac{1}{n} \sum_{i=1}^{n} z_{im}^2$$

- Can be shown that $\sum_{j=1}^{p} \mathrm{Var}(X_j) = \sum_{m=1}^{M} \mathrm{Var}(Z_m)$ where $M = \min(n-1, p)$

Proportion of Variance Explained

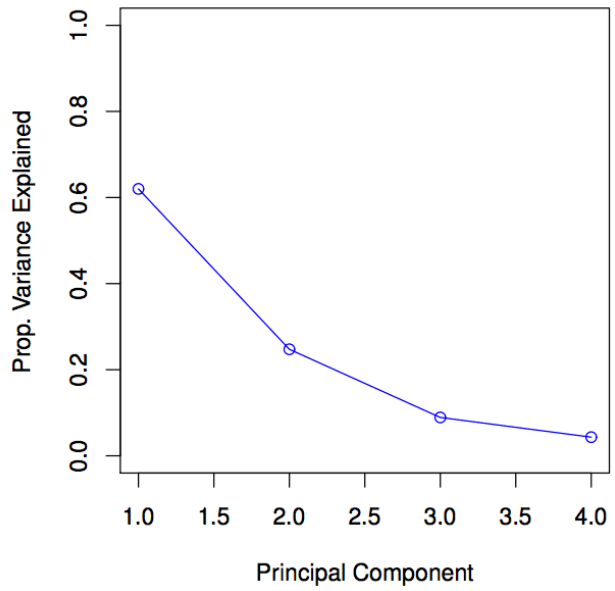
- PVE of m-th component

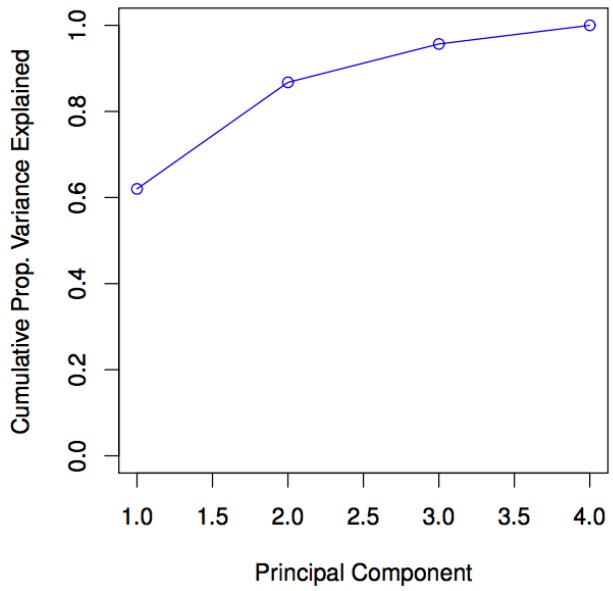
$$0 < \frac{\sum_{i=1}^{n} z_{im}^2}{\sum_{j=1}^{p} \sum_{i=1}^{n} x_{ij}^2} < 1$$



Proportion of Variance Explained

- Back to our original goal, which is predicting y!
- We've constructed principle components $Z_1, \ldots, Z_m$. How do we decide how many to use?





Your good friend, cross-validation ☺

We can think of the number of components as a "tuning parameter", sort of like λ in Lasso and Ridge