Data Mining
W4240 Sections 001, 003/004

L. A. Hannah

Columbia University, Department of Statistics

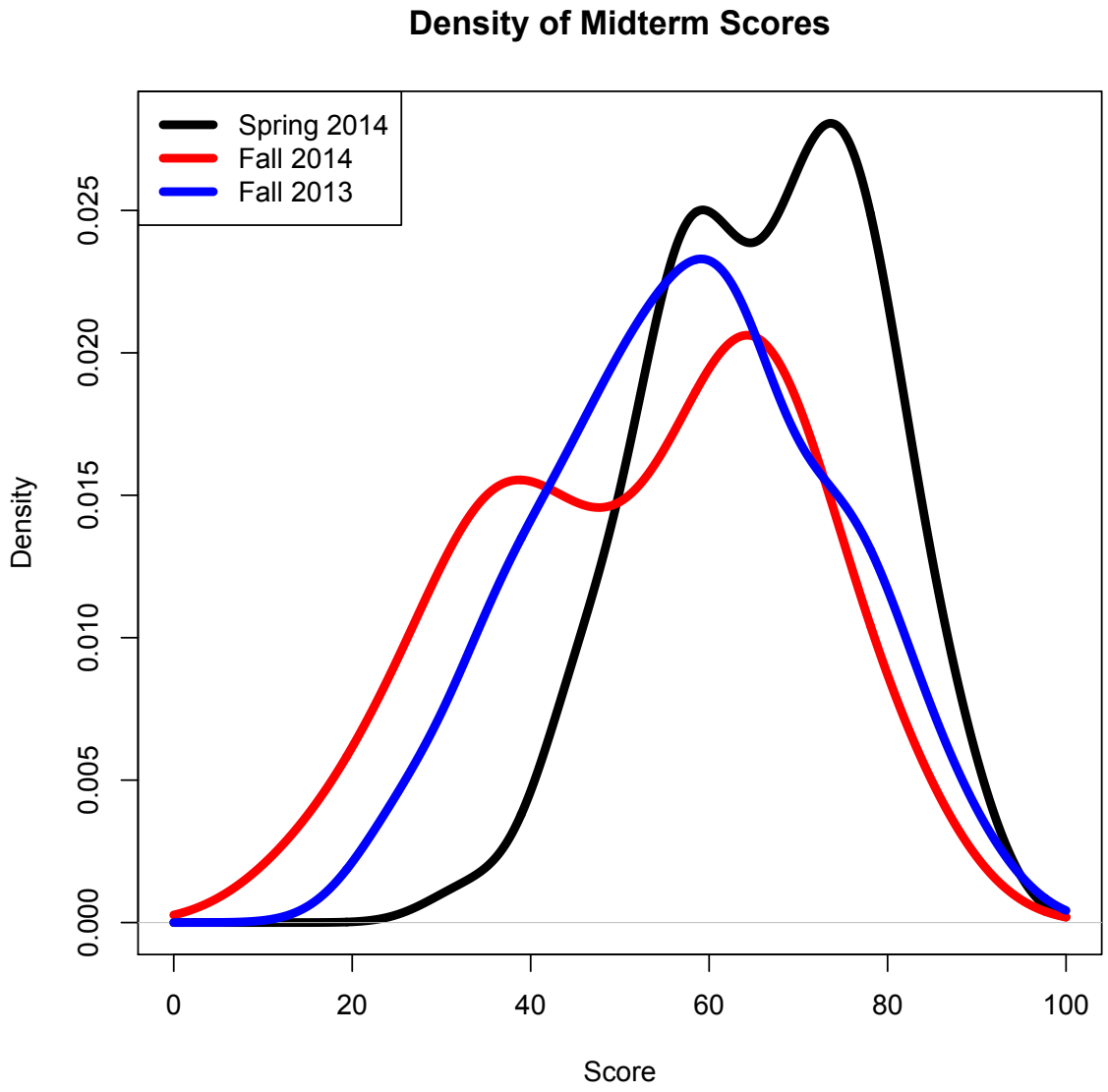
October 23, 2014

Outline

Outline

Midterm

Scores were low:



Midterm

Course grade: 30% homework, 30% midterm, 40% final

Usual curve:

A	A-	B+	B	below B
88+	85 to 87.99	80 to 84.99	70 to 79.99	discretion

Note: I reserve the right to change grade levels depending on tests, class performance, etc.

Midterm

If you are worried about your grade:

- switch to P/F (promise: 35% will be F cutoff)
- drop (GSAS deadline has passed, SEAS deadline is 11/14)
- withdraw (shows "W" on transcript)

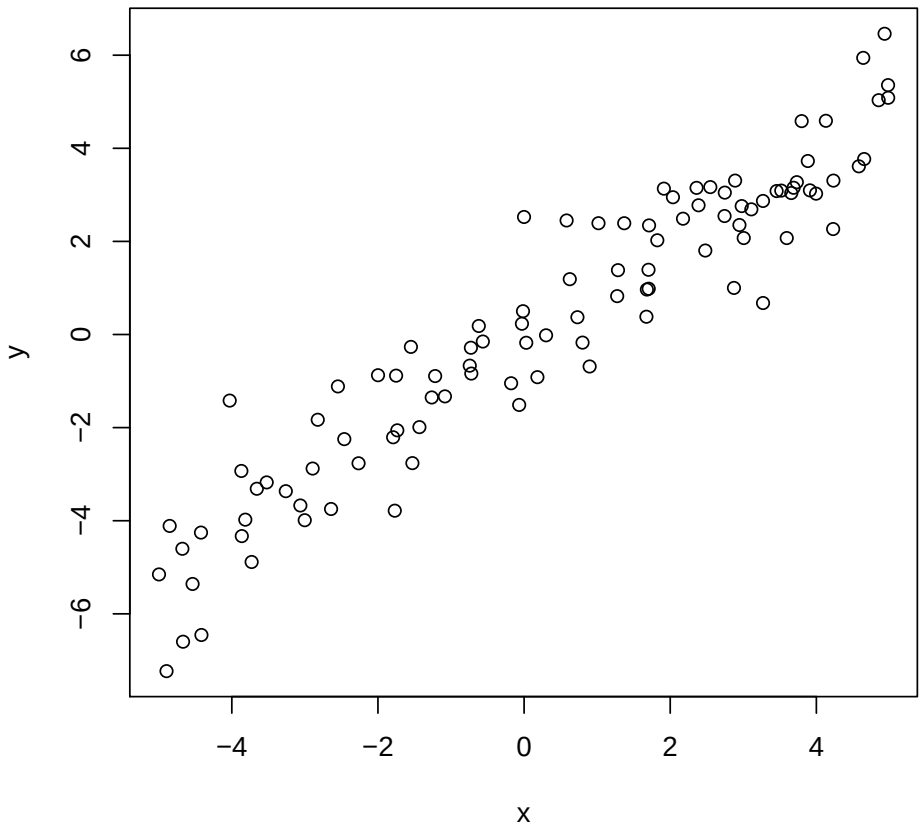
Homework 4

Homework 4 is now posted

- more code, less coding
- you will implement a naive Bayes filter to predict authors of Federalist Papers
- papers are in raw .txt form
- we give you code to clean and transform into bag of words
- ...and some code to transform bag of words into log probabilties
- you get to write code that turns log probabilities into naive Bayes classifier
- data in homework .zip

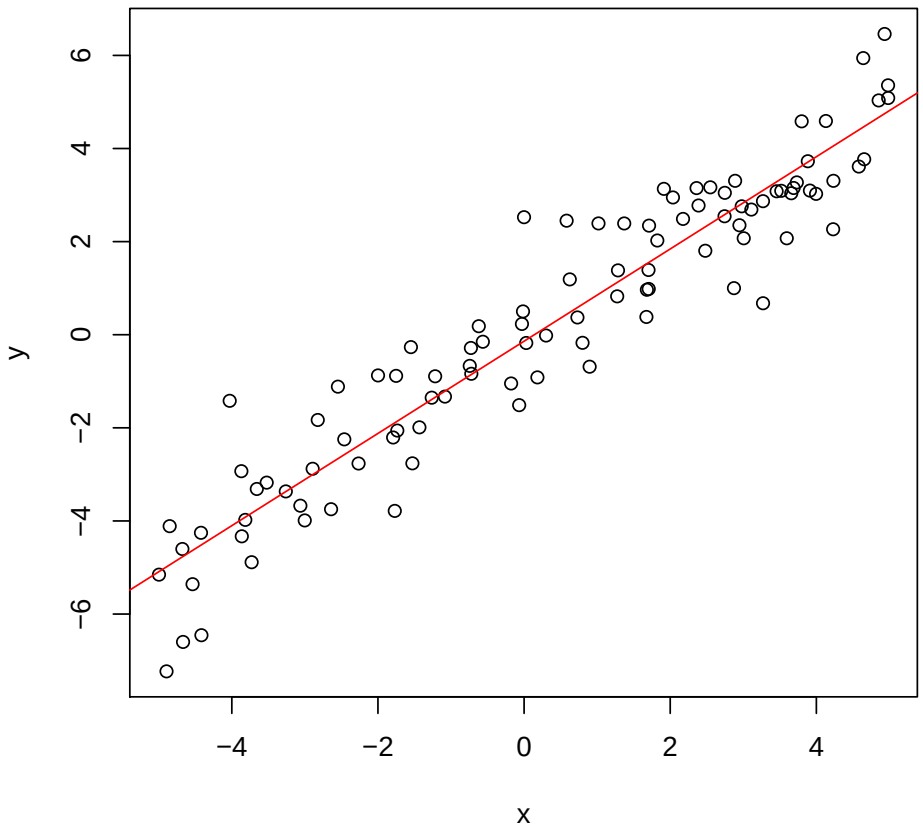
Outline

Linear Regression



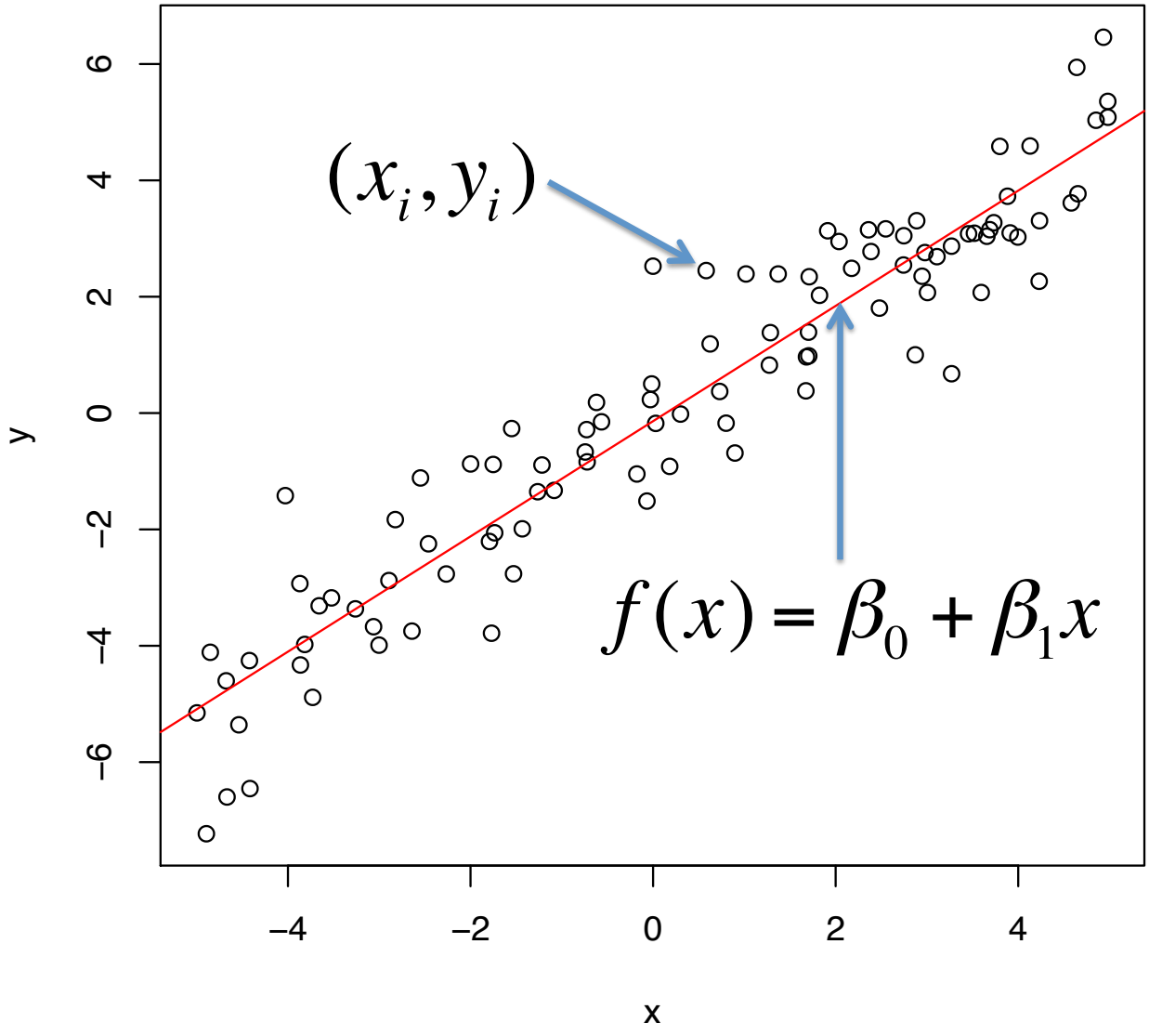
Training data are the set of inputs and outputs, $\mathcal{T} = \{(x_i, y_i)\}_{i=1}^{n}$

Linear Regression



In *linear regression*, the goal is to predict y from x using a linear function

Linear Regression



Linear Regression

Let's begin with some linear regression in R.

```
> n <- 100
> p <- 95
> x <- rnorm(n*p)
> dim(x) <- c(n,p)
> y <- x[,1] - 1.2*x[,2] + rnorm(n)
> fit.lm <- lm(y ~ x)
```

What are the coefficients? What about the residuals? Let's do this a few times.

High Dimensional Data

This is an example of a high-dimensional problem: $n \approx p$.

What are some legitimate assumptions for this type of problem?

- how many covariates actually matter?
- why would some not matter?
- should we fit a simple model or a complex model?
- how can we do it?

High Dimensional Data

This is an example of a high-dimensional problem: $n \approx p$.

What are some legitimate assumptions for this type of problem?

- how many covariates actually matter?
- why would some not matter?
- should we fit a simple model or a complex model?
- how can we do it?

Note: this $n \approx p$ problem motivates *subset selection*, but it is useful in many settings.

Outline

Subset Selection

Pick the best k ($\leq p$) covariates to use in linear regression

Subset Selection

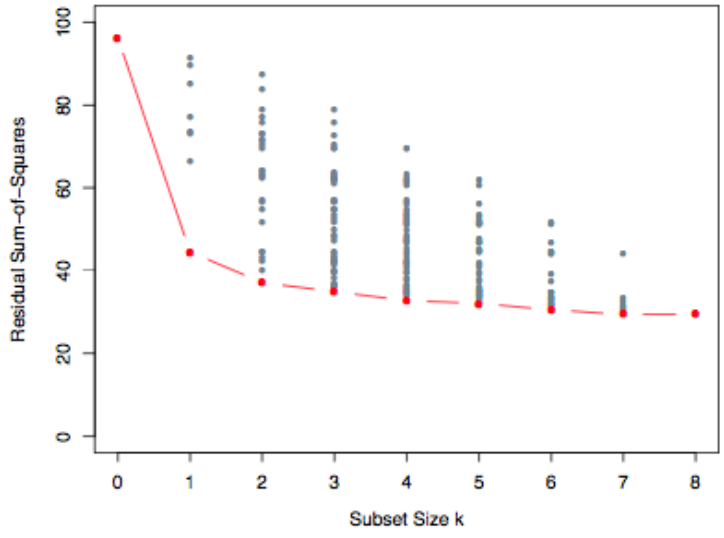
Pick the best k ($\leq p$) covariates to use in linear regression

Why?

- *Predictive Accuracy:* Linear least squares estimator has low bias, high variance. Reduce number of covariates, get a bit more bias but much less variance.
- *Interpretability:* Which variables matter? Which do not? Interpretability allows your model to say something about the data vs. just giving a prediction.

Subset Selection

How to pick the best k ($\leq p$) covariates for linear regression?



Best Subset Selection:

- enumerate possible subsets in a smart way for each k
- for each k, select subset that minimizes RSS
- pick best k: cross-validation or other model selection methods
- good method for $p < 30$ or 40

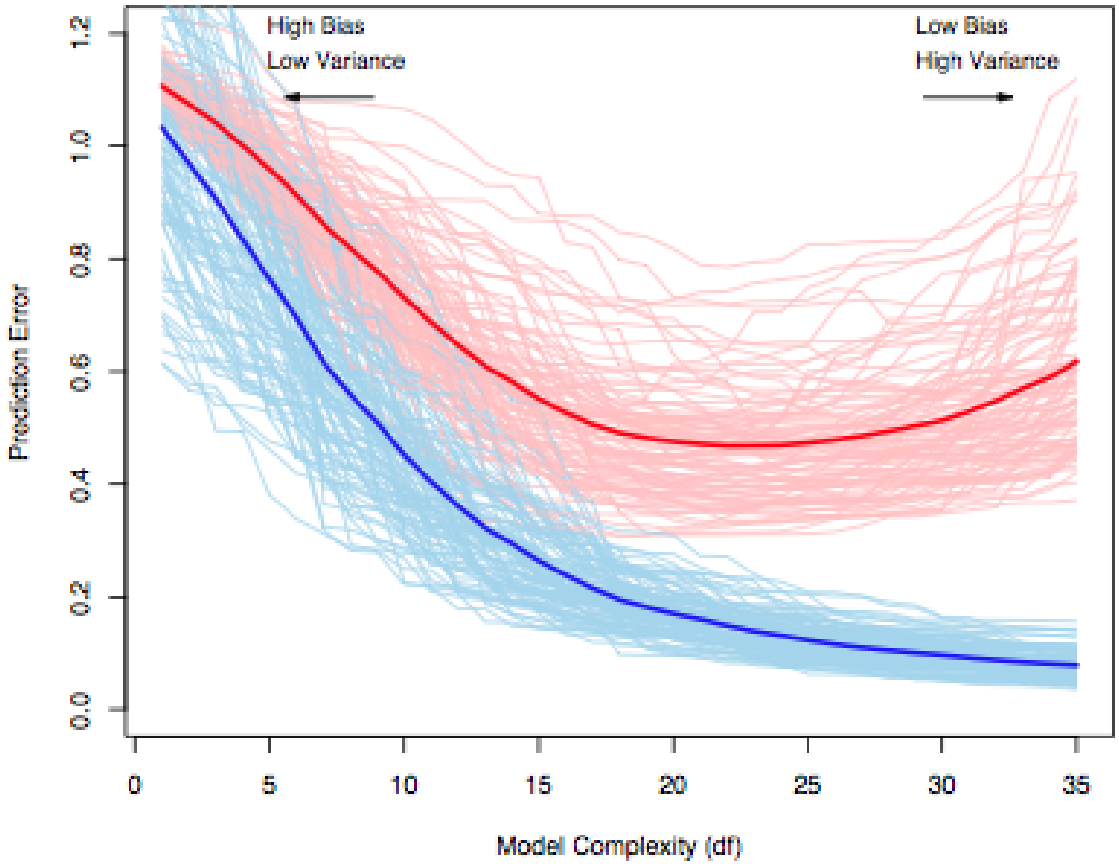
Model Selection

Cross-validation is not always the answer:

- here n is small compared to p *by definition*
- cross-validation may be too expensive since you have to fit all possible model combinations

Other methods like AIC and BIC adjust training error to try to estimate testing error

Training Error



The training error is *optimistic*: it under estimates the testing error. By how much? (Use corrected training error in place of testing error!)

Outline

Training Error Optimism

Training data: $\mathcal{T} = \{(x_1, y_1), \ldots, (x_n, y_n)\}$

New data: X^0, Y^0

Generalization error (extra-sample error):

$$\mathrm{Err}_{\mathcal{T}} = \mathbb{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0)) \,|\, \mathcal{T}]$$

Expected error (we asked about this Tuesday re: bootstrap):

$$\mathrm{Err} = \mathbb{E}_{\mathcal{T}}\mathbb{E}_{X^0, Y^0}[L(Y^0, \hat{f}(X^0)) \,|\, \mathcal{T}]$$

Training error:

$$\mathrm{Err}_{train} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}(x_i))$$

Training Error Optimism

To understand training error,

$$\mathrm{Err}_{train} = \frac{1}{n}\sum_{i=1}^{n} L(y_i, \hat{f}(x_i)),$$

look at *in-sample error* (not a training error!):

$$\mathrm{Err}_{in} = \frac{1}{n}\sum_{i=1}^{n} \mathbb{E}_{Y^0}[L(Y^0, \hat{f}(x_i)) \,|\, \mathcal{T}]$$

(Fix covariates, randomize responses.)

The *optimism* is the difference between Err_{in} and Err_{train}:

$$\mathrm{op} \equiv \mathrm{Err}_{in} - \mathrm{Err}_{train}$$

The average optimism is the expectation over the training sets

$$\mathbb{E}_y(\mathrm{op})$$

Training Error Optimism

In-sample error vs. training sample error vs. extra-sample error:

- **Extra-sample error:** expected error over new covariates and new responses
  - need to approximate distribution of responses and covariates
  - (directly approximated by cross-validation)
- **In-sample error:** expected error over new responses for given covariates
  - current covariate sample approximates true distribution
  - expectation over *new* responses eliminates bias from correlation between observed responses and fitted responses
  - (approximates extra-sample error as n gets large)
- **Training sample error:** error averaged over training samples
  - correlation between y_i and $\hat{y}_i$ causes underestimate of error
  - but, hey, it is easy to compute

Training Error Optimism

Can show for loss functions,

$$\mathbb{E}_y(\text{op}) = \frac{2}{n}\sum_{i=1}^{n}\text{Cov}(\hat{y}_i, y_i)$$

If method overfits, this value will be high.

$$\mathbb{E}_y(\text{Err}_{in}) = \mathbb{E}_y(\text{Err}_{train}) + \frac{2}{n}\sum_{i=1}^{n}\text{Cov}(\hat{y}_i, y_i)$$

In the case of a linear model,

$$\mathbb{E}_y(\text{Err}_{in}) = \mathbb{E}_y(\text{Err}_{train}) + \frac{2p}{n}\sigma_\epsilon^2$$

Can use this to get in-sample estimates of prediction error

Estimating In-Sample Prediction Error

Model selection criteria vs. cross-validation:

- **Cross-validation:**
  - possibly more accurate
  - no need for asymptotic approximations (is n large enough to justify asymptotics?)
  - more flexible (can be used for things other than MLE)
- **Model selection criteria:**
  - often easy to compute
  - theoretically justifiable

Outline

Estimating In-Sample Prediction Error

In general, an estimate of the in-sample error is the training sample error plus an estimate of the optimism,

$$\hat{\mathrm{Err}}_{in} = \mathrm{Err}_{train} + \hat{\mathrm{op}}$$

Suppose that we use a log-likelihood loss function ($-$squared error $=$ Gaussian log-likelihood). The *Akaike Information Criterion* is an asymptotic approximation for Err_{in}:

$$-2\mathbb{E}[\log \Pr_{\hat{\theta}}] \approx -\frac{2}{n}\sum_{i=1}^{n} \log \Pr_{\hat{\theta}}(y_i) + 2\frac{d(\alpha)}{n}$$

Here $\hat{\theta}$ is the MLE estimate. The second term is an approximation of the bias between $2\mathbb{E}[\log \Pr_{\hat{\theta}}]$ and $\frac{2}{n}\sum_{i=1}^{n} \log \Pr_{\hat{\theta}}(y_i)$.

Estimating In-Sample Prediction Error

Let's compute the expectation when we have Gaussian errors:

$$-\frac{2}{n}\sum_{i=1}^{n}\log \Pr_{\hat{\theta}}(y_i) + 2\frac{d(\alpha)}{n}$$

$$= -\frac{2}{n}\sum_{i=1}^{n}\log\left[\frac{1}{\sqrt{2\pi\hat{\sigma}_\epsilon^2}}\exp\left\{-\frac{1}{2\hat{\sigma}_\epsilon^2}(m_{\hat{\theta}}(y_i)-y_i)^2\right\}\right] + 2\frac{d(\alpha)}{n}$$

$$(\text{the term } \log(2\pi\hat{\sigma}_\epsilon^2) \text{ is constant across models} \Rightarrow \text{removed})$$

$$= \frac{1}{n\hat{\sigma}_\epsilon^2}\sum_{i=1}^{n}(m_{\hat{\theta}}(y_i)-y_i)^2 + 2\frac{d(\alpha)}{n}$$

$$AIC(\alpha) = \text{Err}_{train}(\alpha) + 2\frac{d(\alpha)}{n}\hat{\sigma}_\epsilon^2$$

Here $\hat{\theta}$ is the MLE estimate. Choose α that minimizes $AIC(\alpha)$.

Estimating In-Sample Prediction Error

Are there other ways to estimate $\hat{\text{op}}$? Of course.

The *Bayesian Information Criterion* uses the approximation $\log(n)\frac{d(\alpha)}{n}\hat{\sigma}_\epsilon^2$ instead of $2\frac{d(\alpha)}{n}\hat{\sigma}_\epsilon^2$,

$$AIC(\alpha) = \text{Err}_{train}(\alpha) + 2\frac{d(\alpha)}{n}\hat{\sigma}_\epsilon^2$$

$$BIC(\alpha) = \frac{n}{\sigma_\epsilon^2}\left[\text{Err}_{train}(\alpha) + (\log n)\frac{d(\alpha)}{n}\hat{\sigma}_\epsilon^2\right]$$

BIC:

- chooses right model size as $n \to \infty$
- ...but chooses too simple models when n is small

AIC:

- chooses better models with small n
- ...but chooses too complicated models when n is large

Adjusted R^2

Recall from linear regression:

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$$

R^2 explains the reduction in variance of a model.... but a model with a large p might be overfitting.

We can adjust the R^2 for the number of explanatory terms relative to the number of data points: with more data, more explanatory terms are acceptable.

$$\bar{R}^2 = 1 - (1 - R^2)\frac{n-1}{n-p-1}$$

The adjusted R^2 corrects for the extra degrees of freedom associated with more predictors.

Estimating In-Sample Prediction Error

Model selection criteria vs. cross-validation:

- **Model selection criteria:**
  - often easy to compute
  - theoretically justifiable
- **Cross-validation:**
  - possibly more accurate
  - no need for asymptotic approximations (is n large enough to justify asymptotics?)
  - more flexible (can be used for things other than MLE)

Outline

Example: Prostate Data

Data in Prostate.txt (also available on ESL website)

Predictors (columns 1–8): lcavol (log cancer volume), lweight (log weight), age, lbph (log amount of benign prostatic hyperplasia), svi (seminal vesicle inversion), lcp (log capsular penetration), gleason, pgg45 (percentage of Gleason scores 4 or 5)

outcome (column 9): lpsa (level of prostate-specific antigen)

train/test indicator (column 10)

```
> prostate <- read.table("Prostate.txt",header=TRUE, sep="\t")
> names(prostate)
 [1] "X"       "lcavol"  "lweight" "age"
 [5] "lbph"    "svi"     "lcp"     "gleason"
 [9] "pgg45"   "lpsa"    "train"
> prostate.train <- prostate[prostate$train==T,2:10]
> prostate.test <- prostate[prostate$train==F,2:10]
```

Example: Prostate Data

```
> prostate.lm <- lm(lpsa ~ lcavol + lweight + age + lbph
   + svi + lcp + gleason + pgg45, data=prostate.train)
> # Other way:
> # prostate.lm <- lm(lpsa ~., data=prostate.train)
> # Exclude intercept by:
> # prostate.lm <- lm(lpsa ~ lcavol + lweight + age + lbph
   + svi + lcp + gleason + pgg45 - 1, data=prostate.train)
> y.pred.lm <- predict(prostate.lm,prostate.test)
> mean((y.pred.lm-prostate.test$lpsa)^2)
[1] 0.521274
```

Note: the data in ESL was scaled before use, so $\hat{\beta}$ differs

Best Subset Selection

Use the package `leaps`

```
> library(leaps)
> prostate.bss <- regsubsets(lpsa ~ ., data=prostate.train)
> # Let's see the outputs
> summary(prostate.bss)
> coef(prostate.bss,1:4)
> plot(prostate.bss, scale="bic")
> # Get a prediction
> coef.bss <- coef(prostate.bss,2)
> y.pred.bss <- coef.bss[1]
  + coef.bss[2]*prostate.test$lcavol
  + coef.bss[3]*prostate.test$lweight
> mean((y.pred.bss-prostate.test$lpsa)^2)
[1] 0.4924823
```

Forward and Backward Subset Selection

What happens if $p > 40$? We can't search all subsets...

Forward stepwise selection:

- start with intercept
- add in one predictor that improves the fit the most
- repeat until we run out of predictors
- select k through cross-validation, AIC, BIC, adjusted R^2
- "fit improvement" determined by $F-$statistics or AIC scores

This is called a *greedy algorithm*

Forward and Backward Subset Selection

Why greedy algorithms?

- computational: only search through $\mathcal{O}(p \min(n, p))$ subsets (at most)
- statistical: more constrained search means some additional estimator bias, but less variance

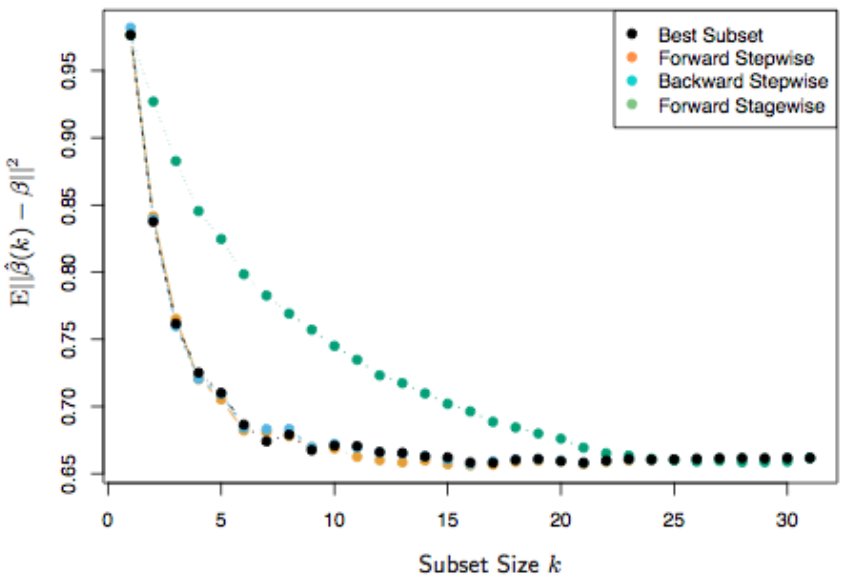


FIGURE 3.6. *Comparison of four subset-selection techniques on a simulated linear regression problem* $Y = X^T\beta + \varepsilon$. *There are* $N = 300$ *observations on* $p = 31$ *standard Gaussian variables, with pairwise correlations all equal to* 0.85. *For* 10 *of the variables, the coefficients are drawn at random from a* $N(0, 0.4)$ *distribution; the rest are zero. The noise* $\varepsilon \sim N(0, 6.25)$, *resulting in a signal-to-noise ratio of* 0.64. *Results are averaged over* 50 *simulations. Shown is the mean-squared error of the estimated coefficient* $\hat{\beta}(k)$ *at each step from the true* β.

Forward and Backward Subset Selection

Backward stepwise selection:

- start with all predictors
- remove one that contributes the least
- repeat until we are left with the intercept
- select k through cross-validation, AIC, BIC, adjusted R^2
- "fit improvement" determined by $F-$statistics or AIC scores
- note: only works if $n > p$

Forward and Backward Subset Selection

Use the function `step` (you can also use `regsubsets()` with `method="forward"` or `method="backward"`)

```
> prostate.fwd <- step(prostate.lm)
> summary(prostate.fwd)
> y.pred.fwd <- predict(prostate.fwd,prostate.test)
> mean((y.pred.fwd-prostate.test$lpsa)^2)
[1] 0.5165135
```

...or we can use `step(...  , direction="backward")`.