Lecture 9 (of 10) – Wed, 18 March 2020.

- This is the last chapter of the module.
- Next week will be a revision lecture – I will make some sort of review of the course and a revision guide and post it on Moodle.
- The exam was written last month and thus the material in this chapter is just as likely as any other material to appear in the exam.
- Please feel free to email me any questions you have about the material: b.marshall@bbk.ac.uk
- Below, if I write [x], it means *please see handwritten equation number x.*
- But if I write <x>, it means *please see equation number x from Chapter 12 of the typed notes.*

Chapter 12: Two Sample Discriminate Analysis

Overview

Here we look at a new way to do prediction for categorical response variables (i.e. those taking a finite number of labels). The method is called *Discriminant analysis*. We previously used a GLM for this task: for example, when there are two labels a Logistic regression GLM can be used. When there are more than two labels it is possible to use a generalisation of logistic regression based on the the multinomial distribution (see **multinom** function in R package **nnet**).

What makes Discriminant analysis different is that it is based on an explicit probability model for the explanatory variables. By using Bayes' Theorem, the optimal way to do prediction is derived – and this is what we call Discriminant analysis. Furthermore, a cost function can be brought into the prediction, allowing certain mistakes to be more costly than others.

Aside: Just because we are using Bayes' Theorem does not mean we are doing Bayesian Statistics (we definitely aren't doing Bayesian Statistics). Bayesian Statistics is a collection of methods that are alternatives to Maximum Likelihood. Thus Bayesian Statistics tells us alternative ways to *estimate models from data*, whereas Discriminant analysis uses Bayes' Theorem to design a method to do *prediction:*

Algorithm of a Statistician:
- *step 1: get data*
- *step 2: estimate model*
- *step 3: use the model to do something useful (e.g. prediction)*

You could use Bayesian Statistics to help with step 2. Discriminant analysis is a way to do step 3.

The rest of the notes (section 12.3 onwards) looks at a particular instance of Discriminant analysis where there are 2 populations and the explanatory variables x are assumed to be ~ Multivariate Normal. Thus we are able to make use of of the multivariate theory from the last few weeks in order to do prediction!

A motivating example:

Suppose the inputs are images of the number 5 or images of the number 6. The objective is to design an algorithm for discriminating between 5s and 6s.

Because there are only two labels, it is possible to use logistic regression:

Get some of images of the number 5 and some images of the number 6 and build a dataset, say coding the response C as 0 if the image is a 5 and as 1 if the image is 6. Given a new image x the job is to predict if C=1 (i.e. if x is a 6).

Recall that an image x is just a grid of N-by-N numbers which we can flatten into a vector. If the images are small and low resolution, say N is about 100, then each image x can be thought of as just a vector of length about 1000.

An initial model to try is given at [1].

- How many parameters are in this model?
- How much data would be needed to get good estimates of the parameters (roughly)?
- What is the meaning of beta1?
- Will this model capture what it means to be a 5 vs a 6?
- How might this be modified to better capture how the label ("5", "6") depends on the image x?
- Would it matter if the fraction of 5s in the training data wasn't actually the fraction of 5s in the population on which the prediction algorithm will be run? (this is a hard question and not examinable)

Logistic regression will do a very bad job on this problem because the way in which the label depends on the image is not easily captured by a *linear predictor*.

It could help to include polynomial terms, and in general it is necessary to design quite elaborate "features" (i.e. new variables) which give some local evidence of the identity of the character in the image. For example, a feature might be designed by drawing horizontal lines across the image and counting each time the line passes through a border between light and dark. Since a 6 has a closed bottom, whereas a 5 has an open bottom, some of the horizontal lines will have 1 more crossing for a 6 than a 5. The feature could therefore be the total count of crossings summed over all horizontal lines, higher values of this newly defined variable increases the (posterior) probability the image is a 6.

After many years of hand designing features in this way, it became apparent that there aren't easily articulated pieces of local evidence which combine in a simple manor to accurately predict the digit in the image with error rates as low as a human. Consequently, this problem motived research into non-linear regression models, which in some sense, automatically discover the appropriate features directly from the data. This is now called Deep Learning.

Deep Learning naturally follows on from GLMs (both historically and mathematically). For a good introduction written by two of the world's leading statisticians see Chapter 18 in the book *Computer Age Statistical Inference* by Efron and Hastie (Chapter 8 covers GLMs).

Discriminant Analysis is an alternative to both Deep Learning and Logistic regression. Furthermore Discriminant Analysis is the optimal solution to this problem because it is derived purely using probability theory (in particular Bayes' Theorem). This means that Logistic regression based on hand crafted features or a Deep Learning model based on many layers is good only in as much as it approximates the Discriminant analysis solution. However the rub is that using Discriminant analysis may require a level of stochastic modelling above and beyond that of both GLMs and Deep Learning.

Another aside: The phrase "automatically discover the appropriate features" which I used to describe Deep Learning is misleading. Deep Learning models can only find certain types of features (as restricted by the model architecture) and these features are by definition only those that are useful for discrimination. There are three remarks I want to make in this regard:

1. If x represents climate measurements, and you applied a Deep Learning model, don't expect it to *automatically discover* the physics underlying the climate. Typically the features discover by Deep Learning models aren't the fundamental aspects of reality that scientists try and uncover.
2. Deep Learning models are sensitive to the precise contents of the training data. For example, if all the 6s in our dataset were written with a slightly thicker nib than the 5s then a typical Deep Learning model for images would hone in on this feature. Using a feature based approach, we could avoid taking advantage of these accidental aspects of the training data. The Deep Learning solution to this problem is to design architectures that are blind to certain features we wish it to ignore. However we now have the same problem we began with but inverted: instead of trying to find useful features to discriminate, we need to hand design architectures to ignore certain features. But finding all the features to ignore might be as hard as finding all the features to include! I guess time will tell.
3. Most of the data encountered by a statistician is of the form of a small (say less than 1000) number of "multi-type" explanatory variables describing the characteristics of the individuals in a population (e.g. age, wage, shoe size). Typically the objective is to ***understand*** the data. In these scenarios a GLM is perfect because it allows you to ask very specific questions of the data (e.g. is y influenced by x1 after accounting for x2?) and get very precise answers, furthermore the resulting model is typically small and interpretable allowing the data to be summarised compactly. A Deep Learning model isn't designed for these sorts of tasks.

Section 1 – What is Discriminant Analysis?

Read section 12.2 of the notes. In particular focus on <1>. Can you derive this formula? You will need to use the results in [2].

Strictly speaking we need to treat the event "X = x" with caution because if X is a continous random variable, Pr(X = x) = 0. Hence you can't use Bayes' Theorem because then you would be dividing by 0. However it turns out that if you apply Bayes' Theorem using the event "x < X <= x + dx" and take the limit as dx → 0, then you arrive at a version of Bayes' Thm which does work for continuous random variables. See your probabilty lecture notes from the Autumn term for details.

Write down a version of <1> but for when there are K populations. See [3] for a start.

Supposing the cost of misclassification was the same for all pairs (i,j). Intuitively, the optimal prediction rule is to find the label j that maximizes p(j|x), since this label has the highest probability of being the true label for x. See [4]. This is called the **m**aximum **a p**osteriori estimator (MAP).

Now let's modify the rule by bringing in the costs. See Theorem 12.1. Then see [5]. Consequently, in the case of 2 populations (K=2), there is a simple formula for the decision rule given at equation <2>. This is the form we will carry over into the next section. But note that both when K = 2 and in the general case K >= 2 (given in [5]), computing which class to predict **does not require calculating the denominator Z(x) in the expression for the posterior.**

Conclusion: optimal prediction (while also taking into account possible misclassification error costs) requires doing the following steps:

Modelling Stage:

- Step 1: for each possible label j, estimate the multivariate probability density (fj) for the explanatory variables x assuming x is of type j.
- Step 2: for each j, work out the fraction of j's in the population (which may be different from the fraction you have in the dataset of course).

Prediction Stage:

- Given new inputs x, if K = 2, compare the ratios given by equation <2>, else if K > 2, compute for each j the expected misclassification cost of predicting j on inputs x, and pick the j with the smallest expected cost.

The Prediction Stage and Step 2 of the Modelling Stage are really very trivial! The Prediction Stage just involves doing K sums and picking the smallest. And Step 2 of the Modelling Stage will typically require simply counting the fraction of j's in some representative database. However Step 1 of the Modelling Stage is extremely hard to do when x is something complicated like an image.

Let's think some more about what is being asked in Step 1.

In the Autumn Term many of you will have taken a course covering Markov Chains (discrete time series) and then AR-MA models (continuous time series). These models can be used for Discriminant Analysis!!!

Example: An archivist finds an old text document buried in a corner of the library and wants to try and find out in which century it was written.

Modelling Step:

- For the centuries j = 1, 2, 3, …, 20 (say), the archivist collects example text documents from century j and estimates a Markov transition probability matrix for century j (call it Pj). This involves computing for each pair of possible words (w1, w2) the probability that the next word is w2 given that the current word is w1. The idea is to calculate these statistics for each century j separately, giving 20 Transition matrices P1, P2, …, P20.

These are our fj's. These transition matrices hopefully capture enough about the way text documents were written in each centruy to allow us to correctly discriminate between centuries.

- Using known ages of the documents in the library, the archivist computes the fraction of j'th century texts in the library. These are the pi terms.

Prediction Step:

Take the discovered document x, and for each century j, evaluate the probability of the text using the transition probability matrix for that century Pj, then multiply this probability with the fractions of j's in the library and then pick the j for which this quantity is largest. (Here we assume equal misclassification costs and hence use MAP.)

A similar procedure could be used for say discriminating certain patterns in financial time series data. For each type of pattern j, collect examples of time series exhibiting that pattern and fit an AR-MA model specific to j. Using the formulas from discriminant analysis, these models can then be used to automatically classify new time series.

The language example highlights the essential difficulty. Language is more complex than a Markov Chain. For 100% optimal predictions we need to use the true fj's. But capturing the true statistics of language (even after nearly 100 years of trying) is still essentially unsolved. But we shouldn't despair, for a Markov Chain might be good enough to discriminate between centuries - if not a first order Markov Chain, then perhaps a second order or third order.

One way to think about this is that we are replacing searching for features (GLM) with searching for realistic probability models for x.

Section 2 – Linear Discriminant Analysis (Section 12.3 in the notes)

In this section we assume that there are two populations and that the data vector X is multivariate normal, where the means are different between the two populations but the covariance matrices are the same. This is the same assumption we made while performing the hypothesis tests and Canonical Variate analysis.

Equation <2> tells us how to do discriminant analysis in the two populations case. Hence all we need to do is calculate the ratio of two multivariate normals when they have a shared covariance matrix. This is what is being worked out in equations <4, 5, 6, 7>.

The main result is equation <7>. This is the version of <2> that emerges when you assume normality. Make sure you can follow the derivation from <2> to <7> when you make the normality assumption.

Yet another aside: The notation D(x,mu) used on page 134 stands for the Mahalanobis Distance, a quantity which we first encountered on page 108. The meaning of this term is context dependent because you need to know what inverse-covariance matrix to put in the middle. On page 108 it is assumed that it is the sample covariance matrix, but here it is assumed to be the shared population covariance matrix. Furthermore, on page 135, D(mu1, mu2)^2 is written to mean the Mahalanobis distance between mu1 and mu2, where you use the shared population covariance matrix Sigma. What this highlights is that the terms "Mahalanobis Distance" (and thus the D notation) is ambiguous and you need to work out the appropriate matrix to use in the middle from the context.

Section 3 – Prediction Error (Section 12.4 in the notes)

Here we do something a little different from what we have seen previously.

The main idea when doing hypothesis tests is to control the Type 1 error probability. The Type 1 error probability is the probability under the null hypothesis of rejecting the null hypothesis. In this case the unknown is a *parameter,* typically a mean. Hence controlling the Type 1 error probability means controlling the probability of making a false statement about this parameter.

In this section, instead of doing a hypothesis test about a parameter we are going to quantify the **prediction error of the algorithm**. Here the unknown is not a parameter, but is rather a random variable (namely C the class label).

What this means is (if x is genuinely normally distributed) then not only does discriminant analysis give the optimal prediction algorithm, but it also tells us how often the method will predict incorrectly. The calculations for computing Pr(wrong prediction) are made simple because of the normality assumption. Please study this calculation carefully.

To better understand what is being calculated in this section, it is possible to write down a more general version in the case where there are K populations. The equations are shown here [6]. In words: discriminant analysis partitions the input space (curly X) into K regions R1, R2, …, RK, such that if x lands in region Rj then discriminant analysis says predict j. Hence an error is made every-time an x lands in Rj but in reality C is not j. Thus Pr(error) can be computed by integrating over all these bad cases.