DME Lab 3: Evaluation in R

15 February, 2012

Hello! In this third and final lab we will do some classification and evaluation using R.

1. This time we will use a dataset available in R. Type the following commands to load the data into the system:

   ```
   > library(MASS)
   > data(cats)
   ```

   This loads the dataset `cats`, which contains three columns regarding the weights of the heart and body of both male and female cats. To know more about this dataset type `help(cats)` or take a look at the data by typing `cats` in the command line.

2. Again, you can get some statistics by doing `summary(cats)`.

3. Make the column names available in the environment: `attach(cats)`

4. Do some conditional histograms of both variables conditioned in the class (`sex`) by running `library(lattice)` and then `histogram(~Bwt | Sex)`. Do the same thing for the heart weight variable.

5. Now, inspect the correlation coefficient of both weight variables: `cor(Bwt,Hwt)`. What do you think?

6. Let's visualise this data in a scatterplot. Run:

   ```
   > col<-c("RED","BLUE")
   > plot(Bwt,Hwt,col=col[Sex])
   ```

   Do you see the two classes? Can you find a good decision boundary for them?

7. Now that you have visualised the data, let's begin with classification. First, let's split the dataset into train and test sets. Run:

   ```
   > index<-1:nrow(cats)
   > testindex<-sample(index, trunc(length(index)/3))
   > testset<-cats[testindex,]
   > trainset<-cats[-testindex,]
   ```

So, we have 1/3 of the dataset for testing and 2/3 for training.

8. Next, we will train an SVM classifier on this data, but first we have to install the package that contains this utility. Run:

   ```
   > install.packages('e1071')
   > library('e1071')
   ```

9. Now, we are ready to train our classifier. Take a look at the description of the `svm` function: `help(svm)`. After you get a sense of this function, run:

   `svm.linear<-svm(Sex~., data = trainset, probability=TRUE, cross=10, kernel="linear")`

   This command creates a SVM classifier based on our trainset, it uses a linear kernel and 10-fold cross validation. Take a look at the results of the classification: `plot(svm.linear, trainset)`.

10. Let's test the model by computing its predictions on the test set. Run:

    `svmlinear.preds<-predict(svm.linear, testset[,-1], probability=TRUE)`

    Inspect the contents of this new variables. What do you see? Is the model accurate?

11. Next, we build a confusion matrix. Type `confmat.linear <- table(pred = svmlinear.preds, true = testset[,1])` and inspect the contents of this new variable. What's the accuracy of this classifier?

12. Try changing the parameters of the SVM classifier:

    ```
    > svm.pol<-svm(Sex~., data = trainset, probability=TRUE, cross=10, kernel="polynomial")
    > plot(svm.pol, trainset)
    > svmpol.preds<-predict(svm.pol, testset[,-1], probability=TRUE)
    > confmat.pol <- table(pred = svmpol.preds, true = testset[,1])
    ```

    And take a look at this new confusion matrix.

13. Now that we are in the evaluation stage, we will plot an ROC curve for our results. Remember that a ROC curve shows the relationship between true positive rate and false positive rate for a binary classifier like our SVM.

    ROC curves are plotted in a 2D graph in which a *true positive rate* is represented in the y axis, whereas the *false positive rate* is represented in the x axis. There are some important points to note in this plot:

    - The point $(0, 0)$ represents a non-positive classification, this means that the classifier's output was negative all the time.
    - The point $(1, 1)$ on the other hand, represents all-positive classification all the time. That is, the output of the classifier was always positive. As you can imagine, in a real-world scenario both situations are incorrect.
    - The point $(0, 1)$ represents *perfect* classification, whereas the point $(1, 0)$ represents *perfect misclassification*, in which case we also succeed in the task (why is that?).

If you'd like to know more about ROC curves, you should take a look at this introduction by Tom Fawcett: `http://www.sciencedirect.com/science/article/pii/S016786550500303X`

Before we can plot ROC curves in R we must install the proper package, so this time do: `install.packages('ROCR')` and follow the instructions for installation. Then, type `library("ROCR")` to load the library.

14. In our case, the ROC curve will be drawn using the probabilities obtained from the SVM classification. Run:

```
> svmlinear.rocr<-prediction(attr(svmlinear.preds,"probabilities")[,2], testset[,1] == "M")
> svmlinear.perf<-performance(svmlinear.rocr, measure = "tpr", x.measure = "fpr")
> svmpol.rocr<-prediction(attr(svmpol.preds,"probabilities")[,2], testset[,1] == "M")
> svmpol.perf<-performance(svmpol.rocr, measure = "tpr", x.measure = "fpr")
> plot(svmlinear.perf,col="BLUE")
> plot(svmpol.perf,add=TRUE,col="RED")
```

15. Next, let's compute the *area under the curve* (AUC). An ROC curve is a visual representation of a classifier's performance. However, to compare classifiers we'd like to have a single scalar value: that's precisely the AUC, whose value lies in the interval $[0, 1]$. Because a random-guessing classifier would produce a line between $(0, 0)$ and $(1, 1)$, which yields an AUC of 0.5, no realistic classifier should have an AUC less than 0.5.

    To compute the AUC of our classifiers curve run:

    `svmlinear.auc<-as.numeric(performance(svmlinear.rocr, measure = "auc", x.measure = "cutoff")@ y.values)`

    Then: `svmpol.auc<-as.numeric(performance(svmpol.rocr, measure = "auc", x.measure = "cutoff")@ y.values)`

    Compare both values.

16. Based on what you've seen so far, which classifier do you think is the best? Can you think of any way to compare them?

17. We will perform McNemar's test to check if these classifiers differ significantly. This test helps to answer the following question: given two classifiers which one of them will be more accurate on new test examples. In this test, the null hypothesis states that both classifiers should have the same error rate. To know more about this test applied for machine learning algorithms read this paper Thomas Dietterich:

    `http://www.mitpressjournals.org/doi/abs/10.1162/089976698300017197`

    To proceed we will need to build a table like the one shown in class. This table must contain the number of instances misclassified by both algorithms (n_{00}), the number of instances misclassified by the first algorithm but not by the second one (n_{01}), the number of instances in the opposite situation (n_{10}), and finally the number of instances misclassified by neither of them (n_{11}). The sum of these four quantities is the total number of instances in the test set.

    Run the following commands to start building such a table:

```
> dummy1<-as.data.frame(predict(svm.linear, testset[,-1]),optional=TRUE)
> dummy2<-as.data.frame(predict(svm.pol, testset[,-1]),optional=TRUE)
```

18. Next, get the instances that were correct from the SVM classifier with linear kernel by running `correct.linear<-(dummy1[,] == testset[,1])`. Do the same for the SVM with polynomial kernel. (And name it `correct.pol`)

19. We will construct the contingency matrix by running the following commands:

    ```
    > n00<-sum(!correct.linear & !correct.pol)
    > n10<-sum(!correct.linear & correct.pol)
    > n01<-sum(correct.linear & !correct.pol)
    > n11<-sum(correct.linear & correct.pol)
    ```

    The above commands compute an elementwise *and* function. You can check this by doing: `head(correct.linear)`, then `head(correct.pol)` and finally `head(!correct.linear & correct.pol)`. You will see that this last command returns the conjunction of both vectors element by element.

20. By running `sum(n00,n10,n01,n11)` you can check that they effectively account for the totality of the test set.

21. To perform McNemar's test just type: `mcnemar.test(matrix(c(n00,n10,n01,n11),nrow=2))`

    Look at the `p-value`. What do you see? What can you say about the comparison of both classifiers and the null hypothesis?

22. Well, this concludes these series of labs. In them, you reviewed briefly the main stages of data mining, namely visualisation, classification and evaluation. Now, you should try to apply what you've learned in a more complex dataset like those proposed for your miniproject. ;-) Again, if you happen to have some spare time, I recommend the following links:

    (a) An interview with the co-author of -probably- the most popular book on artificial intelligence who's now head of Google's research:
    `http://www.wired.com/epicenter/2012/01/googles-go-to-ai-guy/`

    (b) What the future would look like in a networked society: `http://youtu.be/R7cuatm_bqw`