Introduction to Computational BioStatistics with R: classification II

Erik Spence

SciNet HPC Consortium

9 November 2021

SciNet compute • calcul CANADA

Today's slides

Today's slides can be found here. Go to the "Introduction to Computational BioStatistics with R" page, under Lectures, "Classification II".

`https://scinet.courses/1182`

SciNet compute • calcul CANADA

Today's class

Today we will visit the following topics:

- Logistic regression.
- ROC curves.
- Support vector machines.

Ask questions!

SciNet compute • calcul CANADA

Classification approaches

There are lots of classification approaches which one might use.

- Decision trees: analyze the features of the data and make 'decisions' about how to 'split' the data into uniform groups.
- Logistic regression: like linear regression, but now we fit a "yes/no" function to the data.
- Naive Bayes: a type of probabilistic analysis.
- kNN: k Nearest Neighbours; use the k nearest neighbours to a data point to predict the category of a new data point.
- Support Vector Machines: essentially a linear model of the data, used for separate groups.
- Neural networks: a weird algorithmic approach to using functions to categorize data.

Today we will go over logistic regression and support vector machines.

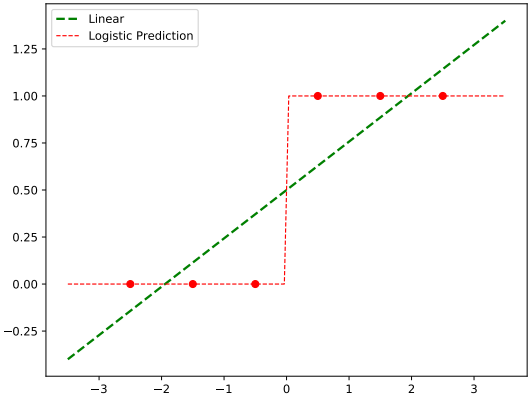
SciNet compute • calcul CANADA

Logistic regression

One way to consider binary classification is to go back to regression, and fit a linear regression to an integer 0/1 variable for classification: over 0.5, True, else False.

This requires a linear separation between the classes to be effective.

However, naive application of linear regression can lead to a number of problems, which grow with the number of dimensions. These are mostly related to the unbounded nature of the function.



Logistic regression, continued

A whole infrastructure exists for "generalized linear models", where the function being fit is not

$$y = \beta_0 + x_1\beta_1 + x_2\beta_2 + \cdots = \mathbf{x} \cdot \vec{\beta}$$

but rather some power or exponential of $\mathbf{x} \cdot \vec{\beta}$.

Consider instead fitting, not the probability p, but rather the log of the odds ratio,

$$\mu = \ln\left(\frac{p}{1-p}\right) = \mathbf{x} \cdot \vec{\beta}$$

We can fit this log-odds equation, and derive

$$p = \frac{e^{\mathbf{x}\cdot\vec{\beta}}}{1 + e^{\mathbf{x}\cdot\vec{\beta}}} = \frac{1}{1 + e^{-\mathbf{x}\cdot\vec{\beta}}}$$

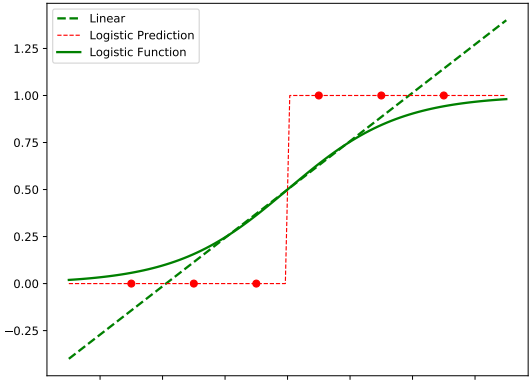
SciNet compute • calcul CANADA

Logistic regression, continued more

$$p = \frac{1}{1 + e^{-\mathbf{x}\cdot\vec{\beta}}}$$

This approach has a number of very nice properties:

- We have a nice, bounded, well-behaved function.
- We can directly calculate the inferred probability of category membership.
- We're essentially fitting a Bernoulli process.

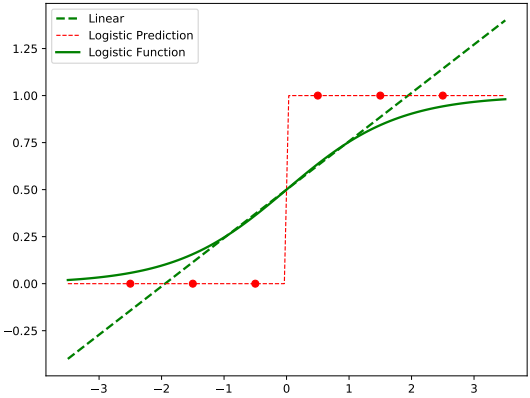


Logistic regression, continued some more

One has to use somewhat different numerical algorithms to fit these curves; typical curve-fitting algorithms deal very poorly with exponentials.

Techniques like expectation maximization (EM) or other well-conditioned iterative methods are often used.

That's fine; they're all hidden beneath whatever logistic or GLM packages you might want to use.



Logistic regression, example

Logistic regression in R is usually performed using a glm.

- You will need to install the 'mlbench' package to get this data.
- The 'complete.cases' function returns a boolean vector, indicating which rows have complete data.
- As always, we split into training and testing data.

```
>
> data(BreastCancer, package = 'mlbench')
>
> bc <- BreastCancer[complete.cases(BreastCancer),]
>
> ind <- sample(c(TRUE, FALSE), nrow(bc),
+               replace = TRUE, prob = c(0.7, 0.3))
>
> train.d <- bc[ind,]
> test.d <- bc[!ind,]
>
> model <- glm(Class ~ Cl.thickness + Cell.size +
+      Cell.shape, family = 'binomial', data = train.d)
>
```

Logistic regression, example, continued

Predictions using logistic regression need some postprocessing.

- Use the "type = 'response'" flag when using predict with logistic regression.
- The returned values are probabilities of 'success'. These must be converted to a classification.
- The ifelse function applies a conditional to each element, and returns a the first argument for a TRUE value, the second for FALSE.

```
> pred <- predict(model, newdata = test.d,
+                 type = 'response')
>
> new.pred <- ifelse(pred > 0.5,
+                    'malignant', 'benign')
>
> conf <- table(test.d$Class, as.factor(new.pred))
>
> conf
          benign malignant
benign       122         4
malignant      4        60
>
> sum(diag(conf)) / sum(conf)
[1] 0.9578947
>
```

Evaluating binary classifiers

Binary classification is a common and important enough special case that its confusion matrix elements have special names, and various quality measures are defined.

	Classified Positive (CP)	Classified Negative (CN)
Actual Positive	True Positive (TP)	False Negative (FN)
Actual Negative	False Positive (FP)	True Negative (TN)

One can always get exactly one of FN or FP to be zero (for example, just classify everything positive, then there will never be any false negatives).

But there is usually a tradeoff between false positives and false negatives.

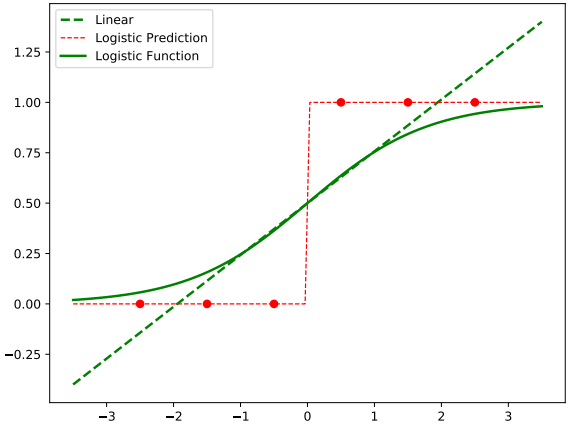
SciNet compute • calcul CANADA

Classification thresholds

In most binary classifiers, there's some equivalent of a threshold you can set. This threshold determines when a given data point moves from one categorization to the other.

For the case of logistic regression, the default threshold is 0.5.

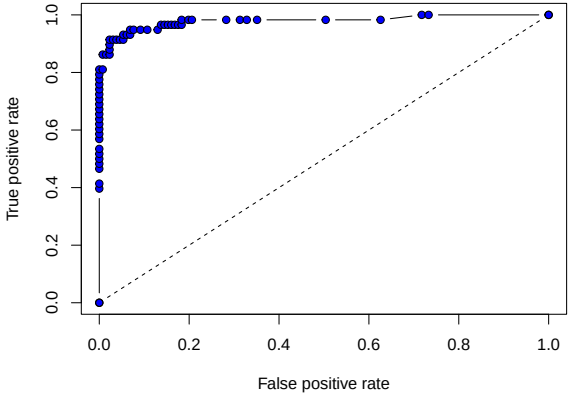
- Set it lower (allow more true, but also false, positives).
- Set it higher (allow more true, but also false, negatives).



ROC curve

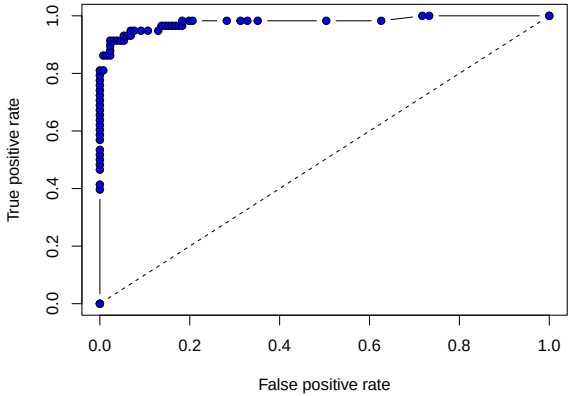
By varying the classification threshold, from 0 to 1, we can get a collection of points for the TPR and FPR. Plotting the two measures on either axis gives a ROC (Receiver Operating Characteristic) curve.

- The diagonal line represents random chance.
- We want our curve to be as high above the diagonal as possible.



ROC curve, continued

```
>
> library(ROCR)
>
> ROC.pred <- prediction(pred, test.d$Class)
> ROC.perf <- performance(ROC.pred,
+                         measure = 'tpr',
+                         x.measure = 'fpr')
>
> plot(ROC.perf, type = 'b', pch = 21,
+      bg = 'blue')
>
> lines(c(0, 1), c(0, 1), lty = 2)
>
```



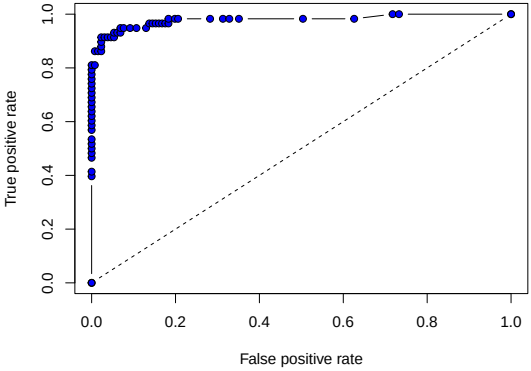
Note that your curve will look different from this one, due to randomness.

ROC curve, continued more

The quality of a classifier is determined by the ROC curve's AUC (area under the curve).

- The worst classifiers will have an AUC near 0.5.
- Good classifiers have an AUC near 1.0.

For the non-binary classification situation, you create "one versus all" ROC curves, with one ROC curve for each category.



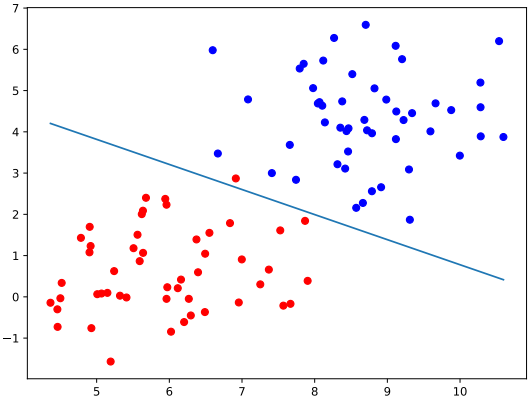

```
> ROC.AUC <- performance(ROC.pred,
+                        measure = 'auc')
> ROC.AUC@y.values
[1] 0.9786706
```

Support Vector Machines

Linear Support Vector Machines (also called Support Vector Classifiers) determine a hyperplane which linearly separates the data set into distinct categories.

The goal of the algorithm is to find the plane that is the farthest from all the closest points, in a k dimensional space.

This just ends up being a minimization problem.



SciNet compute • calcul CANADA

Support Vector Machines, example

To demonstrate the use of Support Vector Machines we will use the heart disease data set.

Note that the heart disease data set has 13 features. Hence, the hyper-plane which passes through the data will be 12D.

```
> heart.data <- read.csv('link below', header = F)
> heart.data$V14 <- as.factor(heart.data$V14)
>
> ind <- sample(c(TRUE, FALSE), nrow(heart.data),
+               replace = TRUE, prob = c(0.7, 0.3))
>
> train.d <- heart.data[ind,]
> test.d <- heart.data[!ind,]
>
> library(caret)
>
> svmFit <- train(V14 ~ .,
+   data = train.d, method = 'svmLinear',
+   preProcess = c('center', 'scale'),
+   trControl = trainControl(method = 'cv', number = 10),
+   tuneLength = 10)
```

Data source: https://dataaspirant.com/wp-content/uploads/2017/01/heart.csv

Support Vector Machines, example, continued

Support Vector Machines take an optional argument, 'C' (the penalty parameter).

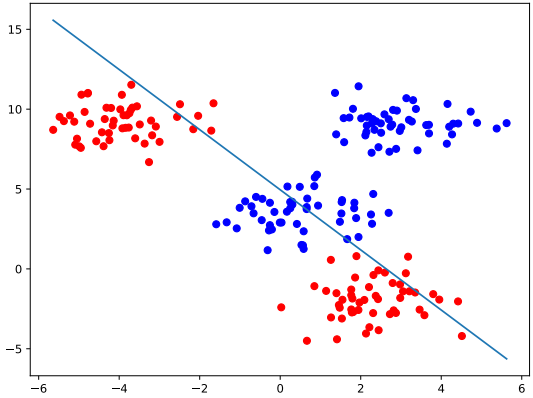
- Large values of C mean that we lack confidence in the data's distribution (noisy data).
- Small values mean the opposite.
- Default value is 1.0.

```
> pred <- predict(svmFit, newdata = test.d)
> confusionMatrix(pred, test.d$V14)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 42  9
         1  5 32

Accuracy :  0.8409
95% CI : (0.7475, 0.9102)
No Information Rate :  0.5341
P-Value [Acc > NIR] : 1.289e-09
.
.
.
>
```

Nonlinear Support Vector Machines

Linear Support Vector Machines are all well and good if your data are linearly separated. But what can we do if we aren't so lucky?

As we can see in the example at right, you can imagine situations where there are clearly defined clusters of data, but fitting linearly is not an option.



SciNet compute • calcul CANADA

Nonlinear Support Vector Machines, continued

We have a technique which is quite good at finding hyperplanes in linearly separated data.

- If the data are not linearly separated, the solution, obviously (!), is to nonlinearly transform the data into a space where it is linearly separable.
- This typically involves adding dimensions to the data which did not previously exist.
- This increases the likelihood of making things linearly separable (Cover's theorem).

Great! But how do we figure out what transformation to apply to the data?

- We don't. We let the SVM kernel do the work for us.
- SVMs use 'kernel functions' to transform the data into the required form.
- There are many types of kernel functions available: linear (which we've already been using), polynomial, radial basis function (RBF), sigmoid, and others.
- Once the hyperplane has been determined in the transformed space, the hyperplane is transformed back into regular data space.

SciNet compute • calcul CANADA

Nonlinear SVM, example

Invoking a nonlinear support vector machine is as simple as specifying "svmRadial" (for the radial basis function kernel).

Because the result has so many dimensions we're not going to try to plot the actual plane.

```
>
> svmNLFit <- train(V14 ~ .,
+   data = train.d, method = 'svmRadial',
+   preProcess = c('center', 'scale'),
+   trControl = trainControl(method = 'cv',
+                            number = 10),
+   tuneLength = 10)
>
```

Nonlinear SVM, example, continued

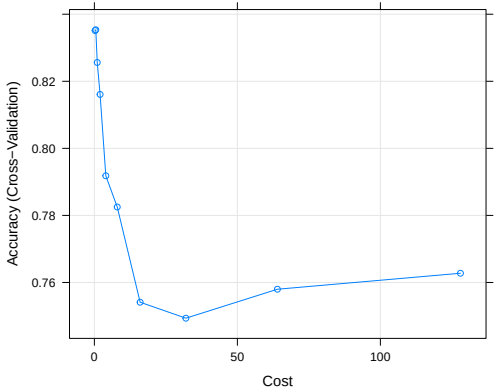
Cross-validation was used to tune the (hidden) hyperparameters. Printing out the model shows how the accuracy changed as a function of the cost ('penalty') hyperparameter.

```
> svmNLFit
Support Vector Machines with Radial Basis Function Kernel

C Accuracy Kappa
0.25 0.8351299 0.6628577
0.50 0.8353680 0.6632606
1.00 0.8256061 0.6434020
2.00 0.8160823 0.6247494
4.00 0.7918182 0.5740152
8.00 0.7825108 0.5573291
16.00 0.7541126 0.5008775
32.00 0.7493506 0.4930295
64.00 0.7579870 0.5100520
128.00 0.7627489 0.5197271

Tuning parameter 'sigma' was held at a value of 0.04483339
Accuracy was used to select the optimal model...
The final model values were sigma = 0.04483339 and C = 0.5 t
```

Nonlinear SVM, example, continued more

We can plot the cross-validation accuracy as a function of the cost parameter.

```
>
> plot(svmNLFit)
>
```



Nonlinear SVM, example, continued even more

As always, we're interested in how well the model does on the test data.

The nonlinear SVM model does slightly better on the test data than the linear SVM, though not much.

```
> NLpred <- predict(svmNLFit, newdata = test.d)
>
> confusionMatrix(NLpred, test.d$V14)
Confusion Matrix and Statistics

          Reference
Prediction  0  1
         0 43  9
         1  4 32

Accuracy :  0.8523
95% CI : (0.7606, 0.9189)
No Information Rate :  0.5341
P-Value [Acc > NIR] : 2.698e-10
⋮
>
```

Summary

You've now seen four classification algorithms: decision trees, logistic regression, kNN and SVM. Some things to remember:

- Logistic regression:
  - not prone to over-fitting.
  - can work well with noisy data.
  - assumes (requires) there is a single smooth boundary between categories.
- SVM:
  - a geometric technique,
  - builds a hyperplane that separates the data into groups.
  - nonlinear SVM projects the data into a higher-dimensional space to build the plane, then returns the plane to regular space.

There are guidelines you can use, but ultimately experience and experimentation is most important.

SciNet compute • calcul CANADA