MSG500/MVE190
Linear Models - Lecture 8

Rebecka Jörnsten
Mathematical Statistics
University of Gothenburg/Chalmers University of Technology

November 22, 2012

1 RECAP

- Ultimate validation of a model is to test its predictive capacity
- Prediction performance is affected by both Bias and Estimation variance
- We combine the two into the criterion *prediction* $MSE = Bias^2$ + *Estimation Variance*
- We can't compute this in real life situations since we don't know the true model (needed to compute the bias)
- With training and test data we can separate model estimation from model validation (prediction)
- Training data: compute $MSE_{train} = RSS/n$ (the fit of the model on the data used to estimate model parameters)
- Test data: compute $pMSE = MSE_{test}$ (the fit of the model to new data *not* used for estimation).
- The $pMSE$ is a substitute for the real *prediction MSE* as defined above.
- We select the model that minimizes the $pMSE$

2 Cross-validation

In the previous lecture we compared backward model selection to the performance of model actually selected to work for prediction. Of course, in practise we won't have access to both training and test data, but as you saw in the demo, you can create training and test data by splitting the data set up. That raises a couple of issues; (i) how much data should be used for training and how much for testing; and (ii) how to split the data up.

We focus on (ii) first. In the previous demo we just randomly split the data 50-50 or 60-40. Now, this means that some observations are used for training and some for testing, and chance decides. Cross-validation is a way of formalizing this idea, but making sure that all data plays both training and testing at least once.

K-fold Cross-validation

- Split the data set into K equal size parts (if the data isn't of size integer$*K$, some folds of data will have one observation more than the others - just keep track of this).
- Enumerate all models you want to investigate: $m_1, \cdots, m_M$
- For $k = 1, \cdots, K$
    1. For observation i *not* in fold k, fit all models $m_1, \cdots, M_m$. Save the corresponding coefficient estimate $\hat{\beta}_k(m_j), j = 1, \cdots, M$

2. Apply the estimated model for prediction to the observations i *in* the fold k and compute the corresponding sum of prediction errors: $prederror_k(m_j) = \sum_{i \in\ fold\ k} \big(y_i - \hat{y}_i(k, m_j)\big)^2$, where $\hat{y}_i(k, m_j) = x_i \hat{\beta}_k(m_j)$. Note, the estimated coefficients $\hat{\beta}_k$ *do not* depend on observations i in fold k.

- Compute the total prediction error sum of squares for all the M models: $prederror(m_j) = \sum_k prederror_k(m_j)$
- Choose the model m^* which minimizes the prediction error: $m^* = argmin_j prederror(m_j)$

Note, that all observations get a chance to be part of the validation or testing of the models. The final selection criterion is the prediction error summed up over all folds. You could also use the prediction MSE, $prederror/n$ where n is the number of observations.

Here's an illustration: We simulate data with $n = 75$ observations from $y = 1 + 2x - .5x^2 + \epsilon$, where $\epsilon \sim N(0, 2)$. We enumerate all the possible models up to order 3: $y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \epsilon$

Model	β_0	β_1	β_2	β_3
1	TRUE	FALSE	FALSE	FALSE
2	TRUE	TRUE	FALSE	FALSE
3	TRUE	FALSE	FALSE	TRUE
4	TRUE	FALSE	TRUE	FALSE
5	TRUE	TRUE	TRUE	FALSE
6	TRUE	TRUE	FALSE	TRUE
7	TRUE	FALSE	TRUE	TRUE
8	TRUE	TRUE	TRUE	TRUE

Table 1: Enumeration of models

In Table 1, I don't allow for the intercept to be absent and there are thus 8 total models to compare.

I try 10-fold cross-validation, i.e. divide the data set into 10 random chunks of the data. To make this easier, I first create a vector enumerating all the observations, then scramble this vector and finally divide it into 10 equal pieces. Note, if my sample size is not an integer value of 10, I let the first folds contain one extra observation until all data observations have been included in a fold.

We now cycle through each fold, withholding part of the data and fitting each model to the rest. We then apply the fitted models to the folds for prediction.

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	40.40	61.78	114.14	84.36	48.33	45.18	24.16	92.53	17.91	50.40	7.72
2	25.23	27.44	73.42	51.14	30.39	36.74	16.18	42.03	18.27	38.51	4.79
3	16.97	42.11	104.26	66.86	33.22	70.63	15.47	69.92	12.31	46.45	6.38
4	42.91	56.06	114.81	78.93	62.02	39.27	21.69	91.83	20.65	47.35	7.67
5	37.96	25.54	65.51	51.78	30.57	26.06	20.04	35.77	26.39	37.30	4.76
6	27.56	25.89	72.92	63.81	30.01	31.53	13.34	44.23	16.88	36.13	4.83
7	20.94	39.83	104.67	93.86	34.59	71.58	13.36	70.86	12.68	44.24	6.76
8	39.40	23.37	63.70	51.04	33.48	16.96	16.92	37.92	25.02	34.56	4.56

Table 2: Prederrors in different folds and total

In Table 2 you can compare the fold results and total prediction error results for the different models (see Table 1). I extract the winning model (with minimum PE) from the table:

```
> winmod <- Models[which.min(PE), ]
> print(winmod)

intercept        b1        b2        b3
     TRUE      TRUE      TRUE      TRUE
```

Let's repeat the exercise with much more data ($n = 1500$).

```
> z <- data.frame(as.matrix(cbind(Prederrors, PE)))
> colnames(z) <- c("Fold1", "Fold2", "Fold3", "Fold4", "Fold5",
+     "Fold6", "Fold7", "Fold8", "Fold9", "Fold10", "PE")
> row.names <- c(seq(1, 8))
> xtable(z, digits = c(0, rep(0, 10), 3), caption = "Prederrors in different folds and total",
+     label = "tab:CVresult2")
```

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	1184	1460	1177	1182	1285	1559	1453	1310	1269	1570	8.967
2	546	667	658	587	727	602	672	660	582	795	4.330
3	841	919	946	777	979	1196	912	896	960	979	6.270
4	1120	1410	1160	1023	1386	1385	1233	1291	1107	1245	8.240
5	479	541	632	538	694	533	595	628	531	693	3.909
6	553	673	661	584	730	592	671	664	574	781	4.321
7	787	815	937	746	947	1302	840	862	1011	983	6.153
8	477	543	631	537	699	530	596	629	528	696	3.911

Table 3: Prederrors in different folds and total

In Table 3 you can compare the fold results and total prediction error results for the different models (see Table 1). I extract the winning model (with minimum PE) from the table:

```
> winmod <- Models[which.min(PE), ]
> print(winmod)
```

```
intercept        b1        b2        b3
     TRUE      TRUE      TRUE     FALSE
```

Next lecture we will talk about the expected prediction error and the sources of variation in this - one being the model estimation and one being the inherent variation in the data (σ^2). You might notice that the PE comes pretty close to the $\sigma^2 = 4$ of the model for large sample sizes, but (if you repeat) the exercise we see more variability for the smaller sample sizes.

Even with this simple model, cross-validation may not always select the correct model - there are some collinearity problems present (x and x^3 are correlated) and the effect of x^2 is rather weak. Repeat the above exercise a couple of times with different sample sizes and noise levels.

Picking K and LOOCV: leave-one-out cross-validation

You can pick the number of folds, K, anyway you want. However, if you check in the literature the most common choices for K are $K = 3, 5, 10$ and n. We will discuss the pros and cons of these choices now.

What is the impact of choosing a small K, say $K = 2$. $K = 2$ means you split the data in half and train/test on each half separately. This boils down to sacrificing quite a lot of data to be saved for testing each model. If you have a huge data set this doesn't really matter too much *but* if your sample size is small you have an even smaller data set to train your models on. The less data you have, the higher the estimation variance of the model parameters (as we know from previous lectures). As a consequence, the prediction performance might deteriorate.

Why do we care? Well, we have talked about bias and variance. If we intend to use the observed prediction error (across folds of data) for model selection, we need this *estimate* of the *expected prediction error* to be accurate. We may suffer from bias, which here would mean that we over- or under-estimate the performance of a model. We may also have high estimation variance of the expected prediction error, which means that our observed quantity may be quite far from the expected one, making model selection an uncertain business. The smaller we make K, the less data we have and so we will bias our prediction error estimate upward.

So, should we pick a large K? The larger K is, the more similar the training data set is in size to the original data set, so we expect training to perform nearly as well for large K as when we use all the data.

However, each of the training sets actually overlap quite a bit, so our final estimate of the prediction error, essentially an average of each fold performance, shares a lot of information. When we use small K we hope that the average over each fold performance cancels out some of the training and test-specific data variabilities. Here, since the training data sets share so much information, we are really only observing a few, separate predictions. This will be reflected in a highly variable estimate of the true prediction error.

Conclusion: small K - we bias our prediction error estimate upward: large K - our estimated prediction error is highly variable. There is a trade-off. In general, I advice you to stay with common choices like $K = 10$ or $K = n$. Now, this extreme $K = n$ option is something that deserves a separate discussion. $K = n$ is called Leave-one-out cross-validation, LOOCV. It can be shown to generate *unbiased* estimates of a model's expected prediction error, but is also notorious for having high estimation variance. LOOCV is still very popular because for linear models we don't actually have to do any refitting at all!

Let's show this. Let's say we look in fold k, meaning observation k is the test data and every other observation is the training data. The fitted value

$$\hat{y}_k = \sum_j h_{kj} y_j$$

when all observations are included, but if we exclude k the predicted value is

$$\hat{y}_k(-k) = \sum_{j \neq k} \frac{h_{kj}}{1 - h_{kk}} y_j$$

(all we are doing is re-weighting the the entries in the Hat-matrix to account for observation k not being included). We can now write:

$$\hat{y}_k(-k) = \sum_{j \neq k} h_{kj} y_j + h_{kk} \hat{y}_k(-k).$$

Now, the squared prediction error for observation k is

$$(y_k - \hat{y}_k(-k))^2 = (y_k - \sum_{j \neq k} h_{kj} y_j - h_{kk} \hat{y}_k(-k))^2 = (y_k - \sum_j h_{kj} y_j + h_{kk}(y_k - \hat{y}_k(-k)))^2$$

But that means that we can write the LOOCV error as

$$\frac{1}{n} \sum_k (y_k - \hat{y}_k(-k))^2 = \frac{1}{n} \sum_k \big(\frac{y_k - \hat{y}_k}{1 - h_{kk}} \big)^2.$$

We can thus compute the LOOCV error from the original fit $\hat{y}_k$ by just re-weighting the residuals by the factor $1 - h_{kk}$. These residuals:

$$\tilde{e}_k = \frac{e_k}{1 - h_{kk}}$$

are called the studentized residuals. Note, the leverage values h_{kk} of course depend on the model we use since $h_{kk} = x_k (X'X)^{-1} x_k'$, where the columns of X are specific for each model.

I apply this strategy to the simulated data above and extract the winning model (with minimum PE):

```
> winmod <- Models[which.min(PE), ]
> print(winmod)

intercept        b1        b2        b3
     TRUE      TRUE      TRUE      TRUE
```

3 Demo 8

We revisit the South-African heart disease data. We start by enumerating all models to investigate.

```
> SA <- data.frame(read.table("SA.dat", sep = "\t", header = T))
> yy <- SA[, 12]
> xx <- SA[, -12]
> rleaps <- regsubsets(xx, yy, int = T, nbest = 250, nvmax = 250,
+     really.big = T, method = c("ex"))
> cleaps <- summary(rleaps, matrix = T)
> Models <- cleaps$which
> Models <- rbind(c(T, rep(F, dim(xx)[2])), Models)
```

Let's try 10-fold cross-validation. First, we create the 10 folds of data:

```
> K <- 10
> ii <- sample(seq(1, length(yy)), length(yy))
> foldsize <- floor(length(yy)/K)
> sizefold <- rep(foldsize, K)
> restdata <- length(yy) - K * foldsize
> if (restdata > 0) {
+     sizefold[1:restdata] <- sizefold[1:restdata] + 1
+ }
```

We will now cycle trough each fold of data, fit each of the models and compute the prediction errors:

```
> Prederrors <- matrix(0, dim(Models)[1], K)
> iused <- 0
> Xmat <- as.matrix(cbind(rep(1, dim(xx)[1]), xx))
> for (k in (1:K)) {
+     itest <- ii[(iused + 1):(iused + sizefold[k])]
+     itrain <- ii[-c((iused + 1):(iused + sizefold[k]))]
+     iused <- iused + length(itest)
+     for (mm in (1:dim(Models)[1])) {
+         betahat <- solve(t(Xmat[itrain, Models[mm, ]]) %*% Xmat[itrain,
+             Models[mm, ]]) %*% t(Xmat[itrain, Models[mm, ]]) %*%
+             yy[itrain]
+         ypred <- Xmat[itest, Models[mm, ]] %*% betahat
+         Prederrors[mm, k] <- sum((yy[itest] - ypred)^2)
+     }
+ }
> PE <- apply(Prederrors, 1, sum)/length(yy)
```

There are more than 1400 models in the above comparison: here are the top 5

```
> jj <- sort.list(PE)[1:5]
> print(as.matrix(Models[jj, ]))

  (Intercept)  age   sbp adiposity obesity typea alcohol alcind tobacco tobind
6        TRUE TRUE FALSE      TRUE   FALSE  TRUE    TRUE  FALSE   FALSE   TRUE
7        TRUE TRUE FALSE      TRUE   FALSE  TRUE    TRUE  FALSE   FALSE   TRUE
5        TRUE TRUE FALSE      TRUE   FALSE FALSE    TRUE  FALSE   FALSE   TRUE
6        TRUE TRUE FALSE      TRUE   FALSE FALSE    TRUE  FALSE   FALSE   TRUE
6        TRUE TRUE FALSE      TRUE    TRUE FALSE    TRUE  FALSE   FALSE   TRUE
   chd famhist
6 TRUE   FALSE
7 TRUE    TRUE
5 TRUE   FALSE
6 TRUE    TRUE
6 TRUE   FALSE

> z <- data.frame(as.matrix(cbind(Prederrors[jj, ], PE[jj])))
> colnames(z) <- c("Fold1", "Fold2", "Fold3", "Fold4", "Fold5",
+     "Fold6", "Fold7", "Fold8", "Fold9", "Fold10", "PE")
> row.names <- c(seq(1, 5))
> xtable(z, digits = c(0, rep(0, 10), 3), caption = "Prederrors in different folds and total",
+     label = "tab:CVldl")
```

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	PE
1	127	87	136	68	127	78	73	40	78	66	2.820
2	128	88	135	63	125	78	73	49	77	64	2.822
3	127	89	135	69	124	78	72	41	78	67	2.822
4	128	90	135	64	121	78	72	50	77	66	2.823
5	128	87	133	68	123	76	74	44	79	68	2.824

Table 4: Prederrors in different folds and total

In Table 4 you can compare the fold results and total prediction error results for the different models (seen in the R output). The winning model is

```
> winmod <- Models[which.min(PE), ]
> print(winmod)

(Intercept)         age         sbp   adiposity     obesity       typea
       TRUE        TRUE       FALSE        TRUE       FALSE        TRUE
    alcohol      alcind     tobacco      tobind         chd     famhist
       TRUE       FALSE       FALSE        TRUE        TRUE       FALSE
```

Let's compare this to the backward model selection results:

```
> mm <- lm(log(ldl) ~ log(age) + sbp + adiposity + log(obesity) +
+     typea + alcohol + alcind + tobacco + tobind + as.factor(chd) +
+     as.factor(famhist), data = SA)
> ss <- step(mm, trace = F)
> print(ss)

Call:
lm(formula = log(ldl) ~ adiposity + log(obesity) + alcohol +
    tobind + as.factor(chd) + as.factor(famhist), data = SA)

Coefficients:
        (Intercept)            adiposity         log(obesity)
          -0.295887             0.019417             0.342913
             alcohol              tobind      as.factor(chd)1
          -0.003449             0.143639             0.154434
as.factor(famhist)2
           0.071745
```

What about the LOOCV model?

```
> PE <- rep(0, dim(Models)[1])
> for (mm in (1:dim(Models)[1])) {
+     modfit <- lm(yy ~ Xmat[, Models[mm, ]] - 1)
+     rstud <- rstudent(modfit) * summary(modfit)$sig
+     PE[mm] <- sum((rstud)^2)/length(yy)
+ }
```

The top 5 LOOCV models are

```
> jj <- sort.list(PE)[1:5]
> print(as.matrix(Models[jj, ]))
```

```
  (Intercept)  age   sbp adiposity obesity typea alcohol alcind tobacco tobind
6        TRUE TRUE FALSE      TRUE   FALSE FALSE    TRUE  FALSE   FALSE   TRUE
7        TRUE TRUE FALSE      TRUE    TRUE FALSE    TRUE  FALSE   FALSE   TRUE
7        TRUE TRUE FALSE      TRUE   FALSE  TRUE    TRUE  FALSE   FALSE   TRUE
8        TRUE TRUE FALSE      TRUE    TRUE  TRUE    TRUE  FALSE   FALSE   TRUE
7        TRUE TRUE FALSE      TRUE   FALSE FALSE    TRUE  FALSE    TRUE   TRUE
   chd famhist
6 TRUE    TRUE
7 TRUE    TRUE
7 TRUE    TRUE
8 TRUE    TRUE
7 TRUE    TRUE
```