NBER Lectures on Machine Learning

An Introduction to Supervised and Unsupervised Learning

July 18th, 2015, Cambridge

Susan Athey & Guido Imbens - Stanford University

Outline

1. Introduction
   (a) Supervised Learning: Classification
   (b) Supervised Learning: Regression
   (c) Unsupervised Learning: Clustering
   (d) Unsupervised Learning: Association Rules
2. Supervised Learning: Regression
   (a) Linear Regression with Many Regressors: Ridge and Lasso

(b) Regression Trees

(c) Boosting

(d) Bagging

(e) Random Forests

(f) Neural Networks and Deep Learning

(g) Ensemble Methods and Super Learners

3. Unsupervised Learning

   (a) Principal Components

   (b) Mixture Models and the EM Algorithm

(c) k-Means Algorithm

(d) Association Rules and the Apriori Algorithm

4. Classification: Support Vector Machines

1. Introduction

• Machine learning methods are about algorithms, more than about asymptotic statistical properties. No unified framework, like maximum likelihood estimation.

• "Does it work in practice?" rather than "What are its formal properties?" Liberating, but also lack of discipline and lots of possibilities.

• Setting is one with large data sets, sometimes many units, sometimes many predictors. Scalability is a big issue

• Causality is de-emphasized. Methods are largely about prediction and fit. That is important in allowing validation of methods through out-of-sample crossvalidation.

• Lots of terminology, sometimes new terms for same things: "training" instead of "estimating", etc.

Key Components

A. Out-of-sample Cross-validation: Methods are validated by assessing their properties out of sample.

This is much easier for prediction problems than for causal problems. For prediction problems we see realizations so that a single observation can be used to estimate the quality of the prediction: a single realization of (Y_i, X_i) gives us an unbiased estimate of $\mu(x) = E[Y_i|X_i = x]$, namely Y_i.

For causal problems we do not generally have unbiased estimates of the true causal effects.

Use "training sample" to "train" (estimate) model, and "test sample" to compare algorithms.

B. Regularization: Rather than simply choosing the best fit, there is some penalty to avoid over-fitting.

• Two issues: choosing the form of the regularization, and choosing the amount of regularization.

Traditional methods in econometrics often used plug-in methods: use the data to estimate the unknown functions and express the optimal penalty term as a function of these quantities. For example, with nonparametric density estimation, researchers use the Silverman bandwith rule.

The machine learning literature has focused on out-of-sample cross-validation methods for choosing amount of regularization (value of penalty).

Sometimes there are multiple tuning parameters, and more structure needs to be imposed on selection of tuning parameters.

C. Scalability: Methods that can handle large amounts of data

• large number of units/observations and/or large number of predictors/features/covariates) and perform repeatedly without much supervision.

The number of units may run in the billions, and the number of predictors may be in the millions.

The ability to parallelize problems is very important (map-reduce).

Sometimes problems have few units, and many more predictors than units: genome problems with genetic information for a small number of individuals, but many genes.

1.a Supervised Learning: Classification

• One example of supervised learning is **classification**

• N observations on pairs (Y_i, X_i), Y_i is element of unordered set $\{0, 1, \ldots, J\}$.

• Goal is to find a function $g(x; \mathbf{X}, \mathbf{Y})$ that assigns a new observation with $X_{N+1} = x$ to one of the categories (less interest in probabilities, more in actual assignment). X_{N+1} is draw from same distribution as X_i, $i = 1, \ldots, N$.

• Big success: automatic reading of zipcodes: classify each handwritten digit into one of ten categories. No causality, pure prediction. Modern problems: face recognition in pictures.

1.b Supervised Learning: Regression

• One example familiar from economic literature is nonparametric regression: **many cases where we need simply a good fit for the conditional expectation.**

• N observations on pairs (Y_i, X_i). Goal is to find a function $g(x; \mathbf{X}, \mathbf{Y})$ that is a good predictor for Y_{N+1} for a new observation with $X_{N+1} = x$.

• Widely used methods in econometrics: kernel regression

$$g(x|\mathbf{X}, \mathbf{Y}) = \sum_{i=1}^{N} Y_i \cdot K\left(\frac{X_i - x}{h}\right) \bigg/ \sum_{i=1}^{N} K\left(\frac{X_i - x}{h}\right)$$

Kernel regression is useful in some cases (e.g., regression discontinuity), but does not work well with high-dimensional x.

• Compared to econometric literature the machine learning literature focuses less on asymptotic normality and properties, more on out-of-sample crossvalidation.

• There are few methods for which inferential results have been established. Possible that these can be established (e.g., random forests), but probably not for all, and not a priority in this literature.

• Many supervised learning methods can be adapted to work for classification and regression. Here I focus on estimating regression functions because that is more familiar to economists.

2.a Linear Regression with Many Regressors: Ridge and Lasso

Linear regression:

$$Y_i = \sum_{k=1}^{K} X_{ik} \cdot \beta_k + \varepsilon_i = X_i'\beta + \varepsilon_i$$

We typically estimate β by ordinary least squares

$$\hat{\beta}_{\text{ols}} = \left(\sum_{i=1}^{N} X_i \cdot X_i' \right)^{-1} \left(\sum_{i=1}^{N} X_i \cdot Y_i \right) = \left(\mathbf{X}'\mathbf{X} \right)^{-1} \left(\mathbf{X}'\mathbf{Y} \right)$$

This has good properties for estimating β given this model (<u>best linear unbiased estimator</u>). But, these are limited optimality properties: with $K \geq 3$ ols is <u>not admissible</u>. The predictions $x'\hat{\beta}$ need not be very good, especially with large K.

What to do with many covariates (large K)? (potentially millions of covariates, and either many observations or modest number of observations, e.g., genome data)

• Simple ols is not going to have good properties. (Like flat prior in high-dimensional space for a Bayesian.)

Zvi Grilliches: "never trust ols with more than five regressors"

• We need some kind of regularization

Vapnik (of "support vector machines" fame): "Regularization theory was one of the first signs of the existence of intelligent inference"

Approaches to Regularization in Regression

- Shrink estimates continuously towards zero.

- Limit number of non-zero estimates: sparse representation.

"bet on the sparsity principle: use a procedure that does well in sparse problems, since no procedure does well in dense problems" (Hastie, Tibshirani and Wainwright, 2015, p. 2)

- Combination of two

Subset Selection

Find the set of t regressors that minimizes the sum of squared residuals

$$\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_0 \quad \text{where} \quad \|\beta\|_0 = \sum_{k=1}^{K} \mathbf{1}_{\beta_k \neq 0}$$

This is hard computationally, and has awkward properties: The single best covariate is not necessarily included in the set of two best covariates.

It is only feasible for modest values of K (does not scale). Greedy versions are available (sequentially selecting covariates).

Ridge Regression: Starting with the regression model

$$Y_i = \sum_{k=1}^{K} X_{ik} \cdot \beta_k + \varepsilon_i = X_i'\beta + \varepsilon_i$$

We estimate β as

$$\widehat{\beta}_{\text{ridge}} = \left(\mathbf{X}'\mathbf{X} + \lambda \cdot \mathbf{I}_K\right)^{-1} \left(\mathbf{X}'\mathbf{Y}\right)$$

We inflate the $\mathbf{X}'\mathbf{X}$ matrix by $\lambda \cdot \mathbf{I}_K$ so that it is positive definite irrespective of K, including $K > N$.

The solution has a nice interpretation: If the prior distribution for β is $\mathcal{N}(0, \tau^2 \cdot \mathbf{I}_K)$, and the distribution of ε_i is normal $\mathcal{N}(0, \sigma^2)$, if $\lambda = \sigma^2/\tau^2$, then $\widehat{\beta}_{\text{ridge}}$ is the posterior mean/mode/median.

The ols estimates are shrunk smoothly towards zero: if the X_{ik} are orthonormal, all the ols coeffs shrink by a factor $1/(1+\lambda)$.

LASSO

“Least Absolute Selection and Shrinkage Operator” (Tibshirani, 1996)

$$\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_1$$

This uses “L_1” norm.

L_p norm is

$$\|x\|_p = \left(\sum_{k=1}^{K} |x_k|^p \right)^{1/p}$$

Andrew Gelman: “Lasso is huge”

Compare L_1 norm to L_2 norm which leads to ridge:

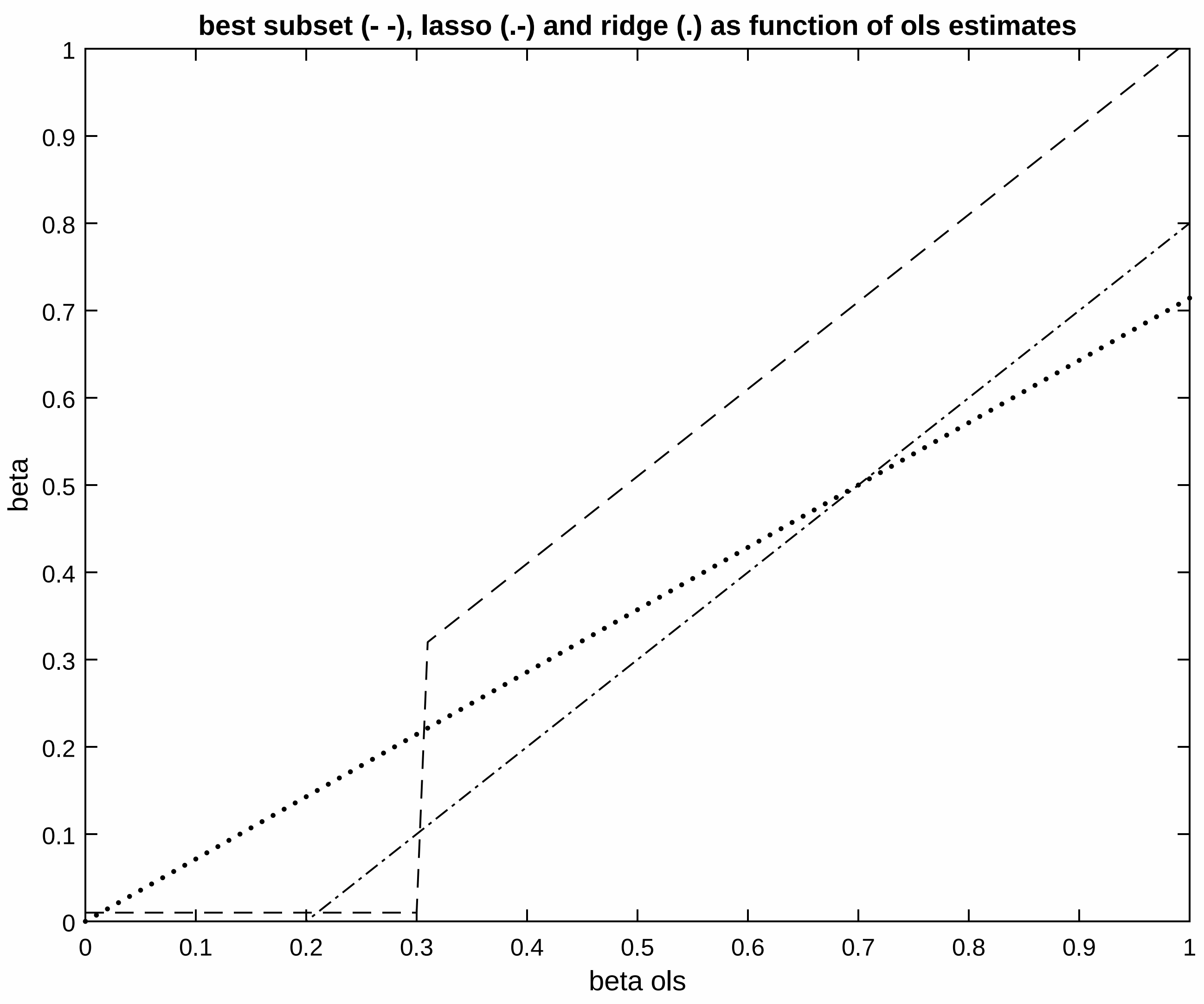
$$\hat{\beta}_{\text{ridge}} = \min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_2^2$$

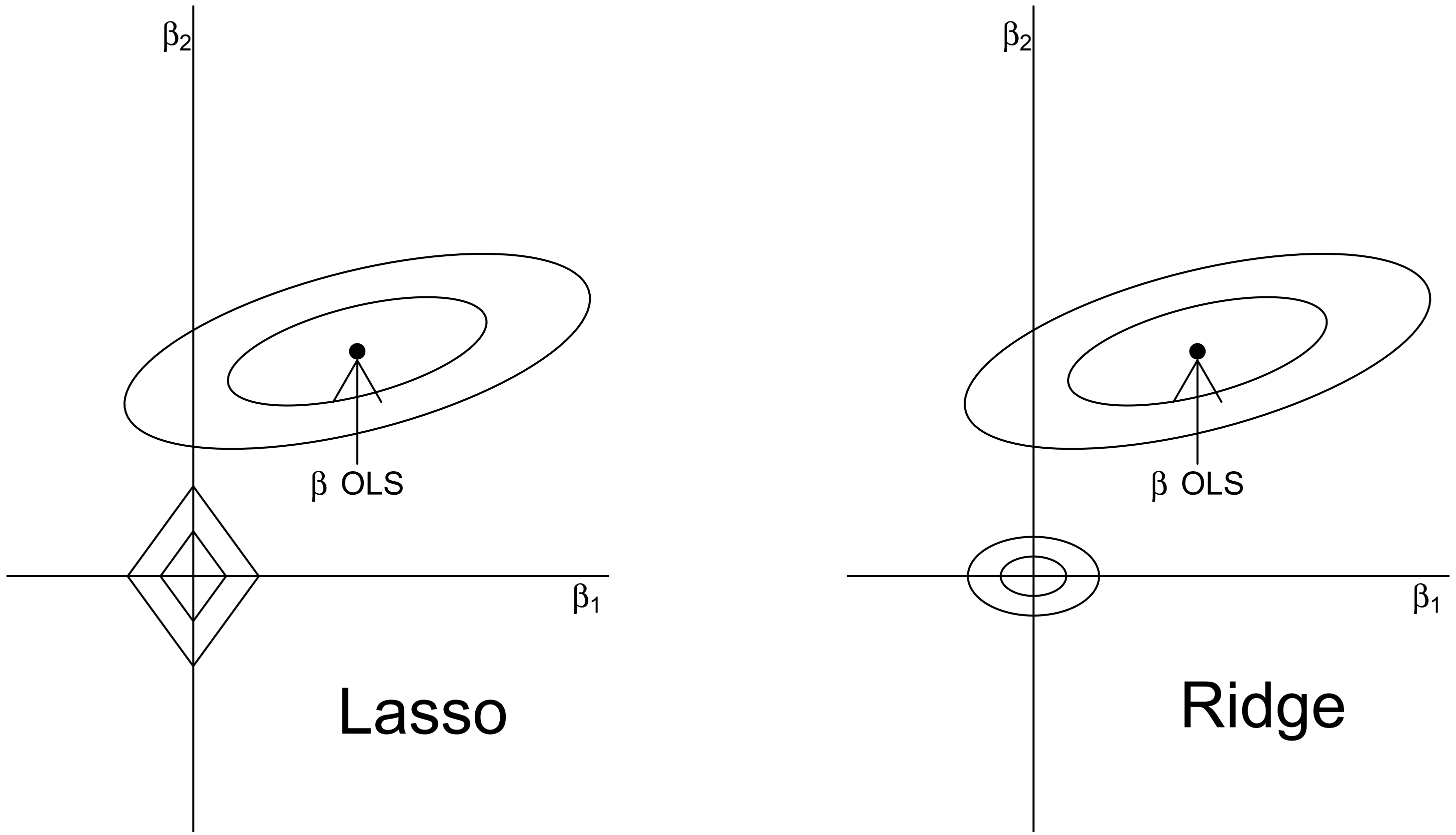
Now all estimates are shrunk towards zero smoothly, no zero estimates. This is easy computationally for modest K.

Or L_0 norm leading to subset selection:

$$\hat{\beta}_{\text{subset}} = \min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_0$$

Non-zero estimates are simple ols estimates. This is computationally challenging (combinatorial problem), but estimates are interpretable.


best subset (- -), lasso (.-) and ridge (.) as function of ols estimates
beta
beta ols
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1


β_2
β_1
β OLS
Lasso
β_2
β_1
β OLS
Ridge

Related Methods

- LARS (Least Angle RegreSsion - "The 'S' suggesting LASSO and Stagewise") is a stagewise procedure that iteratively selects regressors to be included in the regression function. It is mainly used as an algorithm for calculating the LASSO coefficients as a function of the penalty parameter.

- Dantzig Selector (Candes & Tao, 2007)

$$\min_{\beta} \left\{ \max_{k=1}^{K} \left| \sum_{i=1}^{N} X_{ik} \cdot (Y_i - X_i \beta) \right| \right\} \qquad \text{s.t. } \|\beta\|_1 \le t$$

LASSO type regularization, but minimizing the maximum correlation between residuals and covariates.

Interesting properties, but does not seem to work well for prediction.

• LASSO is most popular of machine learning methods in econometrics.

See Belloni, Chernozhukov, and Hansen Journal of Economic Perspective, 2014 for general discussion and referencesto economics literature.

What is special about LASSO?

LASSO shrinks some coefficients exactly to zero, and shrinks the others towards zero.

(Minor modification is <u>relaxed</u> or <u>post</u> LASSO which uses LASSO to selecte non-zero coefficients and then does simple ols on those covariates without shrinking them towards zero.)

• Interpretability: few non-zero estimates, you can discuss which covariates matter, unlike ridge.

• Good properties when the true model is sparse (bet on sparsity).

• No analytic solution, unlike ridge/L_2 regression, but computationally attractive in very large data sets (convex optimization problem) where $\mathbf{X}'\mathbf{X} + \lambda \cdot \mathbf{I}_K$ may be too large to invert.

• If we have a lot of regressors, what do we think the distribution of the "true" parameter values is? Probably there are some regressors that matter a lot, and a lot that matter very little. Shrinking the large estimates a lot, as ridge does, may not be effective.

• If the distribution of coefficients is very thick-tailed, LASSO may do much better than ridge. On the other hand, if there are a lot of modest size effects, ridge may do better. (see discussion in original Tibshirani 1996 paper)

• Focus on covariate selection has awkward aspect. Consider the case where we estimate a population mean by LASSO:

$$\min_{\mu} \sum_{i=1}^{N} (Y_i - \mu)^2 + \lambda \cdot |\mu|.$$

The estimate is zero if $|\overline{Y}| \leq c$, $\overline{Y} - c$ if $\overline{Y} > c$, and $\overline{Y} + c$ if $\overline{Y} < -c$.

Relaxed LASSO here is zero if $|\overline{Y}| \leq c$, and $\overline{Y}$ if $|\overline{Y}| > c$. This is like a super-efficient estimator, which we typically do not like.

- Performance with highly correlated regressors is unstable.

Suppose $X_{i1} = X_{i2}$, and suppose we try to estimate

$$Y_i = \beta_0 + \beta_1 \cdot X_{i1} + \beta_2 \cdot X_{i2} + \varepsilon_i.$$

Ridge regression would lead to $\widehat{\beta}_1 = \widehat{\beta}_2 \approx (\beta_1 + \beta_2)/2$ (plus some shrinkage).

Lasso would be indifferent between $\widehat{\beta}_1 = 0$, $\widehat{\beta}_2 = \beta_1 + \beta_2$ and $\widehat{\beta}_1 = \beta_1 + \beta_2$, $\widehat{\beta}_2 = 0$.

LASSO as a Bayesian Estimator

We can also think of LASSO being the mode of the posterior distribution, given a normal linear model and a prior distribution that has a Laplace distribution

$$p(\beta) \propto \exp\left(-\lambda \cdot \sum_{k=1}^{K} |\beta_k|\right)$$

• For a Bayesian using the posterior mode rather than the mean is somewhat odd as a point estimate (but key to getting the sparsity property, which generally does not hold for posterior mean).

• Also it is less clear why one would want to use this prior rather than a normal prior which would lead to an explicit form for the posterior distribution.

• Related Bayesian Method: spike and slab prior.

In regression models we typically use normal prior distributions, which are conjugate and have nice properties.

More in line with LASSO is to use a prior that is a mixture of a distribution that has point mass at zero (the spike) and a flatter component (the slab), say a normal distribution:

$$f(\beta) = \begin{cases} p & \text{if } \beta = 0, \\ (1-p) \cdot \frac{1}{(2\pi\tau^2)^{K/2}} \exp\left(-\frac{\beta'\beta}{2\tau^2}\right) & \text{otherwise.} \end{cases}$$

Implementation and Choosing the LASSO Penalty Parameter

We first standardize the X_i so that each component has mean zero and unit variance. Same for Y_i, so no need for intercept. We can rewrite the problem

$$\min_\beta \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \|\beta\|_1$$

as

$$\min_\beta \sum_{i=1}^{N} (Y_i - X_i\beta)^2 \quad \text{s.t. } \sum_{k=1}^{K} |\beta_k| \leq t \cdot \sum_{k=1}^{K} |\hat{\beta}_k^{\text{ols}}|$$

Now t is a scalar between 0 and 1, with 0 corresponding to shrinking all estimates to 0, and 1 corresponding to no shrinking and doing ols.

Typically we choose the penalty parameter λ (or t) through crossvalidation. Let $I_i \in \{1, \ldots, B\}$ be an integer indicating the B-th crossvalidation sample. (We could choose $B = N$ to do leave-one-out crossvalidation, but that would be computationally difficult, so often we randomly select $B = 10$ crossvalidation samples).

For crossvalidation sample b, for $b = 1, \ldots, B$, estimate $\hat{\beta}_b(\lambda)$

$$\hat{\beta}_b(\lambda) = \arg\min_{\beta} \sum_{i:I_i \neq b} (Y_i - X_i\beta)^2 + \lambda \cdot \sum_{k=1}^{K} |\beta_k|$$

on all data with $B_i \neq b$. Then calculate the sum of squared errors for the data with $B_i = b$:

$$Q(b, \lambda) = \sum_{i:I_i = b} \left(Y_i - X_i'\hat{\beta}_b(\lambda)\right)^2$$

We can calculate the average of this over the crossvalidation samples and its standard error:

$$\overline{Q}(\lambda) = \frac{1}{B}\sum_{b=1}^{B} Q(b,\lambda), \qquad \text{se}(\lambda) = \left(\frac{1}{B^2}\sum_{b=1}^{B}\Big(Q(b,\lambda) - \overline{Q}(\lambda)\Big)^2\right)^{1/2}$$

We could choose

$$\hat{\lambda}_{\text{min}} = \arg\min_{\lambda} \overline{Q}(\lambda)$$

That tends to overfit a bit, and so Hastie, Tibshirani and Friedman (2009) recommend using the largest λ (sparsest model) such that

$$\overline{Q}(\lambda) \leq \overline{Q}(\hat{\lambda}_{\text{min}}) + \text{se}(\hat{\lambda}_{\text{min}}))$$

Oracle Property

If the true model is sparse, so that there are few (say, P_N) non-zero coefficients, and many (the remainder $K_N - P_N$) zero coefficients, and P_N goes to infinity slowly, whereas K_N may go to infinity fast (e.g., proportional to N), inference as if you know a priori exactly which coefficients are zero is valid. Sample size needs to be large relative to

$$P_N \cdot \left(1 + \ln\left(\frac{K_N}{P_N}\right)\right)$$

In that case you can ignore the selection of covariates part, that is not relevant for the confidence intervals. **This provides cover for ignoring the shrinkage and using regular standard errors.** Of course in practice it may affect the finite sample properties substantially.

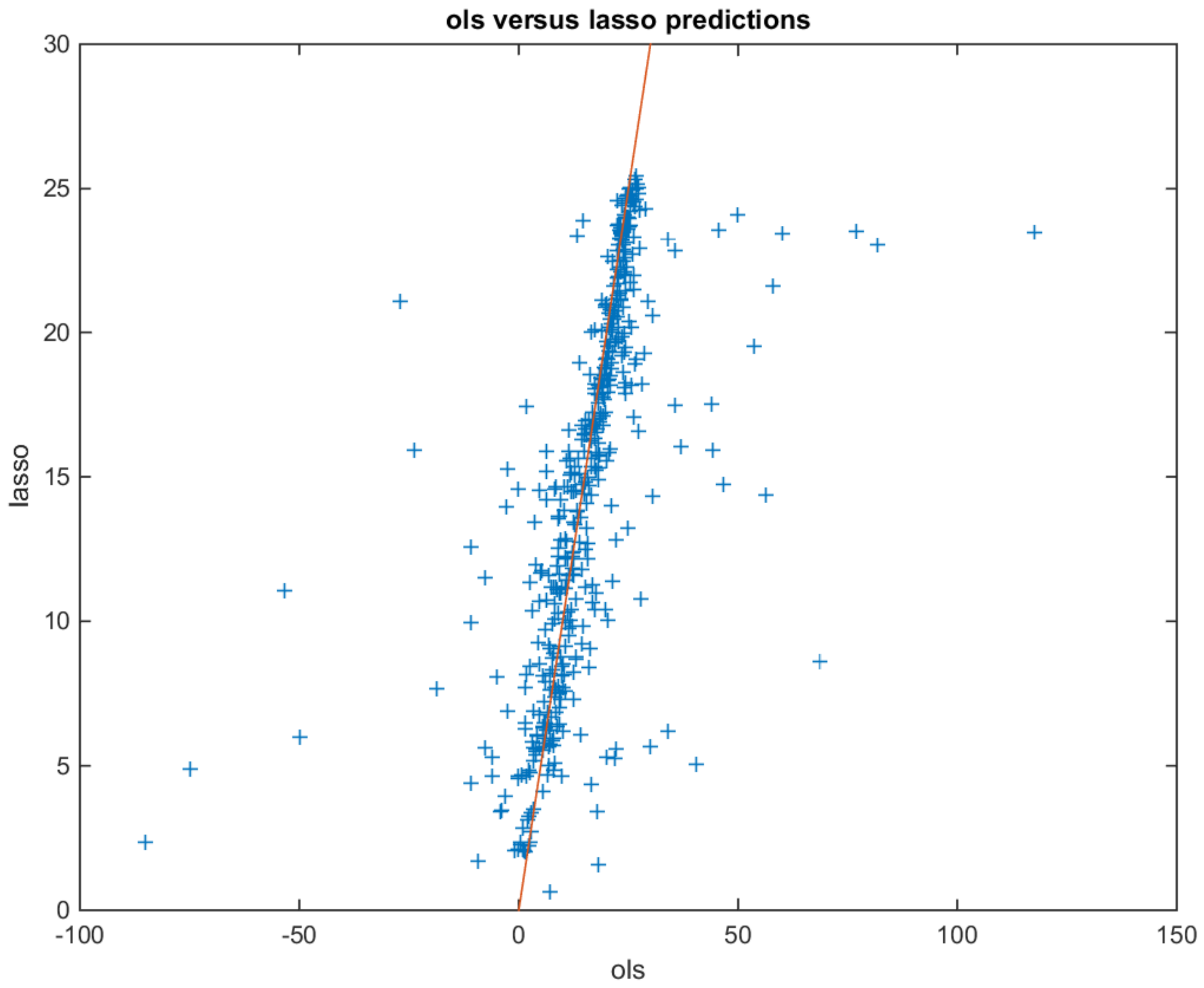
Example

Data on earnings in 1978 for 15,992 individuals. 8 features, indicators for African-America, Hispanic, marital status, no-degree, and continous variables age, education, earnings in 1974 and 1975. Created additional features by including all interactions up to third order, leading to 121 features.

Training sample 7,996 individuals.

LASSO selects 15 out of 121 features, including 3 out of 8 main effects (education, earnings in 1974, earnings in 1975).

Test sample root-mean-squared error
OLS: 1.4093
LASSO: 0.7379


ols versus lasso predictions
lasso
ols
0
5
10
15
20
25
30
-100
-50
0
50
100
150

Elastic Nets

Combine L_2 and L_1 shrinkage:

$$\min_{\beta} \sum_{i=1}^{N} (Y_i - X_i\beta)^2 + \lambda \cdot \left(\alpha \cdot \|\beta\|_1 + (1-\alpha) \cdot \|\beta\|_2^2\right)$$

Now we need to find two tuning parameters, the total amount of shrinkage, λ, and the share of L_1 and L_2 shrinkage, captured by α. Typically α is confined to a grid with few (e.g., 5) values.

2.b Nonparametric Regression: Regression Trees (Breiman, Friedman, Olshen, & Stone, 1984)

- The idea is to partition the covariate space into subspaces where the regression function is estimated as the average outcome for units with covariate values in that subspace.

- The partitioning is sequential, one covariate at a time, always to reduce the sum of squared deviations from the estimated regression function as much as possible.

- Similar to adaptive nearest neighbor estimation.

Start with estimate $g(x) = \overline{Y}$. The sum of squared deviations is

$$Q(g) = \sum_{i=1}^{N} (Y_i - g(X_i))^2 = \sum_{i=1}^{N} \left(Y_i - \overline{Y}\right)^2$$

For covariate k, for threshold t, consider splitting the data depending on whether

$$X_{i,k} \leq t \quad \text{versus} \quad X_{i,k} > t$$

Let the two averages be

$$\overline{Y}_{\text{left}} = \frac{\sum_{i:X_{i,k} \leq t} Y_i}{\sum_{i:X_{i,k} \leq t} 1} \qquad \overline{Y}_{\text{right}} = \frac{\sum_{i:X_{i,k} > t} Y_i}{\sum_{i:X_{i,k} > t} 1}$$

Define for covariate k and threshold t the estimator

$$g_{k,t}(x) = \begin{cases} \overline{Y}_{\text{left}} & \text{if } x_k \leq t \\ \overline{Y}_{\text{right}} & \text{if } x_k > t \end{cases}$$

Find the covariate k^* and the threshold t^* that solve

$$(k^*, t^*) = \arg\min_{k,t} Q(g_{k,t}(\cdot))$$

Partition the covariate space into two subspaces, by whether $X_{i,k^*} \leq t^*$ or not.

Repeat this, splitting the subspace that leads to the biggest improvement in the objective function.

Keep splitting the subspaces to minimize the objective function, with a penalty for the number of splits (leaves):

$$Q(g) + \lambda \cdot \#(\text{leaves})$$

• Result is flexible step function with properties that are difficult to establish. No confidence intervals available.

Selecting the Penalty Term

To implement this we need to choose the penalty term λ for the number of leaves.

We do essentially the same thing as in the LASSO case. Divide the sample into B crossvalidation samples. Each time grow the tree using the full sample excluding the b-th cross validation sample, for all possible values for λ, call this $g(b, \lambda)$. For each λ sum up the squared errors over the crossvalidation sample to get

$$Q(\lambda) = \sum_{b=1}^{B} \sum_{i:I_i=b} (Y_i - g(b, \lambda))^2$$

Choose the λ that minimizes this criterion and estimate the tree given this value for the penalty parameter. (Computational tricks lead to focus on discrete set of λ.)

Pruning The Tree

Growing a tree this way may stop too early: splitting a particular leaf may not need to an improvement in the sum of squared errors, but if we split anyway, we may find subsequently profitable splits.

For example, suppose that there are two binary covariates, $X_{i1}, X_{i,2} \in \{0, 1\}$, and that

$$\mathbb{E}[Y_i|X_{i,1} = 0, X_{i,2} = 0] = \mathbb{E}[Y_i|X_{i,1} = 1, X_{i,2} = 1] = -1$$

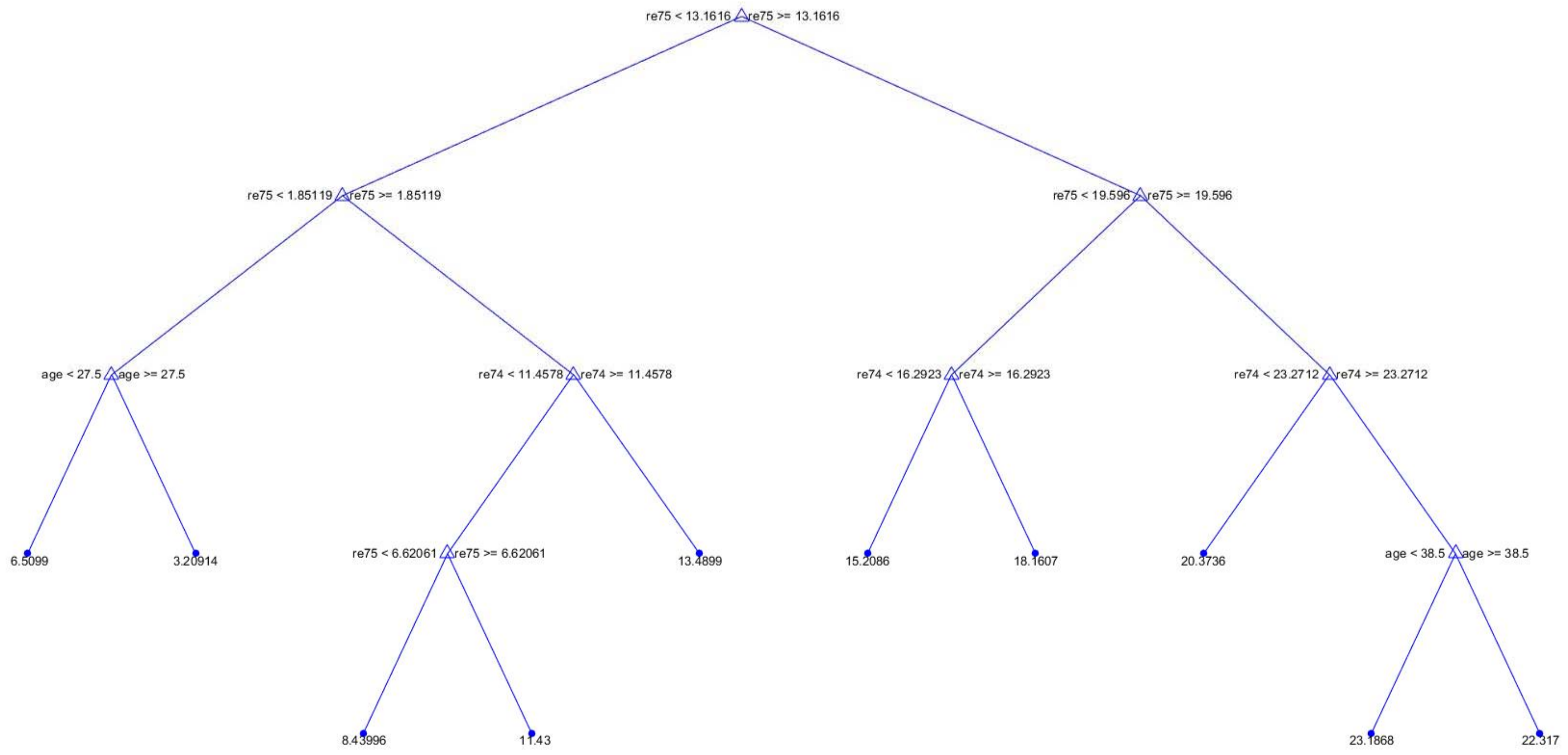
$$\mathbb{E}[Y_i|X_{i,1} = 0, X_{i,2} = 1] = \mathbb{E}[Y_i|X_{i,1} = 0, X_{i,2} = 1] = 1$$

Then splitting on X_{i1} or X_{i2} does not improve the objective function, but once one splits on either of them, the subsequent splits lead to an improvement.

This motivates **pruning** the tree:

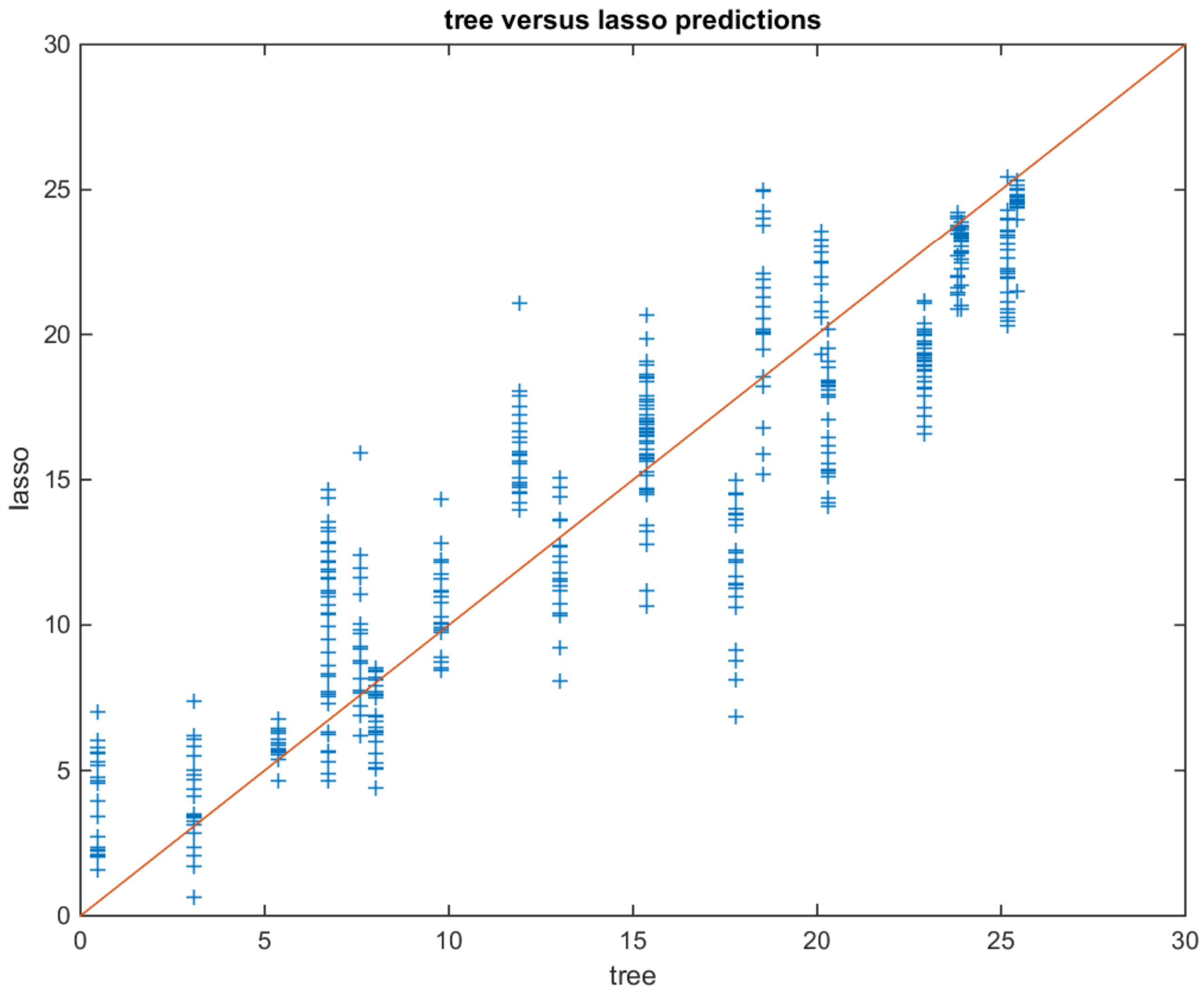
• First grow a big tree by using a deliberately small value of the penalty term, or simply growing the tree till the leaves have a preset small number of observations.

• Then go back and prune branches or leaves that do not collectively improve the objective function sufficiently.


re75 < 13.1616
re75 >= 13.1616
re75 < 1.85119
re75 >= 1.85119
re75 < 19.596
re75 >= 19.596
age < 27.5
age >= 27.5
re74 < 11.4578
re74 >= 11.4578
re74 < 16.2923
re74 >= 16.2923
re74 < 23.2712
re74 >= 23.2712
6.5099
3.20914
re75 < 6.62061
re75 >= 6.62061
13.4899
15.2086
18.1607
20.3736
age < 38.5
age >= 38.5
8.43996
11.43
23.1868
22.317

Tree leads to 37 splits

Test sample root-mean-squared error
OLS: 1.4093
LASSO: 0.7379
Tree: 0.7865


tree versus lasso predictions
lasso
tree
0
5
10
15
20
25
30

2.c Boosting

Suppose we have a simple, possibly naive, but easy to compute, way of estimating a regression function, a so-called **weak learner**.

Boosting is a general approach to repeatedly use the weak learner to get a good predictor for both classification and regression problems. It can be used with many different weak learners, trees, kernels, support vector machines, neural networks, etc.

Here I illustrate it using regression trees.

Suppose $g(x|\mathbf{X},\mathbf{Y})$ is based on a very simple regression tree, using only a single split. So, the algorithm selects a covariate $k(\mathbf{X},\mathbf{Y})$ and a threshold $t(\mathbf{X},\mathbf{Y})$ and then estimates the regression function as

$$g_1(x|\mathbf{X},\mathbf{Y}) = \begin{cases} \overline{Y}_{\text{left}} & \text{if } x_{k(\mathbf{X},\mathbf{Y})} \le t(\mathbf{X},\mathbf{Y}) \\ \overline{Y}_{\text{right}} & \text{if } x_{k(\mathbf{X},\mathbf{Y})} > t(\mathbf{X},\mathbf{Y}) \end{cases}$$

where

$$\overline{Y}_{\text{left}} = \frac{\sum_{i:X_{i,k}\le t} Y_i}{\sum_{i:X_{i,k}\le t} 1} \qquad \overline{Y}_{\text{right}} = \frac{\sum_{i:X_{i,k}> t} Y_i}{\sum_{i:X_{i,k}> t} 1}$$

Not a very good predictor by itself.

Define the residual relative to this weak learner:

$$\varepsilon_{1i} = Y_i - g_1(X_i|\mathbf{X}, \mathbf{Y})$$

Now apply the **same** weak learner to the **new** data set $(\mathbf{X}, \varepsilon_1)$.

Grow a second tree $g_2(\mathbf{X}, \varepsilon_1)$ based on this data set (with single split), and define the new residuals as

$$\varepsilon_{2i} = Y_i - g_1(X_i|\mathbf{X}, \mathbf{Y}) - g_2(X_i|\mathbf{X}, \varepsilon_1)$$

Re-apply the weak learner to the data set $(\mathbf{X}, \varepsilon_2)$.

After doing this many times you get an additive approximation to the regression function:

$$\sum_{m=1}^{M} g_m(x|\mathbf{X}, \varepsilon_{m-1}) = \sum_{k=1}^{K} h_k(x_k) \quad \text{where} \quad \varepsilon_0 = \mathbf{Y}$$

Had we used a weak learner with two splits, we would have allowed for second order effects $h(x_k, x_l)$.

In practice researchers use **shallow** trees, say with six splits (implicitly allowing for 6-th order interactions), and grow many trees, e.g., 400-500.

Often the depth of the initial trees is fixed in advance in an ad hoc manner (difficult to choose too many tuning parameters optimally), and the number of trees is based on prediction errors in a test sample (similar in spirit to cross-validation).

2.d Bagging (Bootstrap AGGregatING) Applicable to many ways of estimating regression function, here applied to trees.

1. Draw a bootstrap sample of size N from the data.

2. Construct a tree $g_b(x)$, possibly with pruning, possibly with data-dependent penalty term.

Estimate the regression function by averaging over bootstrap estimates:

$$\frac{1}{B}\sum_{b=1}^{B} g_b(x)$$

If the basic learner were linear, than the bagging is ineffective. For nonlinear learners, however, this smoothes things and can lead to improvements.

2.e Random Forests (Great general purpose method)

Given data $(\mathbf{X}, \mathbf{Y})$, with the dimension of $\mathbf{X}$ equal to $N \times K$, do the same as bagging, but with a different way of constructing the tree given the bootstrap sample: Start with a tree with a single leaf.

1. Randomly select L regressors out of the set of K regressors
2. Select the optimal cov and threshold among L regressors
3. If some leaves have more than N_{min} units, go back to (1)
4. Otherwise, stop

Average trees over bootstrap samples.

• For bagging and random forest there is recent research suggesting asymptotic normality may hold. It would require using small training samples relative to the overall sample (on the order of $N/(\ln(N)^K)$, where K is the number of features.

See Wager, Efron, and Hastie (2014) and Wager (2015).

2.f Neural Networks / Deep Learning

Goes back to 1990's, work by Hal White. Recent resurgence.

Model the relation between X_i and Y_i through hidden layer(s) of Z_i, with M elements $Z_{i,m}$:

$$Z_{i,m} = \sigma(\alpha_{0m} + \alpha'_{1m} X_i), \qquad \text{for } m = 1, \ldots, M$$

$$Y_i = \beta_0 + \beta'_1 Z_i + \varepsilon_i$$

So, the Y_i are linear in a number of transformations of the original covariates X_i. Often the transformations are sigmoid functions $\sigma(a) = (1 + \exp(-a))^{-1}$. We fit the parameters α_m and β by minimizing

$$\sum_{i=1}^{N} (Y_i - g(X_i, \alpha, \beta))^2$$

• Estimation can be hard. Start with α random but close to zero, so close to linear model, using gradient descent methods.

• Difficulty is to avoid overfitting. We can add a penalty term

$$\sum_{i=1}^{N} (Y_i - g(X_i, \alpha, \beta))^2 + \lambda \cdot \left(\sum_{k,m} \alpha_{k,m}^2 + \sum_k \beta_k^2 \right)$$

Find optimal penalty term λ by monitoring sum of squared prediction errors on test sample.

2.g Ensemble Methods, Model Averaging, and Super Learners

Suppose we have M candidate estimators $g_m(\cdot|\mathbf{X}, \mathbf{Y})$. They can be similar, e.g., all trees, or they can be qualitatively different, some trees, some regression models, some support vector machines, some neural networks. We can try to combine them to get a better estimator, often better than any single algorithm.

Note that we do not attempt to select a single method, rather we look for weights that may be non-zero for multiple methods.

Most competitions for supervised learning methods have been won by algorithms that combine more basic methods, often many different methods.

One question is how exactly to combine methods.

One approach is to construct weights $\alpha_1, \ldots, \alpha_M$ by solving, using a test sample

$$\min_{\alpha_1,\ldots,\alpha_M} \sum_{i=1}^{N} \left(Y_i - \sum_{m=1}^{M} \alpha_m \cdot g_m(X_i) \right)^2$$

If we have many algorithms to choose from we may wish to regularize this problem by adding a LASSO-type penalty term:

$$\lambda \cdot \sum_{m=1}^{M} |\alpha_m|$$

That is, we restrict the ensemble estimator to be a weighted average of the original estimators where we shrink the weights using an L_1 norm. The result will be a weighted average that puts non-zero weights on only a few models.

Test sample root-mean-squared error
OLS: 1.4093
LASSO: 0.7379
Tree: 0.7865
Ensemble: 0.7375

Weights ensemble:
OLS 0.0130
LASSO 0.7249
Tree 0.2621

3. Unsupervised Learning: Clustering

We have N observations on a M-component vector of features, X_i, $i = 1, \ldots, N$. We want to find patterns in these data.

Note: there is no outcome Y_i here, which gave rise to the term "unsupervised learning."

• One approach is to reduce the dimension of the X_i using principal components. We can then fit models using those principal components rather than the full set of features.

• Second approach is to partition the space into a finite set. We can then fit models to the subpopulations in each of those sets.

3.a Principal Components

• Old method, e.g., in Theil (Principles of Econometrics)

We want to find a set of K N-vectors $Y_1, \ldots, Y_K$ so that

$$X_i \approx \sum_{k=1}^{K} \gamma_{ik} Y_k$$

or, collectively, we want to find approximation

$$\mathbf{X} = \Gamma \mathbf{Y}, \qquad \Gamma \text{ is } M \times K, \; M > K$$

This is useful in cases where we have many features and we want to reduce the dimension without giving up a lot of information.

First normalize components of X_i, so average $\sum_{i=1}^{N} X_i/N = 0$, and components have unit variance.

First principal component: Find N-vector $\mathbf{Y}_1$ and the M vector Γ that solve

$$\mathbf{Y}_1, \Gamma = \arg\min_{\mathbf{Y}_1, \Gamma} \text{trace}\left((\mathbf{X} - \mathbf{Y}_1 \Gamma)'(\mathbf{X} - \mathbf{Y}_1 \Gamma) \right)$$

This leads to $\mathbf{Y}_1$ being the eigenvector of the $N \times N$ matrix $\mathbf{X}\mathbf{X}'$ corresponding to the largest eigenvalue. Given $\mathbf{Y}$, Γ is easy to find.

Subsequent $\mathbf{Y}_k$ correspond to the subsequent eigenvectors.

3.b Mixture Models and the EM Algorithm

Model the joint distribution of the L-component vector X_i as a mixture of parametric distributions:

$$f(x) = \sum_{k=1}^{K} \pi_k \cdot f_k(x;\theta_k) \qquad f_k(\cdot;\cdot) \text{ known}$$

We want to estimate the parameters of the mixture components, θ_k, and the mixture probabilities π_k.

The mixture components can be multivariate normal, of any other parametric distribution. Straight maximum likelihood estimation is very difficult because the likelihood function is multi-modal.

The EM algorithm (Dempster, Laird, Rubin, 1977) makes this easy as long as it is easy to estimate the θ_k given data from the k-th mixture component. Start with $\pi_k = 1/K$ for all k.

Create starting values for θ_k, all different.

Update weights, the conditional probability of belonging to cluster k given parameter values (E-step in EM):

$$w_{ik} = \frac{\pi_k \cdot f_k(X_i;\theta_k)}{\sum_{m=1}^{K} \pi_m \cdot f_m(X_i;\theta_m)}$$

Update θ_k (M-step in EM)

$$\theta_k = \arg\max_{\theta} \sum_{i=1}^{N} w_{ik} \cdot \ln f_k(X_i;\theta)$$

Algorithm can be slow but is very reliable. It gives probabilities for each cluster. Then we can use those to assign new units to clusters, using the highest probability.

Used in duration models in Gamma-Weibull mixtures (Lancaster, 1979), non-parametric Weibull mixtures (Heckman & Singer, 1984).

3.c The k-means Algorithm

1. Start with k arbitrary centroids c_m for the k clusters.

2. Assign each observation to the nearest centroid:

$$I_i = m \quad \text{if} \quad \|X_i - c_m\| = \min_{m'=1} \|X_i - c_{m'}\|$$

3. Re-calculate the centroids as

$$c_m = \sum_{i:I_i=m} X_i \Bigg/ \sum_{i:I_i=m} 1$$

4. Go back to (2) if centroids have changed, otherwise stop.

- k-means is fast

- Results can be sensitive to starting values for centroids.

3.d. Mining for Association Rules: The Apriori Algorithm

Suppose we have a set of N customers, each buying arbitrary subsets of a set $\mathcal{F}_0$ containing M items. We want to find subsets of k items that are bought together by at least L customers, for different values of k.

This is very much a data mining exercise. There is no model, simply a search for items that go together. Of course this may suggest causal relationships, and suggest that discounting some items may increase sales of other items.

It is potentially difficult, because there are M choose k subsets of k items that could be elements of $\mathcal{F}_k$. The solution is to do this sequentially.

You start with $k = 1$, by selecting all items that are bought by at least L customers. This gives a set $\mathcal{F}_1 \subset \mathcal{F}_0$ of the M original items.

Now, for $k \geq 2$, given $\mathcal{F}_{k-1}$, find $\mathcal{F}_k$.

First construct the set of possible elements of $\mathcal{F}_k$. For a set of items F to be in $\mathcal{F}_k$, it must be that any set obtained by dropping one of the k items in F, say the m-th item, leading to the set $F_{(m)} = F/\{m\}$, must be an element of $\mathcal{F}_{k-1}$.

The reason is that for F to be in $\mathcal{F}_k$, it must be that there are at least L customers buying that set of items. Hence there must be at least L customers buying the set $F_{(m)}$, and so $F_{(m)}$ is an $k-1$ item set that must be an element of $\mathcal{F}_{k-1}$.

5. Support Vector Machines and Classification

Suppose we have a sample (Y_i, X_i), $i = 1, \ldots, N$, with $Y_i \in \{-1, 1\}$.

We are trying to come up with a classification rule that assigns units to one of the two classes -1 or 1.

One conventional econometric approach is to estimate a logistic regression model and assign units to the groups based on the estimated probability.

Support vector machines look for a boundary $h(x)$, such that units on one side of the boundary are assigned to one group, and units on the other side are assigned to the other group.

Suppose we limit ourselves to linear rules,

$$h(x) = \beta_0 + x\beta_1,$$

where we assign a unit with features x to class 1 if $h(x) \geq 0$ and to -1 otherwise.

We want to choose β_0 and β_1 to optimize the classification. The question is how to quantify the quality of the classification.

Suppose there is a hyperplane that completely separates the $Y_i = -1$ and $Y_i = 1$ groups. In that case we can look for the β, such that $\|\beta\| = 1$, that maximize the <u>margin</u> M:

$$\max_{\beta_0,\beta_1} M \quad \text{s.t.} \;\; Y_i \cdot (\beta_0 + X_i\beta_1) \geq M \;\forall i, \quad \|\beta\| = 1$$

The restriction implies that each point is at least a distance M away from the boundary.

We can rewrite this as

$$\min_{\beta_0,\beta_1} \|\beta\| \quad \text{s.t.} \;\; Y_i \cdot (\beta_0 + X_i\beta_1) \geq 1 \;\forall i.$$

Often there is no such hyperplane. In that case we define a penalty we pay for observations that are not at least a distance M away from the boundary.

$$\min_{\beta_0,\beta_1} \|\beta\| \quad \text{s.t.} \;\; Y_i \cdot (\beta_0 + X_i\beta_1) \geq 1 - \varepsilon_i, \;\; \varepsilon_i \geq 0 \;\; \forall i, \quad \sum_i \varepsilon \leq C.$$

Alternatively

$$\min_{\beta} \frac{1}{2} \cdot \|\beta\| + C \cdot \sum_{i=1} \varepsilon_i$$

subject to

$$\varepsilon_i \geq 0, \qquad Y_i \cdot (\beta_0 + X_i\beta_1) \geq 1 - \varepsilon_i$$

With the linear specification $h(x) = \beta_0 + x\beta_1$ the support vector machine leads to

$$\min_{\beta_0,\beta_1} \sum_{i=1}^{N} \lfloor 1 - Y_i \cdot (\beta_0 + X_i\beta_1) \rfloor_+ + \lambda \cdot \|\beta\|^2$$

where $\lfloor a \rfloor_+$ is a if $a > 0$ and zero otherwise, and $\lambda = 1/C$.

Note that we do not get probabilities here as in a logistic regression model, only assignments to one of the two groups.

Thank You!