EECS E6690: Statistical Learning for Biological and Information Systems Lecture 4: PCA, Nonlinear Models and Model Validation

Prof. Predrag R. Jelenković
Time: Tuesday 4:10-6:40pm

Dept. of Electrical Engineering
Columbia University , NY 10027, USA
Office: 812 Schapiro Research Bldg.
Phone: (212) 854-8174
Email: predrag@ee.columbia.edu
URL: http://www.ee.columbia.edu/~predrag

Last lecture: Linear Model Selection and Regularization

Motivation: Find the smallest/simplest model

- Problem with complex models
  - Overfitting (especially for $n \le p$)
  - High testing error
- Model interpretability
  - Hard to interpret model with many predictors
  - Focus on most important variables
- Approaches for reduction:
  - Subset selection
  - Shrinkage methods
  - Dimension reduction techniques (today)

Last lecture: Bias-Variance trade-off

- "Test error": test variable $y_0 = f(x_0) + \epsilon_0$

$$\begin{aligned}
\mathsf{Err}(x_0) &= \mathbb{E}\left(y_0 - \hat{f}(x_0)\right)^2 \\
&= \mathbb{E}\left(f(x_0) + \epsilon - \hat{f}(x_0)\right)^2 \\
&= \sigma^2 + \mathbb{E}\left(f(x_0) - \mathbb{E}\hat{f}(x_0) - \hat{f}(x_0) + \mathbb{E}\hat{f}(x_0)\right)^2 \\
&= \sigma^2 + \left(f(x_0) - \mathbb{E}\hat{f}(x_0)\right)^2 + \mathsf{Var}(\hat{f}(x_0)) \\
&= \sigma^2 + \left(\mathsf{Bias}(\hat{f}(x_0))\right)^2 + \mathsf{Var}(\hat{f}(x_0))
\end{aligned}$$

- For linear models, sample estimation leads to (see equation (7.12) in the [ESL] book)

$$\frac{1}{n}\sum_{i=1}^{n} \mathsf{Err}(x_i) = \sigma^2 + \frac{1}{n}\sum_{i=1}^{n}\left(\mathsf{Bias}(\hat{f}(x_i))\right)^2 + \frac{p}{n}\sigma^2$$

- Increasing p reduces the bias but increases the variance

Last lecture: Subset selection

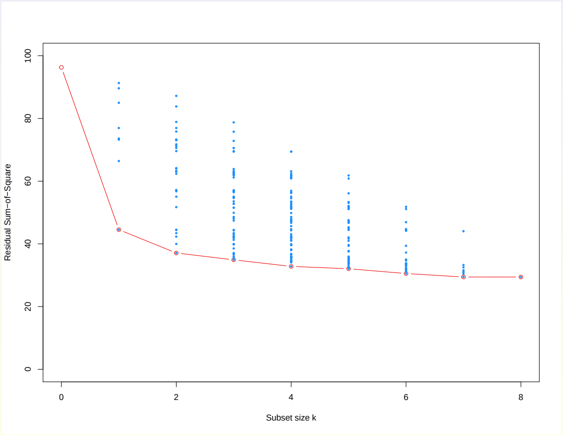
Deciding on the important variables:

- Need a criteria that balance training error and model size
- Several approaches
  - All (or best) subsets selection
    - Consider all 2^p models
    - Infeasible when p is large
  - Forward selection
    - Start with a null model – no predictors
    - Add predictors one-by-one
    - Stopping criterion
  - Backward selection
    - Start with a full model – p predictors
    - Eliminate predictors one-by-one
    - Stopping criterion

Last lecture: Best subset selection

```
# Making the plot
plot( 0:8, prostate.models.best.rss, ylim=c(0, 100),
        type="b", xlab="Subset size k", ylab="Residual Sum-of-Square", col="red2" )
points( prostate.models.size, prostate.models.rss, pch=20, col="dodgerblue",cex=0.7 )
```

Redo this with "glmnet" library: glmnet() function does both ridge (alpha=0) and lasso (alpha=1), and in-between.



Last lecture: Suboptimal (greedy) selection

Forward (or backward) stepwise selection

- Reduce computational complexity by forfeiting optimality
- Algorithm
  - Let $\mathcal{M}_0$ denote the null model (no predictors, sample mean prediction)
  - for $k = 0, 1, \ldots, p - 1$
    - Fit all $p - k$ models that augment the predictors in $\mathcal{M}_k$ with one additional predictor
    - Let $\mathcal{M}_{k+1}$ be the best of these $p - k$ models in terms of the smallest RSS (equivalently the largest R^2)
  - Select the best model from among $\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_p$ using some criterion
- Greedy
- Number of models is $1 + p(p + 1)/2$
- If $p > n$, we can construct $\mathcal{M}_0, \ldots, \mathcal{M}_n$ models only

Last lecture: Regularization (shrinkage) methods

- An alternative to subset selection
- Idea: Regularize/constrain coefficients

- Two widely-used methods:
  - Ridge regression
  - LASSO (least absolute shrinkage and selection operator)

Last lecture: Ridge

- OLS: minimize RSS

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

- Ridge regression: minimize (RSS + shrinkage penalty)

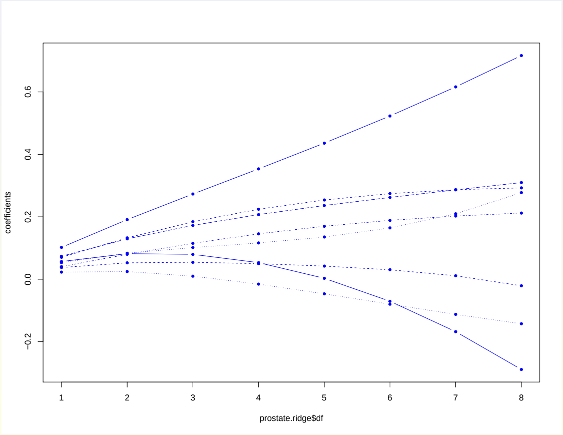
$$\text{RSS} + \lambda \sum_{j=1}^{p} \beta_j^2$$

- λ is a tuning parameter: β_j^λ for each λ
- β_0 is not in the penalty

Prostate: Ridge regression

```
# Calling simple.ridge() function which is part of "ElemStatLearn" package
# Ridge functions in other packages:
# MASS: lm.ridge()
# mda : gen.ridge()
prostate.ridge <- simple.ridge( train[,1:8], train[,9], df=1:8 )

# plot
matplot( prostate.ridge$df, t(prostate.ridge$beta), type="b",
        col="blue", pch=20, ylab="coefficients" )
```



Last lecture: Lasso

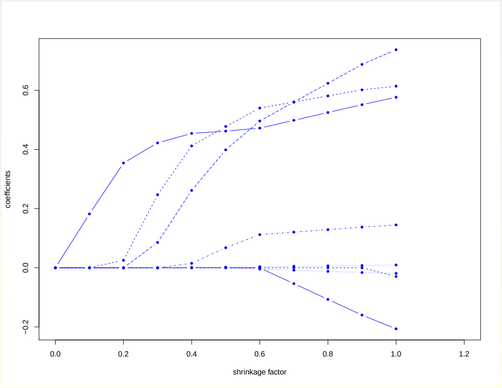
- Ridge regression: Still p (shrunk) predictors
- Inference

- Lasso: minimize
$$\text{RSS} + \lambda \sum_{j=1}^{p} |\beta_j|$$
- Rationale for absolute values: Some of β_j's will be equal to 0
- Data normalization

Prostate: Lasso

```
# loading package "lasso2"
library(lasso2)
prostate.lasso <- l1ce( lpsa ~ ., data=train, trace=TRUE, sweep.out=~1,
        bound=seq(0,1,by=0.1) )
prostate.lasso.coef <- sapply(prostate.lasso, function(x) x$coef)
colnames(prostate.lasso.coef) <- seq( 0,1,by=0.1 )

# plot
matplot( seq(0,1,by=0.1), t(prostate.lasso.coef[-1,]), type="b",
        xlab="shrinkage factor", ylab="coefficients",
        xlim=c(0, 1.2), col="blue", pch=20 )
```



Lasso vs. Ridge regression

- Equivalent formulations
  - Ridge (ℓ_2 penalty):

$$\min_{\boldsymbol{\beta}} \text{RSS} \quad \text{subject to } \sum_{j=1}^{p} \beta_j^2 \leq s$$

  - Lasso (ℓ_1 penalty):

$$\min_{\boldsymbol{\beta}} \text{RSS} \quad \text{subject to } \sum_{j=1}^{p} |\beta_j| \leq s$$

  - Best subset selection (ℓ_0 penalty):

$$\min_{\boldsymbol{\beta}} \text{RSS} \quad \text{subject to } \sum_{j=1}^{p} 1_{\{\beta_j \neq 0\}} \leq s$$

Last lecture: Model selection measures

Indirect measures: C_p, AIC and BIC

- Model with d predictors
- Mallow's C_p:

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

  where $\hat{\sigma}$ is an estimator for the variance of noise (estimated on a model containing all predictors)

- Akaike information criteria (AIC):

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

- Bayesian AIC (BIC):

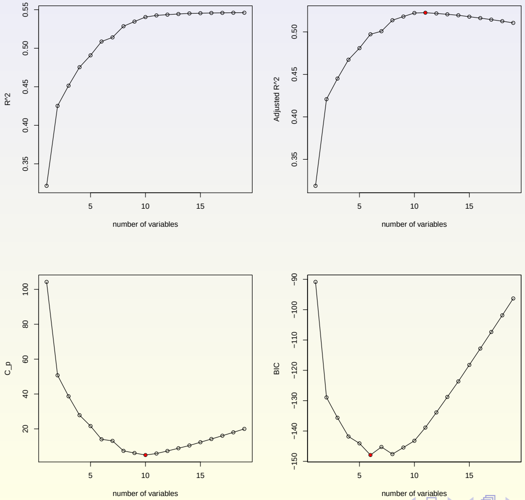
$$\mathsf{BIC} = \frac{1}{n}(\mathsf{RSS} + d\hat{\sigma}^2 \log n)$$

- **Heuristic:** select a model with the lowest C_p, AIC, BIC, or

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

Example

```
> regfit.full<-regsubsets(Salary~.,data=Hitters,nvmax = 19)
> reg.summary<-summary(regfit.full)
> names(reg.summary)
[1] "which"  "rsq"    "rss"    "adjr2"  "cp"     "bic"    "outmat" "obj"
> reg.summary$rsq
 [1] 0.3214501 0.4252237 0.4514294 0.4754067 0.4908036 0.5087146 0.5141227 0.5285569 0.5346124 0.5404950
[11] 0.5426153 0.5436302 0.5444570 0.5452164 0.5454692 0.5457656 0.5459518 0.5460945 0.5461159
```



Dimension Reduction

- Subset selection, shrinkage methods:
  - Original predictors used
- Dimensionality reduction idea
  - Represent/approximate X with a vector Z having less dimensions
  - Then, apply regression to Z
- Many approaches for doing this
- Common approach: Principal Component Analysis (PCA)

Dimension Reduction: PCA

-
  - *Transform the predictors*. Let $Z = (Z_1, \ldots, Z_q)$ represent $q < p$ linear combinations of the original p predictors:

    $$Z_m = \sum_{j=1}^{p} \phi_{m,j} X_j$$

    for some constants $\phi_{m,1}, \ldots, \phi_{m,p}$
  - *Use Ordinary Least Squares (OLS)*: fit the linear regression

    $$\hat{y}_i = \theta_0 + \sum_{m=1}^{q} \theta_m z_{i,m}$$

- If $\{\phi_{m,i}\}$ are chosen appropriately, dimension reduction can outperform the OLS regression

Dimension reduction

- Regression coefficients

$$\begin{aligned}\sum_{m=1}^{q}\theta_m z_{i,m} &= \sum_{m=1}^{q}\theta_m\sum_{j=1}^{p}\phi_{m,j}x_{i,j}\\ &= \sum_{j=1}^{p}\sum_{m=1}^{q}\theta_m\phi_{m,j}x_{i,j}\\ &= \sum_{j=1}^{p}\beta_j x_{i,j}\end{aligned}$$

where

$$\beta_j = \sum_{m=1}^{q}\theta_m\phi_{m,j}$$

- PCA is equivalent to imposing constraints on β_j's in OLS

Finding Principal Components

PCA: Unsupervised method - will be covered more after the midterm
Good for high-dimensional data

Few words here: Finding the first principal component

- Look for the linear combination of the sample feature of the form

$$z_{i1} = \phi_{11} x_{i1} + \phi_{21} x_{i2} + \cdots + \phi_{p1} x_{ip}$$

- *First loading vector/principal component*: $\boldsymbol{\phi}_1 = (\phi_{11}, \phi_{21}, \ldots \phi_{p1})$
- Assume x_i-s are centered $(\sum x_i = 0)$
- Look for $\boldsymbol{\phi}_1$ that has the largest sample variance, i.e.

$$\max_{\boldsymbol{\phi}_1} \frac{1}{n} \sum_{i=1}^{n} z_{i1}^2 = \max_{\boldsymbol{\phi}_1} \frac{1}{n} \sum_{i=1}^{n} (\phi_{11} x_{i1} + \phi_{21} x_{i2} + \cdots + \phi_{p1} x_{ip})^2$$

  subject to $\sum_{j=1}^{p} \phi_{j1}^2 = 1$ (i.e., $\boldsymbol{\phi}_1$ is a unit vector)
- This optimization problem can be solved via eigen-decomposition

First Principal Components: Geometric interpretation

- Loading vector ϕ_1 represents the direction along which the data varies the most
- If we project $x_1, \ldots, x_n$ onto ϕ_1, the projected values are the PC scores z_{i1} since

$$z_{i1} =< x_i, \phi_1 >$$

Second and higher principal components

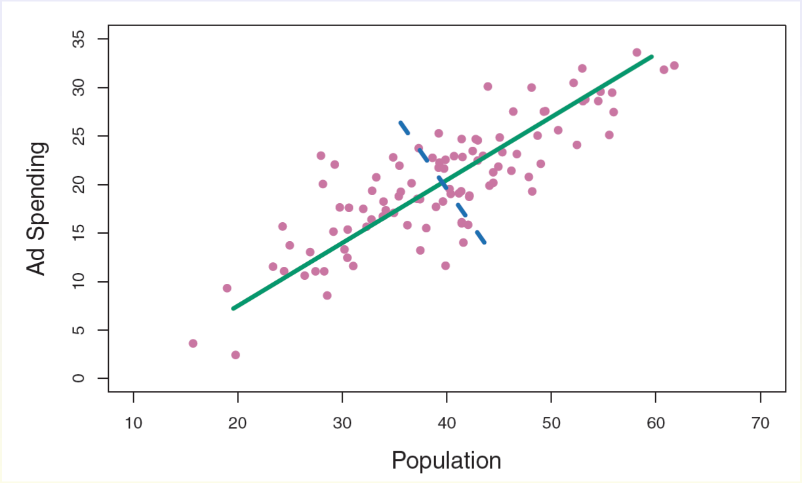
- After ϕ_1 has been determined, we look for ϕ_2 in a similar way, but with the additional constraint that ϕ_1, ϕ_2 are uncorelated

$$< \phi_1, \phi_2 >= 0$$

- We continue this procedure until we find as many PC as we want

PCA: Example

Two PC-s: Solid line: First PC; Dashed line: Second PC



PCA as Eigenvalue-Eigenvector Decomposition

Consider finding $k \leq p$ principal components: $\boldsymbol{\phi}_1, \boldsymbol{\phi}_2, \ldots, \boldsymbol{\phi}_k$

$$\boldsymbol{U} = \begin{bmatrix} \boldsymbol{\phi}_1 & \boldsymbol{\phi}_2 & \cdots & \boldsymbol{\phi}_k \end{bmatrix}, \quad \boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k.$$

Then, the projection of a data point $\boldsymbol{x}_i, 1 \leq i \leq n$ is given by

$$\hat{\boldsymbol{x}}_i = (\boldsymbol{x}_i \cdot \boldsymbol{\phi}_1)\boldsymbol{\phi}_1 + \cdots (\boldsymbol{x}_i \cdot \boldsymbol{\phi}_k)\boldsymbol{\phi}_k = \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i,$$

implying

$$\|\hat{\boldsymbol{x}}_i\|^2 = \boldsymbol{x}_i^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{U}^\top \boldsymbol{x}_i = \boldsymbol{x}_i^\top \boldsymbol{U} U^\top \boldsymbol{x}_i.$$

Note that $\hat{\boldsymbol{x}}_i$ minimizes the distance $\|\boldsymbol{x}_i - \hat{\boldsymbol{x}}_i\|$, and thus, finding k principle components is equivalent to finding $\boldsymbol{U}$ that maximizes

$$\begin{aligned} M &= \max_{\boldsymbol{U}:\boldsymbol{U}^\top \boldsymbol{U} = \boldsymbol{I}_k} \sum_{i=1}^{n} \boldsymbol{x}_i^\top \boldsymbol{U} U^\top \boldsymbol{x}_i \\ &= \text{trace}\left(\boldsymbol{U}^\top \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{U}\right), \quad (\text{using } \boldsymbol{x}^\top \boldsymbol{y} = \text{trace}(\boldsymbol{x}\boldsymbol{y}^\top)) \end{aligned}$$

PCA as Eigenvalue-Eigenvector Decomposition

Now, if $\boldsymbol{A} = \sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top$, the optimization problem becomes

$$M = \max_{\boldsymbol{U}:\boldsymbol{U}^\top\boldsymbol{U}=\boldsymbol{I}_k} \operatorname{trace}\left(\boldsymbol{U}^\top \boldsymbol{A} \boldsymbol{U}\right)$$

Note that $\boldsymbol{A}$ is a symmetric matrix, and therefore orthogonally diagonalizable with $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_n \geq 0$. If $\boldsymbol{U}$ is composed of k eigenvectors that correspond to the k largest eigenvalues, then

$$M \geq \sum_{i=1}^{k} \lambda_i .$$

On the other hand, we can diagonalize $\boldsymbol{A} = \boldsymbol{V}^\top \boldsymbol{\Lambda} \boldsymbol{V}$, where $\boldsymbol{V}$ is an orthogonal matrix, yielding

$$\begin{aligned} M &= \max_{\boldsymbol{U}:\boldsymbol{U}^\top\boldsymbol{U}=\boldsymbol{I}_k} \operatorname{trace}\left(\boldsymbol{U}^\top \boldsymbol{V}^\top \boldsymbol{\Lambda} \boldsymbol{V} \boldsymbol{U}\right) = \max_{\boldsymbol{B}:\boldsymbol{B}^\top\boldsymbol{B}=\boldsymbol{I}_k} \operatorname{trace}\left(\boldsymbol{B}^\top \boldsymbol{\Lambda} \boldsymbol{B}\right) \\ &= \sum_{i=1}^{p} \sum_{j=1}^{k} b_{ij}^2 \lambda_i = \sum_{i=1}^{p} \lambda_i \sum_{j=1}^{k} b_{ij}^2 \leq \sum_{i=1}^{k} \lambda_i \quad \text{(Note: } \sum_{i=1}^{p} b_{ij}^2 = 1, \sum_{j=1}^{k} b_{ij}^2 \leq 1) \end{aligned}$$

$\Rightarrow M = \sum_{i=1}^{k} \lambda_i$: Hence, the first k principal components correspond to the eigenvectors of the k largest eigenvalues of $\boldsymbol{A}$.

Beyond Linear - More Linear ☺: Basis Expansion

Basis expansion

- Map data $\boldsymbol{x}$ into higher dimensional space $\mathbb{R}^d, d > p$: $\boldsymbol{\phi}(\boldsymbol{x}) : \mathbb{R}^p \to \mathbb{R}^d$, i.e.,

$$\phi(\boldsymbol{x}) = (\phi_1(\boldsymbol{x}), \phi_2(\boldsymbol{x}), \dots, \phi_d(\boldsymbol{x}))$$

- Fit linear regression on $\boldsymbol{\phi}(\boldsymbol{x})$ in the higher dimensional space

Example: Polynomial regression of degree q

- Map data $\boldsymbol{x} = (x_1, \dots, x_p)$ into

$$\boldsymbol{\phi}(\boldsymbol{x}) = (x_1, x_1^2, \dots, x_1^q, x_2, x_2^2, \dots, x_2^q, \dots, x_p, x_p^2, \dots, x_p^q, \dots)$$

- Fit linear regression in the higher dimensional, $\mathbb{R}^{pq}$, space
  - Exponential growth of dimensionality: if $p = 10$ and $q = 10$

Quadratic Expansion

Example:

- Consider data with two predictors: $x_i = (x_{i1}, x_{i2})$
- We can obtain the preceding kernel by considering feature map

$$\phi(x_i) = (1, \sqrt{2}x_{i1}, \sqrt{2}x_{i2}, x_{i1}^2, \sqrt{2}x_{i1}x_{i2}, x_{i2}^2)$$

  if $x \in \mathbb{R}^p$, then $\phi(x) \in \mathbb{R}^{2p}$

- Then, the inner product

$$\begin{aligned}\langle \phi(x_i), \phi(x_j) \rangle &= 1 + 2x_{i1}x_{j1} + 2x_{i1}x_{j1} + x_{i1}^2 x_{j1}^2 \\ &\quad + 2x_{i1}x_{i2}x_{j1}x_{j2} + x_{i2}^2 x_{j2}^2 \\ &= 1 + 2\langle x_i, x_j \rangle + \langle x_i, x_j \rangle^2 \\ &= (1 + \langle x_j, x_i \rangle)^2 = K(x_i, x_j)\end{aligned}$$

Quadratic Kernel: $K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^2$

K remains $n \times n$ matrix: does not grow with basis expansion

Polynomial Kernel and Dual Solution for Ridge

Polynomial Kernel:

$$K(x_i, x_j) = (1 + \langle x_j, x_i \rangle)^d$$

K is $n \times n$, doesn't grow with d; recall, n is the number of data points x_i. Computing $\boldsymbol{K}$ is of the order $O(n^2 p)$.

- Dual solution - recall dual form for $\hat{\boldsymbol{y}}$

$$\hat{\boldsymbol{y}} = \boldsymbol{K}\boldsymbol{\alpha},$$

- Dual Ridge: find $\boldsymbol{\alpha}$ that minimizes

$$\|\boldsymbol{y} - \hat{\boldsymbol{y}}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_2^2 = \|\boldsymbol{y} - \boldsymbol{K}\boldsymbol{\alpha}\|_2^2 + \lambda\|\boldsymbol{\alpha}\|_2^2$$

- Solution

$$\boldsymbol{\alpha} = (\lambda\boldsymbol{I} + \boldsymbol{K})^{-1}\boldsymbol{y}$$

  which has computational complexity $O(n^3)$, and does not depend on the polynomial basis expansion parameter d. Hence the total complexity is $O((p+n)n^2) \approx O(n^3)$, when $p \ll n$, and does not depend on d.

Piecewise Polynomial: Splines

Example: Piecewise linear

- Assume $x \in \mathbb{R}$, i.e., $p = 1$
- Consider points: $w_1 \leq w_2 \leq \cdots \leq w_n$ - called knots
- Consider basis

$$b_0(x) = 1, b(x) = x, b_i = (x - w_i)^+$$

- Then, fitting linear regression to these bases will result in piecewise linear approximation
- Relation to Neural Nets: Tow layer neural network corresponds to free knot linear spline (studied in statistics since the 70s):
  - By free knot, one means that knots/weights w_i and linear regression coefficients β_j are optimized at the same time
  - This problem is not convex

In general, basis can be chosen to be anything: trigonometric functions, wavelets, etc.

Where are we now?

Recall, that we started with a general supervised Statistical (Machine) Learning problems

$$\boxed{Y = f(X)}$$

Problem: Estimate f from training data $\{(x_i, y_i)\}$, and then use it in general for inference and prediction

- First, we assumed some characteristics of the approximation function. e.g.: $\hat{f}$ is linear or nonlinear via basis expansion
- Then, we used the training data $\{(x_i, y_i)\}$ to find the best fit of $\hat{f}$ to the training data by minimizing the square error, RSS, or some other error function.
- For linear regression and Gaussian noise, we used classical tools from statistics, χ^2, t-statistics and F-statistics, to characterize the confidence of our model.

Where are we now and what are we doing next?

- Searching for simpler models:
  - Selecting a model using t-statistics and F-statistics
  - Trying all combinations of predictors - not scalable - and then using indirect measures (C_p, AIC, BIC, etc.).
  - Introducing penalties: Ridge or Lasso - scalable
  - Preprocessing data - dimensionality reduction
- Bias - Variance: in general the preceding procedures introduce bias, but the hope is that at an expense of a small bias we have a bigger reduction in variance
- How do we select the best model?
  - For linear and Gaussian: we could use the t-statistics and F-statistics
  - We could use the indirect measures: C_p, AIC, BIC, etc.
  - Are there more direct general procedures to test and select the best model?

Model validation: Training vs. test error

- Select a statistical learning method, e.g.: linear model
- Training error: the average error from using the method to predict the response on the observations used in its training

- **Test error**: the average error from using the method to predict the response on a **new observation**

- Ideally: a large designated test set – seldomly available

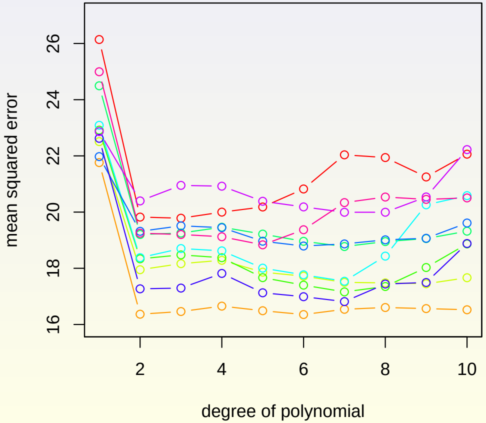
Validation-set approach

- Randomly divide the available samples into:
  - training set
  - validation set
- Random split into two halves
- Fit a model using the training set
- Use the model to predict the responses in the validation set
- The validation-set error is an estimate of the test error
- Drawbacks
  - The error estimate can be variable – depends on the split
  - Only a subset of observations used to fit the model
  - Tends to overestimate the test error for the model fit on the entire data set

Example: Auto data set

```
> set.seed(1)
> train<-sample(392,196) #pick randomly 1/2 sample
> err<-rep(0,10)
> for(i in 1:10) {
+   lm.fit<-lm(mpg~poly(horsepower,i),data = auto, subset = train)
+   err[i]<-mean((auto$mpg-predict(lm.fit,auto))[-train]^2)}
```

Different "train" set produces different results



K-fold cross-validation

- Popular approach
- Used in model selection

- Procedure
    - Randomly divide observations into K equal-sized parts
    - Leave out part k, fit a model using the remaining $K - 1$ parts
    - Use the left-out part to estimate the error
    - Repat for all k
    - Combine results

K-fold cross-validation

- K parts: $C_1, C_2, \ldots, C_K$
- $\cup_k C_k = \{1, \ldots, n\}$
- n_k: the number of observations in part k
- Compute

  $$\text{CV}_{(K)} = \sum_{k=1}^{K} \frac{n_k}{n} \text{MSE}_k$$

  where

  $$\text{MSE}_k = \frac{1}{n_k} \sum_{i \in C_k} (y_i - \hat{y}_i)^2$$

  and $\hat{y}_i$ is the prediction for observation i obtained from the data without part k

- $K = n$: leave-one out cross-validation (LOOCV)

LOOCV

Linear model example: we can compute CV error

- **No randomness** – all subsets of size $(n-1)$ considered
- $\boldsymbol{X}$ and $\boldsymbol{y}$:
  - observation i: $\boldsymbol{X}_i$ and y_i
  - no observation i: $\boldsymbol{X}_{(i)}$ and $\boldsymbol{y}_{(i)}$
- Coefficients
  - observation i omitted:

$$\boldsymbol{X}_{(i)}^{\top}\boldsymbol{X}_{(i)}\hat{\boldsymbol{\beta}}_{(i)} = \boldsymbol{X}_{(i)}^{\top}\boldsymbol{y}_{(i)}$$

  - all observations used:

$$(\boldsymbol{X}_{(i)}^{\top}\boldsymbol{X}_{(i)} + \boldsymbol{X}_i^{\top}\boldsymbol{X}_i)\hat{\boldsymbol{\beta}} = \boldsymbol{X}_{(i)}^{\top}\boldsymbol{y}_{(i)} + \boldsymbol{X}_i^{\top}y_i$$

and

$$\hat{\boldsymbol{\beta}} = \hat{\boldsymbol{\beta}}_{(i)} + (\boldsymbol{X}_{(i)}^{\top}\boldsymbol{X}_{(i)})^{-1}\boldsymbol{X}_i^{\top}(y_i - \hat{y}_i),$$

where $\hat{y}_i$ is the prediction for y_i using all observations

$$\hat{y}_i = \boldsymbol{X}_i\hat{\boldsymbol{\beta}} = \boldsymbol{X}_i(\boldsymbol{X}^{\top}\boldsymbol{X})^{-1}\boldsymbol{X}^{\top}\boldsymbol{y}$$

LOOCV

- Some algebra and

$$(\boldsymbol{X}_{(i)}^\top \boldsymbol{X}_{(i)})^{-1} = (\boldsymbol{X}^\top \boldsymbol{X} - \boldsymbol{X}_i^\top \boldsymbol{X}_i)^{-1}$$
$$= (\boldsymbol{X}^\top \boldsymbol{X})^{-1} + \frac{(\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}_i^\top \boldsymbol{X}_i (\boldsymbol{X}^\top \boldsymbol{X})^{-1}}{1 - \boldsymbol{X}_i (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}_i^\top}$$

  yield

$$\mathsf{CV}_{(n)} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \boldsymbol{X}_i \hat{\boldsymbol{\beta}}_{(i)})^2$$
$$= \frac{1}{n} \sum_{i=1}^{n} \left(\frac{y_i - \hat{y}_i}{1 - h_i} \right)^2$$

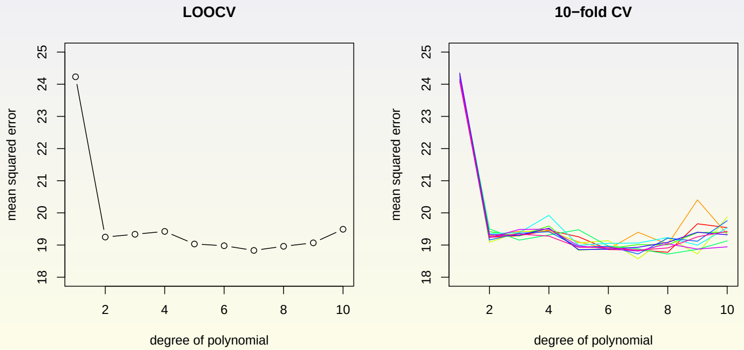
  where $h_i = \boldsymbol{X}_i (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}_i^\top$

- **Weighted sum of squared residuals** (Provides validation for using modified RSS for some of the indirect measures).

Example: Auto data set

```
> library(boot)
> cv.err<-rep(0,10)
> for(i in 1:10) {
+   glm.fit<-glm(mpg~poly(horsepower,i), data=auto)
+   cv.err[i]<-cv.glm(auto,glm.fit)$delta[1]
+ }

> set.seed(1)
> for(i in 1:10) {
+   glm.fit<-glm(mpg~poly(horsepower,i), data=auto)
+   cv.err[i]<-cv.glm(auto,glm.fit,K=10)$delta[1]
+ }
```



What if we are interested in computing other statistics, not just the test error, for the non-Gaussian or nonlinear case?

Bootstrap

- Setup
  - Population model that produces an outcome Y
  - Observations $\boldsymbol{Z}$ from this population model
  - Statistic $T(\boldsymbol{Z})$
  - Distribution of $T(\boldsymbol{Z})$
- Idea
  - The distribution of $T(\boldsymbol{Z})$ can be estimated by sampling $\boldsymbol{Z}$ from the population model
  - Resample with replacement from $\boldsymbol{Z}$ to "approximate" sampling from the population model
- Why?
  - Only samples $\boldsymbol{Z}$ available
  - No information on the population model

Bootstrap: Basic algorithm

- Input
  - A sample of data $\boldsymbol{Z} = (\boldsymbol{Z}_1, \ldots, \boldsymbol{Z}_n)$
  - An estimation rule $\hat{T}$ for Statistic T

- Algorithm
  1. Generate bootstrap samples $\boldsymbol{Z}^{*1}, \boldsymbol{Z}^{*2}, \ldots, \boldsymbol{Z}^{*B}$
     - Create $\boldsymbol{Z}^{*b}$ by selecting n points from $\boldsymbol{Z}$
     - A particular $\boldsymbol{Z}_i$ can appear in $\boldsymbol{Z}^{*b}$ multiple times
  2. Evaluate the estimator on each $\boldsymbol{Z}^{*b}$:

$$\hat{T}_b = \hat{T}(\boldsymbol{Z}^{*b})$$

- The empirical distribution of $\{\hat{T}_1, \ldots, \hat{T}_B\}$ is an estimate of the distribution of $T(\boldsymbol{Z})$
- Bootstrap distribution
- Overlap between $\boldsymbol{Z}$ and $\boldsymbol{Z}^{*b}$?

Example: Variance estimation of the meadian

- The median of $x_1, \ldots, x_n$, $x_i \in \mathbb{R}$, is found by sorting the numbers and taking the middle one (or averaging the two middle ones)
- How good is the estimate median$(x_1, \ldots, x_n)$?
  - Find it's variance
  - How?

1. Generate bootstrap datasets $\boldsymbol{Z}^{*1}, \ldots, \boldsymbol{Z}^{*B}$
2. Calculate:

$$\hat{T}_{\text{mean}} = \frac{1}{B} \sum_{b=1}^{B} \text{median}(\boldsymbol{Z}^{*b})$$

and

$$\hat{T}_{\text{var}} = \frac{1}{B-1} \sum_{b=1}^{B} \left(\text{median}(\boldsymbol{Z}^{*b}) - \hat{T}_{\text{mean}} \right)^2$$

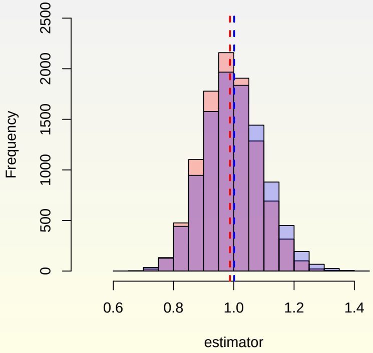
When it works?

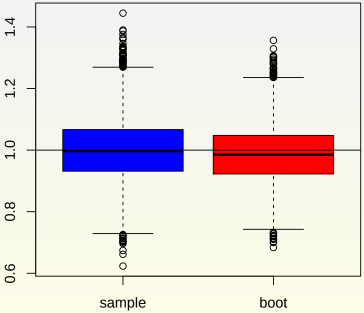
- The bootstrap distribution should be close to the sampling distribution
  - The number of bootstrap samples B should be large enough
  - The original data sample $\boldsymbol{Z}$ should be large enough to be "representative" of the population model

- Few assumptions about the population model
- Can yield inaccurate results (e.g., "extreme value" statistics)
- Usually reliable for estimating standard errors for estimators
  - Standard errors
    - The bootstrap estimate of the standard error is the standard deviation of the bootstrap distribution
  - Confidence intervals
    - $(1-\gamma)$ confidence interval
    - t_α – the αth quantile of the bootstrap distribution
    - $(1-\gamma)$ percentile bootstrap interval: $[t_{\gamma/2}, t_{1-\gamma/2}]$

Example: Mean

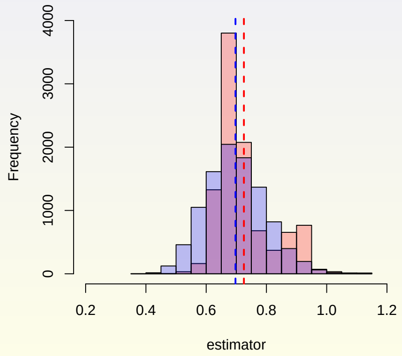
```
> library(stats)
> set.seed(100)
> z<-rexp(100,rate=1)
> boot.fn=function(data,index)
+   return(mean(data[index]))
> boot.fn(z,1:100)
[1] 0.9874761
> boot.fn(z,sample(100,100,replace=T))
[1] 1.184723
> boot.fn(z,sample(100,100,replace=T))
[1] 1.131222
> library(boot)
> boot.out<-boot(z,boot.fn,1000)
```

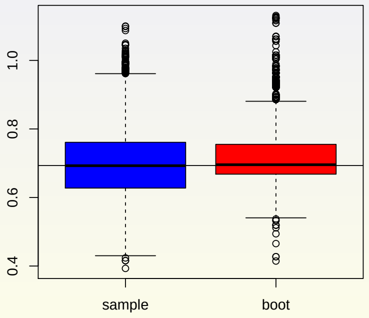




Example: Median

```
> boot.fn=function(data,index)
+   return(median(data[index]))
> boot.fn(z,1:100)
[1] 0.6955635
> boot.fn(z,sample(100,100,replace=T))
[1] 0.6373702
> boot.fn(z,sample(100,100,replace=T))
[1] 0.652271
> boot.out<-boot(z,boot.fn,1000)
```





Bootstrap: Regression modeling

- n observations, response $\boldsymbol{y}$, covariates $\boldsymbol{X}$
- Bootstrap standard errors for OLS coefficients using case resampling:
  - For $b = 1, \ldots, B$
  - Draw sample uniformly at random, with replacement, from observations $(\boldsymbol{X}, \boldsymbol{y})$. Let the ith outcome in the bth sample be $(\boldsymbol{X}_i^{*b}, y_i^{*b})$
  - Compute $\hat{\boldsymbol{\beta}}^{*b}$ given $(\boldsymbol{X}^{*b}, \boldsymbol{y}^{*b})$
- Bootstrap distribution of $\hat{\boldsymbol{\beta}}$ to compute standard errors

Example: When analytics is unavailable, use bootstrap

Example: Heteroskedasticity

- Suppose $X_i \sim \mathcal{N}(0,1)$ and $y_i = X_i + \epsilon_i$, for $i = 1, \ldots, n$, where $\epsilon_i \sim \mathcal{N}(0, X_i^4)$
- Non-constant error variance
- Standard assumptions violated
- Standard errors?

Example: Heteroskedasticity

```
> n<-1000
> set.seed(1)
> x<-rnorm(n); y<-x+x^2*rnorm(n); df<-data.frame(x,y)
>
> lm.fit<-summary(lm(y~x,data = df))
> lm.fit$coefficients
              Estimate Std. Error   t value    Pr(>|t|)
(Intercept) 0.02807759 0.06502742  0.4317807 6.659940e-01
x           1.18461640 0.06286099 18.8450180 5.229894e-68
>
> coef.boot=function(data,indices) {
+   coef.fit<-lm(y~x,data=data[indices,])
+   return(coef(coef.fit))
+ }
>
> boot.out<-boot(df,coef.boot, 50000)
> boot.out

ORDINARY NONPARAMETRIC BOOTSTRAP


Call:
boot(data = df, statistic = coef.boot, R = 50000)


Bootstrap Statistics :
      original        bias    std. error
t1* 0.02807759 -0.0008122518  0.06504296
t2* 1.18461640 -0.0014536288  0.14104734
```

Reading:

ISL: Read Chapters 5 and 7

ESL: Read Chapter 5, 7, and specifically Sections 5.1-5.2 and 7.10-7.11 for this lecture.

Homework 1: Due today, Sep 27, by 11:59pm.

Homework 2: Due Fri, Oct 7th, by 11:59pm.

Midterm planned for Oct 25th