Machine Learning for Microeconometrics
Part 2 - Flexible methods

A. Colin Cameron
U.C.-Davis
.

presented at CINCH Academy 2019
The Essen Summer School in Health Economics
and at
Friedrich Alexander University, Erlangen-Nurnberg

April 2019

Introduction

- Basics used OLS regression
  - though with potentially rich set of regressors with interactions ....
- Now consider remaining methods
  - for supervised learning (y and $\mathbf{x}$)
  - and unsupervised learning (y only).
- Again based on the two books by Hastie and Tibsharani and coauthors.
- These slides present many methods for completeness
  - the most used method in economics is random forests.

- The course is broken into three sets of slides.
- Part 1: Basics
  - variable selection, shrinkage and dimension reduction
  - focuses on linear regression model but generalizes.
- **Part 2: Flexible methods**
  - nonparametric and semiparametric regression
  - flexible models including splines, generalized additive models, neural networks
  - regression trees, random forests, bagging, boosting
  - classification (categorical y) and unsupervised learning (no y).
- Part 3: Microeconometrics
  - OLS with many controls, IV with many instruments, ATE with heterogeneous effects and many controls.
- Parts 1 and 2 are based on the two books given in the references
  - *Introduction to Statistical Learning*
  - *Elements of Statistical Learning.*
- While most ML code is in R, these slides use Stata.

Flexible methods

- These slides present many methods
- Which method is best (or close to best) varies with the application
  - e.g. deep learning (neural nets) works very well for Google Translate.
- In forecasting competitions the best forecasts are **ensembles**
  - a weighted average of the forecasts obtained by several different methods
  - the weights can be obtained by OLS regression in a test sample
    - ★ e.g. given three forecast methods minimize w.r.t. τ_1 and τ_2 $\sum_{i=1}^{n}\{y_i - \tau_1\widehat{y}_i^{(1)} - \tau_2\widehat{y}_i^{(2)} - (1-\tau_1-\tau_2)\widehat{y}_i^{(3)}\}^2$.

Overview

1. Nonparametric and semiparametric regression
2. Flexible regression (splines, sieves, neural networks,...)
3. Regression trees and random forests
    1. Regression trees
    2. Bagging
    3. Random forests
    4. Boosting
4. Classification (categorical y)
    1. Loss function
    2. Logit
    3. k-nearest neighbors
    4. Discriminant analysis
    5. Support vector machines
5. Unsupervised learning (no y)
    1. Principal components analysis
    2. Cluster analysis

1.1 Nonparametric regression

- Nonparametric regression is the most flexible approach
  - but it is not practical for high p due to the curse of dimensionality.
- Consider explaining y with scalar regressor x
  - we want $\widehat{f}(x_0)$ for a range of values x_0.
- With many observations with $x_i = x_0$ we would just use the average of y for those observations
  - $\widehat{f}(x_0) = \frac{1}{n_0}\sum_{i:x_i=x_0}^{n} y_i = \frac{\sum_{i=1}^{n}\mathbf{1}[x_i=x_0]y_i}{\sum_{i=1}^{n}\mathbf{1}[x_i=x_0]}$
- Rewrite as

$$\widehat{f}(x_0) = \sum_{i=1}^{n} w(x_i, x_0)y_i, \text{ where } w(x_i, x_0) = \frac{\mathbf{1}[x_i=x_0]}{\sum_{j=1}^{n}\mathbf{1}[x_j=x_0]}.$$

Kernel-weighted local regression

- In practice there are not many observations with $x_i = x_0$.
- Nonparametric regression methods borrow from nearby observations
  - **k-nearest neighbors**
    - ★ average y_i for the k observations with x_i closest to x_0.
  - **kernel-weighted local regression**
    - ★ use a weighted average of y_i with weights declining as $|x_i - x_0|$ increases.
- Then the original kernel regression estimate is

$$\widehat{f}(x_0) = \sum_{i=1}^{n} w(x_i, x_0, \lambda) y_i.$$

  - where $w(x_i, x_0, \lambda) = w(\frac{x_i, -x_0}{\lambda})$ are kernel weights
  - and λ is a bandwidth parameter to be determined.

Kernel weights

- A kernel function is continuous and is symmetric at zero

with $\int K(z)dz = 1$ and $\int zK(z)dz = 0$
e.g. $K(z) = (1 - |z|) \times \mathbf{1}(|z| < 1)$

- The kernel weights are

$$w(x_i, x_0, \lambda) = w\left(\frac{x_i - x_0}{\lambda}\right) = \frac{K\left(\frac{x_i - x_0}{\lambda}\right)}{\sum_{j=1}^{n} K\left(\frac{x_i - x_0}{\lambda}\right)}.$$

- The bandwidth λ is chosen to shrink to zero as $n \longrightarrow \infty$.
- The estimator $\widehat{f}(x_0)$ is biased for $f(x_0)$.

Local constant and local linear regression

- The local constant estimator $\widehat{f}(x_0) = \widehat{\alpha}_0$ where α_0 minimizes

$$\sum_{i=1}^{n} w(x_i, x_0, \lambda)(y_i - \alpha_0)^2$$

  - this yields $\widehat{\alpha}_0 = \sum_{i=1}^{n} w(x_i, x_0, \lambda) y_i$.
- The local linear estimator $\widehat{f}(x_0) = \widehat{\alpha}_0$ where α_0 and β_0 minimize

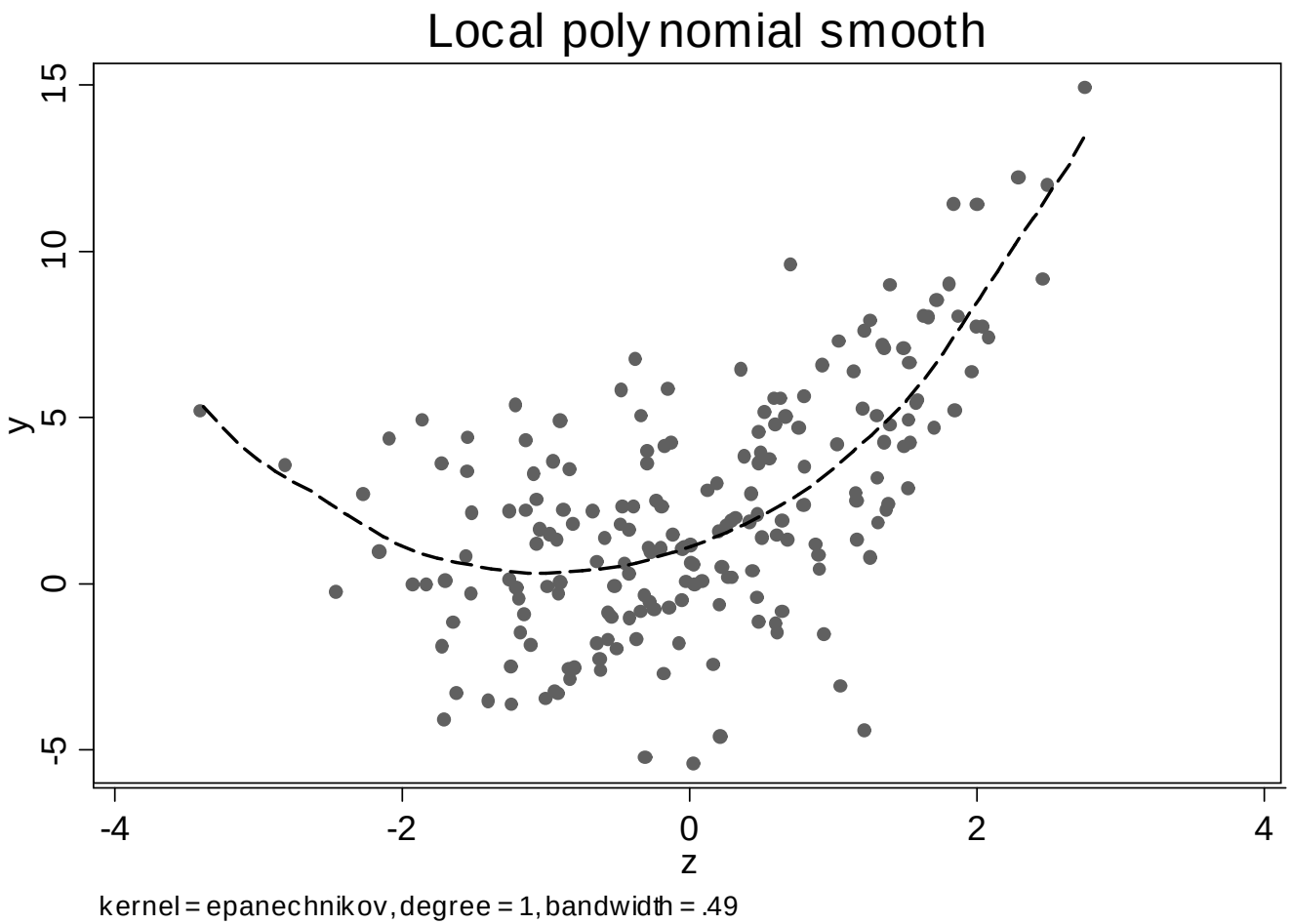
$$\sum_{i=1}^{n} w(x_i, x_0, \lambda)\{y_i - \alpha_0 - \beta_0(x_i - x_0)\}^2.$$

- Stata commands
  - `lpoly` uses a plug-in bandwidth value λ
  - `npregress` is much richer and uses LOOCV bandwidth λ.
- Can generalize to local maximum likelihood that maximizes over θ_0

$$\sum_{i=1}^{n} w(x_i, x_0, \lambda) \ln f(y_i, x_i, \theta_0).$$

Local linear example

- `lpoly y z, degree(1)`



1.2 Curse of Dimensionality

- Nonparametric methods do not extend well to multiple regressors.
- Consider p-dimensional $\mathbf{x}$ broken into bins
  - for $p = 1$ we might average y in each of 10 bins of x
  - for $p = 2$ we may need to average over 10^2 bins of (x_1, x_2)
  - and so on.
- On average there may be few to no points with high-dimensional $\mathbf{x}_i$ close to $\mathbf{x}_0$
  - called the curse of dimensionality.
- Formally for local constant kernel regression with bandwidth λ
  - bias is $O(\lambda^2)$ and variance is $O(n\lambda^p)$
  - optimal bandwidth is $O(n^{-1/(p+4)})$
    - ★ gives asymptotic bias so standard conf. intervals not properly centered
  - convergence rate is then $n^{-2/(p+4)} << n^{-0.5}$

1.3 Semiparametric Models

- Semiparametric models provide some structure to reduce the nonparametric component from K dimensions to 1 dimension.
  - Econometricians focus on partially linear models and on single-index models.
  - Statisticians use generalized additive models and project pursuit regression.
- Machine learning methods can outperform nonparametric and semiparametric methods
  - so wherever econometricians use nonparametric and semiparametric regression in higher-dimensional models it may be useful to use ML methods.

Partially linear model

- A partially linear model specifies

$$y_i = f(\mathbf{x}_i, \mathbf{z}_i) + u_i = \mathbf{x}_i'\boldsymbol{\beta} + g(\mathbf{z}_i) + u_i$$

  - simplest case z (or x) is scalar but could be vectors
  - the nonparametric component is of dimension of $\mathbf{z}$.

- The differencing estimator of Robinson (1988) provides a root-n consistent asymptotically normal $\widehat{\boldsymbol{\beta}}$ as follows
  - $E[y|\mathbf{z}] = E[\mathbf{x}|\mathbf{z}]'\boldsymbol{\beta} + g(\mathbf{z})$ as $E[u|\mathbf{z}] = 0$ given $E[u|\mathbf{x}, \mathbf{z}] = 0$
  - $y - E[y|\mathbf{z}] = (\mathbf{x} - E[\mathbf{x}|\mathbf{z}])'\boldsymbol{\beta} + u$ subtracting
  - so OLS estimate $y - \widehat{m}_y = (\mathbf{x} - \widehat{\mathbf{m}}_{\mathbf{z}})'\boldsymbol{\beta}+$ error.

- Robinson proposed nonparametric kernel regression of y on $\mathbf{z}$ for $\widehat{m}_y$ and $\mathbf{x}$ on $\mathbf{z}$ for $\widehat{\mathbf{m}}_{\mathbf{x}}$
  - recent econometrics articles instead use a machine learner such as LASSO
  - in general need $\widehat{m}$ converges at rate at least $n^{-1/4}$.

Single-index model

- Single-index models specify

$$f(\mathbf{x}_i) = g(\mathbf{x}_i'\boldsymbol{\beta})$$

  - with $g(\cdot)$ determined nonparametrically
  - this reduces nonparametrics to one dimension.
- We can obtain $\widehat{\boldsymbol{\beta}}$ root-n consistent and asymptotically normal
  - provided nonparametric $\widehat{g}(\cdot)$ converges at rate $n^{1/4}$.
- The recent economics ML literature has instead focused on the partially linear model.

Generalized additive models and project pursuit

- Generalized additive models specify $f(\mathbf{x})$ as a linear combination of scalar functions

$$f(\mathbf{x}_i) = \alpha + \sum_{j=1}^{p} f_j(x_{ij})$$

  - where x_j is the j^{th} regressor and $f_j(\cdot)$ is (usually) determined by the data
  - advantage is interpretability (due to each regressor appearing additively).
  - can make more nonlinear by including interactions such as $x_{i1} \times x_{i2}$ as a separate regressor.

- Project pursuit regression is additive in linear combinations of the $x's$

$$f(\mathbf{x}_i) = \sum_{m=1}^{M} g_m(\mathbf{x}_i'\boldsymbol{\omega}_m)$$

  - additive in derived features $\mathbf{x}'\boldsymbol{\omega}_m$ rather than in the $x_j's$
  - the $g_m(\cdot)$ functions are unspecified and nonparametrically estimated.
  - this is a multi-index model with case $M = 1$ being a single-index model.

How can ML methods do better?

- In theory there is scope for improving nonparametric methods.
- k-nearest neighbors usually has a fixed number of neighbors
  - but it may be better to vary the number of neighbors with data sparsity
- Kernel-weighted local regression methods usually use a fixed bandwidth
  - but it may be better to vary the bandwidth with data sparsity.
- There may be advantage to basing neighbors in part on relationship with y.

2. Flexible Regression

- Basis function models
  - global polynomial regression
  - splines: step functions, regression splines, smoothing splines
  - wavelets
  - polynomial is global while the others break range of x into pieces.
- Other methods
  - neural networks.

2.1 Basis Functions

- Also called series expansions and sieves.
- General approach (scalar x for simplicity)

$$y_i = \beta_0 + \beta_1 b_1(x_i) + \cdots + \beta_K(x_i) + \varepsilon_i$$

  - where $b_1(\cdot), ..., b_K(\cdot)$ are basis functions that are fixed and known.
- Global polynomial regression sets $b_j(x_i) = x_i^j$
  - typically $K \leq 3$ or $K \leq 4$.
  - fits globally and can overfit at boundaries.
- Step functions: separately fit y in each interval $x \in (c_j, c_{j+1})$
  - could be piecewise constant or piecewise linear.
- Splines smooth so that not discontinuous at the cut points.
- Wavelets are also basis functions, richer than Fourier series.

Global Polynomials Example

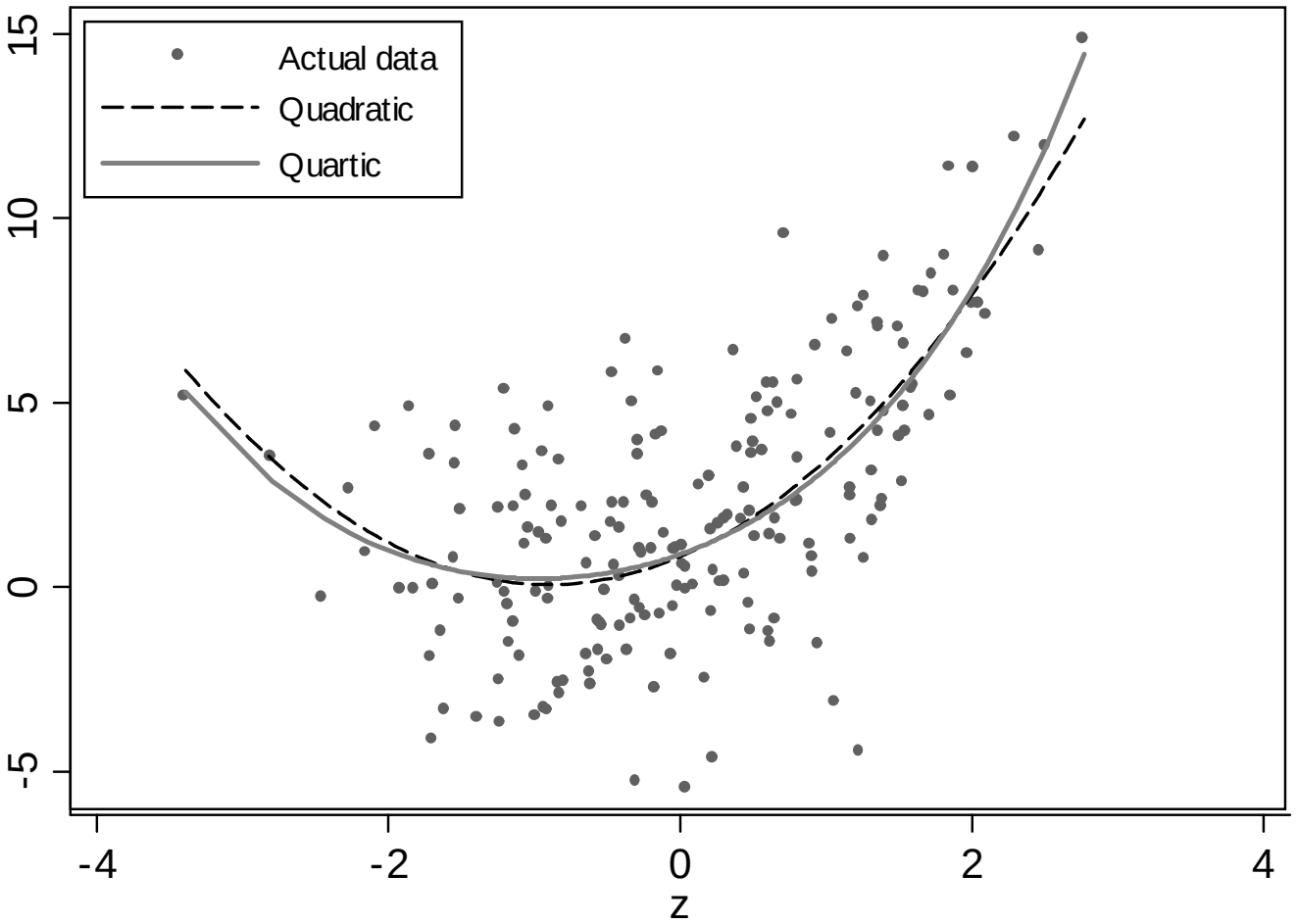
- Generated data: $y_i = 1 + 1 \times x1 + 1 \times x2 + f(z) + u$ where $f(z) = z + z^2$.

```
. * Generated data: y = 1 + 1*x1 + 1*x2 + f(z) + u where f(z) = z + z^2
. clear

. set obs 200
number of observations (_N) was 0, now 200

. set seed 10101

. generate x1 = rnormal()

. generate x2 = rnormal() + 0.5*x1

. generate z = rnormal() + 0.5*x1

. generate zsq = z^2

. generate y = 1 + x1 + x2 + z + zsq + 2*rnormal()

. summarize
```

Variable	Obs	Mean	Std. Dev.	Min	Max
x1	200	.0301211	1.014172	-3.170636	3.093716
x2	200	.0226274	1.158216	-4.001105	3.049917
z	200	.0664539	1.146429	-3.386704	2.77135
zsq	200	1.312145	1.658477	.0000183	11.46977
y	200	2.164401	3.604061	-5.468721	14.83116

Global Polynomials Example (continued)

- Fit quartic in z with (x_1 and x_2) omitted and compare to quadratic
  - `regress y c.z##c.z##c.z##c.z, vce(robust)`
  - quartic chases endpoints.



2.2 Regression Splines

- Begin with step functions: separate fits in each interval (c_j, c_{j+1})
- Piecewise constant
  - $b_j(x_i) = 1[c_j \leq x_i < c_{j+1}]$
- Piecewise linear
  - intercept is $1[c_j \leq x_i < c_{j+1}]$ and slope is $x_i \times 1[c_j \leq x_i < c_{j+1}]$
- Problem is that discontinuous at the cut points (does not connect)
  - solution is splines.

Piecewise linear spline

- Begin with piecewise linear with two knots at c and d

$$\begin{aligned} f(x) &= \alpha_1 1[x < c] + \alpha_2 x 1[x < c] + \alpha_3 1[c \leq x < d] \\ &\quad + \alpha_4 x 1[c \leq x < d] + \alpha_5 1[x \geq d] + \alpha_6 x 1[x \geq d]. \end{aligned}$$

- To make continuous at c (so $f(c-) = f(c)$) and d (so $f(d-) = f(d)$) we need two constraints

$$\begin{aligned} \text{at } c: &\quad \alpha_1 + \alpha_2 c = \alpha_3 + \alpha_4 c \\ \text{at } d: &\quad \alpha_3 + \alpha_4 d = \alpha_5 + \alpha_6 d. \end{aligned}$$

- Alternatively introduce the Heaviside step function

$$h_+(x) = x_+ = \begin{cases} x & x > 0 \\ 0 & \textit{otherwise}. \end{cases}$$

- Then the following imposes the two constraints (so have $6 - 2 = 4$ regressors)

$$f(x) = \beta_0 + \beta_1 x + \beta_2 (x - c)_+ + \beta_2 (x - d)_+$$

Spline Example

- Piecewise linear spline with two knots done manually.

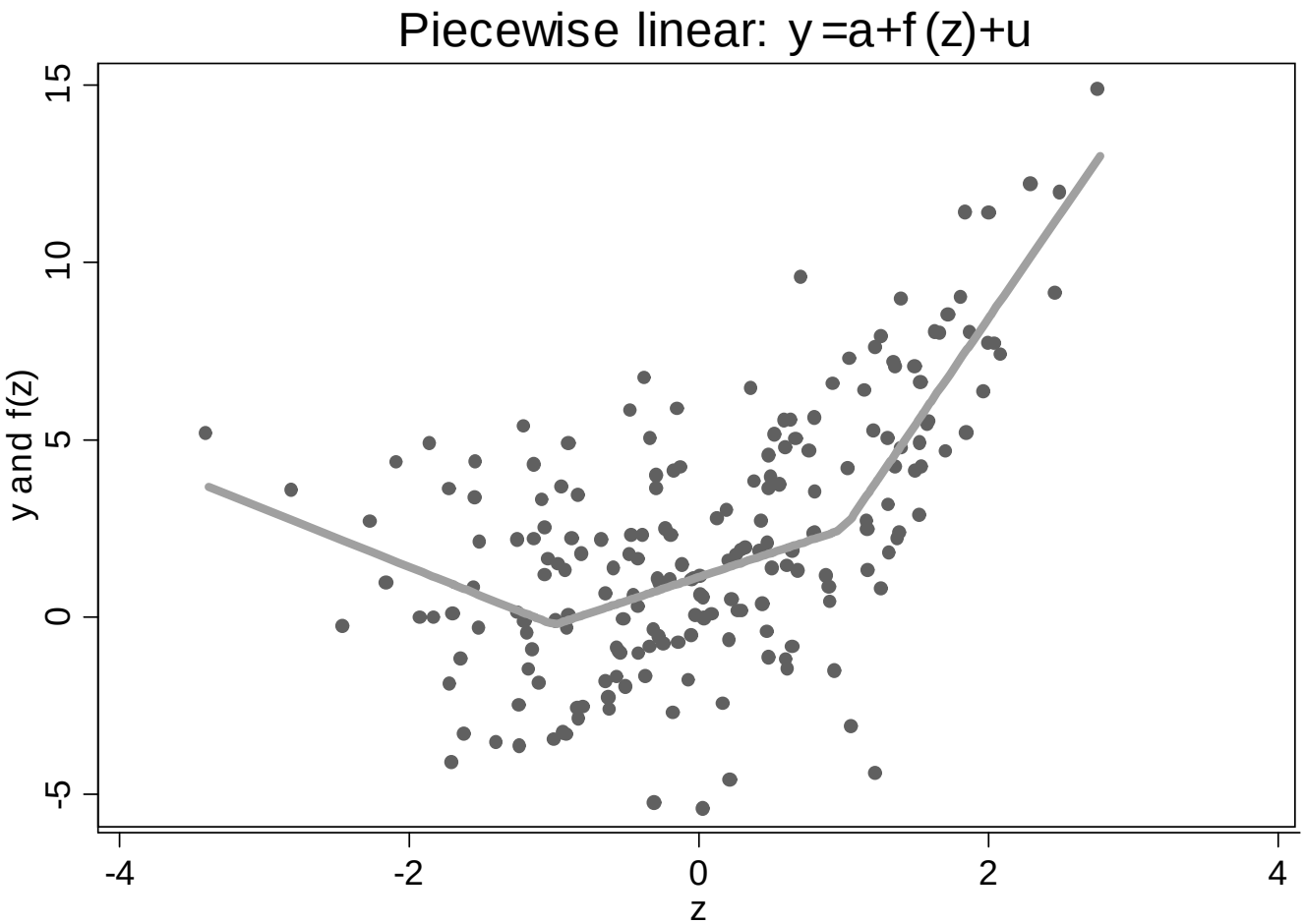
```
. * Create the basis function manually with three segments and knots at -1 and 1
. generate zseg1 = z

. generate zseg2 = 0

. replace zseg2 = z - (-1) if z > -1
(163 real changes made)

. generate zseg3 = 0

. replace zseg3 = z - 1 if z > 1
(47 real changes made)

.
. * Piecewise linear regression with three sections
. regress y zseg1 zseg2 zseg3
```

Source	SS	df	MS		
Model	1253.3658	3	417.7886	Number of obs	= 200
Residual	1331.49624	196	6.79334818	F(3, 196)	= 61.50
				Prob > F	= 0.0000
				R-squared	= 0.4849
				Adj R-squared	= 0.4770
Total	2584.86204	199	12.9892565	Root MSE	= 2.6064

y	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
zseg1	-1.629491	.6630041	-2.46	0.015	-2.937029	-.3219535
zseg2	2.977586	.8530561	3.49	0.001	1.295239	4.659933
zseg3	4.594974	.9164353	5.01	0.000	2.787634	6.402314
_cons	-1.850531	.9204839	-2.01	0.046	-3.665855	-.0352065

Spline Example (continued)

- Plot of fitted values from piecewise linear spline has three connected line segments.



Spline Example (continued)

- The `mkspline` command creates the same spline variables.

```
. * Repeat piecewise linear using command mkspline to create the basis functions
. mkspline zmk1 -1 zmk2 1 zmk3 = z, marginal

. summarize zseg1 zmk1 zseg2 zmk2 zseg3 zmk3, sep (8)
```

Variable	Obs	Mean	Std. Dev.	Min	Max
zseg1	200	.0664539	1.146429	-3.386704	2.77135
zmk1	200	.0664539	1.146429	-3.386704	2.77135
zseg2	200	1.171111	.984493	0	3.77135
zmk2	200	1.171111	.984493	0	3.77135
zseg3	200	.138441	.3169973	0	1.77135
zmk3	200	.138441	.3169973	0	1.77135

- To repeat earlier results: `regress y zmk1 zmk2 zmk3`
- And to add regressors: `regress y x1 x2 zmk1 zmk2 zmk3`

Cubic Regression Splines

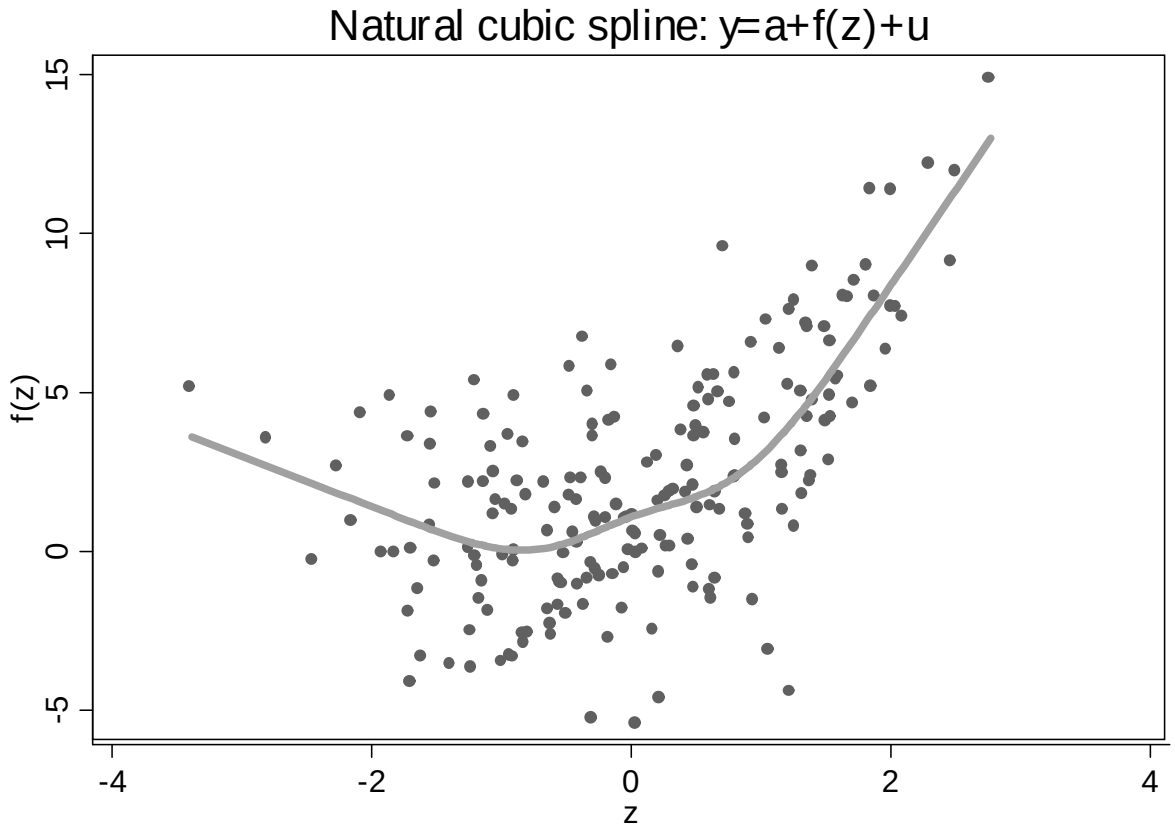
- This is the standard.
- Piecewise cubic model with K knots
  - require $f(x)$, $f'(x)$ and $f''(x)$ to be continuous at the K knots
- Then can do OLS with

$$f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x - c_1)_+^3 + \cdots + \beta_{(3+K)} (x - c_K)_+^3$$

  - for proof when $K = 1$ see ISL exercise 7.1.
- This is the lowest degree regression spline where the graph of $\widehat{f}(x)$ on x seems smooth and continuous to the naked eye.
- There is no real benefit to a higher-order spline.
- Regression splines overfit at boundaries.
  - A natural or restricted cubic spline is an adaptation that restricts the relationship to be linear past the lower and upper boundaries of the data.

Spline Example

- Natural or restricted cubic spline with five knots at the 5, 27.5, 50, 72.5 and 95 percentiles
  - `mkspline zspline = z, cubic nknots(5) displayknots`
  - `regress y zspline*`



Other Splines

- Regression splines and natural splines require choosing the cut points
  - e.g. use quintiles of x.
- Smoothing splines avoid this
  - use all distinct values of x as knots
  - but then add a smoothness penalty that penalizes curvature.
- The function $g(\cdot)$ minimizes

$$\sum_{i=1}^{n}(y_i - g(\mathbf{x}_i))^2 + \lambda \int_a^b g''(t)dt \text{ where } a \leq \text{all } x_i \leq b.$$

  - $\lambda = 0$ connects the data points and $\lambda \rightarrow \infty$ gives OLS.
  - Stata addon command `gam` (Royston and Ambler) does this but only for MS Windows Stata.
- User-written `bspline` command (Newson 2012) enables generation of a range of bases including B splines.
- For multivariate splines use multivariate adaptive regression splines (MARS).

2.3 Wavelets

- Wavelets are used especially for signal processing and extraction
  - they are richer than a Fourier series basis
  - they can handle both smooth sections and bumpy sections of a series.
  - they are not used in cross-section econometrics but may be useful for some time series.
- Start with a mother or father wavelet function $\psi(x)$
  - example is the Haar function $\psi(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} < x < 1 \\ 0 & \text{otherwise} \end{cases}$
- Then both translate by b and scale by a to give basis functions $\psi^{ab}(x) = |a|^{-1/2}\psi(\frac{x-b}{a})$.

2.4 Neural Networks

- A neural network is a richer model for $f(\mathbf{x}_i)$ than project pursuit
  - but unlike project pursuit all functions are specified
  - only parameters need to be estimated.
- A neural network involves a series of nested logit regressions.
- A single hidden layer neural network explaining y by $\mathbf{x}$ has
  - y depends on $\mathbf{z}'s$ (a hidden layer)
  - $\mathbf{z}'s$ depend on $\mathbf{x}'s$.
- A neural network with two hidden layers explaining y by $\mathbf{x}$ has
  - y depends on $\mathbf{w}'s$ (a hidden layer)
  - $\mathbf{w}'s$ depend on $\mathbf{z}'s$ (a hidden layer)
  - $\mathbf{z}'s$ depend on $\mathbf{x}'s$.

Two-layer neural network

- y depends on M $\mathbf{z}'s$ and the $\mathbf{z}'s$ depend on p $\mathbf{x}'s$

$$\begin{aligned} f(\mathbf{x}) &= \beta_0 + \mathbf{z}'\boldsymbol{\beta} && \text{is usual choice for } g(\cdot) \\ z_m &= \frac{1}{1+\exp[-(\alpha_{0m}+\mathbf{x}'\alpha_m)]} && m = 1, ..., M \end{aligned}$$

- More generally we may use

$$\begin{aligned} f(\mathbf{x}) &= h(T) && \text{usually } h(T) = T \\ T &= \beta_0 + \mathbf{z}'\boldsymbol{\beta} \\ z_m &= g(\alpha_{0m} + \mathbf{x}'\alpha_m) && \text{usually } g(v) = 1/(1+e^{-v}) \end{aligned}$$

- This yields the nonlinear model

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

Neural Networks (continued)

- Neural nets are good for prediction
  - especially in speech recognition (Google Translate), image recognition, ...
  - but very difficult (impossible) to interpret.
- They require a lot of fine tuning - not off-the-shelf
  - we need to determine the find the number of hidden layers, the number of M of hidden units within each layer, and estimate the $\alpha's$, $\beta's$,....
- 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.
  - initially back propagation was used
  - now use gradient methods with different starting values and average results or use bagging.
- Deep learning uses nonlinear transformations such as neural networks
  - deep nets are an improvement on original neural networks.

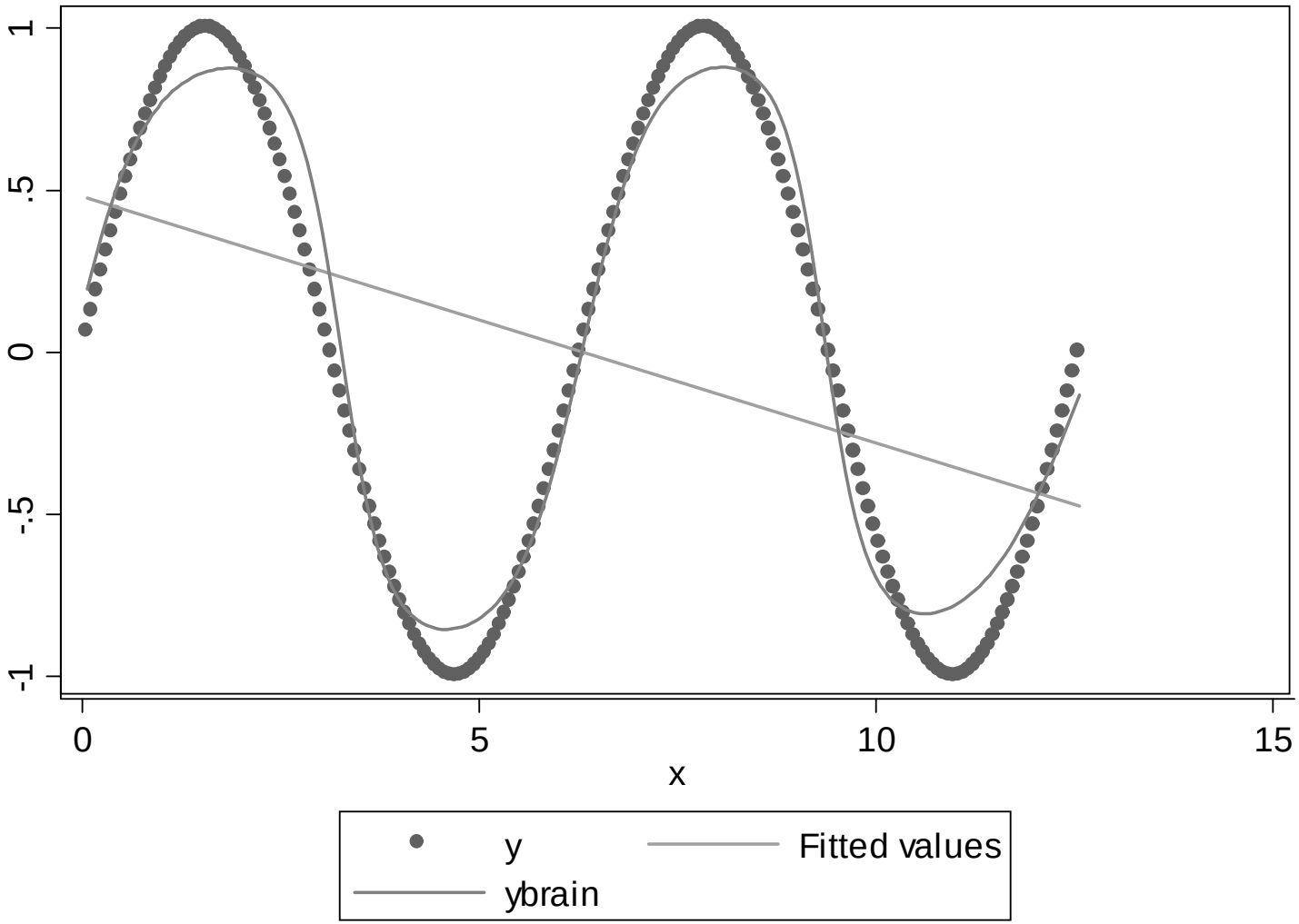
Neural Networks Example

- This example uses user-written Stata command `brain` (Doherr)

```
. * Example from help file for user-written brain command
. clear

. set obs 200
number of observations (_N) was 0, now 200

. gen x = 4*_pi/200 *_n

. gen y = sin(x)

. brain define, input(x) output(y) hidden(20)
Defined matrices:
   input[4,1]
  output[4,1]
  neuron[1,22]
   layer[1,3]
   brain[1,61]

. quietly brain train, iter(500) eta(2)

. brain think ybrain

. sort x

. twoway (scatter y x) (lfit y x) (line ybrain x)
```

Neural Networks Example (continued)

- We obtain



- This figure from ESL is for classification with K categories

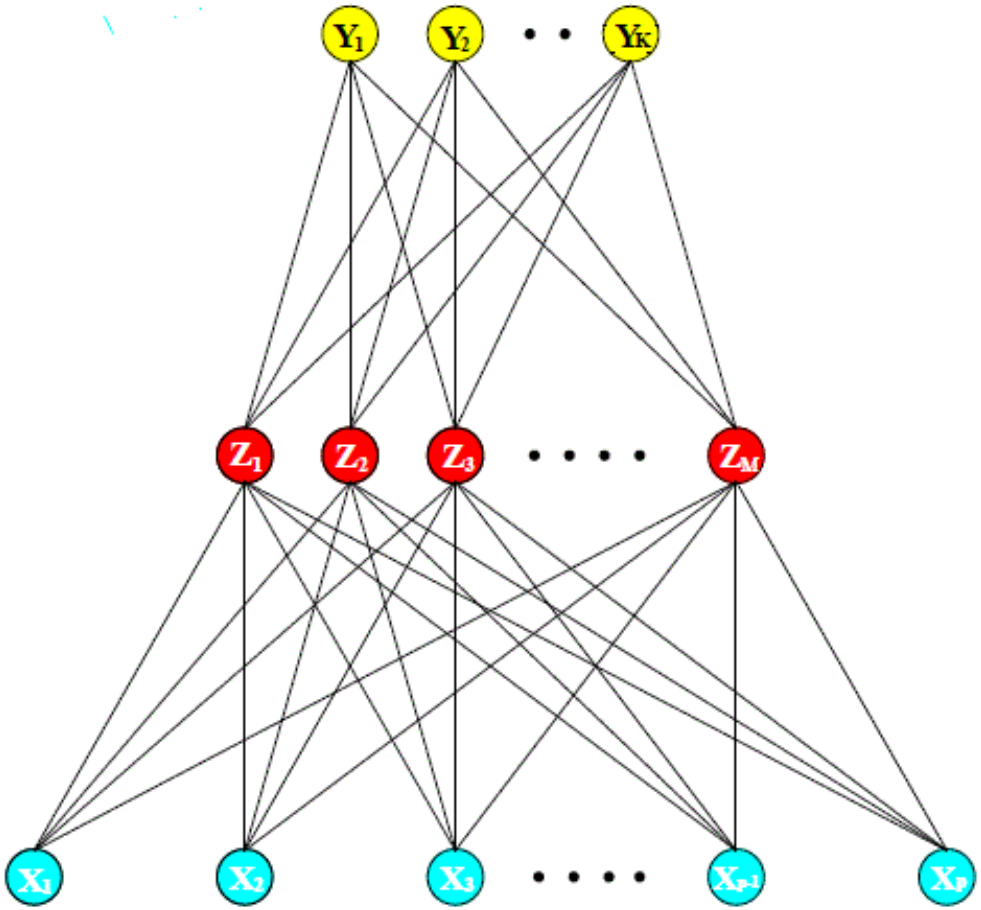


FIGURE 11.2. *Schematic of a single hidden layer, feed-forward neural network.*

3. Regression Trees and Random Forests: Overview

- Regression Trees sequentially split regressors **x** into regions that best predict y
  - e.g., first split is income $<$ or $>$ \$12,000
    second split is on gender if income $>$ \$12,000
    third split is income $<$ or $>$ \$30,000 (if female and income $>$ \$12,000).
- 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 9a greedy method).
- Better methods are then given
  - bagging (bootstrap averaging) computes regression trees for different samples obtained by bootstrap and averages the predictions.
  - random forests use only a subset of the predictors in each bootstrap sample
  - boosting grows trees based on residuals from previous stage
  - bagging and boosting are general methods (not just for trees).

3.1 Regression Trees

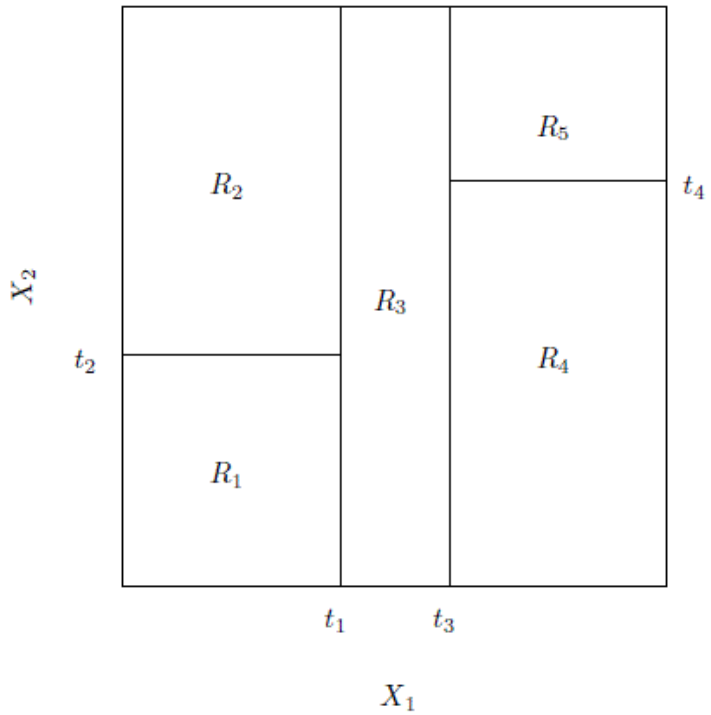
- Regression trees
  - sequentially split $\mathbf{x}'s$ into rectangular regions in way that reduces RSS
  - then $\widehat{y}_i$ is the average of $y's$ in the region that $\mathbf{x}_i$ falls in
  - with J blocks RSS$= \sum_{j=1}^{J} \sum_{i \in R_j} (y_i - \bar{y}_{R_j})^2$.
- Need to determine both the regressor j to split and the split point s.
  - For any regressor j and split point s, define the pair of half-planes $R1(j,s) = \{X | X_j < s\}$ and $R2(j,s) = \{X | X_j \geq s\}$
  - Find the value of j and s that minimize

$$\sum_{i:\mathbf{x}_i \in R1(j,s)} (y_i - \bar{y}_{R1})^2 + \sum_{i:\mathbf{x}_i \in R1(j,s)} (y_i - \bar{y}_{R1})^2$$

    where $\bar{y}_{R1}$ is the mean of y in region $R1$ (and similar for $R2$).
  - Once this first split is found, split both $R1$ and $R2$ and repeat
  - Each split is the one that reduces RSS the most.
  - Stop when e.g. less than five observations in each region.

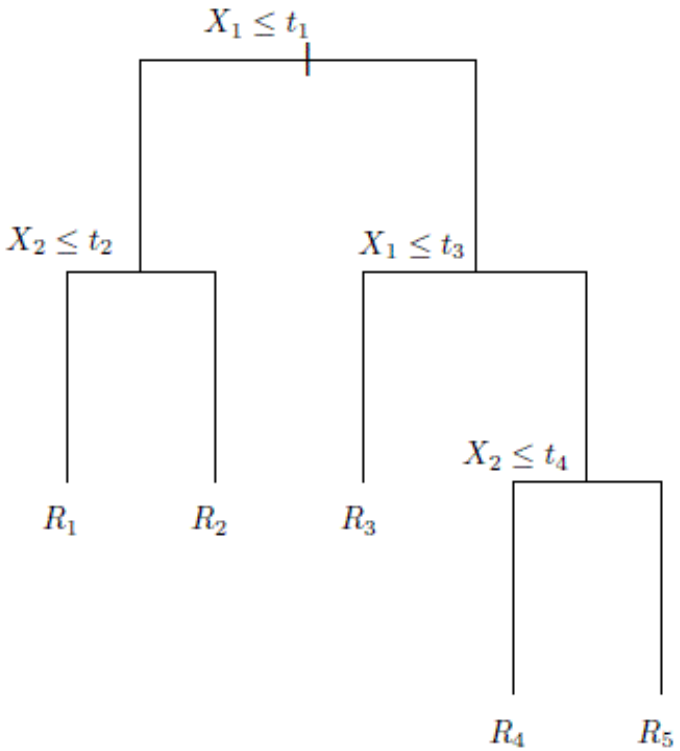
Tree example from ISL page 308

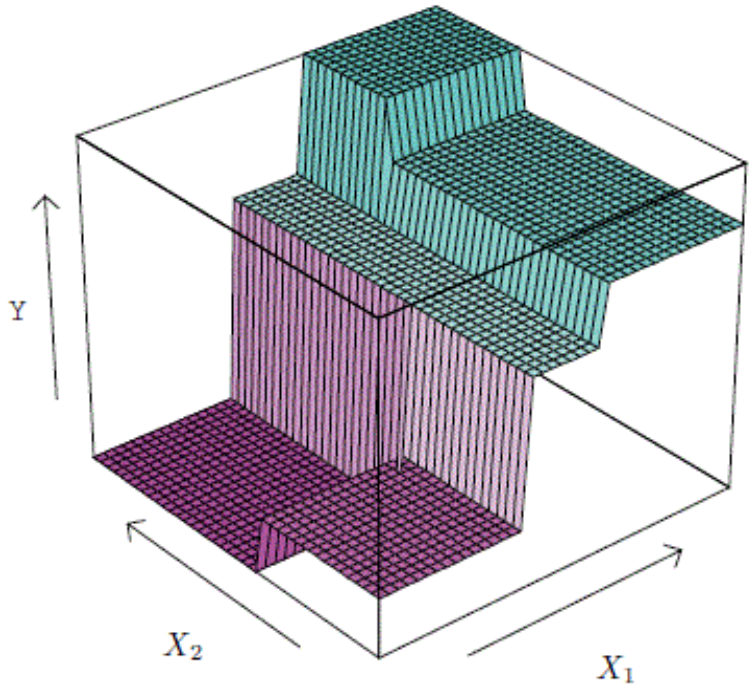
- (1) split X1 in two; (2) split the lowest X1 values on the basis of X2 into R1 and R2; (3) split the highest X1 values into two regions (R3 and R4/R5); (4) split the highest X1 values on the basis of X2 into R4 and R5.



Tree example from ISL (continued)

- The left figure gives the tree.
- The right figure shows the predicted values of y.



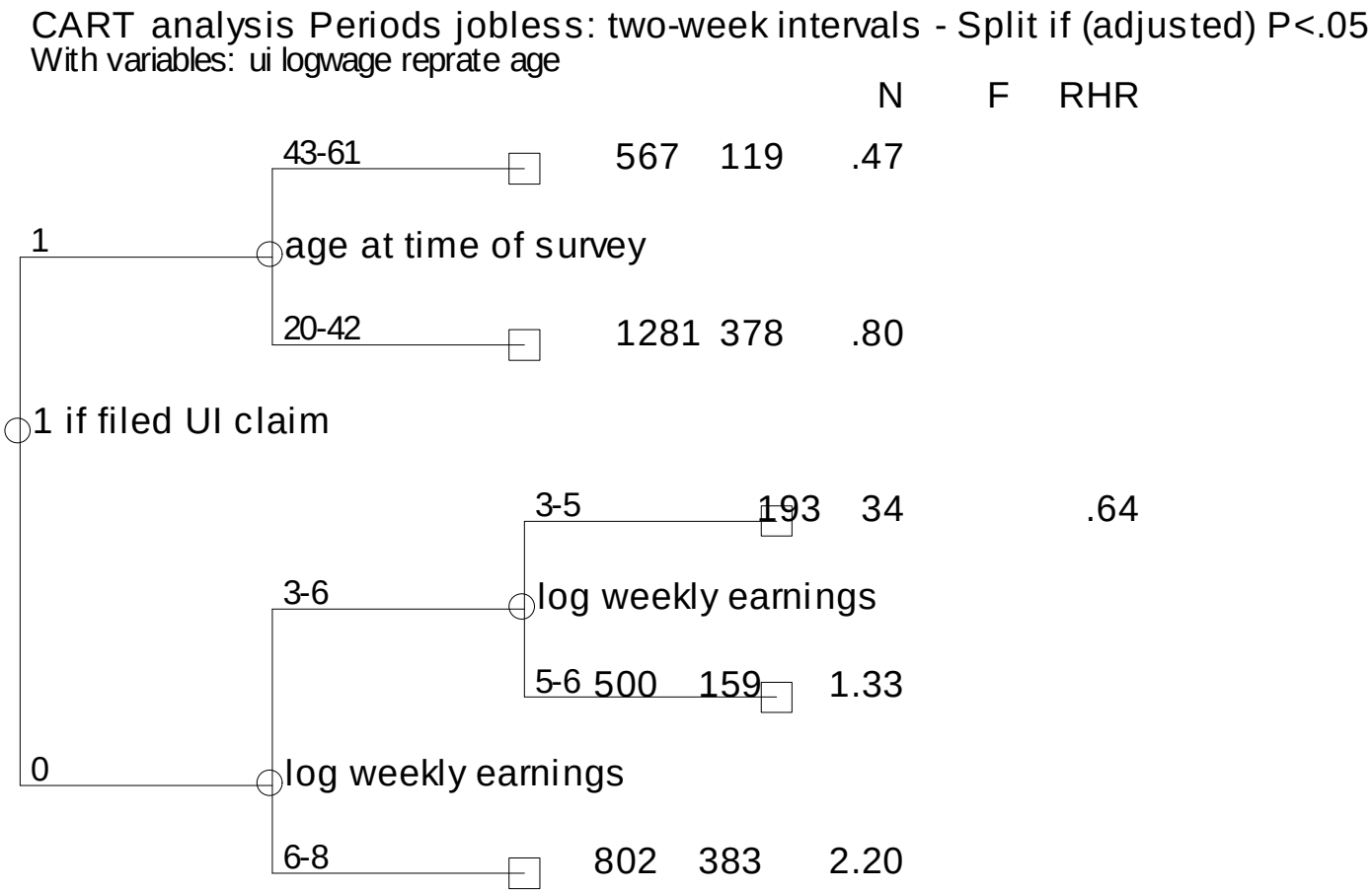


Regression tree (continued)

- The model is of form $f(X) = \sum_{j=1}^{J} c_m \times \mathbf{1}[X \in R_j]$.
- The approach is a topdown greedy approach
  - top down as start with top of the tree
  - **greedy** as at each step the best split is made at that particular step, rather than looking ahead and picking a split that will lead to a better tree in some future step.
- This leads to overfitting, so prune
  - use cost complexity pruning (or weakest link pruning)
  - this penalizes for having too many terminal nodes
  - see ISL equation (8.4).

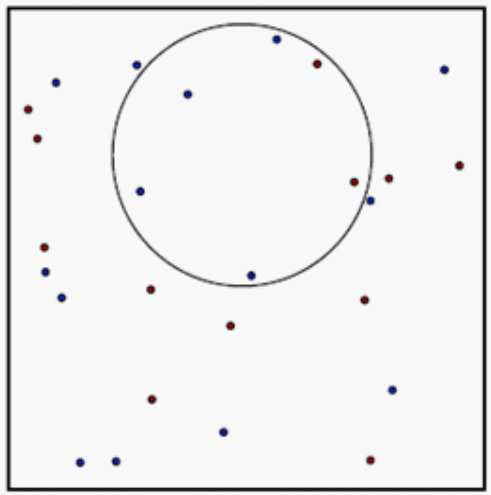
Regression tree example

- The only regression tree add-on to Stata I could find was `cart`
  - for duration data that determined tree using statistical significance.
  - I used it just to illustrate what a tree looks like.

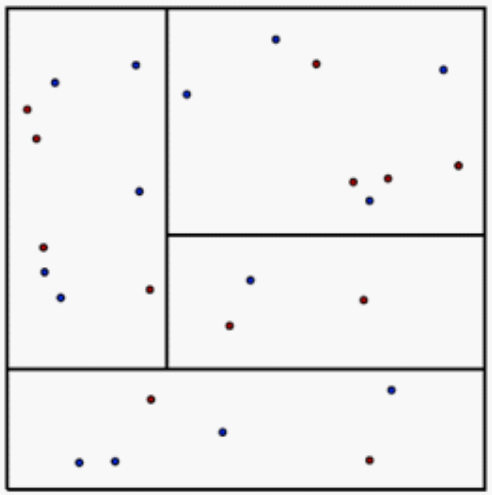


Tree as alternative to k-NN or kernel regression

- Figure from Athey and Imbens (2019), "Machine Learning Methods Economists should Know About"
  - axes are x_1 and x_2
  - note that tree used explanation of y in determining neighbors
  - tree may not do so well near boundaries of region
    - ★ random forests form many trees so not always at boundary.



Euclidean neighborhood, for KNN matching.



Tree-based neighborhood.

Improvements to regression trees

- Regression trees are easy to understand if there are few regressors.
- But they do not predict as well as methods given so far
  - due to high variance (e.g. split data in two then can get quite different trees).
- Better methods are given next
  - bagging
    - ★ bootstrap aggregating averages regression trees over many samples
  - random forests
    - ★ averages regression trees over many sub-samples
  - boosting
    - ★ trees build on preceding trees.

3.2 Bagging (Bootstrap Aggregating)

- Bagging is a general method for improving prediction that works especially well for regression trees.
- Idea is that averaging reduces variance.
- So average regression trees over many samples
  - the different samples are obtained by bootstrap resample with replacement (so not completely independent of each other)
  - for each sample obtain a large tree and prediction $\widehat{f}_b(x)$.
  - average all these predictions: $\widehat{f}_{\text{bag}}(x) = \frac{1}{B}\sum_{b=1}^{B}\widehat{f}_b(x)$.
- Get test sample error by using out-of-bag (OOB) observations not in the bootstrap sample
  - $\Pr[i^{th} \text{ obs not in resample}] = (1-\frac{1}{n})^n \rightarrow e^{-1} = 0.368 \simeq 1/3$.
  - this replaces cross validation.
- Interpretation of trees is now difficult so
  - record the total amount that RSS is decreased due to splits over a given predictor, averaged over all B trees.
  - a large value indicates an important predictor.

3.3 Random Forests

- The B bagging estimates are correlated
  - e.g. if a regressor is important it will appear near the top of the tree in each bootstrap sample.
  - the trees look similar from one resample to the next.
- Random forests get bootstrap resamples (like bagging)
  - but within each bootstrap sample use only a random sample of $m < p$ predictors in deciding each split.
  - usually $m \simeq \sqrt{p}$
  - this reduces correlation across bootstrap resamples.
- Simple bagging is random forest with $m = p$.

Random Forests (continued)

- Random forests are related to kernel and k-nearest neighbors
  - as use a weighted average of nearby observations
  - but with a data-driven way of determining which nearby observations get weight
  - see Lin and Jeon (JASA, 2006).
- Susan Athey and coauthors are big on random forests.

Random Forests example: data

```
. * Data for 65-90 year olds on supplementary insurance indicator and regressors
. use mus203mepsmedexp.dta, clear

. drop if ltotexp == .
(109 observations deleted)

. global zlist suppins phylim actlim totchr age female income

. describe ltotexp $zlist

              storage   display    value
variable name   type    format     label      variable label
------------------------------------------------------------------------------
ltotexp         float   %9.0g                 ln(totexp) if totexp > 0
suppins         float   %9.0g                 =1 if has supp priv insurance
phylim          double  %12.0g                =1 if has functional limitation
actlim          double  %12.0g                =1 if has activity limitation
totchr          double  %12.0g                # of chronic problems
age             double  %12.0g                Age
female          double  %12.0g                =1 if female
income          double  %12.0g                annual household income/1000

. summarize ltotexp $zlist, sep(0)

    Variable |        Obs        Mean    Std. Dev.       Min        Max
-------------+---------------------------------------------------------
     ltotexp |      2,955    8.059866    1.367592   1.098612   11.74094
     suppins |      2,955    .5915398    .4916322          0          1
      phylim |      2,955    .4362098    .4959981          0          1
      actlim |      2,955    .2879865    .4529014          0          1
      totchr |      2,955    1.808799    1.294613          0          7
         age |      2,955    74.24535    6.375975         65         90
      female |      2,955    .5840948    .4929608          0          1
      income |      2,955    22.68353    22.60988         -1     312.46
```

Random Forests example: OLS estimates

- Most important are suppins, actlim, totchr and phylim

```
. regress ltotexp $zlist, vce(robust)

Linear regression                               Number of obs     =      2,955
                                                F(7, 2947)        =     126.97
                                                Prob > F          =     0.0000
                                                R-squared         =     0.2289
                                                Root MSE          =     1.2023
```

ltotexp	Coef.	Robust Std. Err.	t	P>|t|	[95% Conf.	Interval]
suppins	.2556428	.0465982	5.49	0.000	.1642744	.3470112
phylim	.3020598	.057705	5.23	0.000	.1889136	.415206
actlim	.3560054	.0634066	5.61	0.000	.2316797	.4803311
totchr	.3758201	.0187185	20.08	0.000	.3391175	.4125228
age	.0038016	.0037028	1.03	0.305	-.0034587	.011062
female	-.0843275	.045654	-1.85	0.065	-.1738444	.0051894
income	.0025498	.0010468	2.44	0.015	.0004973	.0046023
_cons	6.703737	.2825751	23.72	0.000	6.149673	7.257802

Random Forests example: random forest estimation

```
. * Random forests using user written randomforest command
. randomforest ltotexp $zlist, type(reg) iter(500) depth(10) ///
>    lsize(5) seed(10101)

. ereturn list

scalars:
       e(Observations) =  2955
           e(features) =  7
         e(Iterations) =  500
          e(OOB_Error) =  .9452256910574954

macros:
                e(cmd) : "randomforest"
            e(predict) : "randomforest_predict"
             e(depvar) : "ltotexp"
         e(model_type) : "random forest regression"

matrices:
         e(importance) :  7 x 1
```

Random Forests example (continued)

```
. * Compute expected values of dep. var.: this also creates e(MAE) and e(RMSE)
. predict yh_rf

. ereturn list

scalars:
       e(Observations) =  2955
           e(features) =  7
         e(Iterations) =  500
          e(OOB_Error) =  .9452256910574954
                e(MAE) =  .7557299298029454
               e(RMSE) =  .9662698028945919

macros:
                e(cmd) : "randomforest"
            e(predict) : "randomforest_predict"
             e(depvar) : "ltotexp"
         e(model_type) : "random forest regression"

matrices:
         e(importance) :  7 x 1
```

Random Forests example: importance

- Most important are `actlim`, `totchr` and `phylim`

```
. * Random forests importance of variables
. matrix list e(importance)

e(importance)[7,1]
         Variable I~e
suppins     .26072259
 phylim     .90198178
 actlim             1
 totchr     .98353393
    age     .29094411
 female     .13192694
 income     .38782944
```

3.4 Boosting

- Boosting is also a general method for improving prediction.
- Regression trees use a greedy algorithm.
- Boosting uses a slower algorithm to generate a sequence of trees
  - each tree is grown using information from previously grown trees
  - and is fit on a modified version of the original data set
  - boosting does not involve bootstrap sampling.
- Specifically (with λ a penalty parameter)
  - given current model b fit a decision tree to model $b's$ residuals (rather than the outcome Y)
  - then update $\widehat{f}(x) =$ previous $\widehat{f}(x) + \lambda \widehat{f}^b(x)$
  - then update the residuals $r_i =$ previous $r_i - \lambda \widehat{f}^b(x_i)$
  - the boosted model is $\widehat{f}(x) = \sum_{b=1}^{B} \lambda \widehat{f}^b(x_i)$.
- Stata add-on `boost` includes file `boost64.dll` that needs to be manually copied into `c:\ado\plus`

Boosting example

- Most important are `totchr` and `phylim`

```
. * Boosting using user-written boost command
. set seed 10101

. capture program boost_plugin, plugin using("C:\ado\personal\boost64.dll")

. boost ltotexp $zlist, influence distribution(normal) trainfraction(0.8) ///
>    maxiter(1000) predict(yh_boost)
influence
Distribution=normal
predict=yh_boost
Trainfraction=.8 Shrink=.01 Bag=.5 maxiter=1000 Interaction=5
Fitting ...
Assessing Influence ...
Predicting ...
bestiter= 862
Test R2= .23428402
trainn= 2364
Train R2= .3122529
Influence of each variable (Percent):
5.7794352 suppins
2.43954 phylim
3.2075646 actlim
36.686562 totchr
11.346784 age
1.7692824 female
38.770832 income
```

Comparison of in-sample predictions

```
. * Compare various predictions in sample
. quietly regress ltotexp $zlist

. predict yh_ols
(option xb assumed; fitted values)

. summarize ltotexp yh*
```

Variable	Obs	Mean	Std. Dev.	Min	Max
ltotexp	2,955	8.059866	1.367592	1.098612	11.74094
yh_rf	2,955	8.060393	.7232039	5.29125	10.39143
yh_rf_half	2,955	8.053512	.7061742	5.540289	9.797389
yh_boost	2,955	7.667966	.4974654	5.045572	8.571422
yh_ols	2,955	8.059866	.654323	6.866516	10.53811

```
. correlate ltotexp yh*
(obs=2,955)
```

	ltotexp	yh_rf	yh_rf_~f	yh_boost	yh_ols
ltotexp	1.0000				
yh_rf	0.7377	1.0000			
yh_rf_half	0.6178	0.9212	1.0000		
yh_boost	0.5423	0.8769	0.8381	1.0000	
yh_ols	0.4784	0.8615	0.8580	0.8666	1.0000

4. Classification: Overview

- $y's$ are now categorical
  - example: binary if two categories.
- Interest lies in predicting y using $\widehat{y}$ (classification)
  - whereas economists usually want $\widehat{\Pr}[y = j|\mathbf{x}]$
- Use (0,1) loss function rather than MSE or ln L
  - 0 if correct classification
  - 1 if misclassified.
- Many machine learning applications are in settings where can classify well
  - e.g. reading car license plates
  - unlike many economics applications.

4. Classification: Overview (continued)

- Regression methods predict probabilities
  - logistic regression, multinomial regression, k-nearest neighbors
  - assign to class with the highest predicted probability (Bayes classifier)
    - ★ in binary case $\widehat{y} = 1$ if $\widehat{p} \geq 0.5$ and $\widehat{y} = 0$ if $\widehat{p} < 0.5$.
- Discriminant analysis additionally assumes a normal distribution for the x's
  - use Bayes theorem to get $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$.
- Support vector classifiers and support vector machines
  - directly classify (no probabilities)
  - are more nonlinear so may classify better
  - use separating hyperplanes of X and extensions.

4.1 A Different Loss Function: Error Rate

- Instead of MSE we use the **error rate**
  - the number of misclassifications

$$\text{Error rate} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[y_i \neq \widehat{y}_i],$$

  - ★ where for K categories $y_i = 0, ..., K-1$ and $\widehat{y}_i = 0, ..., K-1$.
  - ★ and indicator $\mathbf{1}[A] = 1$ if event A happens and $= 0$ otherwise.

- The **test error rate** is for the n_0 observations in the test sample

$$\text{Ave}(\mathbf{1}[y_0 \neq \widehat{y}_0]) = \frac{1}{n_0}\sum_{i=1}^{n_0} \mathbf{1}[y_{0i} \neq \widehat{y}_{0i}].$$

- Cross validation uses number of misclassified observations. e.g. LOOCV is

$$\text{CV}_{(n)} = \frac{1}{n}\sum_{i=1}^{n} Err_i = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}[y_i \neq \widehat{y}_{(-i)}].$$

Classification Table

- A classification table or confusion matrix is a $K \times K$ table of counts of $(y, \widehat{y})$
- In 2×2 case with binary $y = 1$ or 0
  - sensitivity is % of $y = 1$ with prediction $\widehat{y} = 1$
  - specificity is % of $y = 0$ with prediction $\widehat{y} = 0$
  - receiver operator characteristics curve (ROC) curve plots sensitivity against 1−sensitivity as threshold for $\widehat{y} = 1$ changes.

Bayes classifier

- The Bayes classifier selects the most probable class
  - the following gives theoretical justification.
- $L(G, \widehat{G}(\mathbf{x})) = \mathbf{1}[y_i \neq \widehat{y}_i]$
  - $L(G, \widehat{G}(\mathbf{x}))$ is 0 on diagonal of $K \times K$ table and 1 elsewhere
  - where G is actual categories and $\widehat{G}$ is predicted categories.
- Then minimize the expected prediction error

$$\begin{aligned} EPE &= E_{G,\mathbf{x}}[L(G, \widehat{G}(\mathbf{x}))] \\ &= E_{\mathbf{x}}\left[\sum_{k=1}^{K} L(G, \widehat{G}(\mathbf{x})) \times \Pr[G_k|\mathbf{x}]\right] \end{aligned}$$

- Minimize EPE pointwise

$$\begin{aligned} f(x) &= \arg\min_{g \in G}\left[\sum_{k=1}^{K} L(G_k, g) \times \Pr[G_k|\mathbf{x}]\right] \\ \partial/\partial c &= \arg\min_{g \in G}[1 - \Pr[g|\mathbf{x}]] \\ &= \max_{g \in G} \Pr[g|\mathbf{x}] \end{aligned}$$

- So select the most probable class.

4.2 Logit

- Directly model $p(\mathbf{x}) = \Pr[y|\mathbf{x}]$.
- Logistic (logit) regression for binary case obtains MLE for

$$\ln\left(\frac{p(\mathbf{x})}{1-p(\mathbf{x})}\right) = \mathbf{x}'\boldsymbol{\beta}.$$

- Statisticians implement using a statistical package for the class of generalized linear models (GLM)
  - logit is in the Bernoulli (or binomial) family with logistic link
  - logit is often the default.
- Logit model is a linear (in $\mathbf{x}$) classifier
  - $\widehat{y} = 1$ if $\widehat{p}(\mathbf{x}) > 0.5$
  - i.e. if $\mathbf{x}'\widehat{\boldsymbol{\beta}} > 0$.

Logit Example

- Example considers supplementary health insurance for 65-90 year-olds.

```
. * Data for 65-90 year olds on supplementary insurance indicator and regressors
. use mus203mepsmedexp.dta, clear

. global xlist income educyr age female white hisp marry ///
>    totchr phylim actlim hvgg

. describe suppins $xlist
```

variable name	storage type	display format	value label	variable label
suppins	float	%9.0g		=1 if has supp priv insurance
income	double	%12.0g		annual household income/1000
educyr	double	%12.0g		Years of education
age	double	%12.0g		Age
female	double	%12.0g		=1 if female
white	double	%12.0g		=1 if white
hisp	double	%12.0g		=1 if Hispanic
marry	double	%12.0g		=1 if married
totchr	double	%12.0g		# of chronic problems
phylim	double	%12.0g		=1 if has functional limitation
actlim	double	%12.0g		=1 if has activity limitation
hvgg	float	%9.0g		=1 if health status is excellent, good or very good

Logit Example (continued)

- Summary statistics

```
. * Summary statistics
. summarize suppins $xlist
```

Variable	Obs	Mean	Std. Dev.	Min	Max
suppins	3,064	.5812663	.4934321	0	1
income	3,064	22.47472	22.53491	-1	312.46
educyr	3,064	11.77546	3.435878	0	17
age	3,064	74.17167	6.372938	65	90
female	3,064	.5796345	.4936982	0	1
white	3,064	.9742167	.1585141	0	1
hisp	3,064	.0848564	.2787134	0	1
marry	3,064	.5558094	.4969567	0	1
totchr	3,064	1.754243	1.307197	0	7
phylim	3,064	.4255875	.4945125	0	1
actlim	3,064	.2836162	.4508263	0	1
hvgg	3,064	.6054178	.4888406	0	1

Logit Example

- Logit model estimates

```
. * logit model
. logit suppins $xlist, nolog

Logistic regression                             Number of obs     =      3,064
                                                LR chi2(11)       =     345.23
                                                Prob > chi2       =     0.0000
Log likelihood = -1910.5353                     Pseudo R2         =     0.0829
```

suppins	Coef.	Std. Err.	z	P>|z|	[95% Conf.	Interval]
income	.0180677	.0025194	7.17	0.000	.0131298	.0230056
educyr	.0776402	.0131951	5.88	0.000	.0517782	.1035022
age	-.0265837	.006569	-4.05	0.000	-.0394586	-.0137088
female	-.0946782	.0842343	-1.12	0.261	-.2597744	.070418
white	.7438788	.2441096	3.05	0.002	.2654327	1.222325
hisp	-.9319462	.1545418	-6.03	0.000	-1.234843	-.6290498
marry	.3739621	.0859813	4.35	0.000	.205442	.5424823
totchr	.0981018	.0321459	3.05	0.002	.0350971	.1611065
phylim	.2318278	.1021466	2.27	0.023	.0316242	.4320315
actlim	-.1836227	.1102917	-1.66	0.096	-.3997904	.0325449
hvgg	.17946	.0811102	2.21	0.027	.0204868	.3384331
_cons	-.1028233	.577563	-0.18	0.859	-1.234826	1.029179

Logit Example (continued)

- Classification table

```
. * Classification table
. estat classification

Logistic model for suppins
```

Classified	True D	True ~D	Total
+	1434	737	2171
-	347	546	893
Total	1781	1283	3064

```
Classified + if predicted Pr(D) >= .5
True D defined as suppins != 0
```

Sensitivity	Pr(+| D)	80.52%
Specificity	Pr(-|~D)	42.56%
Positive predictive value	Pr(D| +)	66.05%
Negative predictive value	Pr(~D| -)	61.14%
False + rate for true ~D	Pr(+|~D)	57.44%
False - rate for true D	Pr(-| D)	19.48%
False + rate for classified +	Pr(~D| +)	33.95%
False - rate for classified -	Pr(D| -)	38.86%
Correctly classified		64.62%

Logit Example (continued)

- Classification table manually
  - error rate $= (737 + 347)/3064 = 0.354$

```
. * Classification table manually
. predict ph_logit
(option pr assumed; Pr(suppins))

. generate yh_logit = ph_logit >= 0.5

. generate err_logit = (suppins==0 & yh_logit==1) | (suppins==1 & yh_logit==0)

. summarize suppins ph_logit yh_logit err_logit
```

Variable	Obs	Mean	Std. Dev.	Min	Max
suppins	3,064	.5812663	.4934321	0	1
ph_logit	3,064	.5812663	.1609388	.0900691	.9954118
yh_logit	3,064	.7085509	.4545041	0	1
err_logit	3,064	.3537859	.4782218	0	1

```
. tabulate suppins yh_logit
```

=1 if has supp priv insurance	yh_logit 0	1	Total
0	546	737	1,283
1	347	1,434	1,781
Total	893	2,171	3,064

4.3 k-nearest neighbors

- k-nearest neighbors (K-NN) for many classes
  - $\Pr[Y = j|\mathbf{x} = \mathbf{x}_0] = \frac{1}{K}\sum_{i \in N_0} \mathbf{1}[y_i = j]$
  - where N_0 is the K observations on $\mathbf{x}$ closest to $\mathbf{x}_0$.
- There are many measures of closeness
  - default is Euclidean distance between observations i and j

$$\left\{\sum_{a=1}^{p}(x_{ai} - x_{ja})^2\right\}^{1/2} \text{ where there are } p \text{ regressors}$$

- Obtain predicted probabilities
  - then assign to the class with highest predicted probability.

k-nearest neighbors example

- Here use Euclidean distance and set $K = 11$

```
. * K-nearest neighbors
. discrim knn $xlist, group(suppins) k(11) notable

Kth-nearest-neighbor discriminant analysis

. predict yh_knn
(option classification assumed; group classification)

. estat classtable, nototals nopercents looclass

Leave-one-out classification table
```

Key
Number

True suppins	LOO Classified 0	1
0	759	524
1	711	1,070
Priors	0.5000	0.5000

k-nearest neighbors example (continued)

- Classification not as good if use leave-one-out cross validation

much better if don't use LOOCV

```
. * K-nn classification table with leave-one out cross validation not as good
. estat classtable, nototals nopercents  // without LOOCV

Resubstitution classification table
```

Key
Number

True suppins	Classified 0	1
0	889	394
1	584	1,197
Priors	0.5000	0.5000

4.4 Linear Discriminant Analysis

- Developed for classification problems such as is a skull Neanderthal or Homo Sapiens given various measures of the skull.
- Discriminant analysis specifies a joint distribution for $(Y, \mathbf{X})$.
- Linear discriminant analysis with K categories
  - assume $\mathbf{X}|Y = k$ is $N(\boldsymbol{\mu}_k, \Sigma)$ with density $f_k(\mathbf{x}) = \Pr[\mathbf{X} = \mathbf{x}|Y = k]$
  - and let $\pi_k = \Pr[Y = k]$
- The desired $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$ is obtained using Bayes theorem

$$\Pr[Y = k|\mathbf{X} = \mathbf{x}] = \frac{\pi_k f_k(\mathbf{x})}{\sum_{j=1}^{K} \pi_j f_j(\mathbf{x})}.$$

- Assign observation $\mathbf{X} = \mathbf{x}$ to class k with largest $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$.

Linear Discriminant Analysis (continued)

- Upon simplification assignment to class k with largest $\Pr[Y = k|\mathbf{X} = \mathbf{x}]$ is equivalent to choosing model with largest **discriminant function**

$$\delta_k(\mathbf{x}) = \mathbf{x}'\Sigma^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k{}'\Sigma^{-1}\boldsymbol{\mu}_k + \ln \pi_k$$

  - use $\widehat{\boldsymbol{\mu}}_k = \bar{\mathbf{x}}_k$, $\widehat{\Sigma} = \widehat{\text{Var}}[\mathbf{x}_k]$ and $\widehat{\pi}_k = \frac{1}{N}\sum_{i=1}^{N} \mathbf{1}[y_i = k]$.
- Called linear discriminant analysis as $\delta_k(\mathbf{x})$ linear in $\mathbf{x}$.

Linear Discriminant Analysis Example

- We have

```
. * Linear discriminant analysis
. discrim lda $xlist, group(suppins) notable

. predict yh_lda
(option classification assumed; group classification)

. estat classtable, nototals nopercents

Resubstitution classification table
```

Key
Number

True suppins	Classified 0	1
0	770	513
1	638	1,143
Priors	0.5000	0.5000

Quadratic Discriminant Analysis

- Quadratic discriminant analysis
  - now allow different variances so $\mathbf{X}|Y = k$ is $N(\boldsymbol{\mu}_k, \Sigma_k)$
- Upon simplification, the Bayes classifier assigns observation $\mathbf{X} = \mathbf{x}$ to class k which has largest

$$\delta_k(\mathbf{x}) = -\frac{1}{2}\mathbf{x}'\Sigma_k^{-1}\mathbf{x} + \mathbf{x}'\Sigma_k^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\boldsymbol{\mu}_k{}'\Sigma_k^{-1}\boldsymbol{\mu}_k - \frac{1}{2}\ln|\Sigma_k| + \ln\pi_k$$

  - called quadratic discriminant analysis as linear in $\mathbf{x}$
- Use rather than LDA only if have a lot of data as requires estimating many parameters.

Quadratic Discriminant Analysis Example

- We have

```
. * Quadratic discriminant analysis
. discrim qda $xlist, group(suppins) notable

. predict yh_qda
(option classification assumed; group classification)

. estat classtable, nototals nopercents

Resubstitution classification table
```

Key
Number

True suppins	Classified 0	1
0	468	815
1	292	1,489
Priors	0.5000	0.5000

LDA versus Logit

- ESL ch.4.4.5 compares linear discriminant analysis and logit
  - Both have log odds ratio linear in X
  - LDA is joint model if Y and X versus logit is model of Y conditional on X.
  - In the worst case logit ignoring marginal distribution of X has a loss of efficiency of about 30% asymptotically in the error rate.
  - If $X's$ are nonnormal (e.g. categorical) then LDA still doesn't do too bad.

ISL Figure 4.9: Linear and Quadratic Boundaries

- LDA uses a linear boundary to classify and QDA a quadratic

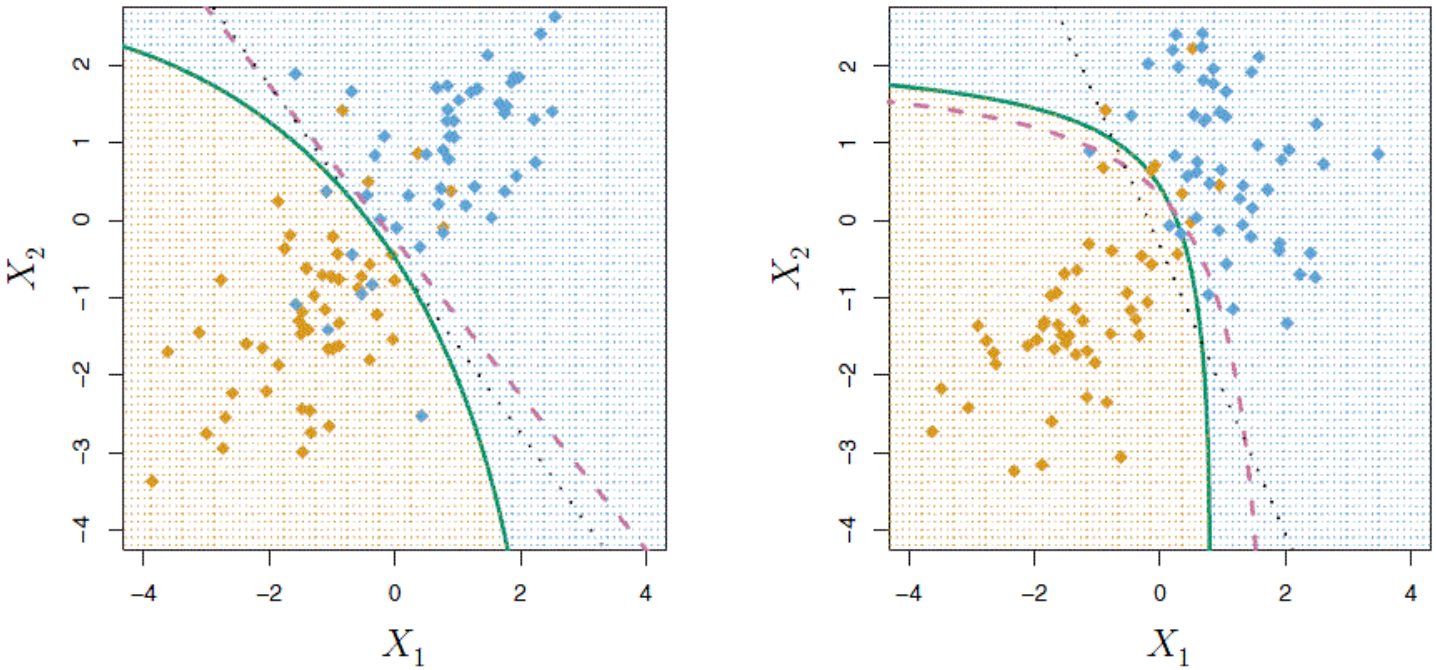


FIGURE 4.9. Left: *The Bayes (purple dashed), LDA (black dotted), and QDA (green solid) decision boundaries for a two-class problem with* $\boldsymbol{\Sigma}_1 = \boldsymbol{\Sigma}_2$. *The shading indicates the QDA decision rule. Since the Bayes decision boundary is linear, it is more accurately approximated by LDA than by QDA.* Right: *Details are as given in the left-hand panel, except that* $\boldsymbol{\Sigma}_1 \neq \boldsymbol{\Sigma}_2$. *Since the Bayes decision boundary is non-linear, it is more accurately approximated by QDA than by LDA.*

4.5 Support Vector Classifier

- Build on LDA idea of linear boundary to classify when $K = 2$.
- Maximal margin classifier
  - classify using a separating hyperplane (linear combination of X)
  - if perfect classification is possible then there are an infinite number of such hyperplanes
  - so use the separating hyperplane that is furthest from the training observations
  - this distance is called the maximal margin.
- Support vector classifier
  - generalize maximal margin classifier to the nonseparable case
  - this adds slack variables to allow some y's to be on the wrong side of the margin
  - $\text{Max}_{\boldsymbol{\beta},\varepsilon} M$ (the margin - distance from separator to training X's) subject to $\boldsymbol{\beta}'\boldsymbol{\beta} \neq \mathbf{1}$, $y_i(\beta_0 + \mathbf{x}_i'\boldsymbol{\beta}) \geq M(1 - \varepsilon_i)$, $\varepsilon_i \geq 0$ and $\sum_{i=1}^{n} \varepsilon_i \leq C$.

Support Vector Machines

- The support vector classifier has a linear boundary
  - $f(\mathbf{x}_0) = \beta_0 + \sum_{i=1}^{n} \alpha_i \mathbf{x}_0' \mathbf{x}_i$, where $\mathbf{x}_0' \mathbf{x}_i = \sum_{j=1}^{p} x_{0j} x_{ij}$.
- The support vector machine has nonlinear boundaries
  - $f(\mathbf{x}_0) = \beta_0 + \sum_{i=1}^{n} \alpha_i K(\mathbf{x}_0, \mathbf{x}_i)$ where $K(\cdot)$ is a kernel
  - polynomial kernel $K(\mathbf{x}_0, \mathbf{x}_i) = (1 + \sum_{j=1}^{p} x_{0j} x_{ij})^d$
  - radial kernel $K(\mathbf{x}_0, \mathbf{x}_i) = \exp(-\gamma \sum_{j=1}^{p} (x_{0j} - x_{ij})^2)$
- Can extend to $K > 2$ classes (see ISL ch. 9.4).
  - one-versus-one or all-pairs approach
  - one-versus-all approach.

ISL Figure 9.9: Support Vector Machine

- In this example a linear or quadratic classifier won't work whereas SVM does.

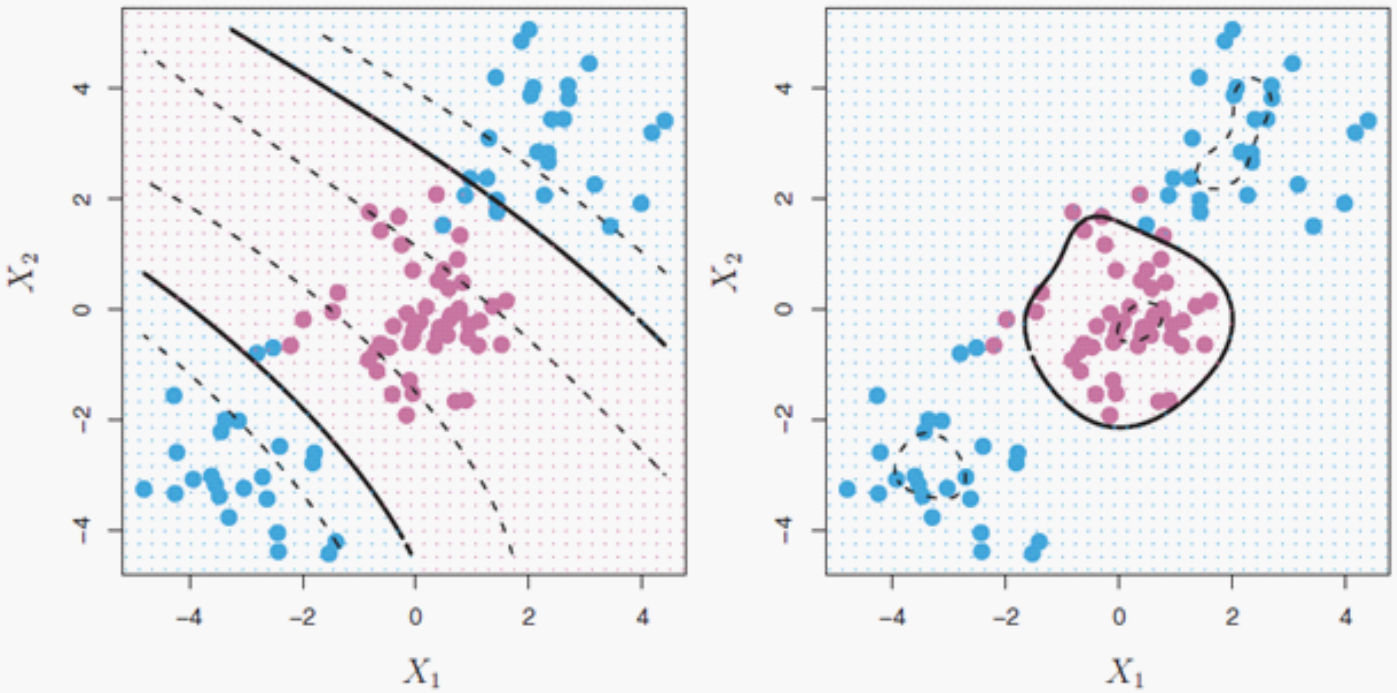


FIGURE 9.9. Left: *An SVM with a polynomial kernel of degree 3 is applied to the non-linear data from Figure 9.8, resulting in a far more appropriate decision rule.* Right: *An SVM with a radial kernel is applied. In this example, either kernel is capable of capturing the decision boundary.*

Support Vector Machines Example

- Use Stata add-on `svmachines` (Guenther and Schonlau)

```
. * Support vector machines - need y to be byte not float and matsize > n
. set matsize 3200

. global xlistshort income educyr age female marry totchr

. generate byte ins = suppins

. svmachines ins income

. svmachines ins $xlist

. predict yh_svm

. tabulate ins yh_svm
```

ins	yh_svm 0	1	Total
0	820	463	1,283
1	224	1,557	1,781
Total	1,044	2,020	3,064

Comparison of model predictions

- The following compares the various category predictions.
- SVM does best but we did in-sample predictions here
  - especially for SVM we should have training and test samples.

```
. * Compare various in-sample predictions
. correlate suppins yh_logit yh_knn yh_lda yh_qda yh_svm
(obs=3,064)
```

	suppins	yh_logit	yh_knn	yh_lda	yh_qda	yh_svm
suppins	1.0000					
yh_logit	0.2505	1.0000				
yh_knn	0.3604	0.3575	1.0000			
yh_lda	0.2395	0.6955	0.3776	1.0000		
yh_qda	0.2294	0.6926	0.2762	0.5850	1.0000	
yh_svm	0.5344	0.3966	0.6011	0.3941	0.3206	1.0000

- Regression trees, bagging, random forests and boosting can be used for categorical data.
  - user-written `boost` applies to Gaussian (normal), logistic and Poisson regression.
  - user-written `randomforest` applies to regression and classification.

5. Unsupervised Learning

- Challenging area: no y, only $\mathbf{x}$.
- Example is determining several types of individual based on responses to many psychological questions.
- Principal components analysis.
- Clustering Methods
  - k-means clustering.
  - hierarchical clustering.

5.1 Principal Components

- Initially discussed in section on dimension reduction.
- Goal is to find a few linear combinations of X that explain a good fraction of the total variance $\sum_{j=1}^{p} Var(X_j) = \sum_{j=1}^{p} \frac{1}{n} \sum_{i=1}^{n} x_{ij}^2$ for mean 0 X's.
- $Z_m = \sum_{j=1}^{p} \phi_{jm} X_j$ where $\sum_{j=1}^{p} \phi_{jm}^2 = 1$ and ϕ_{jm} are called factor loadings.
- A useful statistic is the proportion of variance explained (PVE)
  - a scree plot is a plot of PVE_m against m
  - and a plot of the cumulative PVE by m components against m.
  - choose m that explains a "sizable" amount of variance
  - ideally find interesting patterns with first few components.
- Easier when used PCA earlier in supervised learning as then observe Y and can treat m as a tuning parameter.
- Stata pca command.

5.2 Cluster Analysis: k-Means Clustering

- Goal is to find homogeneous subgroups among the X.
- K-Means splits into K distinct clusters where within cluster variation is minimized.
- Let $W(C_k)$ be measure of variation
  - Minimize$_{C_1,...,C_k} \sum_{k=1}^{K} W(C_k)$
  - Euclidean distance $W(C_k) = \frac{1}{n_k} \sum_{i,i' \in C_k} \sum_{j=1}^{p} (x_{ij} - x_{i'j})^2$
- Global maximum requires K^n partitions.
- Instead use algorithm 10.1 (ISL p.388) which finds a local optimum
  - run algorithm multiple times with different seeds
  - choose the optimum with smallest $\sum_{k=1}^{K} W(C_k)$.

ISL Figure 10.5

- Data is $(x_1.x_2)$ with $K = 2, 3$ and 4 clusters identified.

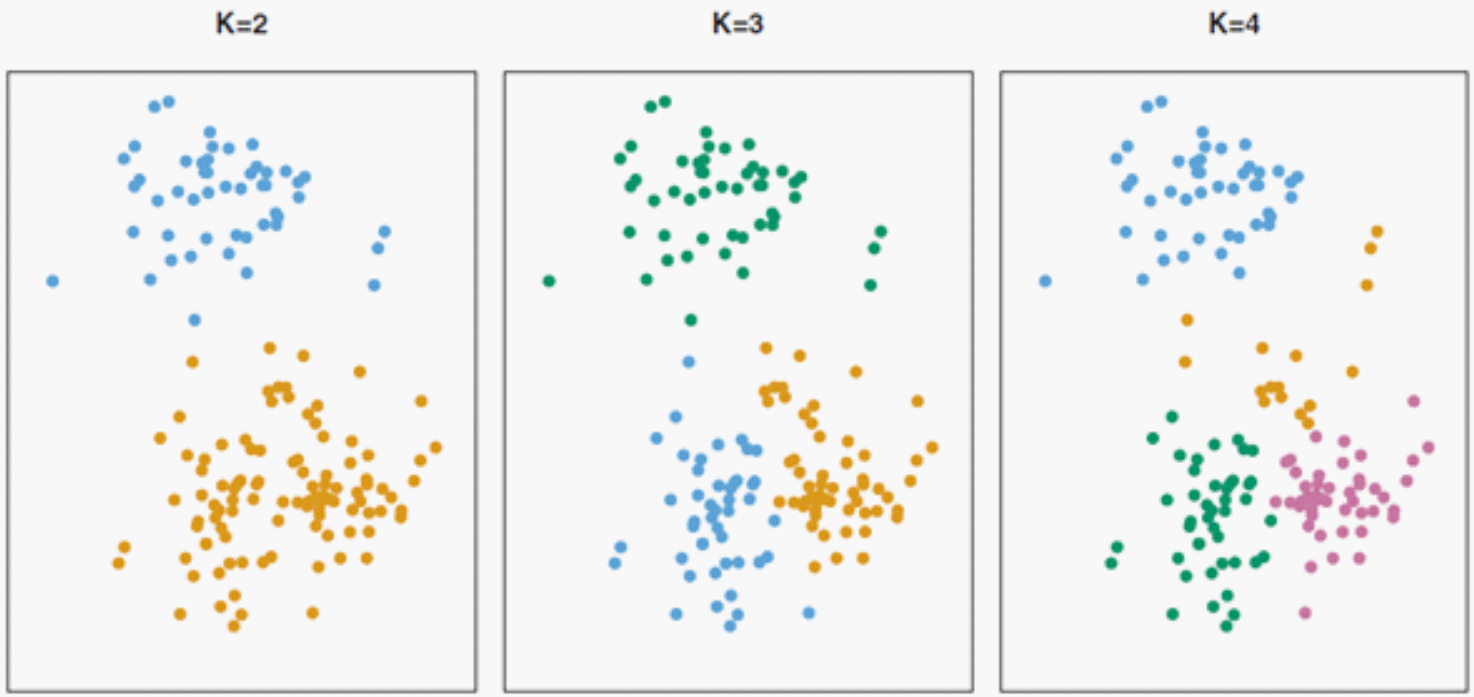


FIGURE 10.5. *A simulated data set with 150 observations in two-dimensional space. Panels show the results of applying K-means clustering with different values of K, the number of clusters. The color of each observation indicates the cluster to which it was assigned using the K-means clustering algorithm. Note that*

k-means clustering example

- Use same data as earlier principal components analysis example.

```
. * k-means clustering with defaults and three clusters
. use machlearn_part2_spline.dta, replace

. graph matrix x1 x2 z     // matrix plot of the three variables

. cluster kmeans x1 x2 z, k(3) name(myclusters)

. tabstat x1 x2 z, by(myclusters) stat(mean)

Summary statistics: mean
  by categories of: myclusters
```

myclusters	x1	x2	z
1	.8750554	.503166	1.34776
2	-.8569585	-1.120344	-.5772717
3	.1691631	.6720648	-.3493614
Total	.0301211	.0226274	.0664539

Hierarchical Clustering

- Do not specify K.
- Instead begin with n clusters (leaves) and combine clusters into branches up towards trunk
  - represented by a dendrogram
  - eyeball to decide number of clusters.
- Need a dissimilarity measure between clusters
  - four types of linkage: complete, average, single and centroid.
- For any clustering method
  - it is a difficult problem to do unsupervised learning
  - results can change a lot with small changes in method
  - clustering on subsets of the data can provide a sense of robustness.

6. Conclusions

- Guard against overfitting
  - use K-fold cross validation or penalty measures such as AIC.
- Biased estimators can be better predictors
  - shrinkage towards zero such as Ridge and LASSO.
- For flexible models popular choices are
  - neural nets
  - random forests.
- Though what method is best varies with the application
  - and best are ensemble forecasts that combine different methods.
- Machine learning methods can outperform nonparametric and semiparametric methods
  - so wherever econometricians use nonparametric and semiparametric regression in higher dimensional models it may be useful to use ML methods
  - though the underlying theory still relies on assumptions such as sparsity.

7. Some R Commands used in ISL

- Splines
  - regression splines: bs(x,knots=c()) in lm() function
  - natural spline: ns(x,knots=c()) in lm() function
  - smoothing spline: function smooth.spline() in spline library
- Local regression
  - loess: function loess
  - generalized additive models: function gam() in gam library
- Tree-based methods
  - classification tree: function tree() in tree library
  - cross-validation: cv.tree() function
  - pruning: function prune.tree()
  - random forest: randomForest() in randomForest library
  - bagging: function randomForest()
  - boosting: gbm() function in library gbm

Some R Commands (continued)

- Basic classification
  - logistic: glm function
  - discriminant analysis: lda() and qda functions in MASS library
  - k nearest neighbors: knn() function in class library
- Support vector machines
  - support vector classifier: svm(... kernel="linear") in e1071 library
  - support vector machine: svm(... kernel="polynomial") or svm(... kernel="radial") in e1071 library
  - receiver operator characteristic curve: rocplot in ROCR library.
- Unsupervised Learning
  - principal components analysis: function prcomp()
  - k-means clustering: function kmeans()
  - hierarchical clustering: function hclust()

8. References

- Undergraduate / Masters level book
  - **ISL:** Gareth James, Daniela Witten, Trevor Hastie and Robert Tibsharani (2013), *An Introduction to Statistical Learning: with Applications in R, Springer*.
  - free legal pdf at http://www-bcf.usc.edu/~gareth/ISL/
  - $25 hardcopy via http://www.springer.com/gp/products/books/mycopy
- Masters / PhD level book
  - **ESL:** Trevor Hastie, Robert Tibsharani and Jerome Friedman (2009), *The Elements of Statistical Learning: Data Mining, Inference and Prediction*, Springer.
  - free legal pdf at http://statweb.stanford.edu/~tibs/ElemStatLearn/index.html
  - $25 hardcopy via http://www.springer.com/gp/products/books/mycopy

References (continued)

- A recent book is
  - EH: Bradley Efron and Trevor Hastie (2016), *Computer Age Statistical Inference: Algorithms, Evidence and Data Science*, Cambridge University Press.
- Interesting book: Cathy O'Neil, Weapons of Math Destruction: How Big Data Increases Inequality and Threatens Democracy.
- My website has some material
  - http://cameron.econ.ucdavis.edu/e240f/machinelearning.html