ApEc 8213: Econometric Analysis III -- Lecture #14

Machine Learning, Part 2
Hansen, Chapter 29, Sections 29.15 – 29.22

I. Introduction

Lecture 13 introduced machine learning, and in particular ridge regression and Lasso regression. Here are some main points from that lecture:

- The focus is on ***predicting*** Y given p X variables, as opposed to estimating structural/causal relationships.

- There are thousands, and even millions of observations (n is the sample size).

- There are hundreds and maybe even thousands of X variables, and it is possible that $p > n$.

- Both ridge regression and Lasso regression "shrink" the estimated coefficients for the X variables in order to reduce variance of the coefficient estimates and to avoid problems when p is close to (or $\geq$) n.

- In general, ridge regression does not remove X variables from the regression by setting some of the coefficients to zero, but Lasso regression does do this.

This lecture will introduce regressions trees, bagging, random forests, and extensions to Lasso.

II. Regression Trees (29.15)

Regression tree estimation predicts *Y* using a (possibly large) number of *X* variables **by splitting the data into subsamples, based on** the values of the *X* **variables**, so that the subsamples have "similar" values of *Y*. **Each "split" divides the data into** two subsamples, called **branches**, and **each branch** is **split into two smaller branches**. The splitting occurs **until the** number of **observations in the branches** hits a pre-specified **minimum size**, and these **"final" branches** are called **leaves** (sometimes these are called "nodes").

The process in the above paragraph is **called growing the tree**. This tends to produce predictions for *Y* that have high standard errors because each leaf has a small number of observations. The **last step** is to **combine branches until a Mallows-type criterion is minimized** (this criterion is the sum of squares of the errors of the predictions for each *Y*, plus a "penalty" for the number of leaves). This last step is **called pruning**.

This is **very non-parametric**. No coefficients are used for the *X* variables to predict *Y*.

Algorithm for "Growing" the Tree

You start with **one *Y* variable** and ***k X* variables**, and a **sample size of *n***. Here are the steps for the "growing" algorithm:

1. **Select a minimum leaf/node size**, denoted by N_{min}, which can be a small number (e.g. 5). You are not allowed to split a "branch" into two smaller branches if one of them has $< N_{\text{min}}$ observations. This rule eventually stops the splitting (growing of the tree).

2. **For a given branch** (which is the whole data set at the start), **choose one of the *X* variables** – call it X_d – **and a parameter, γ, to minimize** the sum of squared errors in the **following equation**:

$$Y = \mu_1 1[X_d \leq \gamma] + \mu_2 1[X_d > \gamma] + e$$

   where μ_1 is the mean of Y for all observations with $X_d \leq \gamma$ and μ_2 is the mean of Y for all observations with $X_d > \gamma$. You need to **do a "grid search" over all possible X_d variables and all possible γ's for the X_d and γ that minimize the sum of e^2 in your "branch"**. The two sets of observations from this splitting are called sub-branches.

3. **For each sub-branch**, which can be denoted by *b*, **perform the same "split"** into **two more sub-branches**. You are very likely to choose a different *X* variable (different *d*). **Keep doing splits until** it is **not possible to do any more splits** for which both branches have $\geq N_{\min}$ observations.

Draw a picture of this with 2 *X* variables.

You could stop here, and for each final sub-branch (leaf, or node), your **prediction of *Y* for the values of *X* that describe that sub-branch** (or leaf, or node) is simply the **mean of the *Y'*s in that sub-branch**. **But** in each of these sub-branches, the number of observations will be quite small, which will lead to **large standard errors of your predicted *Y'*s**, so you need to "go backwards" by "pruning", which means that you combine sub-branches.

Algorithm for "Pruning" the Tree

1. **Specify** a **Mallows-type information criterion**:

$$C = \sum_{i=1}^{n} \hat{e}_i^{\,2} + \alpha N$$

   where *N* is the number of leaves and α is a penalty parameter.

2. Compute *C*.

3. For each leaf, recalculate *C* after "removing" that leaf. **The leaf whose "removal" generates the biggest drop in *C* is "pruned"** by recombining it with its "sibling" leaf to "re-assemble" the sub-branch from which the two leaves were generated. **For each calculation of *C*, you do not remove the leaf** that generates the biggest drop in *C*; instead, you keep it, but **you combine it** with its "sibling" and calculate *C*.

4. Recalculate *C* after pruning the leaf pair, and repeat Step 3 until there is no longer any leaf pair that, when pruned, reduces *C* (the drop in *C* by reducing *N* by 1 is outweighed by the increase in $\sum_{i=1}^{n} \hat{e}_i{}^2$).

How do you select *α*? According to Hansen, the penalty parameter α is usually selected by using a **K-fold cross-validation procedure** (see Lecture 13) using a "Mallows-type" criterion. That is for a given value of α, you **randomly split your sample into 5, 10 or 20 random sub-samples**. **For each subsample, use all the *other* sub-samples to predict *Y*** using a random forest. Use the predictions for all of the sub-samples to calculate the sum of the squares of the prediction errors. Do this for many values of α, and **choose the *α* that gives the smallest sum of the prediction errors**. Presumably you estimate the regression tree with different values of α, and then you calculate the sum of the squared prediction errors for each of those values of α, and you choose that α with the smallest sum of squared prediction errors.

Hansen says (p.958) that it is **“difficult” to calculate the sampling distribution of the predicted *Y’*s** from regression trees, in part because of “strong correlation between the placement of the sample splits [the α’s] and the estimated means” in the sub-branches. **One way to** greatly reduce this correlation, and to **calculate the standard errors of the predicted *Y*’s**, is to split, usually randomly, the sample into two samples of equal size and use one sample to generate the splits and the other sample to calculate (unbiased) means within each sub-branch. This is called the “honest tree” method. See Wager and Athey (2018) for details.

Regression trees can be estimated in R using “rpart”. For Stata, you can download “crtrees” (type “help crtrees” to get the link.

III. Bagging (29.16)

The **predicted values of *Y* conditional on** values of ***X* in a regression tree are discontinuous** step functions. **Smoother predicted values** of *Y*, which will also have a lower variance, can be **generated using “bagging”**, which stands for **b**ootstrap **agg**rega**ting**). The basic idea is to draw a large number *B* of bootstrap samples, and estimate a regression tree for each bootstrapped sample. Then **average the predicted values of *Y*** (conditional on *X*) from these regression trees **over all the bootstrap samples**. This will produce a much smoother estimate of *Y* conditional on *X*.

Bagging estimation is implemented as follows. Define $m(x) = \mathrm{E}[Y \mid X = x]$, which is the "true" conditional expectation function for *Y*. Let $\widehat{m}(x)$ be an estimate of $m(x)$, such as a regression tree estimate. Then, let $\widehat{m}_b{}^*(x)$ be an estimate of $m(x)$ from a bootstrap sample. The bagging estimate of $m(x)$ is:

$$\widehat{m}_{\text{bag}}(x) = (1/B)\sum_{b=1}^{B} \widehat{m}_b{}^*(x)$$

where *B* is the number of bootstrap samples. On pages 958-959, Hansen gives an example that he says will help understand how bagging works. This is optional.

The **main benefit of bagging** is that it **reduces variance in the prediction of *Y*** (relative to regression tree estimation). Another benefit is that it is convenient for doing cross-validation (CV) for a model. For a "typical" non-parametric bootstrap sample, about 37% of the original sample is not in the bootstrap sample. These "out-of-bag" observations can be used for out-of-sample predictions. See pp.959-960 of Hansen for further discussion, and for an estimator for the variance of $\widehat{m}_{\text{bag}}(x)$.

IV. Random Forests (29.17)

Random forests were introduced by Breiman in 2001. They are a modification of bagged regression trees. The **shortcoming of bagged regression trees** is that the **individual bootstrapped regression trees tend to produce** similar, and thus **positively correlated**, **estimates** because the bootstrap samples are all drawn from the same "total" sample. This positive correlation **leads to high variance of the overall** (averaged across bootstrapped samples) **bagged regression tree estimates**.

Random forests "decorrelate" the bootstrap regression trees by "introducing extra randomness". Hansen says that "Random forests … have effectively displaced simple regression trees." This "decorrelation" is done as follows.

Random Forest Algorithm

1. **Choose a minimum leaf size** $N_{\min}$ (typically 5), a "**minimal split fraction**" $\alpha \in [0, 1)$, and a "**sampling number** m that is $< p$ (typically $m = p/3$).

2. For $b = 1, 2, \ldots B$ (i.e. for each bootstrap sample):

a) Draw a nonparametric bootstrap sample.

b) "**Grow" a regression tree on that sample** as follows:

   i) **Select *m* variables at random** from the *p* regressors.

   ii) Among these *m* variables, **select the one that produces the best regression split**, where each split subsample (sub-branch) has at least N_{min} observations and at least a fraction α of the observations in the branch.

   iii) Split the bootstrap sample using the split in ii).

c) **Stop** when each leaf has between N_{min} and $2N_{\text{min}} - 1$ observations.

3. The **estimate of *Y* conditional on *X* = *x*** is:

$$\hat{m}_{\text{rf}}(x) = (1/B) \sum_{b=1}^{B} \hat{m}_b(x)$$

Choosing *m* < *p* variables reduces the correlation between the bootstrapped regression trees, and so **reduces the variance of the bootstrap average $\hat{m}_{\text{rf}}(x)$**.

The variance and standard errors of $\widehat{m}_{\mathrm{rf}}(x)$ can be calculated using the "infinitesimal jackknife", the details of which are in Wager, Hastie and Efron (2014).

Another method of calculating the variance and standard errors is discussed on p.961 of Hansen. Wager and Athey (2018) showed that, under very general conditions, $\widehat{m}_{\mathrm{rf}}(x)$ is a consistent estimate of $m(x)$, and is asymptotically normally distributed.

Random forest estimation can be done in R using the "randomForest" command. For Stata, there is a command "rforest" that can be downloaded.

V. Ensembling

In Section 29.18, Hansen briefly describes "ensembling", which is a method for averaging across different machine learning algorithms. However, he says that "the theoretical literature is thin. Much of the advice … is based on empirical performance."

Here is an example of how ensembling works, slightly modified from Athey and Imbens (2019, *Annual Review of Economics*). Suppose that you have three machine learning models for Y, predicted using various X variables: Ridge regression, Random forest, and Lasso.

Their predictions for Y can be denoted by $\hat{Y}^{\text{RR}}$, $\hat{Y}^{\text{RF}}$ and $\hat{Y}^{\text{LASSO}}$, respectively. The goal is to take a weighted average of these three predictions, choosing the weights (denoted by p) to minimize the sum of squares of the prediction errors. This can be expressed as:

$$(\hat{p}^{\text{RR}}, \hat{p}^{\text{RF}}, \hat{p}^{\text{LASSO}}) = \underset{\hat{p}^{\text{RR}},\ \hat{p}^{\text{RF}},\ \hat{p}^{\text{LASSO}}}{\arg\min} \sum_{i=1}^{N^{test}} (Y_i - p^{\text{RR}}\hat{Y}^{\text{RR}} - p^{\text{RR}}\hat{Y}^{\text{RR}} - p^{\text{LASSO}}\hat{Y}^{\text{LASSO}})^2$$

where N^{test} is a set of observations "held back" that were not used to estimate any of the three models. Usually, one imposes $\hat{p}^{\text{RR}} + \hat{p}^{\text{RF}} + \hat{p}^{\text{LASSO}} = 1$ and $\hat{p}^{\text{RR}} \geq 0$, $\hat{p}^{\text{RF}} \geq 0$ and $\hat{p}^{\text{LASSO}} \geq 0$.

VI. Lasso IV (29.19)

A somewhat more structural **use** of **machine learning** is to use it **to estimate the first stage of an IV/2SLS regression**. The linear model is:

$$Y = X'\beta + e \quad \text{E}[e|\, X] \neq 0$$

$$X = \Gamma' Z + U \quad \text{E}[e|\, Z] = 0,\ \ \text{E}[U\,|\, Z] = 0$$

where β is a $k\times1$ vector (and k is fixed), and Γ is $p\times n$, with large p.

If $p \geq n$, then the 2SLS estimator equals the OLS estimator [**why?**]. If $p < n$ but p is very large, 2SLS will likely have a "many weak instruments" problem. So instead of using OLS for the first stage, it may be best to use Lasso or post-Lasso for the first stage (the paper by Belloni, Chen, Chernozhokov and C. Hansen, 2012, focuses on Lasso).

This is done as follows. Use Lasso to estimate $X_j = \gamma_j Z + U_j$ separately for each variable in X. The $\boldsymbol{\hat{\gamma}_j}$**'s from** each of these **Lasso regressions are "stacked"** into the matrix $\hat{\Gamma}_{\text{Lasso}}$ and are used to **generate the predicted values of** $\boldsymbol{X}$, which are called $\boldsymbol{\hat{X}}_{\text{Lasso}} = \boldsymbol{Z}\hat{\Gamma}_{\text{Lasso}}$. The Lasso IV estimator is:

$$\hat{\beta}_{\text{Lasso-IV}} = (\boldsymbol{\hat{X}}_{\text{Lasso}}'\boldsymbol{X})^{-1}(\boldsymbol{\hat{X}}_{\text{Lasso}}'\boldsymbol{Y})$$

An alternative method to estimate β using Lasso is to **split the sample randomly in half**, and use one half ("A") to estimate $\hat{\Gamma}_{\text{Lasso}}$ and the other half ("B") to estimate β.

To start, use sample A to estimate Γ, and call this $\hat{\Gamma}_{\text{Lasso,A}}$. Then use this estimate, with the data in sample B, to predict the X variables: $\boldsymbol{\hat{X}}_{\text{Lasso,B}} = \boldsymbol{Z}_{\text{B}}\hat{\Gamma}_{\text{Lasso,A}}$. The estimate of β is then:

$$\hat{\beta}_{\text{Lasso,B}} = (\boldsymbol{\hat{X}}_{\text{Lasso,B}}'\boldsymbol{X}_{\text{B}})^{-1}(\boldsymbol{\hat{X}}_{\text{Lasso,B}}'\boldsymbol{Y}_{\text{B}})$$

Next, start with sample B to estimate Γ, call it $\widehat{\Gamma}_{\text{Lasso,B}}$, and then generate an estimate of X using the Z variables in sample A to generate $\widehat{\boldsymbol{X}}_{\text{Lasso,A}} = \boldsymbol{Z}_{\text{A}}\widehat{\Gamma}_{\text{Lasso,B}}$. Finally, combine these two estimate of $\widehat{\boldsymbol{X}}$ to get an averaged estimate of β:

$$\hat{\beta}_{\text{Lasso-SSIV}} = (\widehat{\boldsymbol{X}}_{\text{Lasso,B}}'\boldsymbol{X}_{\text{B}} + \widehat{\boldsymbol{X}}_{\text{Lasso,A}}'\boldsymbol{X}_{\text{A}})^{-1}(\widehat{\boldsymbol{X}}_{\text{Lasso,B}}'\boldsymbol{Y}_{\text{B}} + \widehat{\boldsymbol{X}}_{\text{Lasso,A}}'\boldsymbol{Y}_{\text{A}})$$

where SSIV indicate split sample IV.

Theorem 29.4 gives the results for the asymptotic distribution for $\widehat{\boldsymbol{\beta}}_{\textbf{Lasso-IV}}$ and $\widehat{\boldsymbol{\beta}}_{\textbf{Lasso-SSIV}}$. Note that there is a typo in Hansen's book. In equation (29.20) replace $(\boldsymbol{Q}^{-1}\Omega\, \boldsymbol{Q}^{-1})$ with $(\boldsymbol{Q}^{-1}\Omega\, \boldsymbol{Q}^{-1})^{-1/2}$.

For $\hat{\beta}_{\text{Lasso-IV}}$, one assumption is $||\Gamma||_0\frac{\log(p)}{\sqrt{n}} \to 0$, while for $\hat{\beta}_{\text{Lasso-SSIV}}$ the corresponding assumption is $||\Gamma||_0\frac{\log(p)}{\sqrt{n}} \to 0$. A practical implication of this is that $\hat{\beta}_{\text{Lasso-SSIV}}$ allows for more instruments (p = number of Z variables).

Hansen also says (p.963) that $\hat{\beta}_{\text{Lasso-SSIV}}$ has an "important disadvantage" that two researchers will get different estimates of $\hat{\beta}_{\text{Lasso-SSIV}}$ because their random split of the data into subsamples will be different, even if they start with the same dataset. I doubt that such differences will be very large, and we know why they occur, so I would not worry about this.

IV Lasso can be implemented in Stata using the downloaded "ivlasso" package.

VII. Double Selection Lasso (29.20)

Recall from the previous lecture that **post-Lasso estimation**, where Lasso is used to narrow down the variables in the regression and then OLS is applied to those variables, **has some "problems"**. Belloni, Chernozhulov and Hansen (2014) proposed, **double selection Lasso, which has better statistical properties**.

Consider the following regression model:

$$Y = D\theta + X'\beta + e, \quad \mathrm{E}[e|\, D, X] = 0 \qquad (29.22)$$

where Y and D are single variables, and X has p variables. The parameter of interest is θ. For example, D may be the impact of participating in some type of program on Y, and θ could be the causal effect of participating in that program.

You want to use Lasso to reduce the number of X variables, and then estimate the above equation by OLS.

This can be done by **double-selection**, which is implemented as follows. Consider predicting D using the X variables:

$$D = X'\gamma + V, \quad \text{where } \mathrm{E}[V \,|\, X] = 0 \qquad (29.23)$$

Insert (29.23) into (29.22) to get the reduced form for Y:

$$Y = X'\eta + U, \quad \text{where } \mathrm{E}[U \,|\, X] = 0 \qquad (29.24)$$

where $\eta = \beta + \gamma\theta$ and $U = V\theta$. The **double-selection Lasso algorithm applies Lasso separately to equations (29.3) and (29.4)** and takes the ***union*** of the X variables that are selected from these two applications of Lasso. Then use OLS to estimate equation (29.22) using this "union" of X variables. (Of course, D is also in this regression.)

Belloni et al. show that under a "sparsity" assumption (which concerns how many variables have coefficients equal to zero, see Section 29.12), then this **OLS estimate of equation (29.22)**, which can be denoted as $\hat{\theta}_{\mathrm{DS}}$, **is asymptotically normally distributed** (DS = Double Selection).

In Stata, double-selection Lasso can be implemented using either the "dsregress" command or the downloadable "pdslasso" package. For R, use the "hdm" package.

VIII. Post-Regularization Lasso

A "**potential**" (Hansen's terminology) **improvement on double-selection Lasso** is "post-regularization Lasso", which was proposed by Chernozhukov, C. Hansen and Spindler (2015). It is sometimes called "partialing-out Lasso". The idea here is similar to the Robinson (1988) method of estimating partially linear models (see Lecture 7). Start with equation (29.22) on page 14:

$$Y = D\theta + X'\beta + e, \quad \mathrm{E}[e|\, D, X] = 0$$

Take the expectations of both sides with respect to *X*:

$$\mathrm{E}[Y \mid X] = \theta\,\mathrm{E}[D \mid X] + X'\beta$$

Subtract the second equation from the first one:

$$Y - \mathrm{E}[Y \mid X] = (D - \mathrm{E}[D \mid X])\theta + e$$

Notice that $X'\beta$ is no longer in the equation.

Assume that $\mathrm{E}[Y \mid X]$ and $\mathrm{E}[D \mid X]$ are linear functions of *X*, so that $\mathrm{E}[Y \mid X] = X'\eta$ and $\mathrm{E}[D \mid X] = X'\gamma$. This gives:

$$Y - X'\eta = (D - X'\gamma])\theta + e \quad (29.25)$$

If we knew η and γ, we could estimate this by OLS. We do not know them, so we must estimate them. Chernozhukov, C. Hansen and Spindler (2015) recommend using Lasso or post-Lasso to (separately) estimate η and γ. More specifically, the recommend the following:

1. Estimate γ by estimating $D = X'\gamma + V$ using Lasso or post-Lasso (denote the penalty parameter by λ_1). Denote the estimated γ by $\hat{\gamma}$. Calculate $\hat{V} = D - X'\hat{\gamma}$.

2. Estimate η by estimating $Y = X'\eta + U$ using Lasso or post-Lasso (denote the penalty parameter by λ_2). Denote the estimated η by $\hat{\eta}$. Calculate $\hat{U} = Y - X'\hat{\eta}$.

3. Regress $\hat{U}$ on $\hat{V}$. Denote the coefficient on $\hat{V}$ by $\hat{\theta}_{\text{PR}}$. **This is our estimate for θ in equation (29.22).**

4. Calculate the "conventional" (heteroscedastic) standard error for $\hat{\theta}_{\text{PR}}$.

So why is this estimate of θ (potentially) better than the estimate using double-selection Lasso (based on (29.22))? The reason is that, if D and some of the X variables are correlated, the moment condition for θ from (29.22) is:

$$m(\theta, \beta) = \mathrm{E}[D(Y - D\theta - X'\beta)] = 0$$

This moment condition is “sensitive” to β in the sense that:

$$\partial m(\theta, \beta)/\partial\beta = - \mathrm{E}[DX']$$

which is nonzero if D and X are correlated.

In contrast, the moment condition for θ in equation (29.25) is:

$$m_{\mathrm{PR}}(\theta, \beta) = \mathrm{E}[(D - X'\gamma)(Y - X'\eta - (D - X'\gamma)\theta)] = 0$$

$$= \mathrm{E}[(D - X'\gamma)(Y - D\theta - X'\beta)] = 0 \quad \text{(see top of p.15)}$$

The “sensitivity” of $m_{\mathrm{PR}}(\theta, \beta)$ with respect to β is 0:

$$\partial m_{\mathrm{PR}}(\theta, \beta)/\partial\beta = \mathrm{E}[(D - X'\gamma)X'] = \mathrm{E}[VX'] = 0$$

Hansen summarizes this result in Theorem 29.5. **The advantage of the post-regularization estimator, $\widehat{\boldsymbol{\theta}}_{\mathbf{PR}}$, over the double-selection estimator, $\widehat{\boldsymbol{\theta}}_{\mathbf{DS}}$, is statistical efficiency**. By allowing for separate sets of X variables to be used to “demean” Y and D, **post-regularization does not include *X* variables that may be irrelevant for predicting *Y* and *D***, which reduces the number of parameters being estimated. In contrast, double-selection Lasso uses the same set of X variables to predict both Y and D.

In Stata, post-regularization Lasso (also known as "partialing-out Lasso") can be implemented using either the "poregress" command or the downloadable "pdslasso" package. For R, use the "hdm" package.

Finally, Section 29.22 in Hansen presents "double/debiased machine learning". It combines post-regularization methods with sample splitting. The relevant paper was published in 2018 by Chernozhukov and 6 co-authors. I ran out of time to present this. See pp.967-968 of Hansen for details.

Hansen also points out that new machine learning methods are being developed almost every year, so do not be surprised if some new methods come out that are "better" than these in some sense.

What you do NOT need to study for the final exam:

1. Lecture 1: Details of Examples of likelihood functions.
2. Lecture 1: Details on Cramér-Rao lower bound.
3. Lecture 8 (time series Part 1):

   a) Different types of growth rates on pages 4-5.
   b) Details of ergodicity and Ergodic Theorem.
4. Lecture 9 (time series Part 2)

   a) Details of "mixing" and associated theorems.
   b) Use of lag operators.
5. Lecture 10 (time series part 3): AR(p), ARMA and ARIMA processes (but you should study MA(1), MA(q) and MA(∞), and ***especially*** AR(1) processes).
6. Lecture 11 (time series part 4). Section VII (Exogeneity and Causality).
7. Lecture 6 (non-parametric density estimation): Details of derivations on page 15).
8. Lecture 13 (Machine learning, Part 1): The 5 properties of p-norms.
9. Finally, focus what are in the lecture notes. For anything in the notes that says "See Hansen for details" or some-thing like "See Section X.X in Hansen for Topic Y", you do NOT have to look at those things for the final exam.