Glmnet

One of the best algorithms in machine learning today.

Outline of Talk

- Background
  - Ordinary Least Squares Regression (and logistic regression)
  - Limitations of OLS
  - Introduce regularization – (coefficient shrinkage)
- Principles of Operation
- Examples of Results

Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

Glmnet – Background

modern linear regression – a lot has changed in 200 years

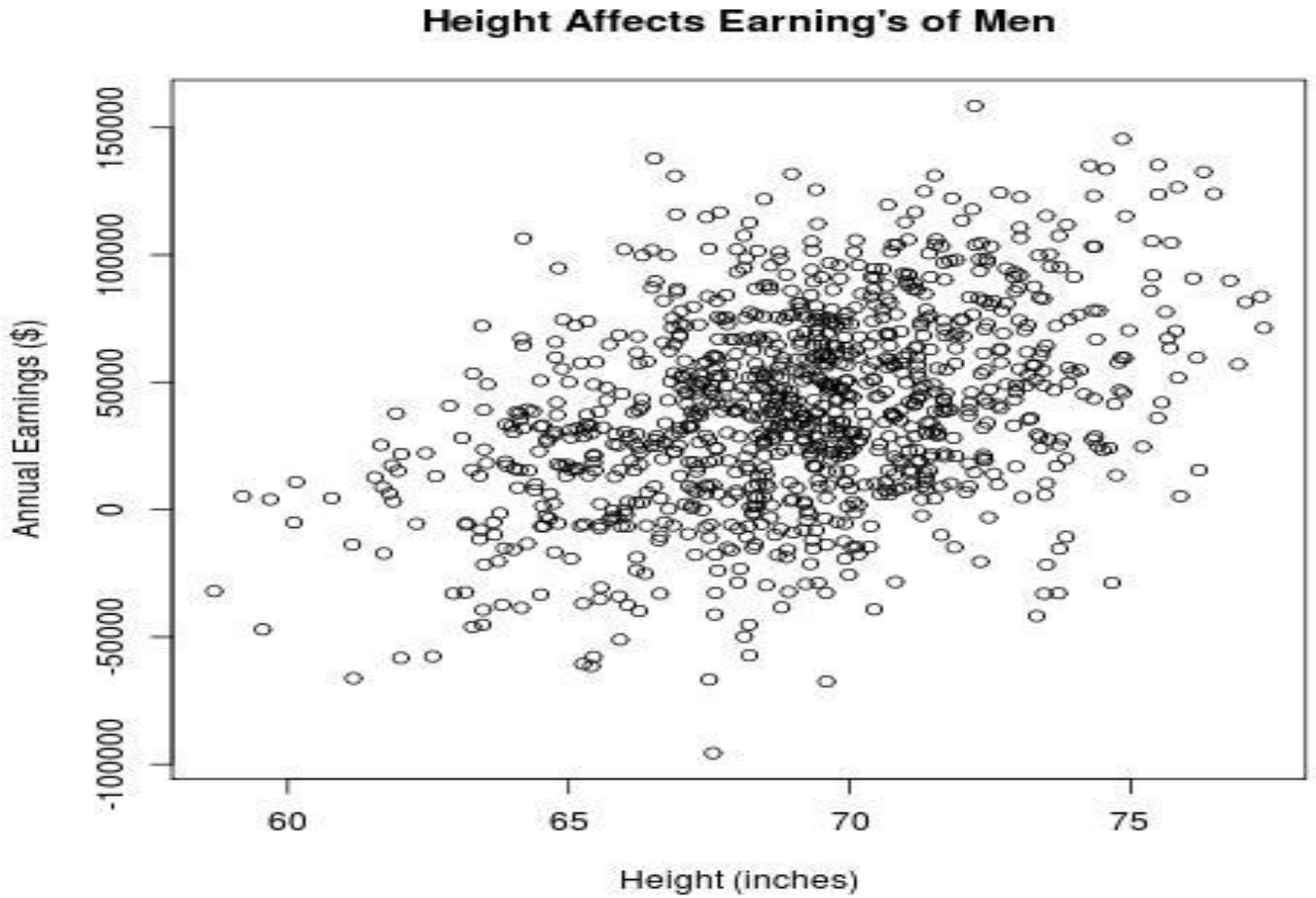
Ordinary Least Squares Regression – First described by Gauss 1794
Seeks to fit a straight line through data so as to minimize sum squared error.

Example: How do men's annual salaries depend on their height?

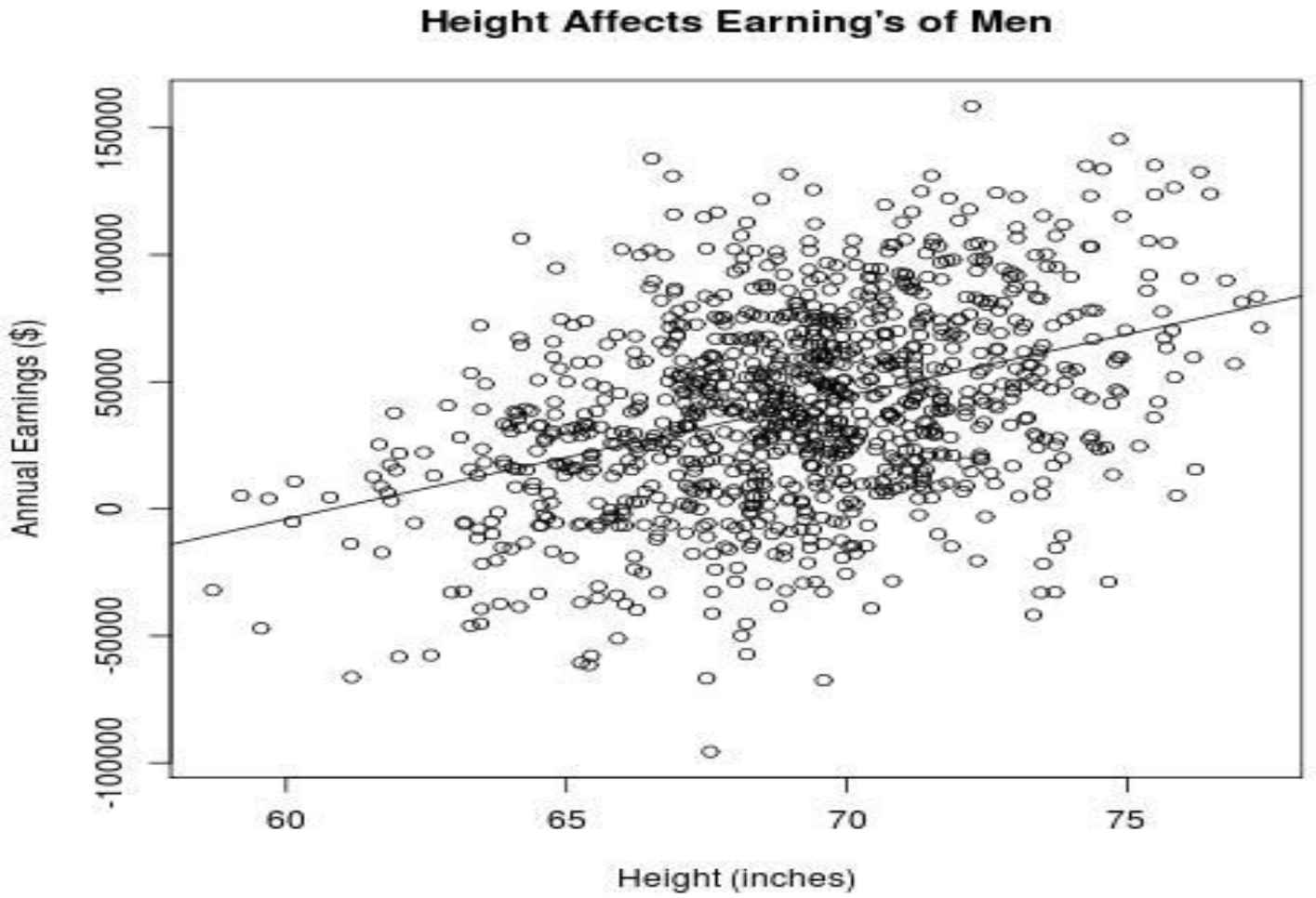
Note: These data are a work of fiction created for purposes of explaining this topic. For the real data see:

Refs. GAO Congressional Report, Nov 20, 2003, http://www.gao.gov/htext/d0435.html
"blink", Malcolm Gladwell, excerpt from http://www.gladwell.com/blink.excerpt2.html

Plot of Men's Yearly Earnings



Regression Line Fit to Data

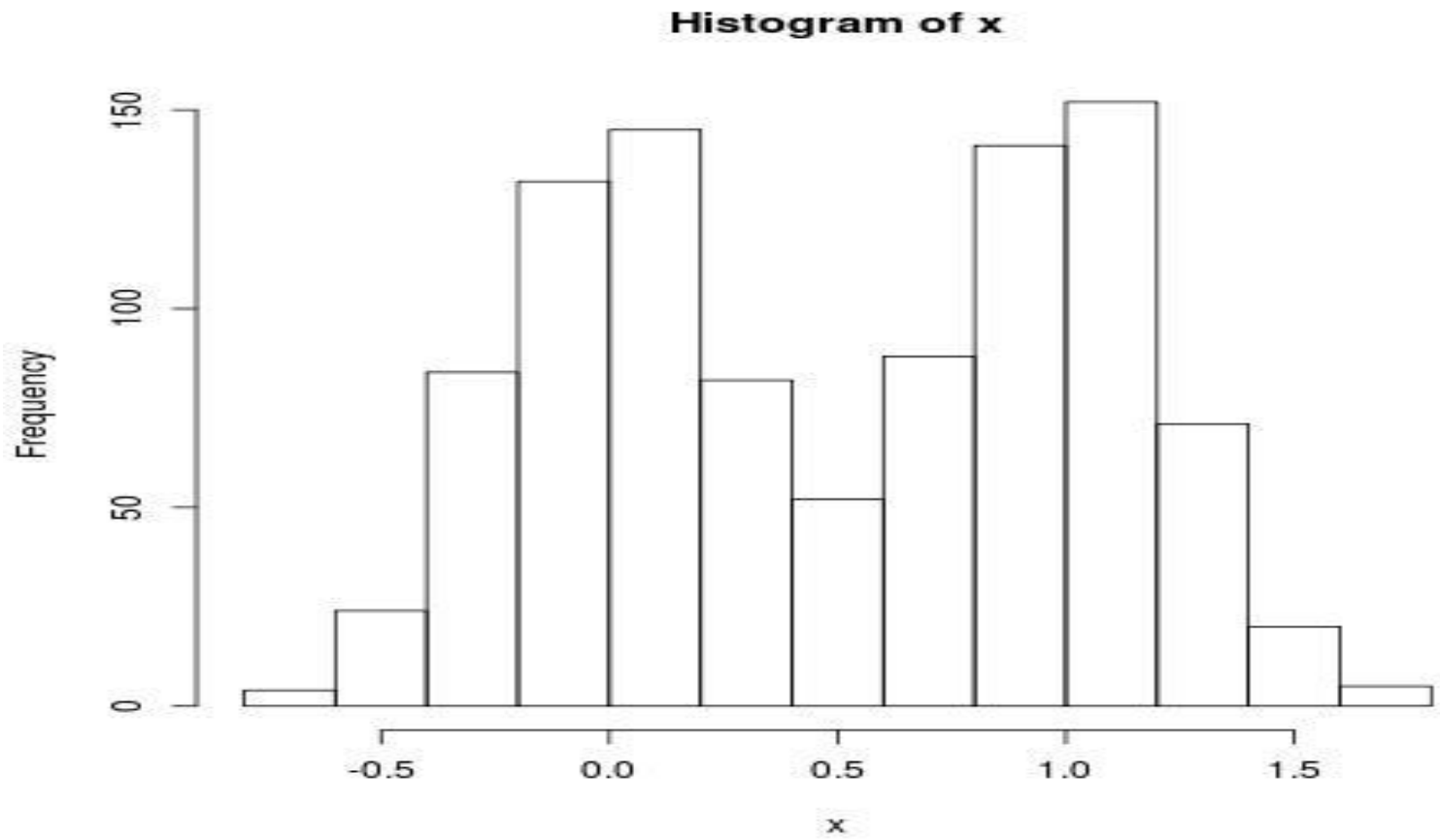


How does linear regression get calculated?

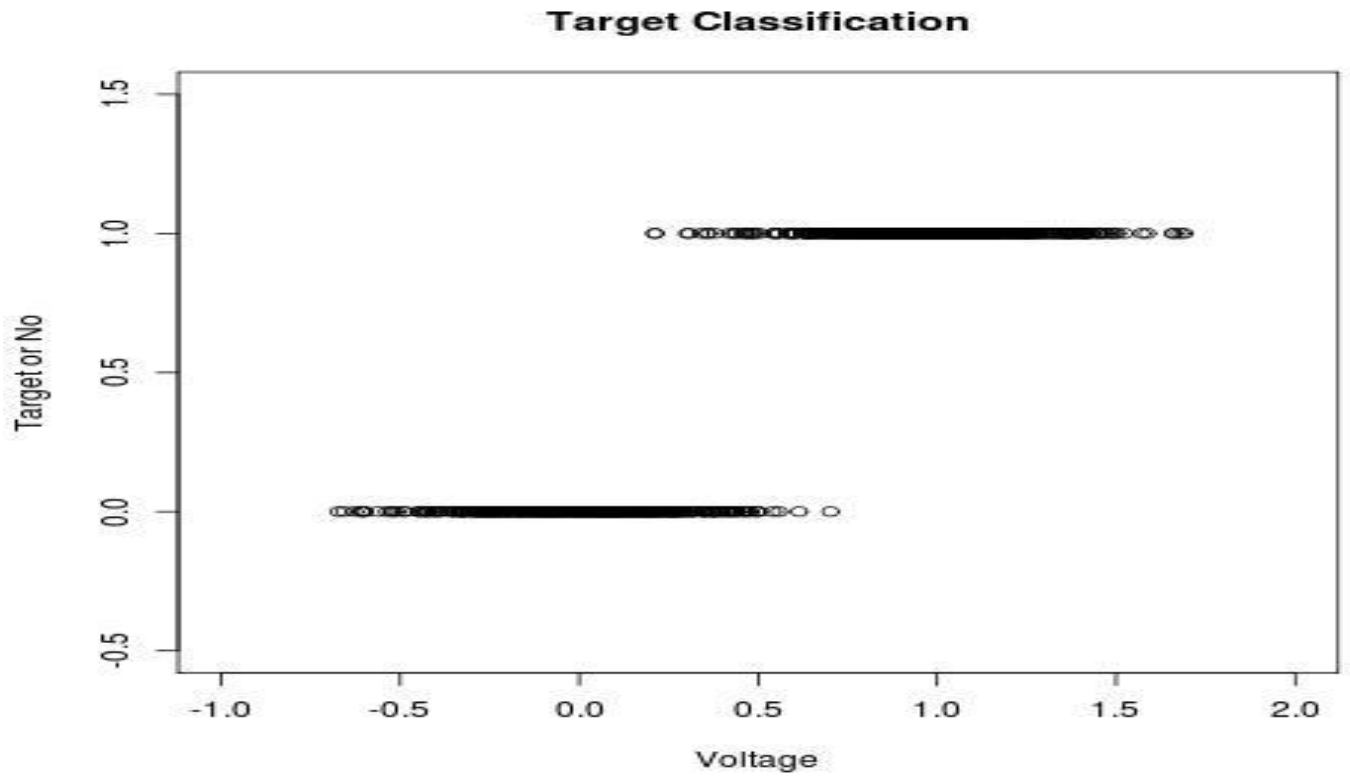
```
#X – list of "M" lists of "P" attributes (height) - X[i][j] for i = 0: (M-1) and j = 0: (P -1)
#Y – list of "M" labels – real numbers (regression) or class identifiers (classification) - earnings
#B – list of "P" coefficients – one for each attribute
#B0 – constant to account for bias between X and Y.

def dotProd(list1, list2)
        sum = 0.0
        for i in range(len(list1)):
                sum += list1[i] * list2[i]
        return sum

def sumSqErr(Y, X, B0, B)
        sum = 0.0
        for i in range(len(Y)):
                yhat = B0 + dotProd(B,X[i])
                error = Y[i] – yhat
                sum += error * error
        return sum*0.5/len(Y)
 #Given Y and X, find B0 and B that minimize sum squared error
```

Regression for Classification Problems

Example: Target Detection – Detector yields 1 volt if target is in view, 0 volts otherwise. Additive noise.



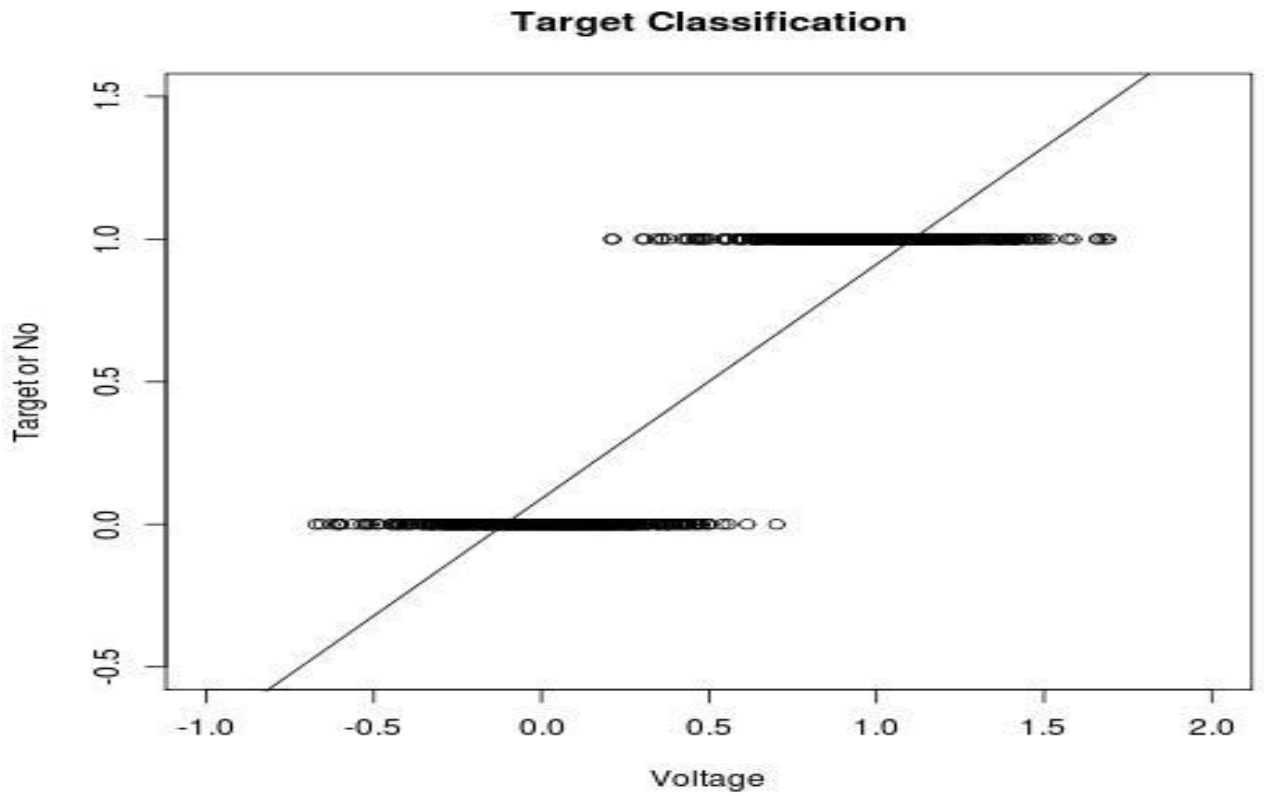
Target Classification with Labels



Could use OLS apparatus

– treat class outcome as 0,1 and fit straight line.

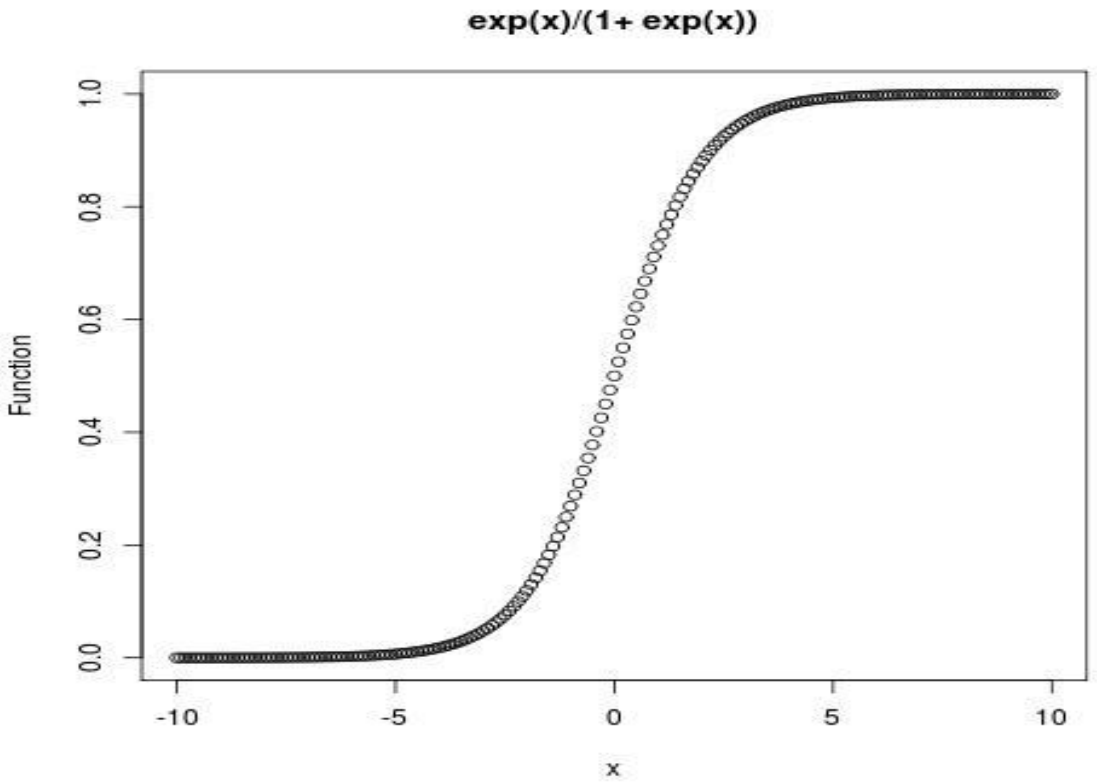
OLS for classification



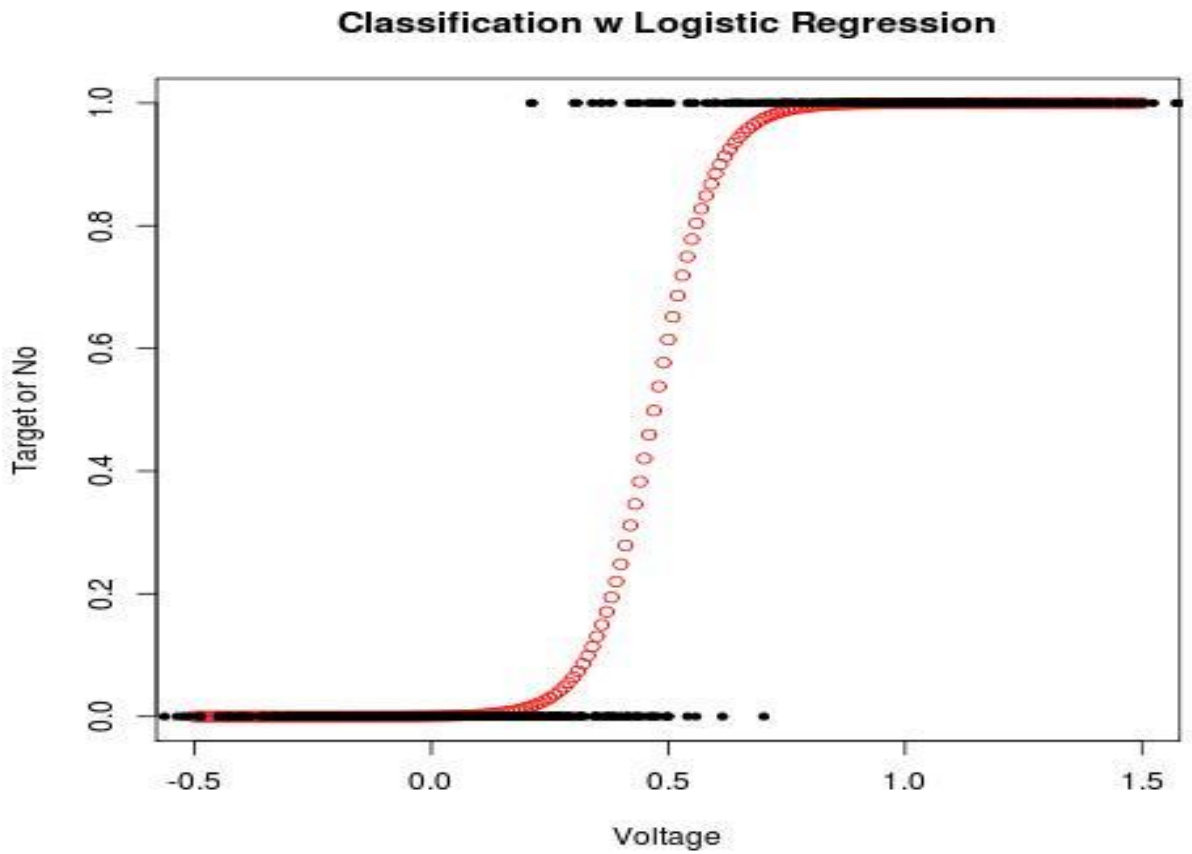
Frequently works just fine.

Logit Function

-Sometimes gives better results

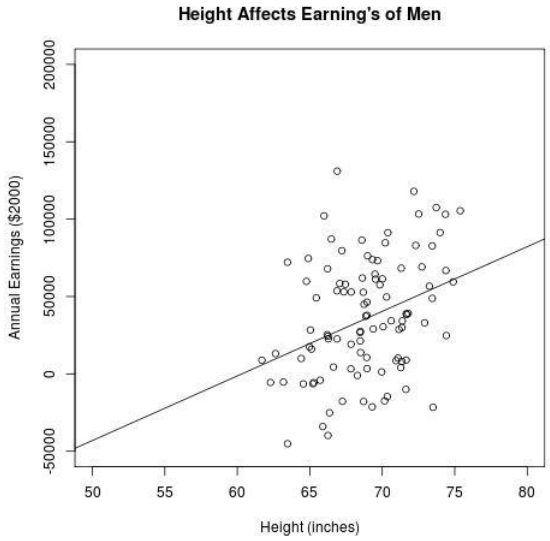


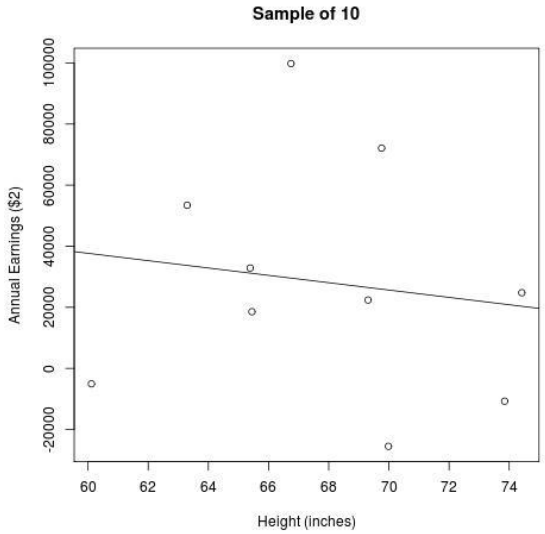
Logistic regression for Classification



Modern Regression – Escape Overfitting

Try fitting fewer points from the earnings vs height plot





Need to hedge your answers to avoid overfitting

More data is better (as long as we can compute answers).

Coefficient shrinkage methods

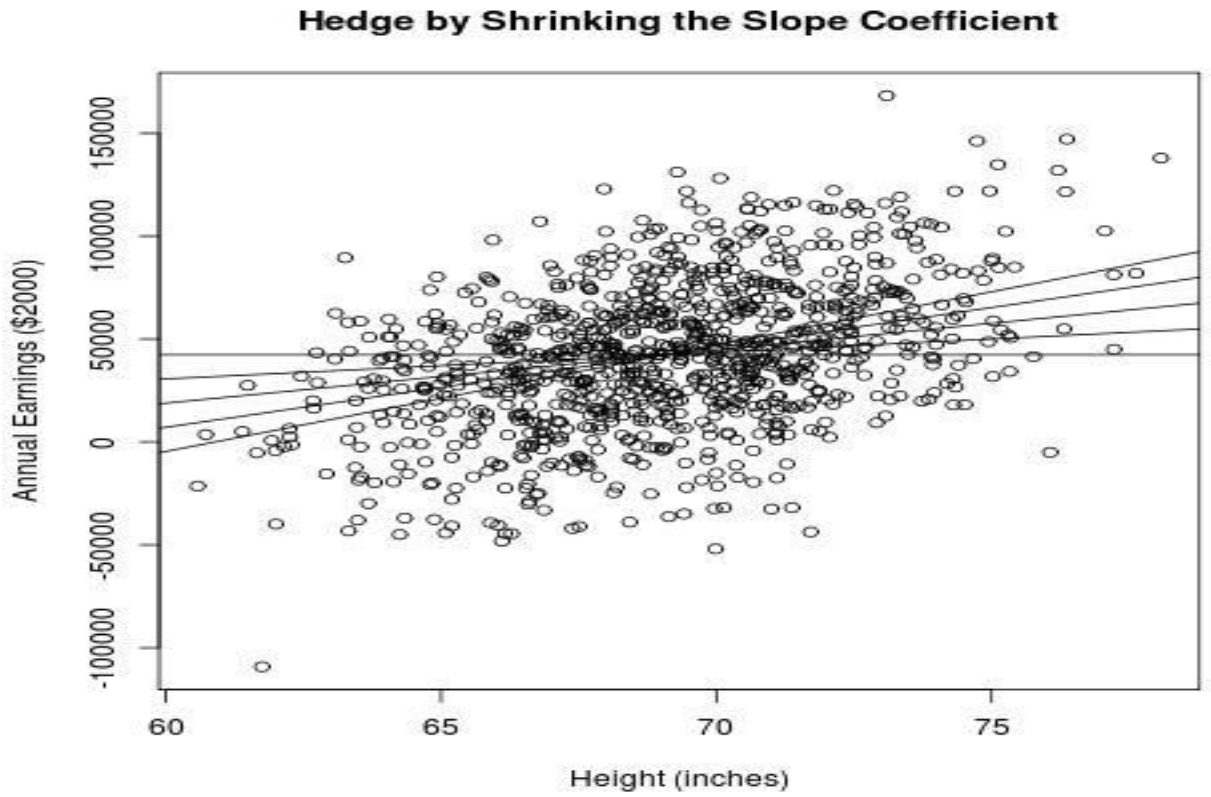
Can be more or less aggressive in use of data.

Shrink coefficients on X (we called that B in the code snip) towards zero.

What does that look like in the earnings vs height example?

Coefficient Shrinkage for earnings vs height

Move the slope of the regression line towards zero slope (horizontal line).



Features of Coefficient Shrinkage

Family of solutions ranging from:

- OLS
- Average output value (regardless of input)

What's the best value of slope parameter?

- Hold out some data from training.
- Use holdout data to resolve slope parameter.

Let's see what it looks like in multiple dimensions.

Coefficient shrinkage in multi-dimensions

First define "small" for a vector of numbers.

Here are two ways:

```
#B is a list containing the coefficients on X (as before). B0 is not included in the sum
#lambda is a positive real number

def lasso(B, lambda)
        sum = 0.0
        for x in B:
                sum += abs(x)
        return lambda*sum

def ridge(B, lambda)
        sum = 0.0
        for x in B:
                sum += x*x
        return lambda*sum
```

glmnet Formulation

Alter ordinary least squares problem formulation.

To the OLS minimization, add a penalty on the size of the coefficients (as a vector).

glmnet authors employ a flexible blend lasso and ridge penalties.

-called elastic net (conceived and explored by Zou and Hastie)

-useful control over solutions (more about that later)

Jerome Friedman, Trevor Hastie, Robert Tibshirani (2010). Regularization Paths for Generalized Linear Models via Coordinate Descent. Journal of Statistical Software, 33(1), 1-22. URL http://www.jstatsoft.org/v33/i01/.

Hui Zou and Trevor Hastie (2005) Regularization and variable selection via the elastic net, *J. R. Statist. Soc.* B (2005) 67, *Part* 2, *pp.* 301–320. http://www.stanford.edu/~hastie/Papers/B67.2%20%282005%29%20301-320%20Zou%20&%20Hastie.pdf

glmnet Formulation – pseudo code

```
def glmnetPenalty(Y, X, B0, B, lambda, alpha)
    penalty = sumSqErr(Y, X, B0, B) + 0.5*(1-alpha)*ridge(B,lambda) + alpha*lasso(B, lambda)
    return penalty
```

Minimize glmnetPenalty with respect to B0 and B.

If lambda = 0.0 you have ordinary least square problem

As lambda -> infinity B -> 0.0.

- Estimate becomes average(Y)
- Ignore X (input data)

Why add this complication?

Eliminate over-fitting (match model complexity to data wealth)

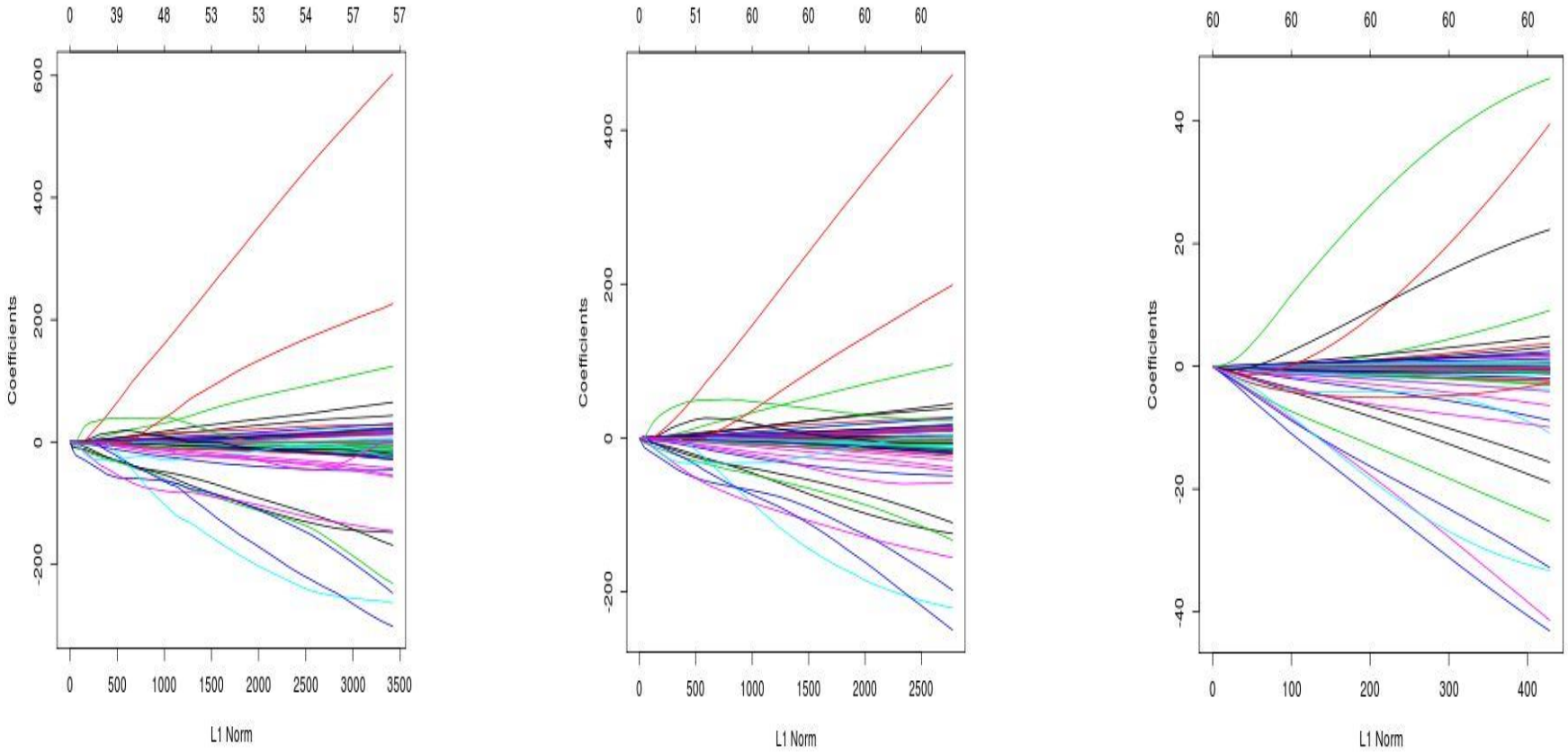
Achieve best generalization – best performance on new data.

Pick member of solution family which gives best performance on held out data (i.e. data not included in training set).

A couple of important details:

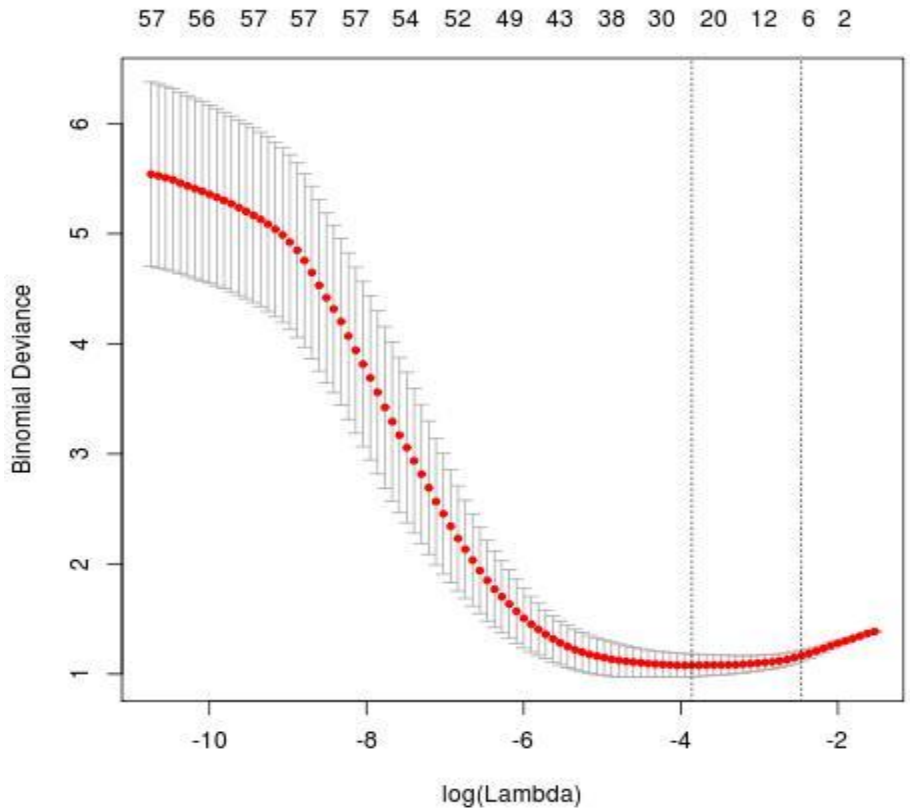
1. Bias value is not included in the coefficient penalty
2. Inputs (attributes) must be scaled. Usual choice is
   mean = 0.0
   standard deviation = 1.0

Coefficient Penalty Function Choices



Coefficient Paths for alpha = 1, 0.2 and 0.0 for sonar data set.

Error vs Model Complexity for Sonar



Performance on hold-out set versus model complexity alpha=1.

glmnet algorithm

We no longer have Gauss's closed form solution. That's where glmnet algorithm comes in.

Basic idea of glmnet algorithm:

- Start with large lambda => All coefficients = 0.0
- Iterate:
  - Make small decrease in in lambda
  - Use coordinate descent to recalculate new B.

Authors demonstrate speed advantage on real datasets that range from x4 to more than x100

-Speed advantage more pronounced on wide attribute spaces and/or large data sets.

Handwritten Zip Code Recognition

Handwritten digits 0 – 9

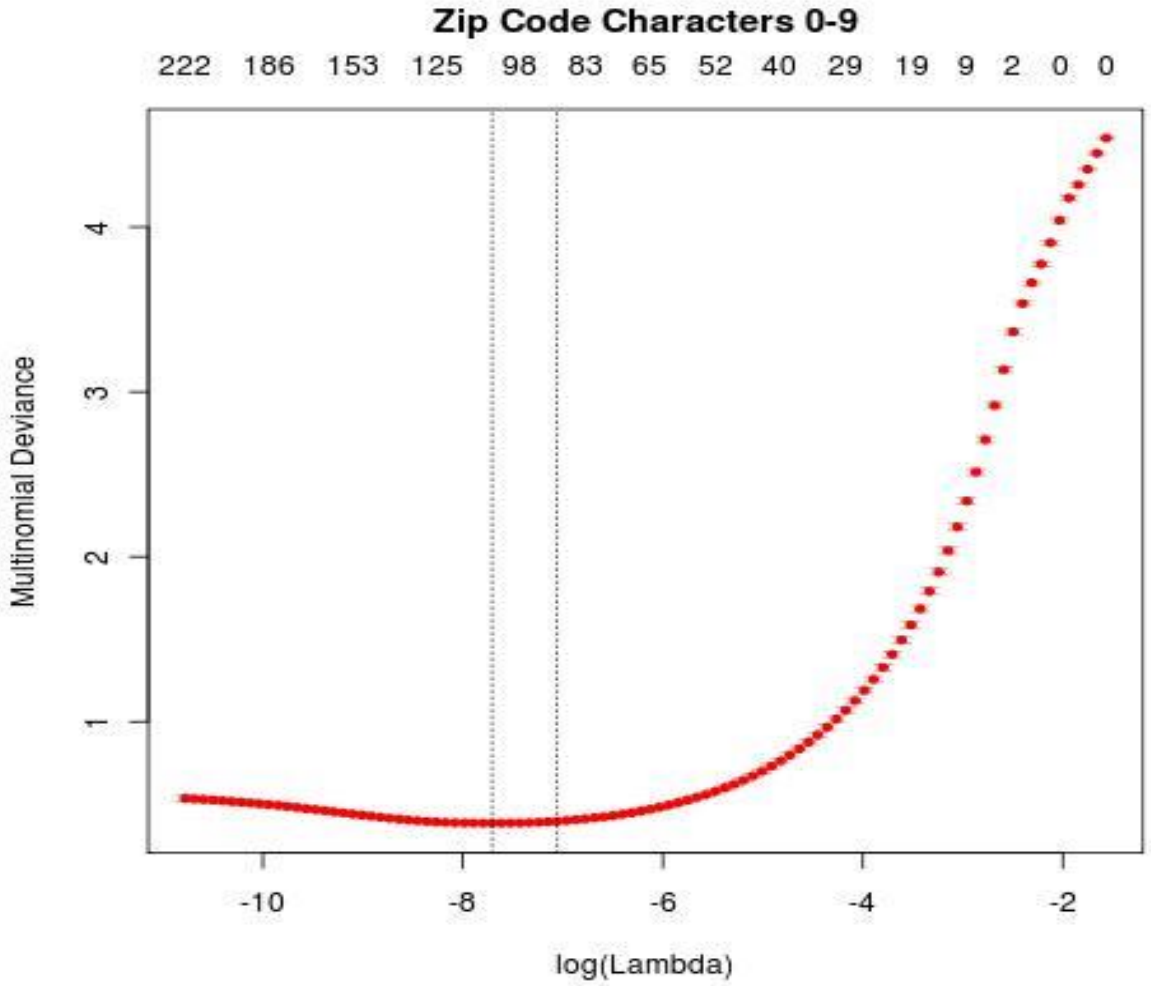
Grey scale levels for 16x16 grid

Roughly 7000 training examples and 2000 test examples

Use R-package glmnet (written by authors of glmnet paper)

zip code data set from Hastie, Tibshirani and Friedman Elements of Statistical Learning, http://www-stat.stanford.edu/~tibs/ElemStatLearn/

Testing Error on Zip Code Digits



Advantages of glmnet

-Rapid generation of entire coefficient paths
-Adapts easily to map-reduce
-Relatively simple implementation – (after training) evaluation requires multiply and sum for each attribute.
-Manageable in real time with wide attribute spaces – e.g. text processing - spam filtering, POST, NER

Algorithm checklist

- Must have parameter for dialing complexity up and down.
- Fast to train
- Doesn't need entire data set in memory at one time
- Handles numeric and categorical input and output (attributes and labels)
- Easily implemented (or already available)
- MR'able