1. Introduction
Nonparametric estimation/inference:

- fundamentally about fitting flexible models to data
- allows data to guide researcher in learning by leveraging weak structure imposed by researcher (e.g. continuity and differentiability)
  - Structure provides dimension reduction by not requiring the researcher to try to learn the value of a function at point $x^*$ by looking only at observations with $x_i = x^*$
- tries to trade off bias and variance in estimation by adapting model complexity to the data at hand
- fundamentally about description/prediction but underlies learning about more “structural” parameters (e.g. treatment effects)

Traditional nonparametric approaches (e.g. kernels, series, ...) perform poorly when size of the input space is large (curse of dimensionality)

- if unwilling to assume much, need LOTS of data before informative conclusions can be drawn
- many more recent nonparametric approaches - high-dimensional models (drawn largely from machine learning, data-mining) - impose more structure (more dimension reduction) but scale better (computationally and as the dimension of the problem increases)
High-dimensional linear model:

\[ Y_i = X'_i \beta + \varepsilon_i = \sum_{j=1}^{p} \beta_j X_{j,i} + \varepsilon_i, \quad \mathbb{E}[\varepsilon_i | X_{1,i}, \ldots, X_{p,i}] = 0, \quad p \gg n \]

Suppose observed data matrix \( X \) is full (row) rank

Least squares estimator solves \( X'X\beta = X'Y \) and yields family of solutions:

\[ \hat{\beta}^w = (X'X)^{-} X'Y + [I - (X'X)^{-} X'X]w \]

for \( A^{-} \) the Moore-Penrose generalized inverse and \( w \) an arbitrary conformable vector

No unique coefficient estimator.
Least squares fitted values in HDLM satisfy

\[ \hat{Y} = X\hat{\beta}^w \]

\[ = X(X'X)^{-1}X'Y + [X - X(X'X)^{-1}X']w \]

\[ = XX'(XX')^{-1}(XX')^{-1}XX'Y + [X - XX'(XX')^{-1}(XX')^{-1}XX']w \]

\[ = Y \]

using \((X'X)^{-1} = X'(XX')^{-1}(XX')^{-1}X\) (property of Moore-Penrose inverse)

I.e. any solution perfectly fits \(Y\) within sample.

Linearity insufficient structure to allow informative estimation and inference when dimensionality gets high enough. Need further regularization/dimension reduction.
2. Selection
Consider model

\[ y_i = x_i' \beta + \varepsilon_i \]

Goal: **Find a linear combination of** \( X \) **that provides a good forecast of** \( Y \)

- Popular additional structure is **sparsity**
  - Of the \( p \gg n \) available predictors, only \( s \ll n \) are needed to obtain a high-quality prediction of \( Y \)
- Want to find which (if any) elements of \( X \) we can drop and still get good forecasts of \( Y \)
- Note that dropping a variable \( \Leftrightarrow \) setting coefficient on that variable to 0
Note that nonparametric series/sieve estimation falls within this framework:

\[ y_i = g(z_i) + u_i \]

\[ = x_i' \beta + r_i + u_i = x_i' \beta + \varepsilon_i \]

- \( z_i \) some low-dimensional set of observed variables
- \( x_i = \{p_k(z_i)\}_{k=1}^p \); e.g. \( x_i = \{1, z_i, z_i^2, z_i^3, \ldots, z_i^p\} \)
- Believe decent forecast available using fewer than \( n \) series terms
- Allows approximation errors (just absorbed in \( \varepsilon \) in these notes) - needs to be dealt with explicitly in the theory
  - E.g. Belloni, Chernozhukov, and Hansen (2014) “Inference on Treatment Effects after Selection amongst High-Dimensional Controls”
How do we decide which variables to drop?

Leaving out variable with strong correlation to signal ⇒ model too simplistic ⇒ bias

Putting in many variables ⇒ hard to learn about all corresponding coefficients ⇒ variance

As with any nonparametrics, want enough variables to capture predictability without over-complicating the model

In principle, want to try all possible combinations and choose the one that does the best job out-of-sample
Best subset algorithm

Let $\bar{p} \leq n$ be a maximum model size.

Best subset algorithm:

- Let $M_0$ be the null-model with no predictors (i.e. forecast for $y_i$ is $\bar{y}$)
- For $k = 1, ..., \bar{p}$
  - Fit all $\binom{p}{k}$ models with exactly $k$ included predictors
  - Choose the best model with $k$ predictors (model with highest $R^2$ under $\ell_2$ loss) and call it $M_k$
- Select a single best model from among $M_0, ..., M_{\bar{p}}$ using CV or IC
  - Obviously choose a model with $\bar{p}$ variables if use unpenalized in-sample loss

An awful lot of models to consider:

- If $p$ is big, $\binom{p}{k}$ is big for any moderate $k$
- Suppose $\bar{p} = p < n$, summing over all $k$ gives $2^p$ models to be considered
- Can be quite unstable (discontinuous in the data) and “greedy”
Choosing $k$ for best subsets: IC

Best subsets computationally demanding - CV can be very cumbersome

Usual implementation of best subsets uses information criterion for selecting $k$

- $Q(\beta) = \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i' \beta)^2$ - i.e. error sum of squares using $\beta$
- Let $\|\beta\|_0$ be the number of non-zero elements in $\beta$
- Suppose $\epsilon \sim N(0, \sigma^2)$ with $\sigma^2$ known $\Rightarrow$ log-likelihood

$$l(\beta) = -\frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i' \beta)^2 + C = -\frac{n}{2\sigma^2} Q(\beta)$$
Choosing $k$ for best subsets: IC

(Ignoring irrelevant constants)

**BIC**($\hat{\beta}(M_k)$):

$$-2l(\hat{\beta}(M_k)) + \log(n)\|\hat{\beta}(M_k)\|_0 = \frac{n}{\sigma^2} \left( Q(M_k) + \log(n)\sigma^2 \frac{k + 1}{n} \right)$$

$$\propto Q(M_k) + \log(n)\sigma^2 \frac{k + 1}{n}$$

**AIC**($\hat{\beta}(M_k)$):

$$-\frac{2}{n}l(\hat{\beta}(M_k)) + 2\frac{\|\hat{\beta}(M_k)\|_0}{n} = \frac{1}{\sigma^2} \left( Q(M_k) + 2\frac{k + 1}{n} \sigma^2 \right)$$

$$\propto Q(M_k) + 2\sigma^2 \frac{k + 1}{n}$$

Note: AIC = $C_p$ in this case.
Choosing $k$ for best subsets: IC

For either $AIC$ or $BIC$ need $\sigma^2$:

- Helpful advice from “Elements” book: “$\hat{\sigma}^2$ is an estimate of the noise variance, obtained from a low-bias model.”
- If $p \ll n$, take estimate from model with all $p$ regressors
- Constructive 2-step procedure:
  - Take $\sigma_0^2 = s_y^2$ (the sample variance of $y$), choose best model based on IC ($\mathcal{M}_{k_0}$)
  - Take $\sigma_1^2 = \hat{\sigma}_{k_0}^2$ (the estimate of the error variance using $\mathcal{M}_{k_0}$), choose the best model based on IC ($\mathcal{M}_{k_1}$)
Choosing \( k \) for best subsets: IC

Note AIC, BIC (and other IC) give solution of same form:

\[
\hat{\beta}_{IC} = \arg \min_{\beta} \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i' \beta)^2 + \lambda \| \beta \|_0
\]

for different choices of \( \lambda \)

- Could find \( \lambda \) by CV
- Relates to other penalized methods to be introduced
- Conventional AIC, BIC \( \lambda \) NOT well-suited to large \( p \) case
  - BIC probably fine if \( p \propto n \)
  - AICc = \( Q(\mathcal{M}_k) + 2\sigma^2 \frac{k+1}{n} \frac{n}{n-k-1} \) also suggested but no great justification (other similar definitions of AICc)
  - Chen and Chen (2008) provide a formal motivation for replacing \( \log(n) \) with \( \log(n) + 2 \log(p) \) in the large \( p \) case
Choosing $k$ for best subsets: IC

Can also just treat $\sigma^2$ as an additional parameter to be estimated along with $\beta$ for each model.

log-likelihood is

$$l(\beta, \sigma^2) = -\frac{n}{2} \log(\sigma^2) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (y_i - x_i' \beta)^2 + C$$

Plugging in to AIC, BIC gives

BIC($\hat{\beta}(M_k)$):

$$-2l(\hat{\beta}(M_k), \hat{\sigma}(M_k)) + \log(n)(\|\hat{\beta}(M_k)\|_0 + 1) \propto \log(Q(M_k)) + \log(n) \frac{k + 2}{n}$$

AIC($\hat{\beta}(M_k)$):

$$-\frac{2}{n} l(\hat{\beta}(M_k), \hat{\sigma}(M_k)) + 2 \frac{\|\hat{\beta}(M_k)\|_0 + 1}{n} \propto \log(Q(M_k)) + 2 \frac{k + 2}{n}$$
Example: Boston Housing Data

Best subsets in Boston housing data (from Harrison and Rubinfield (1978) - conveniently included in the R library “MASS”)

- Outcome: \textit{medv} - median home value
- Predictors: 13 raw predictors in data
- 38 total variables (+ intercept) considered
  - cubic in \textit{crim} (per capita crime rate)
  - dummy for \textit{zn} > 0, cubic in \textit{zn} (proportion residential land zoned for lots over 25,000 ft$^2$)
  - cubic in \textit{indus} (proportion of non-retail business acres)
  - \textit{chas} (Charles River dummy)
  - cubic in \textit{nox} (nitrogen oxide concentration)
  - cubic in \textit{rm} (average rooms per dwelling)
  - cubic in \textit{age} (proportion of owner-occupied units built pre 1940)
  - cubic in \textit{dis} (weighted mean of distances to 5 employment centers)
  - cubic in \textit{rad} (index of highway accessibility)
  - cubic in \textit{tax} (property tax rate per $10,000)
  - cubic in \textit{ptratio} (pupil-teacher ratio by town)
  - cubic in \textit{black} (1000*(proportion black - .63)$^2$)
  - cubic in \textit{lstat} (percent lower SES)
Code: SubsetHousingExample.R

Use 400 observations. Set 106 aside for out-of-sample comparison.

Look at choosing subset size in a few ways

- AIC, BIC
  - Treating variance as parameter to be estimated
  - Treating variance as known and filling in with estimate from largest model
  - Treating variance as known and filling in with iterative estimate

- 10-Fold CV
AIC and BIC (treating variance as parameter to be estimated)
Results:

- AIC (all): $k = 23$, Validation MSE = 30.85
- BIC (Treating variance as parameter to be estimated): $k = 16$, Validation MSE = 31.77
- BIC (Variance from largest model): $k = 16$, Validation MSE = 31.77
- BIC (iterative variance): $k = 12$, Validation MSE = 33.33
- 10-Fold CV: $k = 29$, Validation MSE = 32.22
- For comparison, Polynomial in only $lstat$ with 10-fold CV order: $p = 5$, Validation MSE = 34.67
Example: Boston Housing Data

Variables chosen in AIC model:
- \( lstat, lstat^2, lstat^3 \)
- \( crim, crim^2, crim^3 \)
- \( chas \)
- \( nox, nox^2, nox^3 \)
- \( rm^2, rm^3 \)
- \( dis, dis^2, dis^3 \)
- \( rad \)
- \( tax, tax^2, tax^3 \)
- \( ptratio^2, ptratio^3 \)
- \( black^2, black^3 \)

Nice thing about variable selection is getting a model that is easy to express
Best subsets is nice but blows up very quickly with $p$

Rather than try all possible models, take a particular path through models

Usual Forward Stepwise Selection Algorithm:

- Let $\bar{p} \leq n$ be a maximum model size.
- Let $\mathcal{M}_0$ be the null-model with no predictors (i.e. forecast for $y_i$ is $\bar{y}$)
- For $k = 1, \ldots, \bar{p}$
  - Fit all $p - k + 1$ models with $k$ included predictors formed by taking all predictors in model $\mathcal{M}_{k-1}$ and trying each predictor not included in $\mathcal{M}_{k-1}$ one at a time
  - Choose the best model with $k$ predictors (model with highest $R^2$ under $\ell_2$ loss) and call it $\mathcal{M}_k$
- Select a single best model from among $\mathcal{M}_0, \ldots, \mathcal{M}_{\bar{p}}$ using CV or IC
Wang (2009) considers forward selection using BIC criterion in very-high dimensional ($p \gg n$) setting.

Note uses BIC $\propto \log(Q(M_k)) + (\log(n) + 2 \log(p)) \frac{k+1}{n}$

Shows that selected model contains true model with probability approaching 1 in large samples under

- normality of $X$ and $\varepsilon$ (and homoskedasticity)
- population covariance matrix of $X$ has eigenvalues uniformly bounded away from 0 and $\infty$
- “beta-min” condition: $\min_{j \in \mathcal{T}} |\beta_j| \geq Cn^{-\xi}$ where $\mathcal{T} = \{j : |\beta_j| > 0\}$
Kozbur (2016) considers forward selection where criterion is based on test-statistics

Algorithm

- Let $\bar{p} \leq n$ be a maximum model size.
- Let $M_0$ be the null-model with no predictors (i.e. forecast for $y_i$ is $\bar{y}$)
- For $k = 1, \ldots, \bar{p}$
  - Fit all $p - k + 1$ models with $k$ included predictors formed by taking all predictors in model $M_{k-1}$ and trying each predictor not included in $M_{k-1}$ one at a time
  - For each of the $j = 1, \ldots, p - k + 1$ models, take the t-statistic for testing the null hypothesis that the relevant newly added variable has 0 coefficient (call it $T_j$)
  - For each $j$, compare $|T_j|$ to “critical value” $\hat{cV}$ defined below. Let $\hat{S}_k = \{j : |T_j| > \hat{cV}\}$.
  - If $\hat{S}_k$ is empty, break. Else, form model $M_k$ by taking the union of the variables in model $M_{k-1}$ and variable $\hat{j} = \text{arg max}\{|T_j| : |T_j| > \hat{cV}\}$
"critical value" for testing based forward selection:

\[ \hat{cv} = c_\tau \hat{\tau}_{j(k-1)} \Phi^{-1}(1 - \alpha/p) \]

\[ \hat{\tau}_{j(k-1)} = \frac{\eta_j'(k-1) \sqrt{\text{diag}(\Psi_{j(k-1)}^\varepsilon)}}{\sqrt{\eta_j'(k-1) \Psi_{j(k-1)}^\varepsilon \eta_{j(k-1)}}} \]

where \( \eta_{j(k-1)} = (1, -\hat{\beta}_{j(k-1)}')' \), \( \hat{\beta}_{j(k-1)} \) are the coefficients from regressing the variable considered at step \( j \) on all the variables included in \( M_{k-1} \), and \( \Psi_{j(k-1)}^\varepsilon \) is a matrix with \( u, v \) entry given by

\[ [\Psi_{j(k-1)}^\varepsilon]_{u,v} = \sum_{i=1}^{n} \hat{\varepsilon}_{i,j(k-1)} x_{u,i} x_{v,i} \]

for \( u, v \) indexing variables in \( M_{k-1} \cup j \) and \( \varepsilon_{i,j(k-1)} \) the residual from regressing \( y \) on the variables in \( M_{k-1} \cup j \)

- Need \( c_\tau > 1 \), Kozbur suggests 1.1
- \( \Phi^{-1}(1 - \alpha/p) \) would be a “Bonferroni” adjusted critical value for multiple testing, Kozbur suggests setting \( \alpha = .05 \)
- \( \hat{\tau}_{j(k-1)} \) is an additional correction factor that is used in the theory to account for sampling variable in the model selection path
- In some preliminary simulations, ignoring \( \hat{\tau}_{j(k-1)} \) and setting \( c_\tau = 1 \) seems to work pretty well
Kozbur (2016) considers testing based forward selection in very-high dimensional ($p \gg n$) setting

Assumptions:

- Let $s = \|\beta\|_0$
- $s + a_n$ for $a_n \to \infty$ dimensional submatrices of $\frac{1}{n} \sum_i x_i x_i'$ and $\frac{1}{n} \sum_i \varepsilon_i^2 x_i x_i'$ have minimum and maximum eigenvalue bounded with probability approaching one
- $\frac{(s+a_n)^2 \log^2(p)}{n} \to 0$ and $\frac{\log^3(p)}{n} \to 0$

Results:

- $\frac{1}{n} \sum_i (x_i' \hat{\beta} - x_i' \beta)^2 = O_p(s \log(p) / n)$
  - Best possible forecast rate
- $\hat{s} = O(s)$
  - Selected model has similar size to true model - note that no guarantee you get the right variables
Forward selection in the housing data.

Doing forward selection allows consideration of a MUCH bigger set of potential variables

Consider:

- Charles River dummy
- Cubic polynomial in all continuous variables including all first and second order interactions
- dummy variables formed by splitting all continuous variables at deciles
  - where possible - otherwise form deciles, take unique elements and form dummies
- interact lstat and dis dummies with sixth order polynomials in lstat and dis and third order polynomials in crim, nox, tax, and ptratio
- interact other dummies with cubic in base variable (e.g. crim dummies with cubic in crim)
- 1240 variables (including some redundant ones)
Choose variables based on BIC, AIC, AICc with iterative variance estimates and testing

- AIC and AICc essentially fail
  - AIC chooses 201 variables (max allowed), validation MSE: 180.52
  - AICc chooses 104 variables, validation MSE: 163.24
  - Potential explanations
    - Penalty not well-justified in large $p$ settings
    - Numeric instability and not pre-cleaning obviously redundant variables (which is annoying bookkeeping)
- BIC chooses 39 variables, validation MSE: 18.77
  - Estimate of s.e. of MSE based on BIC model: 7.79
- Testing chooses 6 variables, validation MSE = 26.18
Nice thing about variable selection:

End up with a single model that is (relatively) easy to work with and talk about

This model WILL NOT be the true model except under extremely strong conditions.
Example: Boston Housing Data

Selected model using forward selection with BIC:

- $lstat \ast indus \ast nox, crim^2 \ast age, crim^2 \ast dis, nox^2 \ast dis, crim \ast rm \ast dis, age^2 \ast dis,$
  $indus \ast nox \ast tax, lstat \ast rm \ast tax, rm^2 \ast tax, indus \ast age \ast tax, lstat \ast ptratio, lstat^2 \ast ptratio,$
  $rm \ast rad \ast ptratio, (lstat \in (7.6, 9.55]) \ast crim^3, (lstat \in (9.55, 11.5]) \ast nox^2,$
  $(lstat \in (15.4, 18.1]) \ast tax^3, (lstat \in (23.2, \infty)) \ast dis^3, (lstat \in (6.29, 7.6]) \ast ptratio^3,$
  $(dis \in (2.69, 3.22]) \ast crim^2, (dis \in (1.61, 1.96]) \ast nox^3$

- $chas, rm^3, rad, lstat^2, indus \in (4.05, 5.96], nox \in (.469, .504], nox \in (.573, .605],$
  $nox \in (.661, .713], rm \in (6.38, 6.54], tax \in (287, 307], (lstat \in (4.59, 6.29]) \ast lstat^5,$
  $(lstat \in (23.2, \infty)) \ast lstat^5, (crim \in (.24, .504]) \ast crim^3, (crim \in (2.05, 5.7]) \ast crim^3,$
  $(indus \in (5.96, 7.26]) \ast indus^3, (nox \in (.605, .661]) \ast nox^3, (rm \in (7.15, \infty)) \ast rm^3,$
  $(tax \in (233, 273]) \ast tax^3, (black \in (369, 381]) \ast black^3$
Can do subset selection with essentially any criterion function (e.g. replace LS with log-likelihood for logit for binary choice)

Are many subset selection methods (e.g. backward elimination, stepwise regression)
  - Rules for moving back (pruning) often *ad hoc*

Bigger problem - get very slow very fast
  - Keeping track of design matrix is a lot of overhead
  - Changing model by one variable can dramatically change fit - discontinuous means need to fit model by model

Related issue: Stability
  - Small change in data may lead to large change in solution
  - High variance in selection may feed through to unstable out-of-sample performance
Validation MSE from forward selection model for each $k$:
3. Penalized Estimation
Key idea in much of contemporary statistics is **regularization**: At a high-level, *regularization* is just introducing additional information to solve an ill-posed inverse problem

- a very long history of use in mathematics
- any useful statistical method is doing some kind of regularization (e.g. mean is not allowed to vary arbitrarily for each observation)
- Many modern statistical methods *explicitly* introduce regularization by directly penalizing model complexity
  - E.g. information criteria for choosing variables
Penalized estimator solves

\[ \hat{f} = \arg \min_{f} L(\text{data}, f) + \lambda C(f) \]

- \( L(\text{data}, f) \) is a loss function that decreases as model fits data better (e.g. sum of squared residuals)
  - In principle, works for essentially any loss function and (theoretical) results available for common ones
- \( C(f) \) is a penalty (cost) function that increases in model complexity
- \( \lambda \) is penalty parameter (price) that controls how fit versus complexity are balanced
Squared Error Loss and Linear Models

Canonical examples:

$$\hat{\beta}_p = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - x_i' \beta)^2 + \lambda p(\beta)$$

- **Ridge**: $p(\beta) = \sum_{j=1}^{p} \psi_j \beta_j^2$
- **LASSO**: $p(\beta) = \sum_{j=1}^{p} |\psi_j \beta_j|$
- "\ell_q": $p(\beta) = \sum_{j=1}^{p} |\psi_j \beta_j|^q$
More elaborate examples:

- **Elastic Net:**

  \[ p(\beta) = \sum_{j=1}^{p} |\psi_{1,j}\beta_j| + \lambda_2 \sum_{j=1}^{p} \psi_{2,j}\beta^2_j \]

- **SCAD:** For \( a > 2 \),

  \[ p(\beta) = \sum_{j=1}^{p} \begin{cases} 
  |\psi_{j}\beta_j| & \text{if } |\psi_{j}\beta_j| \leq \lambda \\
  -\frac{|\psi_{j}\beta_j|^2 - 2a\lambda|\psi_{j}\beta_j| + \lambda^2}{2\lambda(a-1)} & \text{if } \lambda < |\psi_{j}\beta_j| \leq a\lambda \\
  \frac{(a+1)\lambda}{2} & \text{if } |\psi_{j}\beta_j| > a\lambda 
  \end{cases} \]

  (a quadratic spline with knots at \( \lambda \) and \( a\lambda \))

- **Lava:** Decompose \( \beta = \delta + \gamma \),

  \[ p(\beta) = \sum_{j=1}^{p} |\psi_{1,j}\delta_j| + \lambda_2 \sum_{j=1}^{p} \psi_{2,j}\gamma^2_j \]
Impose regularization by shrinking coefficients towards 0 (in principle could shrink to other value)

Complexity measured by size of estimated coefficients

Builds in a belief that model is not too complicated in that coefficients should not be “too big”

Scale of variables (and thus coefficients) very important for this belief
  - Standard implementations assume homoscedasticity and standardize data \textit{ex ante}
  - Take all the $\psi$’s = 1 in this case
  - More generally, proper choice of $\psi$ can allow for heteroscedasticity, non-Gaussian errors, dependence
    - To my knowledge, only formally worked out for LASSO under heteroscedasticity (BCH 2012) and clustering (BCHK 2016)

Key tradeoffs are type of shrinkage and computational complexity
Penalty Functions

All have $\lambda = 1; a = 3$ for SCAD
Shrinkage functions offer another device for thinking about what different penalties do.

Idea: Consider a simple case where problems have explicit solution in terms of MLE.

Shrinkage function captures what penalty function does to MLE in that setting.

E.g. consider least squares estimation with orthonormal input matrix so each $\hat{\beta}_{MLE}$ obtained by marginal regression.
\( \lambda = 1 \) for all cases, \( a = 3 \) for SCAD, \( \lambda_2 = 1 \) for lava
Penalized estimators have equivalent formulation in terms of a constrained optimization problem:

E.g. LS criterion:

\[
\hat{\beta}_P = \arg \min_{\beta} \sum_{i=1}^{n} (y_i - x_i' \beta)^2
\]

subject to \( p(\beta) \leq M \)

Equivalent to penalized formulation when \( M \) and \( \lambda \) set appropriately

Easy to see how kinked penalties lead to variable selection in this framework
Contours of LS criterion in 2-D case and shape of constraint region (LASSO - diamond, Ridge - circle). (Figure 6.7 from ISL.)
Choice of penalty parameter

With penalty selected to produce desired shrinkage/regularization, still need to choose appropriate tuning parameters (e.g. $\lambda$)

CV standard in statistics

Can also do some theory in this case to get some guidance (e.g. BCH 2012, BCHK 2016)
  - Probably especially useful in non-iid/dependent data settings

Convex penalties are computationally very convenient
  - E.g. fast/efficient algorithms for LASSO produce solution path for ALL $\lambda$
Example: Baseball Data

Let’s look at ridge and LASSO in building a model to predict a baseball player’s salary given performance metrics.

Code: Hitters_Example2.R

- $Y \ (Salary)$: Salary in $1000 in 1987
- $X_1 \ (AtBat)$: Number of at bats in 1986
- ...
- $X_{19} \ (NewLeague)$: Player’s league at start of 1987 (American or National)

All x-variables standardized
Example: Baseball Data

Plot Ridge coefficients against $\log(1/\lambda)$

Move smoothly between (essentially) 0 and (essentially) the unpenalized values
Example: Baseball Data

Can choose which value of $\lambda$ to use based on CV:
Example: Baseball Data

Plot **LASSO** coefficients against log(1/\(\lambda\))

Move (fairly) smoothly between (exactly) 0 and (essentially) the unpenalized values

Coefficients get zeroed out along the path
Can choose which value of $\lambda$ to use based on CV:

best lambda is: 0.87
How do CV-min Ridge and LASSO coefficients compare to unpenalized?

![Graph comparing linear coefficients to ridge and lasso coefficients](image-url)
Example: Baseball Data

How do CV-min Ridge and LASSO fitted values compare to unpenalized?

- y
- Linear
- Ridge
- Lasso
Pretty close agreement in this example

LASSO model is slightly more parsimonious than ridge or full

<table>
<thead>
<tr>
<th></th>
<th>Int.</th>
<th>AtBat</th>
<th>Hits</th>
<th>HmRun</th>
<th>Runs</th>
<th>RBI</th>
<th>Walks</th>
<th>Years</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>535.9</td>
<td>-291.6</td>
<td>338.5</td>
<td>37.9</td>
<td>-60.7</td>
<td>-27.0</td>
<td>135.3</td>
<td>-16.7</td>
</tr>
<tr>
<td>Ridge</td>
<td>535.9</td>
<td>-258.0</td>
<td>278.4</td>
<td>11.6</td>
<td>-19.6</td>
<td>-3.3</td>
<td>120.2</td>
<td>-44.6</td>
</tr>
<tr>
<td>LASSO</td>
<td>535.9</td>
<td>-286.4</td>
<td>310.1</td>
<td>12.5</td>
<td>-29.9</td>
<td>0</td>
<td>123.4</td>
<td>-34.8</td>
</tr>
<tr>
<td>OLS</td>
<td>-391.8</td>
<td>86.9</td>
<td>-14.2</td>
<td>481.7</td>
<td>261.2</td>
<td>-214.3</td>
<td>31.3</td>
<td>-58.5</td>
</tr>
<tr>
<td>Ridge</td>
<td>-180.8</td>
<td>138.21</td>
<td>53.95</td>
<td>276.2</td>
<td>132.2</td>
<td>-174.8</td>
<td>31.0</td>
<td>-60.9</td>
</tr>
<tr>
<td>LASSO</td>
<td>-171.2</td>
<td>0</td>
<td>14.5</td>
<td>389.9</td>
<td>192.5</td>
<td>-195.9</td>
<td>24.2</td>
<td>-58.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>Put Outs</th>
<th>Assists</th>
<th>Err</th>
<th>New Lea</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>78.9</td>
<td>53.8</td>
<td>-22.2</td>
<td>-12.4</td>
</tr>
<tr>
<td>Ridge</td>
<td>78.6</td>
<td>45.3</td>
<td>-24.4</td>
<td>-14.0</td>
</tr>
<tr>
<td>LASSO</td>
<td>79.0</td>
<td>43.2</td>
<td>-19.3</td>
<td>-5.9</td>
</tr>
</tbody>
</table>
Look at same set of variables considered for forward selection.

Consider Ridge and LASSO

Choose penalty parameter using 10-fold CV
Example: Boston Housing Data

CV function for Ridge:

Lines give CV-minimizing value and largest value within 1 estimated standard error
Example: Boston Housing Data

CV function for LASSO:

Lines give CV-minimizing value and largest value within 1 estimated standard error
Performance on validation data set:

- Ridge(CV-Min): Validation MSE = 29.23
- Ridge(1SE): Validation MSE = 35.83
- LASSO(CV-Min): Validation MSE = 19.43
- LASSO(1SE): Validation MSE = 20.89
Example: Riboflavin Production

Classic high-dimensional example

\[ n = 71 \text{ observations, } p = 4088 \text{ right-hand-side variables} \]

Data:
- \( Y \) - log(riboflavin production)
- \( X_j \) - log(expression level of gene\( j \)), \( j = 1, \ldots, 4088 \)

Goal: Uncover genes likely associated with riboflavin production
Example: Riboflavin Production

5-Fold CV function for LASSO:
Example: Riboflavin Production

5-Fold CV function for SCAD:

![Graph showing 5-Fold CV function for SCAD with variables selected and cross-validation error on the y-axis and log(λ) on the x-axis. The graph includes a dotted line indicating the selected variables.](image-url)
Estimated (absolute value of) coefficients (blue - lasso-cv-min, red - lasso-cv-1se, green - scad-cv-min):

23, 40, and 16 variables selected by LASSO (1SE), LASSO (MIN), SCAD (MIN)
Look at (simulated) LASSO performance in a couple of simple settings:

**Simulation Design 1:**
- $Y = X_1 + .25\varepsilon$
- $(X_1, \ldots, X_p, \varepsilon)' \sim N(0, I_{p+1})$
- $n = 100, \quad p \in \{1, 5, 20, 50, 90, 100, 200\}$

**Simulation Design 2:**
- $Y = \exp(-X_1/2) + .75X_11(X_1 > 0) + .1(\sum_{j=2}^{5} X_j) + .1(X_2 - X_3 + X_4 - X_5)^2 + .1\varepsilon$
- $(X_1, \ldots, X_p, \varepsilon)' \sim N(0, I_{p+1})$
- $n \in \{100, 200\}, \quad p \in \{1, 5, 20, 50, 100, 200\}$
Increasing $p$ Simulation

Consider (i) lasso using only linear terms and (ii) lasso using nonlinear terms formed by “basis” expansion

Note that there’s a huge computational bottleneck in large $p$ cases.

- Allowing for nonlinearity and interactions in all the terms blows up the computational overhead quickly
- E.g. with $p$ variables and only allowing second order effects there are $p + p(p + 1)/2$ terms - (20300 terms with $p = 200$) - That’s a really big design matrix
- Allow all first and second order terms + dummies for quartiles interacted with linear term
Let’s look at results in linear model first.

LASSO penalty parameter allowing nonlinear terms selected by 5-fold CV

<table>
<thead>
<tr>
<th></th>
<th>$p = 1$</th>
<th>$p = 5$</th>
<th>$p = 20$</th>
<th>$p = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
</tr>
<tr>
<td>OLS</td>
<td>0.056</td>
<td>0.064</td>
<td>0.058</td>
<td>0.069</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.056</td>
<td>0.064</td>
<td>0.057</td>
<td>0.066</td>
</tr>
<tr>
<td>LASSO.NL</td>
<td>0.051</td>
<td>0.068</td>
<td>0.055</td>
<td>0.070</td>
</tr>
</tbody>
</table>
Nonlinear case with $n = 100$

LASSO penalty parameter allowing nonlinear terms selected by 5-fold CV

<table>
<thead>
<tr>
<th></th>
<th>$p = 1$</th>
<th></th>
<th>$p = 5$</th>
<th></th>
<th>$p = 20$</th>
<th></th>
<th>$p = 50$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
</tr>
<tr>
<td>OLS</td>
<td>0.439</td>
<td>0.552</td>
<td>0.518</td>
<td>0.531</td>
<td>0.657</td>
<td>0.601</td>
<td>0.895</td>
<td>0.955</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.420</td>
<td>0.577</td>
<td>0.416</td>
<td>0.577</td>
<td>0.424</td>
<td>0.577</td>
<td>0.379</td>
<td>0.591</td>
</tr>
<tr>
<td>LASSO.NL</td>
<td>0.253</td>
<td>0.407</td>
<td>0.038</td>
<td>0.016</td>
<td>0.169</td>
<td>0.072</td>
<td>0.339</td>
<td>0.339</td>
</tr>
</tbody>
</table>
Nonlinear case with $n = 200$

LASSO penalty parameter allowing nonlinear terms selected by 5-fold CV

<table>
<thead>
<tr>
<th></th>
<th>$p = 1$</th>
<th></th>
<th>$p = 5$</th>
<th></th>
<th>$p = 20$</th>
<th></th>
<th>$p = 50$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
</tr>
<tr>
<td>OLS</td>
<td>0.270</td>
<td>0.534</td>
<td>0.330</td>
<td>0.503</td>
<td>0.341</td>
<td>0.541</td>
<td>0.445</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.286</td>
<td>0.535</td>
<td>0.321</td>
<td>0.503</td>
<td>0.307</td>
<td>0.526</td>
<td>0.331</td>
</tr>
<tr>
<td>LASSO.NL</td>
<td>0.230</td>
<td>0.369</td>
<td>0.015</td>
<td>0.017</td>
<td>0.031</td>
<td>0.032</td>
<td>0.086</td>
</tr>
</tbody>
</table>

For $p = 100$ and $p = 200$, computation and storage of matrices is a headache even in these tiny examples

Consider *ex ante* screening to eliminate very unlikely candidates before analysis
4. Screening
Screening

- Idea: Look at marginal correlation (for linear models)
- Rank variables by marginal correlations and only consider the $d$ variables with the highest correlation
- Fan and Lv (2008) suggest $d = n - 1$ or $d = n / \log(n)$
- Can be shown to work theoretically (in the sense of not discarding relevant variables) under some strong conditions
- Intuitive and natural idea that can make life much easier in very big problems
Increasing $p$ Simulation

Do SIS with $d = 5n$

- NOT one of Fan and Lv’s suggestions
- tractable at this scale
- decrease possibility of spuriously excluding a variable from consideration
- Cheating a little by screening on the whole data set (understate CV)
Increasing $p$ Simulation

Linear model results:

<table>
<thead>
<tr>
<th></th>
<th>$p = 50$</th>
<th></th>
<th>$p = 100$</th>
<th></th>
<th>$p = 200$</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
</tr>
<tr>
<td>OLS</td>
<td>0.134</td>
<td>0.119</td>
<td>1.337</td>
<td>3.011</td>
<td>0.798</td>
<td>0.554</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.059</td>
<td>0.070</td>
<td>0.059</td>
<td>0.071</td>
<td>0.059</td>
<td>0.071</td>
</tr>
<tr>
<td>LASSO.NL</td>
<td>0.060</td>
<td>0.076</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>SC.LASSO.NL</td>
<td>0.057</td>
<td>0.077</td>
<td>0.056</td>
<td>0.095</td>
<td>0.064</td>
<td>0.091</td>
</tr>
</tbody>
</table>

Not a huge loss relative to LASSO with everything in $p = 50$
Nonlinear case with $n = 100$

<table>
<thead>
<tr>
<th>Method</th>
<th>CV (p=50)</th>
<th>MSE (p=50)</th>
<th>CV (p=100)</th>
<th>MSE (p=100)</th>
<th>CV (p=200)</th>
<th>MSE (p=200)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLS</td>
<td>0.895</td>
<td>0.955</td>
<td>3.395</td>
<td>11.809</td>
<td>2.781</td>
<td>2.712</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.379</td>
<td>0.591</td>
<td>0.407</td>
<td>0.578</td>
<td>0.408</td>
<td>0.578</td>
</tr>
<tr>
<td>LASSO.NL</td>
<td>0.339</td>
<td>0.339</td>
<td>.</td>
<td>.</td>
<td>.</td>
<td>.</td>
</tr>
<tr>
<td>SC.LASSO.NL</td>
<td>0.283</td>
<td>0.279</td>
<td>0.247</td>
<td>0.426</td>
<td>0.194</td>
<td>0.473</td>
</tr>
</tbody>
</table>
Nonlinear case with $n = 200$

<table>
<thead>
<tr>
<th>Method</th>
<th>$p = 50$</th>
<th>$p = 100$</th>
<th>$p = 200$</th>
</tr>
</thead>
<tbody>
<tr>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
</tr>
<tr>
<td>OLS</td>
<td>0.445</td>
<td>0.607</td>
<td>0.709</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.331</td>
<td>0.524</td>
<td>0.355</td>
</tr>
<tr>
<td>LASSO.NL</td>
<td>0.086</td>
<td>0.060</td>
<td>.</td>
</tr>
<tr>
<td>SC.LASSO.NL</td>
<td>0.077</td>
<td>0.069</td>
<td>0.144</td>
</tr>
</tbody>
</table>
5. Post-Penalized Estimation
Penalized estimators with convex penalty functions computationally convenient (coupled with convex loss)

Shrink all coefficients towards 0
  ▶ Great for coefficients that are really zero (or vanishingly small)
  ▶ May lead to substantial biases for non-zero coefficients

Non-convex penalties (e.g. SCAD, $\ell_p$ with $p < 1$) motivated by desire to mitigate this bias

Adaptive LASSO alters penalty loadings based on first step estimate to alleviate bias
Post-Penalized Estimation (Belloni and Chernozhukov (2013))

Computationally simple and intuitive idea to undo shrinkage bias by applying unpenalized estimator using only variables selected to have non-zero coefficient

E.g.

$$\hat{\beta}_L = \arg \min _{\beta} \sum _{i=1} ^n (y_i - x_i^T \beta) ^2 + \lambda \| \beta \|_1$$

Post-LASSO estimator

$$\hat{\beta}_{PL} = \arg \min _{\beta : \beta_j = 0 \ \forall \ j \text{ such that } \hat{\beta}_L,j = 0} \sum _{i=1} ^n (y_i - x_i^T \beta) ^2$$
Results in Belloni and Chernozhukov (2013) suggest Post-LASSO works at least as well as LASSO and sometimes much better.

Seems to work well with theoretically driven plug-in penalty given in Belloni and Chernozhukov (2013).

Be careful with cross-validation.
Let’s look at linear model simulation again, just using linear terms.

Code: PLASSOIncreaseP_Example9.R

<table>
<thead>
<tr>
<th></th>
<th>p = 1</th>
<th></th>
<th>p = 5</th>
<th></th>
<th>p = 20</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
</tr>
<tr>
<td>OLS</td>
<td>0.056</td>
<td>0.064</td>
<td>0.058</td>
<td>0.069</td>
<td>0.076</td>
<td>0.074</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.056</td>
<td>0.064</td>
<td>0.057</td>
<td>0.066</td>
<td>0.058</td>
<td>0.066</td>
</tr>
<tr>
<td>P.LASSO</td>
<td>0.056</td>
<td>0.064</td>
<td>0.056</td>
<td>0.064</td>
<td>0.056</td>
<td>0.064</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>p = 50</th>
<th></th>
<th>p = 100</th>
<th></th>
<th>p = 200</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
<td>CV</td>
<td>MSE</td>
</tr>
<tr>
<td>OLS</td>
<td>0.134</td>
<td>0.119</td>
<td>1.337</td>
<td>3.011</td>
<td>0.798</td>
<td>0.554</td>
</tr>
<tr>
<td>LASSO</td>
<td>0.059</td>
<td>0.070</td>
<td>0.059</td>
<td>0.071</td>
<td>0.059</td>
<td>0.071</td>
</tr>
<tr>
<td>P.LASSO</td>
<td>0.056</td>
<td>0.064</td>
<td>0.056</td>
<td>0.064</td>
<td>0.056</td>
<td>0.064</td>
</tr>
</tbody>
</table>

Oracle performance! - i.e. as good as using true model (for each $p$)

Note that this will not hold up when “linear” model is an approximation
6. A Little Lasso Theory
Lasso Problem

Recall that lasso estimates parameters by solving

\[
\hat{\beta}_p = \arg \min_{\beta} \sum_{i=1}^{n} (y_i - x_i' \beta)^2 + \lambda \sum_{j=1}^{p} |\psi_j \beta_j|
\]

Lasso problem is convex (has a unique solution) but is not differentiable

Can find solution by looking at subdifferential

- subderivative of function \( f(\cdot) \) at point \( x_0 \) is a set of vectors \( v \) such that \( f(x) - f(x_0) \geq v'(x - x_0) \)
- at a point where a function is differentiable, subdifferential is the conventional gradient
- a convex function is minimized at the point where 0 is included in the subdifferential
Scalar Lasso problem

Specialize to case where \( \dim(x_i) = 1, \frac{1}{n} \sum_i x_i^2 = 1, \psi_1 = 1 \), so lasso solves

\[
\hat{\beta}_P = \arg\min_{\beta} Q(\beta) = \arg\min_{\beta} \sum_{i=1}^{n} (y_i - \beta x_i)^2 + \lambda |\beta|
\]

Subdifferential:

\[
\partial Q(\beta) = -2x' y + 2n\beta + \lambda \text{ if } \beta > 0 \\
= -2x' y + 2n\beta - \lambda \text{ if } \beta < 0 \\
\in -2x' y + 2n\beta + s\lambda \text{ for } s \in [-1, 1] \text{ if } \beta = 0
\]

Estimator \( \hat{\beta}_P \) found at point where 0 is in the subdifferential at that point:

\[
\hat{\beta}_P = \begin{cases} 
\frac{1}{n} x' y - \frac{1}{2n} \lambda & \text{if } \frac{1}{n} x' y - \frac{1}{2n} \lambda > 0 \\
\frac{1}{n} x' y + \frac{1}{2n} \lambda & \text{if } \frac{1}{n} x' y + \frac{1}{2n} \lambda < 0 \\
0 & \text{if } |\frac{1}{n} x' y| \leq \frac{1}{2n} \lambda
\end{cases}
\]
Intuition for penalty parameter choice

Estimate $\beta$ to be exactly 0 whenever $|\frac{1}{n} x' y| \leq \frac{1}{2n} \lambda$

A desirable property would be that we get $\hat{\beta} = 0$ when $\beta$ really is 0 with high-probability

- get this by choosing any $\lambda$ “big enough”
- but big $\lambda$ implies more shrinkage on non-zero coefficients

Implies choosing $\lambda$ such that

$$\Pr(|\frac{1}{\sqrt{n}} x' \varepsilon| \leq \frac{1}{2\sqrt{n}} \lambda) \to 1$$

- $\frac{1}{\sqrt{n}} x' \varepsilon \sim N(0, \sigma^2)$ [assuming, e.g., iid sampling, $\varepsilon \perp x$, and $E[\varepsilon^2] = \sigma^2$]
- Suggests choosing $\lambda = 2\sqrt{n}\sigma \Phi^{-1}(1 - \gamma_n/2)$ for $\gamma_n \to 0$
Look back at general problem

\[ \hat{\beta}_P \in \arg \min_b \frac{1}{n} \sum_{i=1}^{n} (y_i - x_i' b)^2 + \lambda \frac{1}{n} \sum_{j=1}^{p} \hat{\phi}_j |b_j|. \]

Need to choose \( \lambda \) and \( \hat{\phi}_j \), \( 1 \leq j \leq p \).

Key to good selection properties of Lasso is choosing these so that

\[ \frac{\lambda \hat{\phi}_j}{n} \geq 2c \left| \frac{1}{n} \sum_{i=1}^{n} x_{j,i} i_{i,i} \right| \quad \text{for each } 1 \leq j \leq p \]

occurs with high probability.
General Intuition for Choice of $\lambda$

1. Previous inequality holding $\iff \lambda/\sqrt{n} \geq 2c \left| \frac{1}{\sqrt{n}\phi_j} \sum_{i=1}^{n} x_{j,i} \epsilon_i \right|$ for each $1 \leq j \leq p$.
   - Setting $\lambda/\sqrt{n}$ large enough to dominate $p$ standard normals would work if $\frac{1}{\sqrt{n}\phi_j} \sum_{i=1}^{n} x_{j,i} \epsilon_i$ were standard normal.
   - $\lambda = 2c\sqrt{n}\Phi^{-1}(1 - \gamma_n/2p)$ with $\gamma_n = o(1)$ will implement this
2. Need $\phi_j$ to be an appropriate measure of the variability of $\frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{j,i} \epsilon_i$

  ▶ “Ideally”: $\hat{\phi}_j = \phi_j$ where

  $$\phi_j^2 = \text{Var} \left( \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{j,i} \epsilon_i \right)$$

  ▶ Suggests using $\hat{\phi}_j$ a consistent estimator of $\text{Var} \left( \frac{1}{\sqrt{n}} \sum_{i=1}^{n} x_{j,i} \epsilon_i \right)$

  ▶ Results for independent, heteroskedastic case given in Belloni, Chernozhukov, and Hansen (2012) using Huber-Eicker-White variance estimator

  ▶ Results for clustered case given in Belloni, Chernozhukov, Hansen, and Kozbur (2016) using clustered variance estimator

  ▶ Both rely on application/extension of moderate deviation theory of Jing, Shao, and Wang (2003)
To implement in practice, need to form $\hat{\phi}_j$.

Feasible iterative procedure:

1. Form initial guess about $\{\epsilon_i\}_{i=1}^n$, $\{\hat{\epsilon}_i\}_{i=1}^n$
   - Simple choice is to set $\hat{\epsilon}_i = y_i$
   - Another choice is to set $\hat{\epsilon}_i = y_i - (x_i^0)'\hat{\beta}^0$ where $x_i^0$ is a (small) set of initial variables thought likely to be important and $\hat{\beta}^0$ are the associated least squares regression coefficients

2. Form $\hat{\phi}_j = \sqrt{\text{Var} \left( \frac{1}{\sqrt{n}} \sum_{i=1}^n x_{j,i} \epsilon_i \right)}$ using $\hat{\epsilon}_i$ in place of $\epsilon_i$

3. Estimate lasso coefficients with $\lambda$ given above and $\hat{\phi}_j \rightarrow \hat{\beta}_P$

4. Update $\hat{\epsilon}_i = y_i - x_i'\hat{\beta}_P$

5. Repeat 2-4 a small number of times.
Some Formal Properties

- Let $s = \|\beta\|_0$
- $a_n s$ for $a_n \to \infty$ dimensional submatrices of $\frac{1}{n} \sum_i x_i x_i'$ have minimum and maximum eigenvalue bounded with probability approaching one
- Lots of bounded eigenvalue moments
- $\frac{s^2 \log^2(p)}{n} \to 0$ and $\frac{\log^3(p)}{n} \to 0$

Results:
- $\frac{1}{n} \sum_i (x_i' \beta - x_i' \hat{\beta}_P)^2 = O_p( s \log(p)/n )$
  - Best possible forecast rate
- $\hat{s} = O(s)$
  - Selected model has similar size to true model - note that no guarantee you get the right variables

Pretty similar to testing based forward selection
7. Concluding Comments
Some remarks on penalized estimation/subset selection:

- Penalized/selection estimators attractive for a variety of reasons
  - Produce results interpretable within familiar modeling frameworks
  - Can perform remarkably well in forecasting (and other contexts) with well-chosen variables and methods
  - Reasonably easy to build in functional restrictions (e.g. monotonicity, shape constraints, etc.)
  - Readily extended to the usual models (e.g. penalized logistic regression)
  - Many extensions, related methods (e.g. fused LASSO and smoothing splines for functional data, group LASSO, etc.)
  - Structural amenable to theoretical analysis

- Have some unappealing feature
  - Bookkeeping to deal with nonlinearities, interactions, etc. with even a few X’s gets annoying
  - Need to construct all the relevant terms - memory/computation intensive
  - Not automatic - e.g. if you didn’t think of the interaction and include it in the set of candidate variables, you won’t find it (other methods - e.g. trees and random forests do this)