Predictive Model Selection and Assessment using R

Feiming Chen

May 6, 2011

Contents

1 Introduction 1

2 Model Selection Based on Cross Validation (CV) 2

3 Examples of Model Selection and Assessment 3

4 R Functions for Model Selection and Assessment 3
   4.1 Model Selection from Comparison of Estimated Test Error Rates . . . . . . . . . . 4
   4.2 Best Model for Future Prediction . . . . . . . . . . . . . . . . . . . . . . . . . . . 6
   4.3 Model Assessment . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
       4.3.1 CV Error . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
       4.3.2 Confusion Matrix . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
       4.3.3 ROC Curve . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 7
       4.3.4 Calibration Plot . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . 10

1 Introduction

In statistics, a prediction problem is about predicting a response (either continuous or discrete) using a set of predictors (some of them may be continuous, others may be discrete). The solution to such a problem is to build a prediction model on a training sample. This process presents two (related) challenges:

Model Selection We often have many candidate models (e.g. regression, tree, neural net, SVM, etc). Each model may have many sub-models specified by hyper-parameters that need to be tuned for optimal prediction performance (e.g. variable selection, shrinkage/penalty factor, smoothing/complexity parameter, Bayesian hyper-priors, etc.).

In order to choose the (approximate) best model, we need to estimate objectively the performance of different models and their sub-models. The performance metric is usually the test error, also called generalization error, which is the expected prediction error over an independent test sample. In contrast, the training error is the expected prediction error over the training sample, which is not an objective performance measure as a model can be over-trained to fit nicely to the training sample but have poor prediction performance on new data.

Model Assessment Once we decide on a best model, we want to estimate its test error by making prediction on a new sample, which provides an objective judgement on the performance of this final model. The estimated test error is typically larger, but closer to truth, than the one obtained at the Model Selection stage. This is because the very action of choosing the best model according to the estimated test errors across all candidate models is tantamount to training, at the risk of overfitting, on the validation part of the data that generates the test errors. For example, if we have 1000 candidate models that have equal
true test errors, then the selected model will, by pure chance, have the lowest estimated test error, which is highly likely to be lower than the true test error. This suggests that the test error of the best model estimated from the Model Selection stage cannot be trusted and it is desirable to estimate it again in an unbiased way by using a new independent and uncontaminated sample.

Besides the test error, other model assessment tools such as ROC Curve or Calibration Plot may be useful.

To estimate the test errors of different models and to assess the final model objectively, we shall ideally split the data set into three parts[1]:

**Training Sample** It is used for model estimation by estimating the model parameters. It can be 50% of the data.

**Validation Sample** It is used for model selection by estimating the test error of each candidate model. It can be 25% of the data.

**Test Sample** It is used for model assessment by estimating the test error of the final chosen model. It can be the remaining 25% of the data.

If the data is insufficient to be split into three parts, we can drop the Test Sample if we can accept (slightly) biased model assessments. In addition, we can drop the Validation Sample and use CV (Cross Validation) or Bootstrap method on the Training Sample to estimate the test errors to do model selection. Basically, the CV or Bootstrap method use the training sample efficiently by generating internal small validation samples. Alternatively we can use analytical estimators such as AIC or BIC, which may not be available for some models. In general, my recommendation is to use 5-fold or 10-fold CV.

## 2 Model Selection Based on Cross Validation (CV)

Because analytical model selection metrics such as AIC or BIC are not universally available for all models (e.g. trees, SVM), we usually use Cross Validation(CV) or Bootstrap for model selection[1]. In general, CV and Bootstrap have similar performance. However, the Bootstrap method (e.g. the “.632+” estimator) is generally more computational intensive than CV. What’s more, the concept of CV is simple and easy to communicate. Thus we recommend CV in general.

### How CV Works
Consider a 10-fold CV, we would split the Training Sample into 10 roughly equal-sized parts. For the 1st part, we fit the model to the other 9 parts of the data, and calculate the prediction error, $\epsilon_1$, of the fitted model on this 1st part of the data; then we repeat this process on the 2nd part of data, producing $\epsilon_2$, and so on and so forth. Note in this way each observation is predicted exactly once, as an out-of-sample prediction. In the end, we will have 10 error estimates: $\epsilon_1, \epsilon_2, \ldots, \epsilon_{10}$, which can be averaged into one $\epsilon$ (CV Error) and which can be used to compute the standard error of $\epsilon$. Alternatively, if the computing resource is ample, then we can, for example, repeat the CV procedures, using different 10-fold partitions each time, for 20 times and estimate the standard error of the CV Error.

In the model selection process, it would be very informative to plot out this CV Error, along with its standard error bar, to facilitate the model comparison. Often we apply a one-standard error rule[1], in which we choose the most parsimonious model whose error is no more than one standard error above the error of the best model.

### Comparison Between 5-fold CV (CV-5) and 10-fold CV (CV-10)

**Bias** If the training sample size is small, CV will in general overestimate the test error (because the model does not see enough data to show its full “strength”). This estimation bias will be negligible if we have sufficient sample size.
For CV-5, each training model uses only 80% of the Training Sample. For CV-10, 90%. Thus the bias of CV-5 error is higher than that of CV-10.

**Variance** For the same reason above (CV-5: 80%; CV-10: 90%), the training models of CV-10 are more similar (correlated) to each other than those of CV-5. Thus the variance of CV-10 error is higher than that of CV-5.

**Computing Time** CV-10, which fits a model 10 times, will take about twice as long to run in computer as CV-5, which fits a model 5 times.

Since it is generally hard to know the bias-variance trade-off between CV-5 and CV-10, we recommend using the computing time to make a choice: if a model takes a long time to fit, use CV-5; otherwise use CV-10.

### 3 Examples of Model Selection and Assessment

We provide some examples and notes to illustrate the process of model selection and assessment.

**Selection of hyper-parameter** Consider a best subset regression of size $p$. We can choose $p_{\text{best}}$ by using CV-10 on Training Sample. Thus we may not need Validation Sample here. Then we fit a new best subset regression of size $p_{\text{best}}$ on the entire Training Sample. The fitted regression is then assessed using the Test Sample, yielding an estimate of test error, say, RMSE (Root Mean Squared Error). When we are satisfied with this model, we can combine Training, Validation, and Test Samples together and fit a new best subset regression of size $p_{\text{best}}$ on the pooled data. This final model can then be deployed in Production, where it is used for prediction on new data set.

**Feature selection** Suppose we have a data of 100 observations and 1000 variables. Naturally we want to select some important variables before modeling. Unless we are using unsupervised learning methods (e.g. PCA, ICA, etc.) to select/summarize the variables, any supervised learning method for selecting variables, where the knowledge of the response is utilized, shall be carried out independently and repeatedly in each training set of the CV procedure. In other words, CV is fair only if variable selection is considered as part of the prediction model.

**ROC curve** When the response is dichotomous, we often use two metrics in lieu of test error:

- Sensitivity = 1 – (Misclassification Error when Response = TRUE)
- Specificity = 1 – (Misclassification Error when Response = FALSE)

There is often a key tuning parameter, called Cut-off, that dictates the trade-off between sensitivity and specificity. The trade-off can be illustrated by the ROC (Receiver Operating Characteristic) curve.

When the cut-off variable is the predicted response probability, we should use the out-of-sample predictions from CV to choose the optimum cut-off. Although it may seem more sensible to treat the cut-off as part of the model parameter and optimize it in each fold of the CV, it is cumbersome and not worth the effort to do so since only one degree of freedom is spent for the cut-off. Indeed, it is best not to have cut-off at all since it causes information loss by converting a continuous variable (predicted probability) into a categorical variable (0-1 response). A Brier score, although less interpretable than sensitivity/specificity, is recommended as the measure of test error. Nevertheless, the cut-off is often needed if sensitivity/specificity are required (e.g. by FDA) to be reported.

### 4 R Functions for Model Selection and Assessment

We use the statistical programming language R[2] to perform model selection and assessment.

---

[1] For CV-5, each training model uses only 80% of the Training Sample. For CV-10, 90%. Thus the bias of CV-5 error is higher than that of CV-10.

[2] For CV-5, each training model uses only 80% of the Training Sample. For CV-10, 90%. Thus the bias of CV-5 error is higher than that of CV-10.
**Example Data** We use Fisher’s Iris data for illustration, where the response *Species* is discrete with three levels and the 4 predictors are on continuous scale.

```r
> data(iris)
```

### 4.1 Model Selection from Comparision of Estimated Test Error Rates

We use the `errorest` function in the R `ipred` package, which by default uses a 10-fold CV to estimate the test error of any given model. The `errorest` function requires a `predict` function that returns a predicted class (or scalar). The return of `errorest` is a single estimated test error from the CV. Thus in order to get a standard error of the CV error, we can repeat it for, say, 20 times.

```r
> library(ipred)
> set.seed(1)  # fix random seed to make this report reproducible!

**CART (Classification and Regression Tree)** The `rpart` function in the R library `rpart` is used to fit a Recursive Partitioning and Regression Trees (a.k.a. CART).

```r
> library(rpart)
> mypredict.rpart <- function(object, newdata)
+   predict(object, newdata, type="class")
> errorest(Species ~ ., data = iris, model = rpart, predict=mypredict.rpart)
```

Call: errorest.data.frame(formula = Species ~ ., data = iris, model = rpart, predict = mypredict.rpart)
10-fold cross-validation estimator of misclassification error
Misclassification error: 0.0533

**Random Forest** The R library `randomForest` implements Breiman’s random forest algorithm (based on Breiman and Cutler’s original Fortran code) for classification and regression.

```r
> library(randomForest)
> errorest(Species ~ ., data = iris, model = randomForest)$error

[1] 0.04
```

**Naive Bayes Classifier** The `naiveBayes` function in the R library `e1071` computes the conditional a-posterior probabilities of a categorical class variable given independent predictor variables using the Bayes rule.

```r
> library(e1071)
> mypredict.naiveBayes <- function(object, newdata) {
+   predict(object, newdata[,-1])  # assume 1st column of "newdata" is
+   # response, rest is predictors.
+ }
> errorest(Species ~ ., data = iris, model = naiveBayes,
+   predict=mypredict.naiveBayes)$error

[1] 0.04667
```
Nearest Neighbor Classifier  The ipredknn function in the R library class implements the k-nearest neighbour classification. By default, $k = 5$.

```r
> library(class)
> mypredict.knn <- function(object, newdata)
+   predict.ipredknn(object, newdata, type="class")
> errorest(Species ~ ., data = iris, model = ipredknn,
+   predict=mypredict.knn)$error
[1] 0.02
```

LVQ (Learning Vector Quantization)  The lvq3 function in the R library class moves examples in a codebook to better represent the training set.

```r
> library(class)
> mymodel.lvq <- function(formula, data) {
+   f <- model.frame(formula, data)  # 1st column is response
+   cd <- lvqinit(f[,[-1]], f[,1])
+   cd3 <- lvq3(f[,[-1]], f[,1], cd)
+   cd3
+ }
> mypredict.lvq <- function(object, newdata)
+   lvqtest(object, newdata[,-1])
> errorest(Species ~ ., data = iris, model = mymodel.lvq,
+   predict=mypredict.lvq)$error
[1] 0.04
```

LDA (Linear Discriminant Analysis)  The lda function in the R library MASS performs the Linear Discriminant Analysis.

```r
> library(MASS)
> mypredict.lda <- function(object, newdata)
+   predict(object, newdata)$class
> errorest(Species ~ ., data = iris, model = lda, predict=mypredict.lda)$error
[1] 0.02
```

FDA (Flexible Discriminant Analysis)  The fda function in the R library mda performs the Flexible Discriminant Analysis.

```r
> library(mda)
> errorest(Species ~ ., data = iris, model = fda, method=mars)$error
[1] 0.03333
```

MDA (Mixture Discriminant Analysis)  The mda function in the R library mda performs the Mixture Discriminant Analysis.

```r
> library(mda)
> errorest(Species ~ ., data = iris, model = mda)$error
[1] 0.02667
```
ANN (Artificial Neural Networks) The \texttt{nnet} function in the R library \texttt{nnet} fits a single-hidden-layer neural network, possibly with skip-layer connections.

\begin{verbatim}
> library(nnet)
> mypredict.nnet <- function(object, newdata)
+   as.factor(predict(object, newdata, type="class"))
> errorest(Species ~ ., data = iris, model = nnet, predict=mypredict.nnet,
+   decay=0.01, size=10, trace=F)$error
[1] 0.03333
\end{verbatim}

SVM (Support Vector Machines) The \texttt{svm} function in the R library \texttt{e1071} is used to train a support vector machine. It can be used to carry out general regression and classification (of \texttt{nu} and epsilon-type), as well as density-estimation.

\begin{verbatim}
> library(e1071)
> errorest(Species ~ ., data = iris, model = svm)$error
[1] 0.03333
\end{verbatim}

Multinomial Logit Model The \texttt{multinom} function in the R library \texttt{nnet} fits a multinomial log-linear model via neural networks.

\begin{verbatim}
> library(nnet)
> mypredict.multinom <- function(object, newdata)
+   as.factor(predict(object, newdata, type="class"))
> errorest(Species ~ ., data = iris, model = multinom,
+   predict=mypredict.multinom, trace=F)$error
[1] 0.02667
\end{verbatim}

4.2 Best Model for Future Prediction

By examining the CV errors and, optionally, their standard errors, of all the candidate models, we can select a best model. The selection shall also be biased toward simpler models, using the principal of Occam’s Razor. For the predictive models we have considered for the \texttt{iris} data set, we have:

\begin{center}
\begin{tabular}{lc}
\hline
Models & 10-fold CV Error \\
\hline
CART & 0.0533 \\
Random Forest & 0.04 \\
Naive Bayes & 0.0467 \\
Nearest Neighbor & 0.02 \\
LVQ & 0.04 \\
LDA & 0.02 \\
FDA & 0.0333 \\
MDA & 0.0267 \\
ANN & 0.0333 \\
SVM & 0.0333 \\
Multinomial Logit & 0.0267 \\
\hline
\end{tabular}
\end{center}

Many models are more or less equivalent. But the LDA (Linear Discriminant Analysis), due to its simplicity and small memory profile (compare to Nearest Neighbor), seems to be a good choice.

Suppose we select LDA as the winning model. Let’s ascertain the standard deviation of its CV error by repeating the 10-fold CV 20 times.
> set.seed(1)
> ans <- replicate(20, errorest(Species ~ ., data = iris, model = lda,
+ predict=mypredict.lda)$error)
> sd(ans)
[1] 0

We get zero spread (this is an unusual example). So the CV error should be quite close to
the true test error.

For future predictions, we can train the LDA model on the entire training set and use it on
future data.

> m <- lda(Species ~ ., data = iris)

4.3 Model Assessment

Various summary and graphical methods are available to assess a (selected) model. In the
following, we use the iris data to illustrate some assessment methods for categorical response
data. When the response variable is continuous, we can use residual diagnostic plots such as
those provided by the plot method for the lm or glm objects.

4.3.1 CV Error

As the starting point, we can report the CV error, the standard error of the CV error, and, if
available, the test error on the Test Sample.

<table>
<thead>
<tr>
<th>Models</th>
<th>CV Error</th>
<th>Standard Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LDA</td>
<td>0.02</td>
<td>0+</td>
</tr>
</tbody>
</table>

4.3.2 Confusion Matrix

For categorical response, we can display the prediction performance via Confusion Matrix, using
the out-of-sample predictions from the CV procedure.

> predicted <- errorest(Species ~ ., data = iris,
+ model = lda, predict=mypredict.lda,
+ est.para= control.errorest(predictions=TRUE))$predictions
> library(xtable) # make LaTeX Table
> xtable(table(iris$Species, predicted), caption="Confusion Matrix")

<table>
<thead>
<tr>
<th></th>
<th>setosa</th>
<th>versicolor</th>
<th>virginica</th>
</tr>
</thead>
<tbody>
<tr>
<td>setosa</td>
<td>50</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>versicolor</td>
<td>0</td>
<td>48</td>
<td>2</td>
</tr>
<tr>
<td>virginica</td>
<td>0</td>
<td>1</td>
<td>49</td>
</tr>
</tbody>
</table>

Table 1: Confusion Matrix

4.3.3 ROC Curve

When the response variable is binary (e.g. positive or negative for a certain disease), the pre-
dicted response is typically a probability. In this case we can use the ROC (Receiver Operating
Characteristics) curve to plot the Sensitivity against Specificity by varying a threshold parameter
(e.g. cut-off). Note if we use the entire training data to fit a model, the ROC curve may appear
too good due to overfitting. It is desirable to use the out-of-sample predictions from the CV
procedure to make the ROC curve.
A Wrapper Function to Make ROC Plot  The R package `ROCR` can make a nice ROC plot. We provide the following wrapper function. In general we want the cut-off to be close to the upper-right corner, where sensitivity and specificity are equal to 1. We overlay a diagonal line (line of no-discrimination), on which the model is equivalent to a random coin toss that bears no correlation with the response.

```r
roc.curve <- function(p, l, cutoff=quantile(p, seq(0,1,by=0.2)), ...) {
## Make a ROC plot.
+ ## Wrapper to plot ROC curve by using R package ROCR
+ ## "p" is a vector of prediction (continuous);
+ ## "l" is a vector of truth (0/"negative" or 1/"positive")
+ ## "cutoff" is a list of values to be plotted on ROC curve.
+ ## "..." is a passed to "plot".
+ ## RETURN: a list of ROC curve coordinates and cut-offs.
+ 
+ par(pty="s")
+ require(ROCR)
+ a <- prediction(p, l)
+ ## ROC for Sensitivity vs. Specificity.
+ plot((pp <- performance(a, "sens", "spec")), colorize=T,
+      print.cutoffs.at=cutoff, text.adj=c(1.2, 1.2), text.cex=0.7, lwd=2, las=1,
+      ...) 
+ grid(col="orange")
+ ## Draw a "line of no-discrimination".
+ ## Sens = P(X=+ | T=+), Spec = P(X=- | T=-),
+ ## if X is independent of T, then Sens + Spec = P(X=)+P(X-) = 1, so the pair
+ ## (Sens, Spec) lies on a diagonal line.
+ abline(c(1, -1), col="gray70", lty=2)
+ par(pty="m")
+ 
+ invisible(pp)
+ }
```

ROC Curve Based on Training Sample  We first make a binary response variable `is.versicolor` from the categorical response `Species` in the `iris` data. Then we build a logistic regression model on the entire training data and make in-sample predictions. The resulting ROC curve will appear too optimistic.

```r
d <- transform(iris, is.versicolor = factor(Species == "versicolor"))[,,-5]
p <- predict(glm(is.versicolor ~ ., data = d, family=binomial),
     type="response")
roc.curve(p, d$is.versicolor,
     main="ROC Curve for Prediction of Versicolor: Training Sample")
```
ROC Curve for Prediction of Versicolor: Training Sample

Predicted Probabilities from CV  An unbiased way to make the ROC Plot is to use the genuine forcasts from the CV procedure. The following functions will get the out-of-sample probability predictions from CV.

```r
> cv.idx <- function(n, k=10) {
+   ## Generate a list of obs indexes for CV: indicating which observations are
+   ## included in each cross-validation sample (as the test sample).
+   ## "k": k-fold CV, default to 10 (10-fold CV)
+   ## "n": sample size.
+   ## RETURN: a list of obs indexes for each fold of CV.
+   require(ipred)
+   cv.sizes <- ipred::kfoldcv(k, n) # equal partition of data to k-fold
+   bracket <- embed(c(0, cumsum(cv.sizes)), 2) # bracket for each fold.
+   idx <- sample.int(n) # a random permutation of indexes.
+   ans <- list()
+   for (i in seq(nrow(bracket)))
+     ans[[i]] <- idx[(bracket[i, 2] + 1) : bracket[i, 1]] # i-fold CV sample idx
+   ans
+ }
> if (F) {
+   cv.idx(100)
+
+ }  
> CV.prob <- function(formula, data, model, predict = predict, k=10, ...) {
+   ## Get k-fold CV Probabilities (out-of-sample predictions).
+   ## "model"/"predict": a model/predict function that has formula interface.
+   ## "..." is passed to "model".
+   +
+   +   n <- nrow(data)
```
ROC Curve Based on CV Sample

With the out-of-sample probability predictions from the CV procedure, the resulting ROC curve will be closer to truth.

```r
> mypredict.glm <- function(object, newdata) predict(object, newdata, type="response")
> pp <- CV.prob(is.versicolor ~ ., data = d,
+ model = glm, predict=mypredict.glm, family=binomial)
> roc.curve(pp, d$is.versicolor,
+ main="ROC Curve for Prediction of Versicolor: CV Sample")
```

4.3.4 Calibration Plot

When we predict a categorical response with probabilities, we want these predicted probabilities to be well calibrated, in the sense that a fraction of about $p$ of the events we predict with
probability $p$ actually occur.

**A R function for making a Calibration Plot** We make a calibration plot where a diagonal red line indicates the ideal calibration. The blue line, estimated from the model predictions, is expected to be as close to the red line as possible. A point-wise 95% confidence band for the blue line is shaded in gray. If, however, the blue line is close to the yellow no-discrimination line ($y = 0.5$) instead, it indicates the failure of the model. The raw binary response is shown as ticks on the bottom and top of the plot.

```r
> calibrate.plot <- function (y, p, main="Calibration Plot") {
+ ## Make Calibration Plot: a fraction of about $p$ of the events we
+ ## predict with probability $p$ actually occur. An ideal calibration
+ ## plot (of Observed Average against Predicted Value) should show a
+ ## straight diagonal line, which connects between (0,0) and (1,1).
+ ## y: the outcome binary 0-1 variable.
+ ## p: the probability predictions (between 0 and 1) estimating $E(y|x)$.
+ + newp <- seq(0, 1, length=100)
+ + yy <- predict(loess(y ~ p, span=1), newp, se=T)
+ + yy.ok <- !is.na(yy$fit)
+ + yy$fit <- yy$fit[yy.ok]
+ + yy$se.fit <- yy$se.fit[yy.ok]
+ + newp <- newp[yy.ok]
+ + se.lower <- yy$fit - 2 * yy$se.fit # confidence band
+ + se.upper <- yy$fit + 2 * yy$se.fit
+ + par(pty="s")
+ + plot(c(0,1), c(0,1), type="n", xlab="Predicted Probability",
+ + ylab="Observed Proportions", xaxs="i", yaxs="i", las=1, main=main)
+ + polygon(c(newp, rev(newp), newp[1]),
+ + c(se.lower, rev(se.upper), se.lower[1]),
+ + col = "gray", border = NA)
+ + rug(p[y == 0], side=1, col="navy")
+ + rug(p[y == 1], side=3, col="navy")
+ + abline(0, 1, col="red") # perfect calibration line
+ + abline(h=0.5, col="yellow", lty=2) # perfect irrelevance (no-discrimination) line
+ + lines(newp, yy$fit, lwd=2, col="blue")
+ + par(pty="m")
+ }
```

**Calibration Plot Based on Training Sample** On a model fitted to a training data, the Calibration Plot will appear too optimistic.

```r
> y <- ifelse(d$is.versicolor == "TRUE", 1, 0)
> calibrate.plot(y, p,
+ main="Calibration Plot for Prediction of Versicolor: Training Sample")
```

11
Calibration Plot for Prediction of Versicolor: Training Sample

Calibration Plot Based on CV Sample By using the true forecasts from the CV procedure, we will make a more realistic Calibration Plot.

> calibrate.plot(y, pp,
+ main="Calibration Plot for Prediction of Versicolor: CV Sample")
References

