A Very Brief Introduction to Machine Learning for Regression

A. Colin Cameron
Univ. of California- Davis

Abstract: These slides attempt to demystify machine learning.
The slides cover standard machine learning methods such as k-fold cross-validation, lasso, regression trees and random forests.
The slides conclude with some recent econometrics research that incorporates machine learning methods in causal models estimated using observational data.

Presented at the third Statistical Methodology in the Social Sciences Conference
University of California - Davis, 2017.
More at http://cameron.econ.ucdavis.edu/e240f/machinelearning.html.

October 27, 2017

Introduction

- The goal is **prediction**.
- **Machine learning** means that no stuctural model is given.
  - Instead the **machine** is given an **algorithm** and **existing data**.
  - These **train** the machine to come up with a prediction model.
  - This **model** is then used to make predictions given new data.
- Various methods guard against **overfitting** the existing data.
- There are many, **many algorithms**
  - a given algorithm may work well for one type of data and poorly for other types.
- Forming data to input can be an art in itself (data carpentry)
  - e.g. what **features** to use for facial recognition.
- What could go wrong?
  - correlation does not imply causation
  - social science models can help here.

Overview

1. Terminology
2. Cross-validation
3. Regression (Supervised learning for continuous y)
    1. Subset selection of regressors
    2. Shrinkage methods: ridge, lasso, LAR
    3. Dimension reduction: PCA and partial LS
    4. High-dimensional data
4. Nonlinear models in including neural networks
5. Regression trees, bagging, random forests and boosting
6. Classification (categorical y)
7. Unsupervised learning (no y)
8. Causal inference with machine learning
9. References

1. Terminology

- Topic is called **machine learning** or statistical learning or data learning or data analytics where data may be big or small.
- **Supervised learning = Regression**
  - We have both outcome y and regressors $\mathbf{x}$
  - 1. **Regression**: y is continuous
  - 2. **Classification**: y is categorical
- **Unsupervised learning**
  - We have no outcome y - only several $\mathbf{x}$
  - 3. **Cluster Analysis**: e.g. determine five types of individuals given many psychometric measures.
- These slides
  - focus on 1.

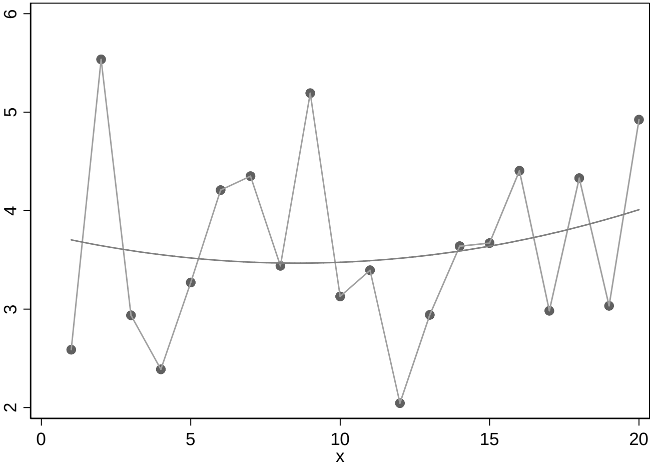
Terminology (continued)

- Machine learning methods guard against overfitting the data.
- Consider two types of data sets
    - 1. **training data set** (or **estimation sample)**
        - ★ used to fit a model
    - 2. **test data set** (or **hold-out sample** or **validation set**)
        - ★ additional data used to determine model goodness-of-fit
        - ★ a test observation $(\mathbf{x}_0, y_0)$ is a previously unseen observation.
- Models are created on 1. and we use the model that does best on 2.

2. Cross Validation

- **Goal:** Predict y given p regressors $x_1, \ldots, x_p$.
- **Criterion**: use **squared error loss** $(y - \widehat{y})^2$
  - some methods adapt to other loss functions.
- **Training data set:** yields the prediction rule $\widehat{f}(x_1, \ldots, x_p)$
  - e.g. OLS yields $\widehat{y} = \widehat{\beta}_0 + \widehat{\beta}_1 x_1 + \cdots + \widehat{\beta}_p x_p$.
- **Test data set:** yields an estimate of the **true prediction error**
  - This is $E[(y_0 - \widehat{y}_0)^2]$ for $(y_0, x_{10}, \ldots, x_{p0})$ not in the training data set.
- Note that we **do not use** the training data set mean squared error
  - $\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \widehat{y}_i)^2$
  - because models overfit in sample (they target y not $E[y|x_1, \ldots, x_p]$)
    - ★ e.g. if $p = n - 1$ then $R^2 = 1$ and $\sum_{i=1}^{n} (y_i - \widehat{y}_i)^2 = 0$.

Bias-variance tradeoff



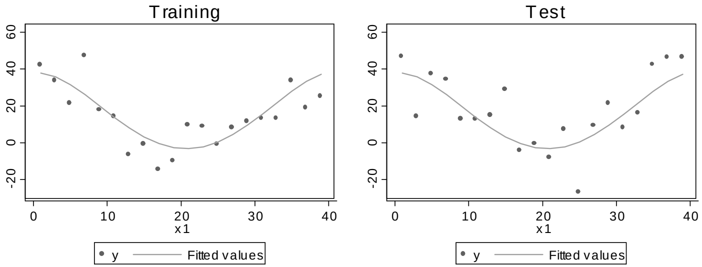
Stata Example

- D.g.p. is quadratic with $n = 40$. Fit OLS polynomial of degree 4.

```
. * Generate data: quadratic with n=40 (total) and n=20 (train) and n=20 (test)
. qui set obs 40

. set seed 10101

. gen x1 = _n - mod(_n+1,2)  // x1 = 1 1 3 3 5 5 .... 39 39

. gen x2 = x1^2

. gen x3 = x1^3

. gen x4 = x1^4

. gen dtrain = mod(_n,2)==1  // dtrain = 1 0 1 0 .... 1 0

. gen y = 2 + 0.1*(x1-20)^2 + rnormal(0,10)

. reg y x1-x4, noheader
------------------------------------------------------------------------------
           y |      Coef.   Std. Err.      t    P>|t|     [95% Conf. Interval]
-------------+----------------------------------------------------------------
          x1 |   .4540487   3.347179     0.14   0.893    -6.341085    7.249183
          x2 |   -.437711   .3399652    -1.29   0.206    -1.127877    .2524551
          x3 |    .020571   .0127659     1.61   0.116    -.0053452    .0464871
          x4 |  -.0002477   .0001584    -1.56   0.127    -.0005692    .0000738
       _cons |   37.91263   9.619719     3.94   0.000     18.38357     57.4417
------------------------------------------------------------------------------
```

Predictions in training and test data sets

- Now fit to only training data $(n_{Train} = 20)$ and plot predictions.
- Quartic model predicts worse in test dataset (right panel)
  - Training data (left): scatterplot and fitted curve $(n_{Test} = 20)$: .
  - Test data (right): scatter plot (different y) and predictions $(n = 20)$.



Single split-sample validation

- Fit polynomial of degree k on training data for $k = 1, \ldots, 4$
  - compute MSE $\sum_i (y_i - \widehat{y}_i)^2$ for training data and test data
- Test MSE is lowest for quadratic
  - Training MSE is lowest for quartic due to overfitting.

```
. * Split sample validation - training and test MSE for polynomials up to deg 4
. forvalues k = 1/4 {
  2.   qui reg y x1-x`k' if dtrain==1
  3.   qui predict y`k'hat
  4.   qui gen y`k'errorsq = (y`k'hat - y)^2
  5.   qui sum y`k'errorsq if dtrain == 1
  6.   qui scalar mse`k'train = r(mean)
  7.   qui sum y`k'errorsq if dtrain == 0
  8.   qui scalar mse`k'test = r(mean)
  9. }

. di _n "MSE linear     Train = " mse1train "  Test = " mse1test _n ///
>    "MSE quadratic  Train = " mse2train "  Test = " mse2test _n ///
>    "MSE cubic      Train = " mse3train "  Test = " mse3test _n ///
>    "MSE quartic    Train = " mse4train "  Test = " mse4test _n

MSE linear     Train = 252.32258  Test = 412.98285
MSE quadratic  Train = 92.781786  Test = 184.43114
MSE cubic      Train = 87.577254  Test = 208.24569
MSE quartic    Train = 72.864095  Test = 207.78885
```

k-fold Cross Validation

- **Problem:** with single-split validation
  - 1. Lose precision due to smaller training set, so may actually overestimate the test error rate (MSE) of the model.
  - 2. Results depend a lot on the particular single split.
- **Solution:** Randomly **divide data into K groups or fold**s of approximately equal size
  - First fold is the validation set
  - Method is fit in the remaining $K-1$ folds
  - Compute MSE for the first fold
  - Repeat K times (drop second fold, third fold, ..) yields

$$\text{CV}_{(K)} = \frac{1}{K}\sum_{j=1}^{K} MSE_{(j)}; \qquad \text{typically } K = 5 \text{ or } K = 10.$$

- Aside: Leave-one-out cross-validation used in bandwidth selection for nonparametric regression (local fit) is the case $K = n$.

k-fold cross validation for full sample

- Begin with all 40 observations
  - Randomly form five folds
  - Five times estimate on four ($n_{Train} = 32$), predict on fifth ($n_{Test} = 8$).
- Following does this for the quadratic model.
  - $CV_{(5)} = \frac{1}{5}(15.27994 + \cdots + 8.444316) = 12.39305.$

```
. * Five-fold cross validation example for quadratic
. set seed 10101

. crossfold regress y x1 x2

             |       RMSE
-------------+-----------
        est1 |   15.27994
        est2 |   16.77849
        est3 |   11.15653
        est4 |   10.30595
        est5 |   8.444316

. * Compute five-fold cross validation measure - average of the above
. matrix RMSEs  = r(est)

. svmat RMSEs, names(rmses)

. sum rmses

    Variable |       Obs        Mean    Std. Dev.       Min        Max
-------------+--------------------------------------------------------
      rmses1 |         5    12.39305    3.501561   8.444316   16.77849
```

Five-fold cross validation for all models

- Do this for polynomials of degree 1, 2, 3 and 4
  - CV measure is lowest for the quadratic.

```
. * Five-fold cross validation measure for polynomials up to degree 4
. forvalues k = 1/4 {
  2.   qui set seed 10101
  3.   qui crossfold regress y x1-x`k'
  4.   qui matrix RMSEs`k' = r(est)
  5.   qui svmat RMSEs`k', names(rmses`k')
  6.   qui sum rmses`k'
  7.   qui scalar cv`k' = r(mean)
  8. }

. di _n "CV(5) for k = 1,..,4 = " cv1 ",  " cv2 ",  "cv3 ",  "cv4

CV(5) for k = 1,..,4 = 12.393046,  12.393046,  12.629339,  12.475117
```

Penalty measures

- Alternative to cross-validation that uses all the data and is quicker.
  - though is more model specific and less universal.
- Focused on loss function squared error or log-likelihood
  - whereas cross-validation approach quite universal.
- Leading examples
  - Akaike's information criterion: $\text{AIC} = -2\ln L + 2p$
  - Bayesian information criterion: $\text{BIC} = -2\ln L + p\ln N$
  - Mallows CP
  - Adjusted R^2 ($\overline{R}^2$)

AIC and BIC penalty measures for full sample

- Compute AIC and BIC for polynomials of degree 1, 2, 3 and 4 ($n = 40$)
- Both AIC and BIC are minimized by the quadratic model.

```
. * Full sample estimates with AIC, BIC penalty - polynomials up to deg 4
.   forvalues k = 1/4 {
  2.   qui reg y x1-x`k'
  3.   qui scalar aic`k' = -2*e(ll) + 2*e(rank)
  4.   qui scalar bic`k' = -2*e(ll) + e(rank)*ln(e(N))
  5. }

. di _n "AIC for k = 1,..,4 = " aic1 ",  " aic2 ",  "aic3 ",  "aic4, ///
>    _n "BIC for k = 1,..,4 = " bic1 ",  " bic2 ",  "bic3 ",  "bic4

AIC for k = 1,..,4 = 348.99841,  314.26217,  316.01317,  315.3112
BIC for k = 1,..,4 = 352.37617,  319.32881,  322.76869,  323.7556
```

3. Regression Methods

- A flexible linear (in parameters) regression model with many regressors may fit well.
- Consider linear regression model with p potential regressors where p is too large.
- Methods that **reduce the model complexity** are
  - choose a subset of regressors
  - shrink regression coefficients towards zero
    - ridge, lasso, LAR
  - reduce the dimension of the regressors
    - principal components analysis.
- Linear regression may predict well if include interactions and powers as potential regressors.

Subset Selection of Regressors

- General idea is for each model size choose best model and then chose between the different model sizes.
- So
  - 1. For $k = 1, 2, ..., p$ choose a "best" model with k regressors
  - 2. Choose among these p models based on model fit with penalty (e.g. CV or AIC) for larger models.
- Methods include
  - best subset
  - forwards stepwise
  - backwards stepwise
  - hybrid.

Variance-bias trade-off

- Consider regression model

$$\begin{aligned} y &= f(\mathbf{x}) + \varepsilon \\ E[\varepsilon] &= 0 \text{ and } \varepsilon \text{ independent of } \mathbf{x} \end{aligned}$$

- For out-of-estimation-sample point $(y_0, \mathbf{x}_0)$ the MSE

$$\begin{aligned} E[(y_0 - \widehat{f}(\mathbf{x}_0))^2] &= Var[\widehat{f}(\mathbf{x}_0)] + \{Bias(\widehat{f}(\mathbf{x}_0))\}^2 + Var(\varepsilon) \\ \text{MSE} &= \text{Variance} + \text{Bias-squared} + \text{Error variance} \end{aligned}$$

- **Lesson 1:** more flexible models have less bias but more variance
- **Lesson 2:** bias can be good if minimizing MSE is our goal.
  - shrinkage estimators exploit this.

Shrinkage Methods

- There is a mean-variance trade-off.
- Shrinkage estimators minimize RSS (residual sum of squares) with a penalty for model size
  - this shrinks parameter estimates towards zero.
- The extent of shrinkage is determined by a **tuning parameter**
  - this is determined by cross-validation or e.g. AIC.
- Ridge and lasso are not invariant to rescaling of regressors, so first standardize the data
  - so x_{ij} below is actually $(x_{ij} - \bar{x}_j)/s_j$.
- Ridge penalty is a multiple of $\sum_{j=1}^{p} \beta_j^2$.
- Lasso penalty is a multiple of $\sum_{j=1}^{p} |\beta_j|$.

Ridge Regression

- Penalty for large models is $\sum_{j=1}^{p} \beta_j^2$.
- The **ridge estimator** $\widehat{\boldsymbol{\beta}}_\lambda$ of $\boldsymbol{\beta}$ minimizes

$$\sum_{i=1}^{n}(y_i - \mathbf{x}_i'\boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} \beta_j^2$$

  - where $\lambda \geq 0$ is a tuning parameter.
- Equivalently, ridge minimizes *RSS* subject to $\sum_{j=1}^{p} \beta_j^2 \leq s$.
- The ridge estimator is

$$\widehat{\boldsymbol{\beta}}_\lambda = (\mathbf{X}'\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}'\mathbf{y}.$$

- Features
  - clearly biased
  - shrinks all coefficients towards zero
  - algorithms exist to quickly compute $\widehat{\boldsymbol{\beta}}_\lambda$ for many values of λ
  - then choose λ by cross validation.

Lasso (Least Absolute Shrinkage And Selection)

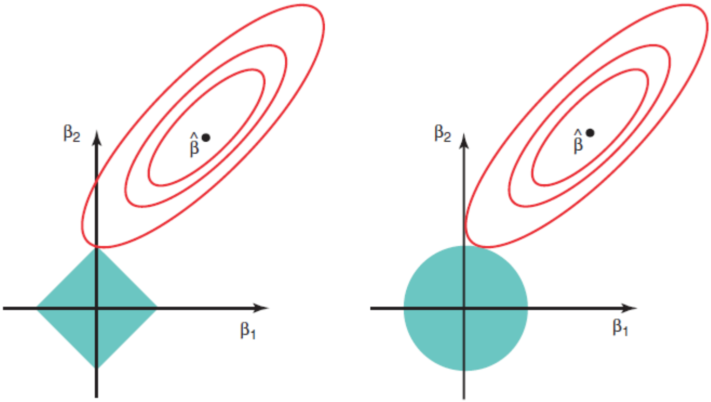
- Penalty for large models is $\sum_{j=1}^{p} |\beta_j|$.
- The **lasso estimator** $\widehat{\boldsymbol{\beta}}_\lambda$ of $\boldsymbol{\beta}$ minimizes

$$\sum_{i=1}^{n} (y_i - \mathbf{x}_i' \boldsymbol{\beta})^2 + \lambda \sum_{j=1}^{p} |\beta_j|$$

  - where $\lambda \geq 0$ is a tuning parameter.
- Equivalently lasso minimizes RSS subject to $\sum_{j=1}^{p} |\beta_j| \leq s$.
- Features
  - drops regressors
  - best when a few regressors have $\beta_j \neq 0$ and most $\beta_j = 0$
  - leads to a more interpretable model than ridge.

Lasso versus Ridge

- $\widehat{\boldsymbol{\beta}} = (\widehat{\beta}_1, \widehat{\beta}_2)$ minimizes residual sum of squares
  - bigger ellipses have larger RSS
  - choose the first ellipse to touch the shaded (constrained) area.
- Lasso (left) gives a corner solution with $\widehat{\beta}_1 = 0$.



Dimension Reduction

- **Reduce** from p regressors to $M < p$ linear combinations of regressors
  - Form $\mathbf{X}^* = \mathbf{XA}$ where $\mathbf{A}$ is $p \times M$ and $M < p$
  - $\mathbf{Y} = \beta_0 + \mathbf{X}\boldsymbol{\beta} + \mathbf{u}$ reduced to
  - $\mathbf{Y} = \beta_0 + \mathbf{X}^*\boldsymbol{\beta} + \mathbf{v}$
    $= \beta_0 + \mathbf{X}\boldsymbol{\beta}^* + \mathbf{v}$ where $\boldsymbol{\beta}^* = \mathbf{A}\boldsymbol{\beta}$.
- Two methods
  - 1. **Principal components**
    - ★ use only $\mathbf{X}$ to form $\mathbf{A}$ (unsupervised)
  - 2. **Partial least squares**
    - ★ also use relationship between $\mathbf{y}$ and $\mathbf{X}$ to form $\mathbf{A}$ (supervised).

High-Dimensional Models

- High dimensional simply means p is large relative to n
  - in particular $p > n$
  - n could be large or small.
- Problems with $p > n$:
  - C_p, AIC, BIC and $\overline{R}^2$ cannot be used.
  - due to multicollinearity cannot identify best model, just one of many good models.
  - cannot use regular statistical inference on training set
- Solutions
  - Forward stepwise, ridge, lasso, PCA are useful in training
  - Evaluate models using cross-validation or independent test data
    - ★ using e.g. MSE or R^2.

4. Nonlinear Models

- Basis function models
  - 1. polynomial regression
  - 2. step functions
  - 3. regression splines
  - 4. smoothing splines, B-splines, ...
  - 5. wavelets
  - polynomial is global while the others break range of x into pieces.
- Other methods
  - local polynomial regression
  - generalized additive models
  - neural networks.

Neural Networks

- Neural network is a very rich parametric model for $f(\mathbf{x})$
  - only parameters need to be estimated
  - as usual guard against overfitting.
- Consider a neural network with two layers
  - Y depends on m $\mathbf{Z}'s$ (a hidden layer) that depend on p $\mathbf{X}'s$.

$$\begin{array}{rll} Z_1 & = g(\alpha_{01} + \mathbf{X}'\boldsymbol{\alpha}_1) & \text{e.g. } g(v) = 1/(1+e^{-v}) \\ \vdots & \quad \vdots \quad \vdots & \\ Z_m & = g(\alpha_{0m} + \mathbf{X}'\boldsymbol{\alpha}_m) & \\ T & = \beta_0 + \sum_{m=1}^{M} \beta_m Z_m & \\ f(\mathbf{X}) & = h(T) & \text{usually } h(T) = T \end{array}$$

- So with above $g(\cdot)$ and $h(\cdot)$

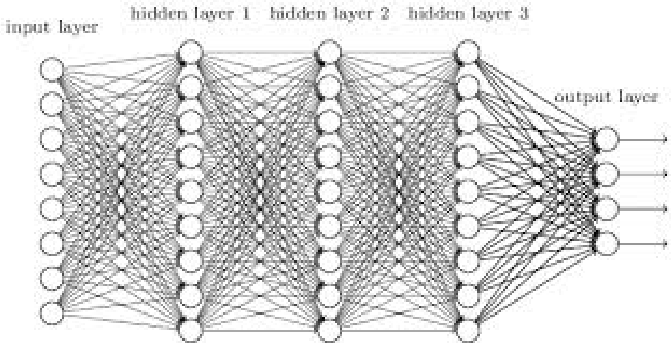
$$f(\mathbf{x}_i) = \beta_0 + \sum_{m=1}^{M} \beta_m \times \frac{1}{1+\exp(-\alpha_{0m} - \mathbf{x}_i'\alpha_m)}.$$

- We need to find the number M of hidden units and estimate the $\alpha's$.

Neural Networks (continued)

- Minimize the sum of squared residuals but need a penalty on $\alpha' s$ to avoid overfitting.
  - Since penalty is introduced standardize $x' s$ to (0,1).
  - Best to have too many hidden units and then avoid overfit using penalty.
- Neural nets are good for prediction
  - especially in speech recognition, image recognition, ...
  - but very difficult (impossible) to interpret.
- Deep learning uses nonlinear transformations such as neural networks
  - deep nets are an improvement on original neural networks
  - e.g. led to great improvement of Google Translate.

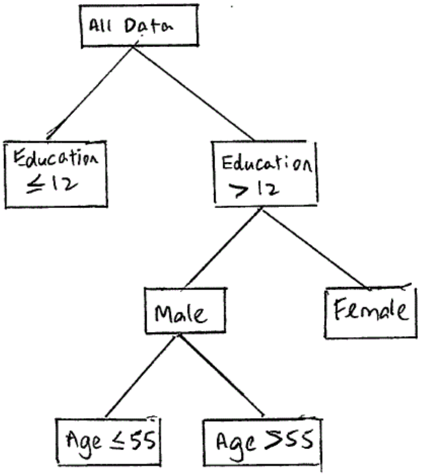
- Off-the-shelf software
  - 1. converts e.g. image or text to y and $\mathbf{x}$ to data input
  - 2. runs the deep net using stochastic gradient descent
  - e.g. CNTK (Microsoft), or Tensorflow (Google) or mxnet
- Inference: neural net gives in-sample $\widehat{y}_i = \psi_i(\mathbf{x}_i)'\widehat{\boldsymbol{\beta}}$
  - so out-of-sample OLS regress y_i on $\psi_i(\mathbf{x})$ gives $\widetilde{\boldsymbol{\beta}}$ and $\text{se}(\widetilde{\boldsymbol{\beta}})$.



5. Regression Trees

- Regression Trees sequentially split regressors $\mathbf{x}$ into regions that best predict y
  - e.g., first split is education $<$ or $>$ 12
    and second split is on gender for education $>$ 12
    and third split is on age $\leq$ 55 or $>$ 55 for male with education $>$ 12
    and could then re-split on education
- Then $\widehat{y}_i = \bar{y}_{R_j}$ is the average of $y's$ in the region that $\mathbf{x}_i$ falls in
  - with J blocks RSS $= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2$.
- Need to determine both the regressor j to split and the split point s.
  - Each split is the one that reduces RSS the most.
  - Stop when e.g. less than five observations in each region.

- Example: annual earnings y depend on education, gender, age, ...



Bagging, Random Forests and Boosting

- Trees do not predict well due to high variance
  - e.g. split data in two then can get quite different trees
  - e.g. first split determines future splits.
  - called a **greedy algorithm** as does not consider future splits.
- **Bagging** (bootstrap averaging) computes regression trees
  - for many different samples obtained by bootstrap
  - then average predictions across the trees.
- **Random forests** use only a subset of the predictors in each bootstrap sample
- **Boosting** grows tree using information from previously grown trees
  - and is fit on a modified version of the original data set
- Bagging and boosting are general methods (not just for trees).

6. Classification Methods

- $y's$ are now categorical (e.g. binary if two categories).
- Use (0,1) loss function
  - 0 if correct classification and 1 if missclassified.
- Methods
  - logistic regression, multinomial regression, k nearest neighbors
  - linear and quadratic discriminant analysis
  - support vector classifiers and support vector machines

7. Unsupervised Learning

- Challenging area: no y, only $\mathbf{X}$.
- Principal components analysis.
- Clustering Methods
  - k means clustering.
  - hierarchical clustering.

8. Causal Inference with Machine Learning

- Focus on **causal estimation** of a key parameter, such as an average marginal effect, after controlling for confounding factors.
- For models with selection on observables (unconfoundedness)
  - e.g. regression with controls or propensity score matches
  - good controls makes this assumption more reasonable
  - so use only use machine learning methods (notably lasso) to select best controls.
- And for instrumental variables estimation with many possible instruments
  - using a few instruments avoids many instruments problem
  - use machine learning methods (notably lasso) to select best instruments
- But valid statistical inference needs to control for this data mining
  - currently active area of econometrics research.

- Commercial example is online website predicting quantity demand change from price change.
- $q(p) = f(p) + e(p)$ where $e(p)$ is error
  - naive machine learners will fit $f(p)$ well
  - but $dq(p)/dp = df(p)/dp + de(p)/dp$
- Suppose $y = g(x) + \varepsilon$ where x is endogenous
  - there are instruments $E[\varepsilon|z] = 0$.
  - then $\pi(z) = E[y|z] = E[g(x)|z] = \int g(x)dF(x|z)$
  - use machine learner to get $\widehat{\pi}(z)$ and $\widehat{F}(x|z)$
  - then solve the above integral equation
- Easier for economists to use off-the-shelf machine learners
  - than for machine learners to learn methods for endogeneity.

9. Big Data

- Hal Varian (2014), "Big Data: New Tricks for Econometrics", JEP, Spring, 3-28.
- Tools for handling big data
  - file system for files split into large blocks across computers
    - Google file system (Google), Hadoop file system
  - database management system to handle large amounts of data across many computers
    - Bigtable (Google), Cassandra
  - accessing and manipulating big data sets across many computers
    - MapReduce (Google), Hadoop.
  - language for Mapreduce / Hadoop
    - Sawzall (Google), Pig
  - Computer language for parallel processing
    - Go (Google - open source)
  - simplified structured query language (SQL) for data enquiries
    - Dremel, Big Query (Google), Hive, Drill, Impala.

10. Conclusion

- Machine learning focuses on prediction
  - guarding for overfitting using validation or AIC/BIC.
- Supervised learning predicts y given $\mathbf{x}$
  - usual regression minimizes MSE $=$ bias$^2 +$ variance
  - classification minimizes (0,1) loss function.
- Most popular machine learning method
  - deep neural nets.
- Economists / econometricians adapt to causal inference using
  - LASSO
  - Random forests.

11. Book References

- **http://cameron.econ.ucdavis.edu/e240f/machinelearning.html**
- Next two books I used have free pdf and $25 softback.
- Undergraduate / Masters level book
  - Gareth James, Daniela Witten, Trevor Hastie and Robert Tibsharani (2013), *An Introduction to Statistical Learning: with Applications in R*, Springer.
- Masters / PhD level book
  - Trevor Hastie, Robert Tibsharani and Jerome Friedman (2009), *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer.
- A recent book
  - Bradley Efron and Trevor Hastie (2016), *Computer Age Statistical Inference: Algorithms, Evidence and Data Science*, Cambridge University Press.
- Interesting general audience book is Cathy O'Neil, Weapons of Math Destruction: *How Big Data Increases Inequality and Threatens Democracy*.

Simpler Articles

- Hal Varian (2014), "Big Data: New Tricks for Econometrics", *Journal of Economic Perspectives*, Spring, 3-28.
- Sendhil Mullainathan and J. Spiess (2017), "Machine Learning: An Applied Econometric Approach", *Journal of Economic Perspectives*, Spring, pp. 87-106.
- A. Belloni, V. Chernozhukov and C. Hansen (2014), "High-Dimensional Methods and Inference on Treatment and Structural Effects in Economics," *Journal of Economic Perspectives* Spring, pp.29-50.
- Following are leaders in causal econometrics and machine learning
  - Victor Chernozhukov, Alex Belloni, Christian Hansen + coauthors
    - use Lasso a lot.
  - SusanAthey and Guido Imbens
    - use random forests a lot.