Crosss Validation of Class Predictions

Kevin R. Coombes

April 23, 2019

Contents

1 Introduction

When building models to make predictions of a binary outcome from omics-scale data, it is especially useful to thoroughly cross-validate those models by repeatedly splitting the data into training and test sets. The *CrossValidate* package provides tools to simplify this procedure.

2 A Simple Example

We start by loading the package

```
> library(CrossValidate)
```

Now we simulate a data set with no structure that we can use to test the methods.

```
> set.seed(123456)
> nFeatures <- 1000
> nSamples <- 60
> pseudoclass <- factor(rep(c("A", "B"), each = 30))
> dataset <- matrix(rnorm(nFeatures * nSamples), nrow = nFeatures)
```

Now we pick a model that we would like to cross-validate. To start, we will use K nearest neighbors (KNN) with $K = 3$.

```
> model <- modeler5NN
```

The we invoke the cross-validation procedure.

```
> cv <- CrossValidate(model, dataset, pseudoclass, frac = 0.6, nLoop = 30)
```

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] 5
[1] 6
[1] 7
[1] 8
[1] 9
[1] 10
[1] 11
[1] 12
[1] 13
[1] 14
[1] 15
[1] 16
[1] 17
[1] 18
[1] 19
[1] 20
[1] 21
[1] 22
[1] 23
[1] 24
[1] 25
[1] 26
[1] 27
[1] 28
[1] 29
[1] 30
```

By default (`verbose = TRUE`), the cross validation procedure prints out a counter for each iteration. This behavior can be overridden by setting `verbose = FALSE`.

```
> summary(cv)
```

```
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.5555556 0.3888889 0.5833333 0.5600000 0.5789474
1st Qu. 0.7222222 0.5000000 0.6250000 0.6250000 0.6593750
Median  0.8055556 0.6111111 0.7083333 0.6666667 0.7647059
Mean    0.7851852 0.6074074 0.6962963 0.6705333 0.7512223
3rd Qu. 0.8888889 0.7222222 0.7500000 0.7331871 0.8221925
Max.    1.0000000 0.7777778 0.8333333 0.8000000 1.0000000
```

```
Validation Accuracy:
             sens       spec       acc       ppv       npv
Min.    0.2500000 0.08333333 0.2916667 0.3529412 0.1428571
1st Qu. 0.5000000 0.33333333 0.4583333 0.4593301 0.4093750
Median  0.6250000 0.41666667 0.5208333 0.5147059 0.5227273
Mean    0.6083333 0.41944444 0.5138889 0.5163261 0.5015829
3rd Qu. 0.7291667 0.58333333 0.5729167 0.5846154 0.5714286
Max.    0.9166667 0.66666667 0.7500000 0.6875000 0.8750000
```

The summary reports the performance separately on the training data and the testing data. In this case, KNN overfits the training data (getting roughly 70% of the "predictions" correct) but is no better than coin toss on the test data.

3 Testing Multiple Models

A primary advantage of defining a common interface to different classification methods is that you can write code that tests them all in exactly the same way. For example, let's suppose that we want to compare the KNN method above to the method of compound covariate predictors. We can then do the following.

```
> models <- list(KNN = modeler5NN, CCP = modelerCCP)
> results <- lapply(models, CrossValidate,
+                   data = dataset, status = pseudoclass,
+                   frac = 0.6, nLoop = 30, verbose = FALSE)
> lapply(results, summary)
```

```
$KNN
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.5000000 0.3333333 0.5833333 0.5555556 0.6086957
1st Qu. 0.6805556 0.6111111 0.6666667 0.6557971 0.6710526
Median  0.7777778 0.6666667 0.6944444 0.6830144 0.7416667
Mean    0.7648148 0.6555556 0.7101852 0.6945616 0.7440793
3rd Qu. 0.8333333 0.7222222 0.7500000 0.7385584 0.7894737
Max.    0.9444444 0.8333333 0.8333333 0.8235294 0.9230769

Validation Accuracy:
             sens       spec       acc       ppv       npv
Min.    0.4166667 0.08333333 0.3750000 0.3846154 0.2000000
1st Qu. 0.5208333 0.33333333 0.4583333 0.4575758 0.4196429
Median  0.5833333 0.33333333 0.4791667 0.4880952 0.4807692
Mean    0.6138889 0.38888889 0.5013889 0.5016199 0.4979113
3rd Qu. 0.6666667 0.50000000 0.5416667 0.5437063 0.5714286
```

```
Max.    0.9166667 0.58333333 0.7083333 0.6666667 0.8000000

$CCP
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
        sens spec acc ppv npv
Min.       1    1   1   1   1
1st Qu.    1    1   1   1   1
Median     1    1   1   1   1
Mean       1    1   1   1   1
3rd Qu.    1    1   1   1   1
Max.       1    1   1   1   1

Validation Accuracy:
             sens       spec       acc       ppv       npv
Min.    0.0000000 0.08333333 0.2500000 0.0000000 0.1250000
1st Qu. 0.3541667 0.41666667 0.4166667 0.4000000 0.4166667
Median  0.4166667 0.45833333 0.4583333 0.4545455 0.4545455
Mean    0.4138889 0.45833333 0.4361111 0.4246593 0.4309654
3rd Qu. 0.5000000 0.58333333 0.4895833 0.4903846 0.4926471
Max.    0.6666667 0.75000000 0.6250000 0.6153846 0.6363636
```

The performance of KNN with this set of training-test splits is simila to the previous set. The CCP metod, by contrast, behaves much worse. It perfectly fits (and so overfits) the training data and consequently actually manages to do *worse* than chance on the test data.

4 Filtering and Pruning

Having a common interface also lets us write code that combines the same modeling method with different algorothms to filter genes (by something like mean expression, for example) or to perform feature selection (using univariate t-tests, for example). Many such methods are prvoided by the *Modeler* package on which *CrossValidate* depends. Here we show how to combine the KNN method with several different methods to preprocess the set of features.

Here we show how to do this the wrong way.

```
> pruners <- list(ttest = fsTtest(fdr = 0.05, ming = 100),
+                 cor = fsPearson(q = 0.90),
+                 ent = fsEntropy(q = 0.90, kind = "information.gain"))
> for (p in pruners) {
+   pdata <- dataset[p(dataset, pseudoclass),]
+   cv <- CrossValidate(model, pdata, pseudoclass, 0.6, 30, verbose=FALSE)
+   show(summary(cv))
+ }
```

```
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.8888889 0.8333333 0.9166667 0.8571429 0.8947368
1st Qu. 1.0000000 0.9444444 0.9444444 0.9419935 1.0000000
Median  1.0000000 0.9444444 0.9722222 0.9473684 1.0000000
Mean    0.9814815 0.9444444 0.9629630 0.9488882 0.9827096
3rd Qu. 1.0000000 1.0000000 0.9722222 1.0000000 1.0000000
Max.    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000

Validation Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.7500000 0.5000000 0.7083333 0.6470588 0.7857143
1st Qu. 0.9166667 0.8333333 0.8437500 0.8333333 0.8701299
Median  0.9166667 0.8333333 0.8750000 0.8571429 0.9230769
Mean    0.9333333 0.8611111 0.8972222 0.8781415 0.9328116
3rd Qu. 1.0000000 0.9166667 0.9583333 0.9230769 1.0000000
Max.    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.8333333 0.7222222 0.8611111 0.7826087 0.8571429
1st Qu. 0.9444444 0.8888889 0.9166667 0.8960526 0.9444444
Median  1.0000000 0.9444444 0.9444444 0.9459064 1.0000000
Mean    0.9759259 0.9314815 0.9537037 0.9380154 0.9761498
3rd Qu. 1.0000000 1.0000000 0.9930556 1.0000000 1.0000000
Max.    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000

Validation Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.7500000 0.5833333 0.7083333 0.6666667 0.7777778
1st Qu. 0.9166667 0.8333333 0.8750000 0.8392857 0.9000000
Median  0.9166667 0.9166667 0.9166667 0.9090909 0.9198718
Mean    0.9305556 0.8694444 0.9000000 0.8839455 0.9305413
3rd Qu. 1.0000000 0.9166667 0.9166667 0.9214744 1.0000000
Max.    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.
```

```
Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.7777778 0.6666667 0.7500000 0.7272727 0.7647059
1st Qu. 0.8888889 0.7361111 0.8055556 0.7687970 0.8595238
Median  0.8888889 0.8333333 0.8611111 0.8421053 0.8888889
Mean    0.9037037 0.8111111 0.8574074 0.8316587 0.8958485
3rd Qu. 0.9444444 0.8888889 0.8888889 0.8888889 0.9364583
Max.    1.0000000 0.9444444 0.9722222 0.9473684 1.0000000

Validation Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.5000000 0.3333333 0.5416667 0.5294118 0.5714286
1st Qu. 0.6666667 0.6041667 0.6666667 0.6428571 0.6887019
Median  0.7500000 0.6666667 0.7083333 0.7207792 0.7207792
Mean    0.7416667 0.6916667 0.7166667 0.7197803 0.7421660
3rd Qu. 0.8333333 0.7500000 0.7500000 0.7500000 0.7837302
Max.    1.0000000 1.0000000 0.8333333 1.0000000 1.0000000
```

We can tell that this method is wrong because the validation accuracy is much better than chance—which is impossible on a dataset without any true structure. The problem is that we have aplied the fature selection method to the combined (training plus test) dataset, which allows information from the test data to creep into the model building step.

Now we can do it the right way, with the feature selection step included inside the cross-validation loop.

```
> for (p in pruners) {
+   cv <- CrossValidate(model, dataset, pseudoclass, 0.6, 30,
+                       prune=p, verbose=FALSE)
+   show(summary(cv))
+ }

---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.9444444 0.9444444 0.9722222 0.9473684 0.9473684
1st Qu. 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Median  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Mean    0.9944444 0.9944444 0.9944444 0.9947368 0.9947368
3rd Qu. 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Max.    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000

Validation Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.1666667 0.1666667 0.2916667 0.2857143 0.2727273
1st Qu. 0.4166667 0.4166667 0.4583333 0.4469697 0.4545455
```

```
Median  0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
Mean    0.5194444 0.4972222 0.5083333 0.5070572 0.5096763
3rd Qu. 0.6666667 0.5833333 0.5833333 0.5870098 0.5674603
Max.    0.8333333 0.8333333 0.6666667 0.7500000 0.7500000
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.9444444 0.9444444 0.9722222 0.9473684 0.9473684
1st Qu. 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Median  1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Mean    0.9981481 0.9888889 0.9935185 0.9894737 0.9982456
3rd Qu. 1.0000000 1.0000000 1.0000000 1.0000000 1.0000000
Max.    1.0000000 1.0000000 1.0000000 1.0000000 1.0000000

Validation Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.2500000 0.1666667 0.2916667 0.2727273 0.2222222
1st Qu. 0.4166667 0.3541667 0.4166667 0.4392361 0.4375000
Median  0.5000000 0.5000000 0.5000000 0.5000000 0.5000000
Mean    0.5111111 0.4944444 0.5027778 0.5156090 0.4961891
3rd Qu. 0.6458333 0.6458333 0.5729167 0.5582386 0.5840336
Max.    0.8333333 0.9166667 0.7083333 0.8333333 0.7500000
---------------
Cross-validation was performed using 60 percent of the data for
training. The data set was randomly split into training and testing
sets 30 times.

Training Accuracy:
             sens      spec       acc       ppv       npv
Min.    0.6666667 0.6666667 0.8055556 0.7500000 0.7391304
1st Qu. 0.8333333 0.8472222 0.8888889 0.8650794 0.8571429
Median  0.9444444 0.9444444 0.9166667 0.9302885 0.9411765
Mean    0.9074074 0.9037037 0.9055556 0.9145730 0.9181831
3rd Qu. 1.0000000 1.0000000 0.9444444 1.0000000 1.0000000
Max.    1.0000000 1.0000000 0.9722222 1.0000000 1.0000000

Validation Accuracy:
             sens       spec       acc       ppv       npv
Min.    0.1666667 0.08333333 0.2500000 0.2727273 0.1428571
1st Qu. 0.4166667 0.33333333 0.5000000 0.5000000 0.5000000
Median  0.5833333 0.41666667 0.5000000 0.5000000 0.5000000
Mean    0.5611111 0.45277778 0.5069444 0.5127854 0.5085964
3rd Qu. 0.6666667 0.64583333 0.5833333 0.5770677 0.6000000
Max.    0.9166667 0.91666667 0.7500000 0.8750000 0.8000000
```