161.326 Statistical Machine Learning
Part 8: Linear Classification

Mark Bebbington

Summary

This part is the equivalent of 4 lecture/lab hours. The statistical concepts will probably be new to all students.

The book has a little material on LDA (Sections 10.1 and 10.2). Thus copies of some pages from Hastie et al can be found in the `reading' folder. The lecture slides follow the material in the same order, with the addition of a lot of worked programming examples in NumPy.

As a general approach, I recommend following through the lecture slides, trying all the programs as you go along (remember to change the "os.chdir" argument to wherever you have put the data file(s)), and reading the relevant pages from Hastie et al once you finish a sub-topic. On the other hand, if another approach works better for you, go for it.

Once you have finished the lecture slides you should be able to do the first question of Assignment 3. The remainder of this document is a commentary on the lecture slides, which should be read in conjunction with them.

Slide

Read sections 10.1 and 10.2 in the text.

3. The idea in classification is that the output is now one of a finite number of possibilities, called *classes*, and we want to identify what region of the input space (values of x) are associated with each output class. One way to do this is to fit a linear model for each class, and put the boundary between two classes at the points x where the two models (i.e, coefficients β) produce the same value.
4. Classification (into 2 classes) using "nearest neighbour" methods.
5. The same problem solved by a linear model. Note that in using a linear model you assume a linear boundary. This can be good and/or bad.
6. You can get nonlinear boundaries by using polynomial functions of the original predictors. There are also much harder ways.
7. The 40% error rate refers to test data, where the model is fitted to separate training data.
8. Note, for later, that y is integer until the line `y = transpose(matrix(y))`
9. The indicator matrix Y has N rows, each of which has one "1" and the rest "0"'s. The "1" is in the column corresponding to the class, y, of that data point. The problem is then a linear regression problem, which we have already seen. The only difference is that our output is a matrix, not a vector. The matrix $\hat{B}$ is the equivalent of $\hat{\beta}$.
10. Note use of `int(y[n])` – see 8. above. Look at Y to see what an indicator matrix looks like.
11. The fitted output ŷ will be a vector, each element corresponding to one class. The largest value indicates the class which the output most likely belongs to.

12. The ".argmax[0]" extension returns the index of the maximum of each row of YhatT, i.e., if 0, the first class is a better fit, if 1, the second class, etc. We add 1 to make them correspond to the classes in the data, which are indexed from 1, not 0. Note error rate of approximately 48%. The whole program is in 'vowel_indreg.py'.
13. This explains why an indicator matrix is used, rather than just regressing on the class vector.
14. See 'ex8_1.py'. We see that the error rate on the test data, using the trained model is 67%! Fitting it to the test data reduces it to about the same as before 50%. We don't have enough data.
15. A discriminant function is any function that gives a "goodness" or "closeness" of fit to a given class, which enables us to *classify* a point x as being in the class to which it is closest.
16. The logit transform can be understood as $\log(p/(1-p)) = \exp(\beta_0+\beta^T X)$, a linear model.
17. LDA uses a Bayesian paradigm, while logistic regression estimates the probability of being in a particular class. LDA is also much more easily extended to more than 2 classes.
18. Revision
19. We suppose that each class is concentrated in the X-space, and that each concentration can be defined by using a multivariate normal distribution.
20. Here we put a Gaussian density down on each class, and classify points as being in whichever class has the greatest density value.
21. This is the formulation as a discriminant function. Our prior distribution is just the unconditioned (on X) proportions of each class in the training data.
22. Here we have both types of error.
23. Here the classes are each pi = 1/11th of the data, and mu is the center of each class in X-space
24. The covariance matrix (same for all classes – see Slide 19).
25. This is the discriminant function (Slide 21)
26. Using the discriminant function to classify each point, and give the percentage that are wrongly classified. Program is in 'vowel_lda.py'
27. This is a 2-D projection of the 12-D data.
28. See 'ex8_2.py'. Error rates of 56% and 28%. Note that you have to be careful with use of N and N2.

You can now read Hastie et al, pp. 79-87

29. This formulation can be generalized to more than 2 classes, but the optimization procedure becomes very curly indeed.
30. Note that, as probabilities sum to one, we are directly estimating the probability that a point lies in a given class.
31. The likelihood is a function of the parameters and the data which gives the likelihood of the data being produced by the given parameters. When optimized over the parameters, it selects the parameter values most likely given the data. In the formulation, it is very important that the classes be y = 0 or 1, as (1-y) has to be 0 or 1 also.
32. The likelihood can be maximized in this case by a simple Newton-Raphson algorithm, where β^{new} is obtained iteratively from β^{old} until successive values are within some tolerance distance.
33. 'adiposity' = 'fattyness' (of heart tissue, presumably). Note that 'famhist' is a 0-1 variable. Type-A behaviour is aggressive, high stress.
34. The data in pictures.
35. Note that we will have an intercept, hence the concatenation. The class is the last column (just for a change).
36. Pretty straight-forward implementation of Slide 32, except for the loop to construct W. Unfortunately, in NumPy, diag(.) of a one column matrix results in the first element of the vector, not a matrix with the vector on the diagonal.

37. Estimated parameters, from which the probabilities can be calculated.
38. Classification is simple – assign it to the class with the highest probability or, as there are only two classes, to that with probability > 0.5. Error rate of 27%. The program is in 'heart_logreg.py'.
39. See `ex8_3.py'. All three error rates are very comparable. Note that the dependent variable y is indexed 0-1, not 1-2, which means that you need to modify 'vowel_indreg.py' and 'vowel_lda.py' in several places!!
40. Compare with Slides 12-14 of Part 2.
41. Note that W is the same animal as in Slide 36.
42. The significant factors appear to be the intercept, tobacco, cholesterol, family history, type-A behaviour and age. But there are a lot of correlations between the predictors. Have a look at the correlations (see Slide 18 of Part 2).
43. See 'ex8_4.py'. We see that all except factor 8 (alcohol) appear to be significant individually. However, there appears to be no strong correlation between significance and error rate. You may need to add a small amount to one element of the matrix ${}_1X^TW_1X$ in order to make in non-singular.

You can now read Hastie et al, pp. 95-99.

This is the end of Part 8.