Lecture 3: Linear methods for prediction (cont)

Alan Lee

Department of Statistics
STATS 784 Lecture 3

July 31, 2017

Outline

Today's agenda

In this lecture we continue our discussion of linear methods for prediction, concentrating on methods for estimating prediction error. We treat

- Cross-validation and bootstrap estimates of prediction error
- Subset selection techniques

Again, we will use the California housing data as a running example.

See *The Elements of Statistical Learning*, Ch7, *Introduction to Statistical Learning* Ch 5, and *Applied Predictive Modelling*, Ch 4.

Methods for estimating PE

We discuss three:

- The test set estimate (requires a new data set with inputs and outputs)
- Cross validation (does not require a new data set, but uses a type of test set estimate)
- The Bootstrap (uses repeated bootstrap samples)

The test set estimate

As we have seen, this is simple but requires a new data set containing both inputs and outputs. It may not be very accurate if the test set is not very large.

Would we be better combining the test and training sets and using the combined set to fit the model? A bigger sample size will give a predictor with smaller variance.

We would then have to use another method (such as CV or bootstrap) to actually estimate the prediction error.

Cross-validation

In cross validation, we split the data randomly into 10 parts, use one part for the test set and the remaining 9 parts for the training set, and estimate the PE with the test set estimator.

We repeat this process using a different tenth as the test set each time. We average the resulting 10 PE estimates to get a final estimate. We can repeat for different random splits and average.

This is (10-fold) cross-validation. (Could also split into 5 parts - 5-fold CV). See later for a discussion about which is best.

Cross-validation in R

There are several options. In the R330 package, there is a function `cross.val` that will calculate a cross-validation estimate of prediction error. In the package `bootstrap`, there is a function `crossval` that can be used with linear models (as well as many others). The package `caret` which we will look at later, is even more comprehensive. For the California data, we have

```
> # cross.val from R330 package
> library(R330)
> cross.val(california, nfold = 10, nrep = 20)
Cross-validated estimate of root
mean square prediction error =  0.3434868
> 0.3434868^2
[1] 0.1179832
```

Compare with apparent error of 0.116821 and a test set error of 0.114515. (Since the sample is so big, there is not much difference between these.)

Choice of k in k-fold CV

Both the conditional and unconditional prediction errors depend on the size of the training set: the bigger the training set the better the predictor and the smaller the PE.

Suppose we want to estimate PE corresponding to a training set of size n. When using k-fold CV, we are using a predictors based on a training sets of size $(k-1)n/k$, and estimating the average error of these predictors, which will be biased upwards.

On the other hand, the estimates of the error are based on test sets of size n/k. These estimates will be less accurate as k increases.
Thus

- For large k, CV(k) is more variable, less biased.
- For small k, CV(k) is more biased, less variable.

$k = 5$ or $k = 10$ is a good compromise choice.

Standard error of CV

The CV estimate is based on random splits of the training data: two successive applications of the function `cross.val` will result in different estimates. We can run the function several times and calculate the standard deviation of the results (this is not quite the true SD as the results from different splits are not independent, but should not be too bad.)

```
> cvvec = numeric(20)
> for(i in 1:20){
  cvvec[i] = cross.val.mod(california, nfold=10,nrep=1)
  }
> mean(cvvec)
[1] 0.1180739
> sd(cvvec)
[1] 0.0002110114
```

There is not much variability in the CV estimate, due to the large sample size. CV10 appears to be biased upward.

The Bootstrap

The basic idea here is to mimic new data by resampling the training set, i.e. taking repeated bootstrap samples from the training set.

Suppose the training set contains n cases. To take a bootstrap sample, we take a random sample of size n with replacement from the training set. Thus, some of the cases in the training set will be duplicated in the bootstrap sample, and others won't be included at all.

The probability a given case won't be included in the bootstrap sample is $(1 - \frac{1}{n})^n$ which for large n is approximately $e^{-1} = 0.368$ (The probability that a case *will* be included is thus about 0.632.)

Bootstrap approaches

A possible bootstrap approach might be to fit the model to a bootstrap sample, and use the original sample as a test set. This doesn't work too well as the test and training sets are too similar.

An alternative is to use the bootstrap sample as both test and training set. This really will be an underestimate. If we take the difference between the two estimates (called the optimism), we get a (possibly too small) estimate of the amount by which the in-sample error underestimates PE.

We could average the optimisms over B bootstrap samples and add the result to the in-sample error to get a corrected PE estimate. This turns out to work quite well.

The .632 estimator

The percentage of the original data that appears in a bootstrap sample is on average about 63.2%, so if we use the bootstrap sample as a training set and the original sample as a test set there will be considerable overlap, and the error estimate will be too small. The estimate is (denoting the predictor based on bootstrap sample b by f_b and the original data set by (x_i, y_i))

$$\frac{1}{B}\sum_{b=1}^{B}\frac{1}{n}\sum_{i=1}^{n}(y_i - f_b(x_i))^2 = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{B}\sum_{b=1}^{B}(y_i - f_b(x_i))^2.$$

For 62% of the samples we are using a data point to estimate the error that was also used to fit the model. A better estimate would be

$$\epsilon^{(0)} = \frac{1}{n}\sum_{i=1}^{n}\frac{1}{|C_i|}\sum_{b\in C_i}(y_i - f_b(x_i))^2,$$

where C_i is the set (of size $|C_i|$) of bootstrap samples that do not contain (x_i, y_i)

The .632 estimator (cont)

It can be shown that this estimate is very similar to "half-fold CV" with $k = 2$. This will tend to be an overestimate of PE. If we average this estimate with the underestimate $\overline{\mathrm{err}}$ we should get a better result. This leads to the "0.632" estimate

$$(1 - 0.632)\overline{\mathrm{err}} + 0.632\epsilon^{(0)}$$

The choice of these weights is motivated by a rather complicated argument we will not go into. (See Efron (1983) if you want the full monty.) A modification (the 0.632+ estimator, Efron 1997) uses different weights.

Summary

Let $PE(\text{test}, \text{training})$ denote the estimate of prediction error obtained by calculating the test set error when using the data set “test” as the test set and the data set “training” as the training set. The different estimates of PE are

- In-sample: $PE = PE(\text{training}, \text{training}) = \overline{err}$
- Test set estimate $PE = PE(\text{test}, \text{training})$
- $\overline{\text{err}}$ + opt: $PE = \overline{err} + \overline{PE(data, bs) - PE(bs, bs)}$
  bs=bootstrap sample
- 0.632: $0(1 - 0.632)\overline{\text{err}} + 0.632\epsilon^{(0)}$

The test set estimate estimates the conditional error, while the bootstrap and CV estimates are better estimates of the unconditional error.

An example

To compare the various estimators, we conducted a small simulation, using data pairs $(x_i, y_i), i = 1, \ldots, n$ sampled from a bivariate normal distribution with means 0, variances 1 and correlation ρ. We want to evaluate the prediction error when using least squares regression (with no intercept) as a predictor. Consider the case where ρ is 0.3 and the training set has sizes $n = 20, 50, 100$. We got the estimates on the next slide (averaged over 1000 replications, with $B = 50$ bootstrap samples in each). Note: LOOCV is leave-one-out CV, PE is the unconditional PE.

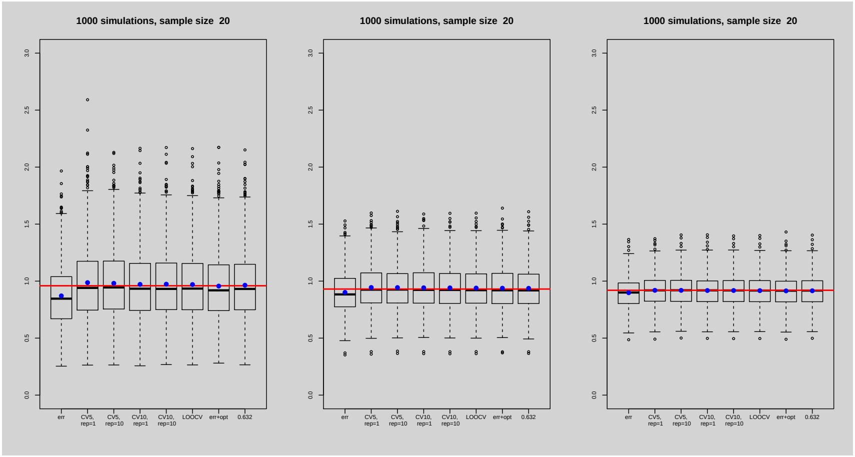
Results

The table shows estimates of PE using different methods, for sample sizes $n = 20,\ 50,\ 100$.

	$n = 20$		$n = 50$		$n = 100$	
	$PE = 0.959$		$PE = 0.930$		$PE = 0.920$	
	Mean	SD	Mean	SD	Mean	SD
$\overline{\text{err}}$	0.871	0.281	0.901	0.184	0.898	0.129
CV5,rep=1	0.987	0.337	0.944	0.196	0.919	0.132
CV5, rep=10	0.981	0.319	0.944	0.194	0.919	0.132
CV10, rep=1	0.972	0.317	0.942	0.194	0.918	0.132
CV10, rep=10	0.973	0.316	0.941	0.193	0.918	0.131
LOOCV	0.970	0.315	0.939	0.192	0.917	0.131
$\overline{\text{err}}$+opt	0.956	0.311	0.938	0.195	0.915	0.131
0.632	0.964	0.312	0.937	0.192	0.915	0.131

The opt+err method looks good. The apparent error $\overline{\text{err}}$ underestimates.

Boxplots of 1000 reps



Points to note

- The apparent error underestimates, but the effect reduces with increasing sample size.
- The CV estimates are overestimating the unconditional PE , but the bootstrap estimates are doing well.
- For $n = 20$ CV5 is slightly more biased than CV10.
- Repeating the CV estimates for different splits is not having much effect for 10 fold, but is slightly improving the estimate for 5-fold.

Bootstrap: the California data

The R 330 package has a function `err.boot` that will calculate a bootstrap estimate of PE. The package `bootstrap` has a function `bootpred` that can be used with linear models (as well as many others). The package `caret` will calculate both CV and bootstrap estimates for a variety of models.

```
> err.boot(california, B=50)
$err
[1] 0.116821
$Err
[1] 0.117288
```

Very similar to the CV10 estimate! NB: err is the apparent error estimate, ERR the err + opt estimate. See the code handout for `bootpred` and `caret`.

The bootstrap package

This package contains functions for more general cross-validation and bootstrapping (not just for linear regression). We will illustrate its use with the California data. First, cross-validation (data in data frame `california.df`):

```
> library(bootstrap)
> theta.fit <- function(x,y){lsfit(x,y)}
> theta.predict <- function(fit,x){cbind(1,x)%*%fit$coef}
> sq.err <- function(y,yhat) { (y-yhat)^2}
>
> y =log(california.df[,1])
> x = california.df[,-1]
> cv10Errs = crossval(x,y, theta.fit, theta.predict,ngroup=10)
> cv10 = mean((y-cv5Errs$cv.fit)^2)
> cv10
[1] 0.1181994
```

The bootstrap package (cont)

And the bootstrap

```
> boot = bootpred(x,y,nboot=200, theta.fit, theta.predict,
+            err.meas=sq.err)
> bootOpt = boot[[1]] + boot[[2]] # boot + opt estimate
> bootOpt
[1] 0.1181497
> boot632 = boot[[3]] # boot632 estimate
> boot632
[1] 0.1177459
```

The caret package

caret = "classification and regression training". It is a package for efficiently computing the PE of several different models, when we are trying to find the best predictor. It will compute both CV and bootstrap estimates. We illustrate its use with some sample code on the next slide. There is a useful website:
https://topepo.github.io/caret/
The book "Applied Predictive Modeling" is also useful. (written by the author of caret) , and see also the Statistical Software article by Max Kuhn "Building Predictive Models in R using the caret Package"(link on the lectures page)

The caret package: CV usage

```
> library(caret)
> CV10 = train(log(medval)~ medinc + medage + rooms +
+         bedrooms + pop + house + lat + long,
+         data = california.df,
+         method = "lm",  # you get this from the website
+         trControl = trainControl(method="cv", number=10,
+         repeats=20))
> CV10
Linear Regression
Resampling results:

  RMSE       Rsquared
  0.3429905  0.6376908
> 0.3429905^2
[1] 0.1176425
```

The caret package: bootstrap usage

```
> boot = train(log(medval)~ medinc + medage + rooms +
+       bedrooms + pop + house + lat + long,
+       data = california.df,
+       method = "lm",  # you get this from the website
+       trControl = trainControl(method="boot", repeats=200))
> boot
> boot
Linear Regression
Resampling results:
  RMSE       Rsquared
  0.3442151  0.6350016
> 0.3442151^2
[1] 0.118484
```

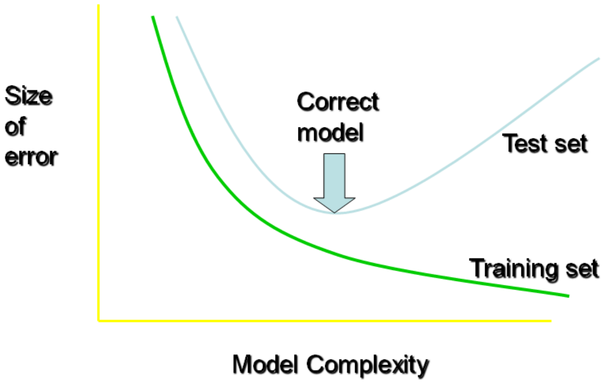
Variable selection

Should we use all the variables available, or just some of them? Of course, we should include all variables we think might be important features, identified by conversations with subject matter experts. But using all available variables might overfit and lead to poor predictions, as we saw in the polynomial example.

Model complexity

When fitting models to a training set, we do not want to use a model that has too many variables. Such a model will attempt to capture not only the true function f but will also model the errors, as in the polynomial example. Future data will have the same f but different errors, which will not be well predicted by a too-complex model. Thus, a too-complex model will have a small training set error, but a large PE (test set error, the error expected when predicting future data.)

Model complexity(cont)



As we add more variables, the in-sample error goes down, but after a certain point, the true PE stars to rise.

Methods for variable selection

In linear prediction (and other kinds a well), we might want to identify a subset of variables that predicts well. There are several approaches possible:

- All possible subsets: If the number of variables is not too great, we could examine all possible subsets of variables (If there are k variables there will be 2^k possible subsets).
- We could add variables one at a time, choosing the variable that gives the best improvement in the estimated PE. This is called **forward selection**.
- We could start with all the variables, then delete variables one at a time, choosing the variable that gives the best improvement in the estimated PE. This is called **backward elimination**.
- A combination of the last two: **stepwise regression**.

Measuring PE inprovement

How do we measure the PE improvement? Since there are a lot of comparisons, CV or the bootstrap may be too computationally intensive.

Using the in-sample error is no good, since adding a variable will **always decrease** the in-sample error.

One quick way is to use a penalized form of the in-sample error, the AIC, which is proportional to the in-sample error plus the quantity $2p\hat{\sigma}^2/n$, where p is the number of estimated coefficients in the predictor, and $\hat{\sigma}^2$ is an estimate of the error variance. This is the method used in R.

Example: the California data, forward selection

```
> null.model = lm(log(medval)~1, data=california.df)
> selected = step(null.model, scope = formula(california),
        direction = "forward", trace=0)
> selected
Call:
lm(formula = log(medval) ~ medinc + medage + house + pop + lat +
    long + bedrooms + rooms, data = california.df)
```

Example: the California data, backward elimination

```
> selected = step(california, scope = formula(california),
        direction = "backward", trace=0)
```

In this case the same model is selected.

Example: the California data, stepwise

```
> selected = step(california, scope = formula(california),
        direction = "both", trace=0)
```

Again, the same model is selected.

Example: the California data, all possible subsets

```
> allpossregs(california)
      rssp sigma2 adjRsq       Cp      AIC      BIC      CV
1 1848.761  0.185  0.430 5815.345 15815.34 15829.77 184.987
2 1773.865  0.177  0.453 5176.804 15176.80 15198.43 177.543
3 1296.345  0.130  0.600 1094.859 11094.86 11123.70 129.780
4 1274.979  0.128  0.607  914.131 10914.13 10950.18 127.674
5 1188.281  0.119  0.633  174.653 10174.65 10217.92 119.747
6 1174.588  0.118  0.637   59.553 10059.55 10110.02 118.360
7 1171.280  0.117  0.638   33.253 10033.25 10090.94 118.056<-----------
8 1168.210  0.117  0.639    9.000 10009.00 10073.89 118.052<-----------
  medinc medage rooms bedrooms pop house lat long
1      1      0     0        0   0     0   0    0
2      1      1     0        0   0     0   0    0
3      1      0     0        0   0     0   1    1
4      1      0     0        1   0     0   1    1
5      1      0     0        1   1     0   1    1
6      1      1     0        1   1     0   1    1
7      1      1     1        1   1     0   1    1<-----------------
8      1      1     1        1   1     1   1    1<-----------------
```

Divide the CV entries by $n/10 = 1000$ to get the actual CV estimates.

References

1. Efron, B. (1983). Estimating the error rate of a prediction rule: some improvements on cross-validation, *Journal of the American Statistical Association* 78: 316-331.
2. Efron, B. and Tibshirani, R.J. (1993). *An Introduction to the Bootstrap*. Chapman and Hall.
3. Efron, B. and Tibshirani, R. (1997). Improvements on cross-validation: the 632+ bootstrap: method, *Journal of the American Statistical Association* 92: 548-560.
4. Hastie, T., Tibshirani, R.J. and Friedman, J. (2009). *The Elements of Statistical Learning, 2nd Ed*. Springer.
5. Kuhn, M. and Johnson, K. (2013). *Applied Predictive Modelling*. Springer
6. Venables, W.N. and Ripley, B.D. (2002) *Modern Applied Statistics with S, 4th Ed.* Springer