Topic 8. Classification

Case 5: Intrusion Detection

using TCP dump data from a local area network

The Plan

- Describe the project.

- Review previous classification tools: nearest neighbor, classification trees

- Discuss limits of prediction tools when used with three or more categories: regression, neural nets

- Introduce new tools that have a probabilistic foundation: linear discriminant analysis, quadratic discriminant analysis, logistic regression.

Intrusion Detection

The intrusion detector learning task is to build a predictive model (a classifier) capable of distinguishing between connections that are intrusions (attacks) and normal connections.

MIT's Lincoln Labs acquired nine weeks of raw TCP dump data for a local-area network (LAN) simulating a typical U.S. Air Force LAN. They operated the LAN as if it were a true Air Force environment but peppered it with multiple attacks. The data here is a version of this dataset.

The raw data was about four gigabytes of compressed binary TCP dump data from seven weeks of network traffic. This was processed into about five million connection records of which I have a half million collected under an early protocol and another half million under a later protocol with more attack types.

Intrusion Detection (continued)

A connection is a sequence of TCP packets, starting and ending at some well defined times, between which data flows to and from a source IP address to a target IP address under some well defined protocol.

There are 42 features in the data set, which were generated partially by domain knowledge and partly through automated procedures.

The last feature, connection.type, is the target wherein each connection is labeled as either normal or by the name of one specific attack type. There are 24 training attack types in the early data, which is the data we shall use.

Attack Types

Attacks fall into four main categories:

- DOS: denial-of-service, e.g. SYN flood (aka Neptune);
- R2L: unauthorized access from a remote machine, e.g. guessing password;
- U2R: unauthorized access to local superuser (root) privileges, e.g., various "buffer overflow" attacks;
- Probing: surveillance and other probing, e.g., port scanning.

A Haphazard Sampling of the Features

Field	Feature	Type	Definition
1	duration	continuous	connect time in seconds
2	protocol.type	discrete	type of protocol, e.g. tcp, udp
9	urgent	continuous	number of urgent packets
11	num.failed.logins	continuous	number of failed login attempts
12	logged.in	discrete	login success (=1) or failure (=0)
17	num.file.creations	continuous	number of file creation operations
23	count	continuous	number of connections to the same host as the current connection in the past two seconds
24	srv.count	continuous	number of connections to the same service as the current connection in the past two seconds
25	serror.rate	continuous	fraction of same-host connections that have “SYN” errors
26	srv.serror.rate	continuous	fraction of same-service connections that have “SYN” errors
29	same.srv.rate	continuous	fraction of same-host connections to the same service in past two seconds
30	diff.srv.rate	continuous	fraction of same-host connections to different services in past two seconds
42	connection.type	discrete	target, coded: normal, neptune (dos), smurf (dos), portsweep (probe), etc.

Project Description

- A complete description of the project is file ntw_doc.html

- A complete listing of features is in ntw_lst.txt and, for attacks, in ntw_atk.txt

- You can access these files by entering

  ftp://ftp.econ.duke.edu/pub/arg/datamine/cases/network/

  in the netsite address dialog box of a browser (e.g. Netscape, Mozilla, Internet Explorer)

Previous Work

Some previous work on these data, as well as a description of the logic behind feature construction is in

> Stolfo, Salvatore J., Wei Fan, Wenke Lee, Andreas Prodromidis, and Philip K. Chan (1999) *Cost-based Modeling and Evaluation for Data Mining With Application to Fraud and Intrusion Detection: Results from the JAM Project*

This article is in directory (folder) cases/network/articles as file wenke-discex00.pdf or wenke-discex00.ps together with the more important cited articles and websites for additional articles and the RIPPER software described in the article.

Previous Work (continued)

The slides from a seminar presentation from Stolfo, Fan, Lee, Prodromidis, and Chan (1999) are at

http://www.cs.columbia.edu/~sal/JAM/PROJECT/MIT/mit-index.html

The RIPPER software they use is a sort of decision tree classifier that produces Prologue type rules that can be used to implement the tree in applications. It was developed at AT&T Research Laboratories. I have a copy of RIPPER, which is C code, which you can use to explore the use of this tool for a class project. The tool comes with a few test data sets that are of some interest.

They apply RIPPER to classification sub-tasks and build a complete classifier using a boosting technique they call a meta-classifier. We shall study boosting later in the course.

Data for Analysis

Of the 494,020 cases 19% are classified normal connections, 22% are neptune attacks, and 57% are smurf attacks for a total of 98%; smurf and neptune attacks are described next. We shall only consider these three categories. Also, we will only use a 10% fraction to keep the sample size small enough that we do not have to resort to memory management methods.

Notice that these percentages are not realistic. This is because the data is experimental data, it is not on-line observational data. This fact will need to be taken account in our analysis.

Smurf Attacks

An Internet Control Message Protocol (ICMP) message requests a reply from its recipient; the message sent by the ping command is an example. In a smurf attack an ICMP message with a falsified IP return address is sent to a broadcast address (e.g. 152.3.250.1). When an ICMP command is sent to a broadcast address, it is broadcast to all machines on the targeted network. Each machine on the network receives this broadcast ICMP message and responds with a reply to the false IP address thereby swamping the machine whose true IP address is the same as the false address. The cure is to configure all routers on the network to not to broadcast ICMP messages. To stop a similar attack strategy from within the network, which need not rely on routers, configure all machines on the network to not respond to ICMP messages.

Neptune Attacks

When a machine attempts to establish a TCP connection (web, telnet, etc.) to a server it sends a SYN message. The server responds with a SYN-ACK message. The connecting machine replies to the server with an ACK message and the connection is then opened. The neptune attack consists in sending many SYN messages (aka SYN flood) but never sending an ACK message in reply to the server's SYN-ACK message. The server has in its memory a data structure describing all pending connections, and too many partially-open connections will cause it to overflow. At best this causes the server to be unable to respond to connection requests, at worse the server crashes. One defense is to increase the size of the data structure dramatically and purge half-open connections quickly.

Data for Analysis (continued)

Using the same methods as for Case 3, Donor Recapture, the data is divided into learning

network/lrn/ntw_lrn.dat

validation

network/val/ntw_val.dat

and testing

network/tst/ntw_tst.dat

samples.

The testing sample is small enough for use with XL-Miner.

Classification Problems

Case 1, Credit Scoring, was a classification problem.

The nearest neighbor method considered then is designed for classification problems and can be applied directly here. Nothing more needs to be said. Similarly for classification trees.

To apply prediction methods to the network intrusion problem, one would proceed as follows:

Classification Problems

Let $Y_{neptune}$ equal 1 if the connection is "neptune" and 0 if not. Using regression, neural nets, or trees, fit the model

$$Y_{neptune} = f(x),$$

to the learning data, where x is the vector of features. Let $\widehat{f}_{neptune}$ denote the model thus obtained. Do the same for Y_{normal} and Y_{smurf}. Given a feature vector x^o for which a classification is desired, compute the prediction

$$\widehat{Y}_{neptune} = \widehat{f}_{neptune}(x^o)$$

Do the same to get $\widehat{Y}_{normal}$ and $\widehat{Y}_{smurf}$.

If $\widehat{Y}_{neptune}$ is the largest then classify x^o as "neptune", else if $\widehat{Y}_{normal}$ is largest classify as "normal", else as "smurf".

Problems with This Approach

Although this approach is common, and often works reasonably well, there are two main problems with it.

- **Masking.** It is unlikely but possible that

  $$\widehat{f}_{smurf}(x) < \min\left\{\widehat{f}_{normal}(x), \widehat{f}_{neptune}(x)\right\}$$

  for every possible configuration of features x so that a connection never gets classified as "smurf".

- **Ignores Structure** The approach completely ignores the probabilistic structure of the problem. Taking this extra knowledge into account ought to improve performance.

The Standard Methods

Taking the probabilistic structure into account leads to the three standard classification methods:

- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Logistic Regression

We shall consider each of them, in turn.

The Standard Measure of Performance

The standard measure of performance, is the classification error rate, cer.

The classification error rate of a procedure is the number of misclassifications in a sample divided by the sample size.

In a sample that had 100 “neptunes,” 100 “normals,” and 100 “smurfs,” were we to misclassify 5 “normals” as “smurfs” and 2 “smurfs” as “neptunes” and classify everything else correctly, then the classification error rate would be

$$\text{cer} = \frac{7}{300} = 0.023$$

Probabilistic Structure

For a given set of features x, a connection can be "neptune", "normal", or "smurf". Whichever it is can be thought of as being determined by tossing a single die: If the die lands 1, then the connection is "neptune"; if 2, then it is "normal"; if 3, then "smurf"; and if 4, 5, or 6, then toss again.

Now think of this die as being loaded and imagine that the features x tell you how much weight has been added to the faces 4, 5, and 6. Because the faces on the opposite sides of a die sum to 7, if x adds extra weight to face 5, then 2, which is a "normal" connection, is the most likely. The more weight added, the more likely "normal" becomes.

Probabilistic Structure (continued)

This dice mechanism is actually an accurate representation of the probability model for a classification problem. What it does is define three conditional probabilities:

$$P(C = \text{“neptune”} \,|\, X = x)$$
$$P(C = \text{“normal”} \,|\, X = x)$$
$$P(C = \text{“smurf”} \,|\, X = x)$$

The first is the probability of the connection C being “neptune” if features X are known to take on the value x. The second and third are the values for “normal” and “smurf” respectively.

Probabilistic Structure (continued)

If the classification error rate is accepted as the appropriate measure of a classification scheme, then, given a set of observed features x^o, the optimal classification strategy is to compute

$$P(C = \text{“neptune”} \,|X = x^o)$$
$$P(C = \text{“normal”} \,|X = x^o)$$
$$P(C = \text{“smurf”} \,|X = x^o)$$

and classify the connection according to whichever is largest. That is, if $P(C = \text{“normal”} \,|X = x^o)$ is the largest of the three, then for that x^o the connection is predicted to be “normal”.

Probabilistic Structure (continued)

The notation

$$P(C = \text{“neptune”}\,|X = x^o)$$
$$P(C = \text{“normal”}\,|X = x^o)$$
$$P(C = \text{“smurf”}\,|X = x^o)$$

is cumbersome. Let us substitute

$$P(C = c|X = x^o)$$

where c=1, 2, 3 for “neptune”, “normal”, and “smurf”, respectively.

Linear Discriminant Analysis

The idea behind linear and quadratic discriminant analysis is to specify a conditional probability density and marginal probabilities that are easy to estimate and then construct $P(C = c|X = x)$ indirectly using Bayes Theorem.

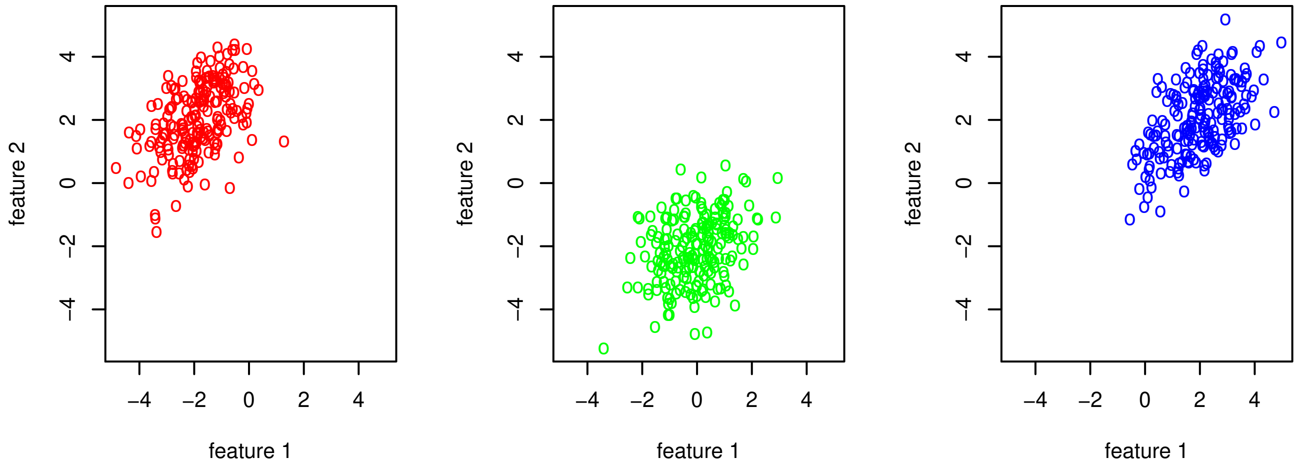
Conditional Probability Density

The training sample can be divided into three groups corresponding to "neptune", "normal", and "smurf" or, equivalently, $c = 1, 2, 3$. The features in each of these three groups its own probability density $f(x|C = c)$.

Linear discriminant analysis assumes that $f(x|C = c)$ is the normal distribution where the vector of features has a different mean for each c but the variances and correlations are the same for all c. Quadratic discriminant analysis allows these variances and correlations to depend on c. That is the only difference between the two methods. The normal density $f(x|C = c)$ is easily determined from the training sample by computing means, variances, and correlations.

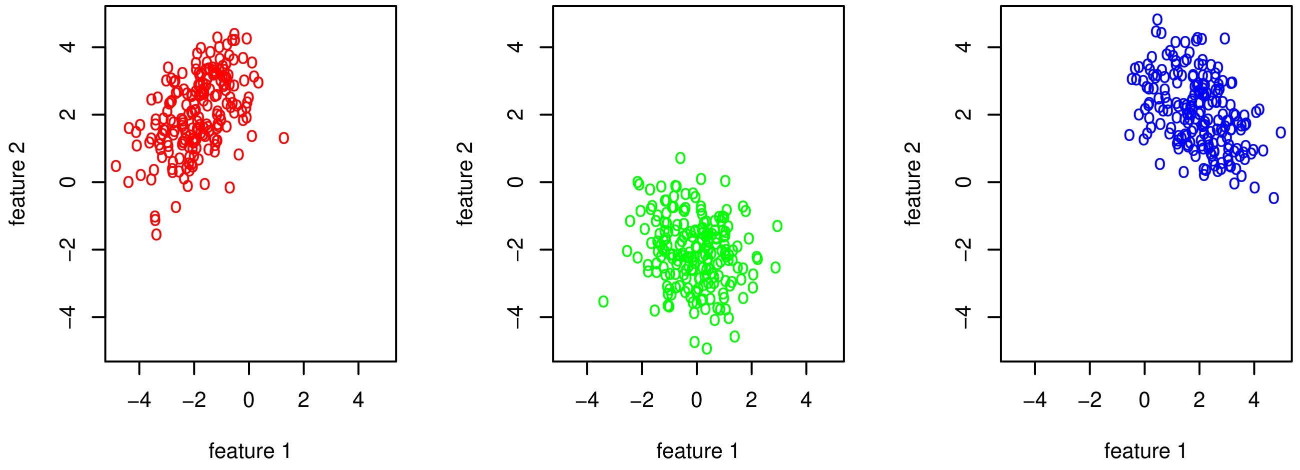
Note that a side effect of the normality assumption is that only continuous features are allowed.

Fig 96. Data That Satisfy LDA Assumptions



Simulated data that satisfy the LDA assumptions of normality and equal variances and correlations within groups.

Fig 97. Data That Satisfy QDA Assumptions



Simulated data that satisfy the QDA assumptions of normality but different variances and correlations across groups.

Marginal Probabilities

Returning now to Case 5, Intrusion Detection, we can ignore the features altogether and estimate the unconditional probability of a "neptune" as

$$\pi_1 = \frac{\text{number of "neptunes" in the training set}}{\text{total number observations in the training set}}$$

Similarly for π_2 and π_3 for "normals" and "smurfs".

Notice that if the training set is experimental data and not observational data, these proportions will not be what one encounters when classifying new cases. For smaller sized training sets one must adjust the π_i to more realistic values. For the larger sized training sets in data mining this will not matter much.

Bayes Theorem

$$P(C = c|X = x) = \frac{f(x|C = c)\pi_c}{\sum_{c=1}^{3} f(x|C = c)\pi_c}$$

Bayes Theorem (continued)

One can work out a detailed expression for $P(C = c|X = x)$ of the previous slide, but there is really no point to it.

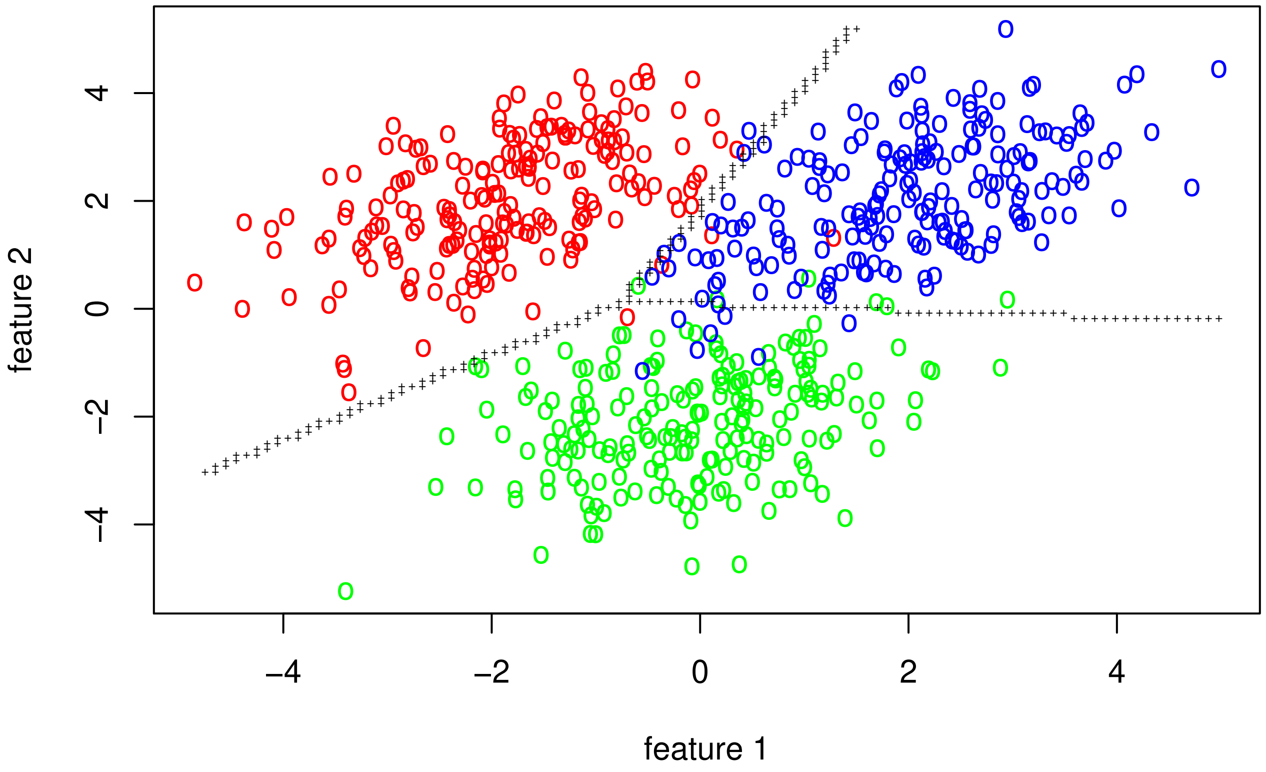
All that is necessary to know is that the boundary between features where cases get classified differently, e.g. the vectors x that satisfy the equation

$$P(C = 1|X = x) = P(C = 2|X = x),$$

is a flat linear surface for linear discriminant analysis and is a curved quadratic surface for quadratic discriminant analysis.

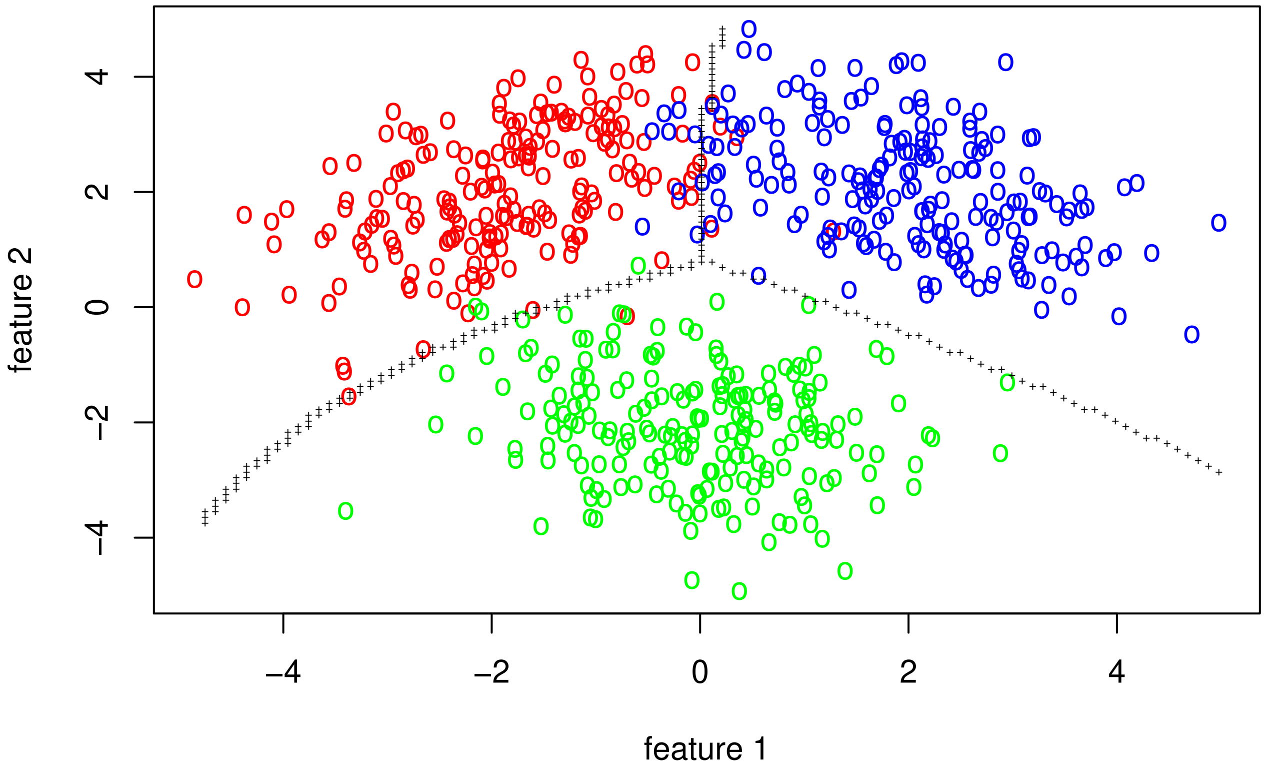
That is where the names come from.

Fig 98. LDA Classification Applied to LDA Data



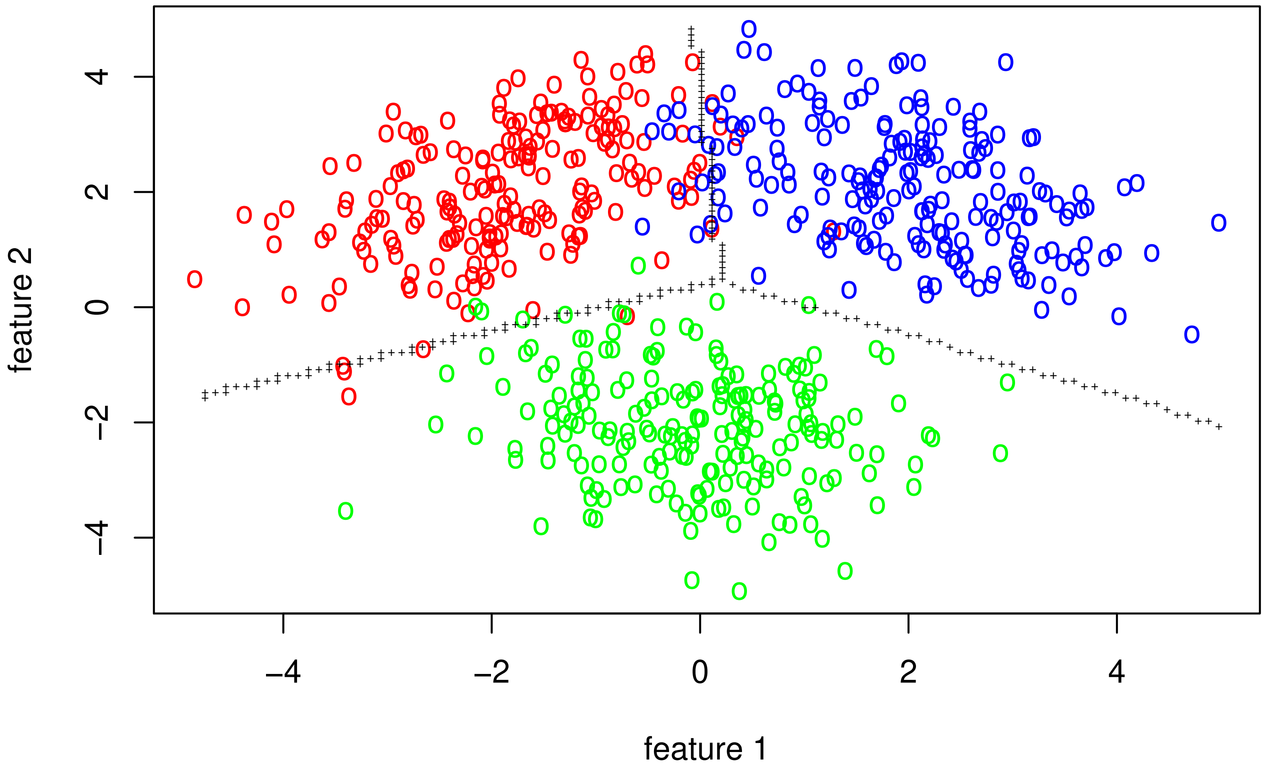
LDA classification method applied to simulated data that satisfy the LDA assumptions.

Fig 99. QDA Classification Applied to QDA Data



QDA classification method applied to simulated data that satisfy the QDA assumptions but not the LDA assumptions.

Fig 100. LDA Classification Applied to QDA Data



LDA classification method applied to simulated data that satisfy the QDA assumptions but not the LDA assumptions.

Feature Selection

As with Case 3, Donor Recapture, we need some automatic means to select features, although with only 41 features, features selection could actually be done by hand.

Because I want to illustrate classifications graphically, I will restrict myself to two continuous features. This actually does little harm because we achieve a classification error rate of 0.0063 in the validation data set, which is surprisingly good.

We will use linear discrimination analysis to try all possible pairs of two continuous features, estimate the linear discriminant rule in the training sample, compute the cer in the validation sample, and select the pair of features with the smallest cer.

LDA Model Selection: Results

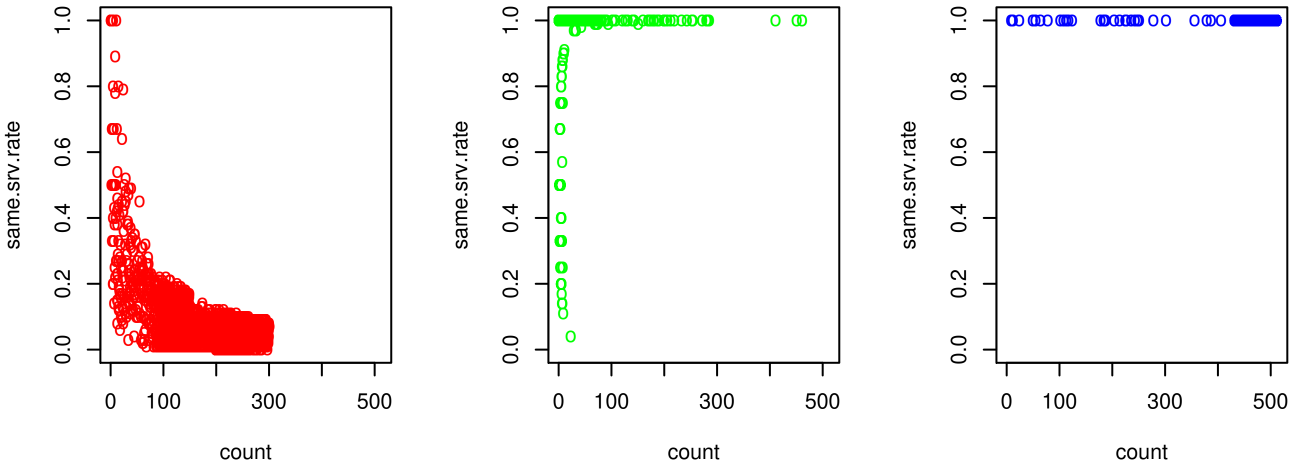
ntw_sel.r.Rout

```
[1] "0.00632845730884947  count  same.srv.rate"
```

The Selected Features

Field	Feature	Definition
23	count	number of connections to the same host as the current connection in the past two seconds
29	same.srv.rate	fraction of same-host connections to the same service in the past two seconds

Fig 101. The Training Data



Neptune attacks shown in red, normal connections in green, and smurf attacks in blue.

Table 28. Verification of Assumptions

	Mean		Standard Deviation		
Data	Count	Same Serv Rate	Count	Same Serv Rate	Correlation
Combined			42.6075	0.06434	-0.24278
Neptune	188.43	0.06949	69.0462	0.06602	-0.44984
Normal	8.0139	0.98622	18.1781	0.08979	0.03288
Smurf	506.90	1	18.6648	0	NA

Verification of Assumptions (continued)

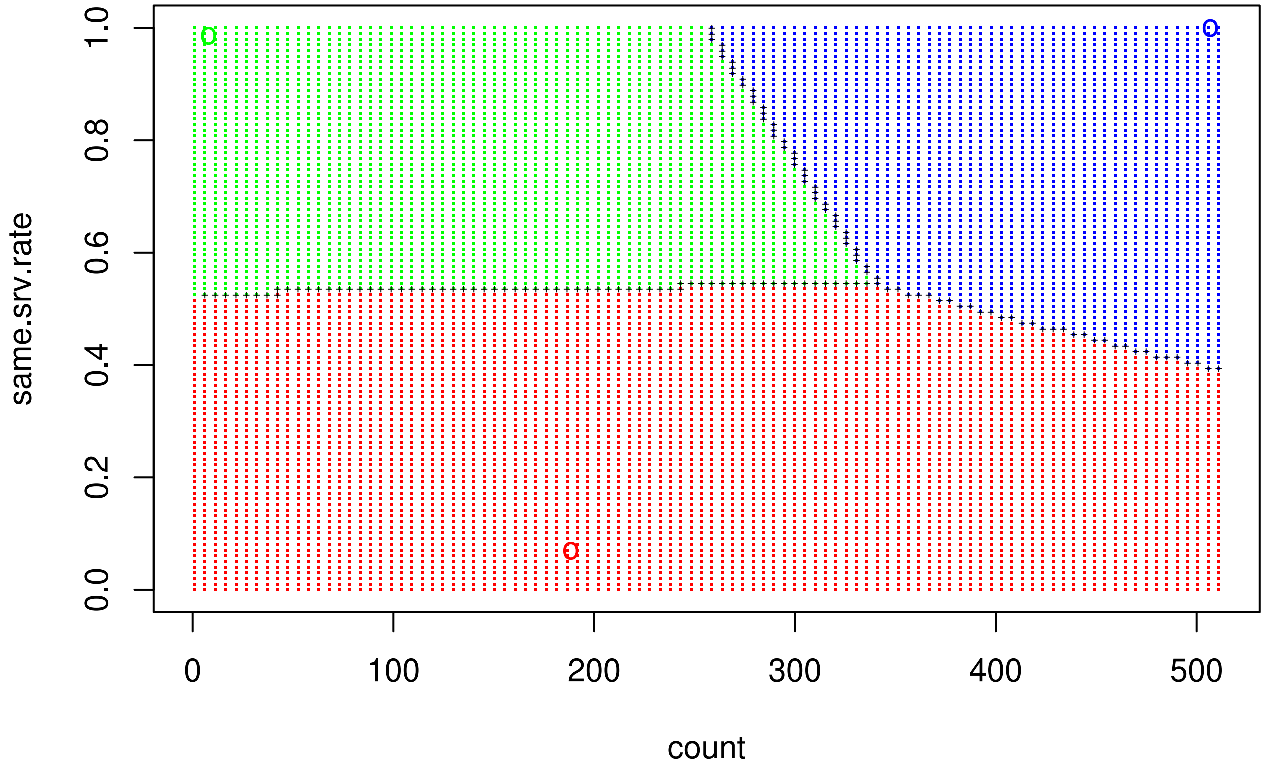
The linear discriminant analysis assumption of equal standard deviations and correlations across groups is clearly violated.

Ordinarily quadratic discrimination analysis would therefore be more appropriate.

But most software, including R, cannot handle a zero standard error. Theoretically it is not a problem, but unless one takes special precautions when writing the code the zero standard error will cause a division by zero with at best no results and at worst unreliable results.

We are stuck with linear discriminant analysis, like it or not.

Fig 102. LDA Classification with Sample Proportions



Neptune attacks shown in red, normal connections in green, and smurf attacks in blue. Prior is (0.2246, 0.2018, 0.5736); validation cer is 0.006328.

The Prior

The prior that we used was the proportion of “neptunes”, “normals”, and “smurfs” found in the training data.

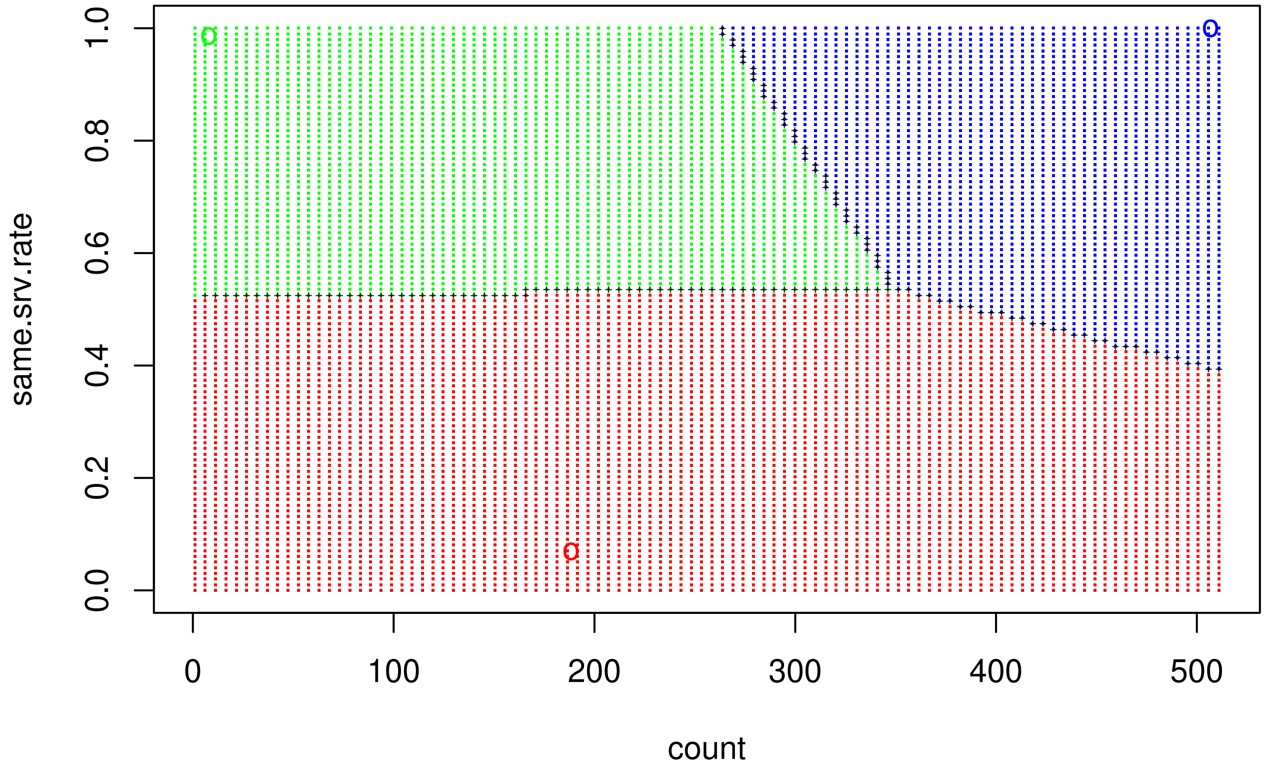
But the training data is experimental data where these proportions were determined by the experimenters. They are not the proportions that would be seen in actual data.

If the classification procedure is to be used online when most connections are normal, then the proportions should be adjusted to more realistic values.

An interesting approach would be to try to adjust them dynamically. For instance, if connections started to become increasingly classified as “smurf”, then the proportion assigned to smurf would be increased.

However, in this application, adjusting them to more realistic values makes hardly any difference.

Fig 103. LDA Classification with a Realistic Prior



Neptune attacks shown in red, normal connections in green, and smurf attacks in blue. Prior is (0.1, 0.8, 0.1).

Curved Boundaries

The fact that quadratic discrimination analysis cannot be used with these training data does not mean that we cannot have curved boundaries if we wish.

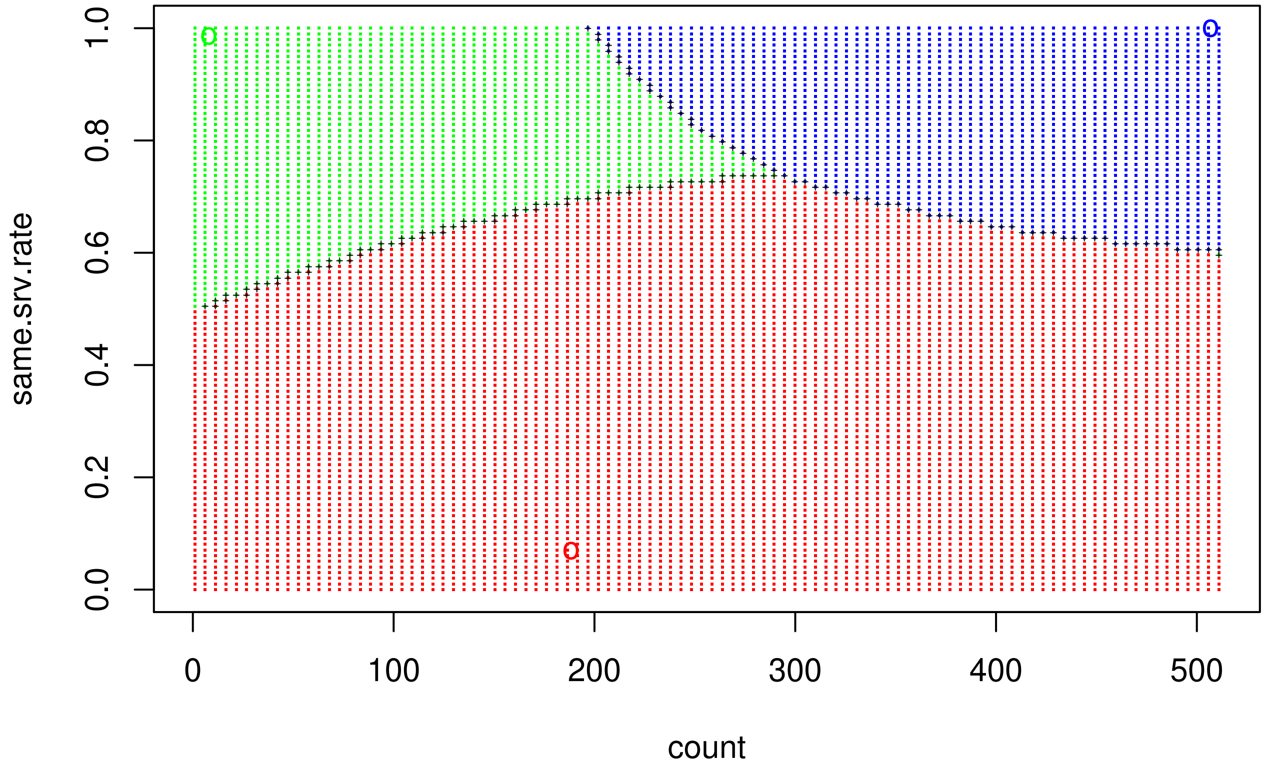
We can use derived features to get them.

To get quadratic boundaries, we can add the features

$$(count)^2 \quad (same.serv.rate)^2 \quad (count)(same.serve.rate)$$

to the linear discriminant analysis.

Fig 104. LDA Classification with Quadratic Boundaries



Neptune attacks shown in red, normal connections in green, and smurf attacks in blue. Prior is (0.2246, 0.2018, 0.5736); validation cer is 0.00508.

Commentary Linear on Discriminant Analysis

Linear discriminant analysis works surprisingly well even when its assumptions of normality and constant standard deviations and correlations across classifications are violated.

Hastie, Tibshirani, and Friedman (2001) speculate that the reason for this is that the variance in trying to determine classification boundaries is inherently so large that a lot of bias can be tolerated. Linear discriminant analysis is low variance because it only depends on the estimate of as many means, standard deviations, and pairwise correlations as there are classifications. If its assumptions are violated, the bias will increase. But apparently never gets large enough to do much harm.

Probabilistic Structure Revisited

For a given set of features x, a connection can be "neptune", "normal", or "smurf". Whichever it is can be thought of as being determined by tossing a single die: If the die lands 1, then the connection is "neptune"; if 2, then it is "normal"; if 3, then "smurf"; and if 4, 5, or 6, then toss again.

Now think of this die as being loaded and imagine that the features x tell you how much weight has been added to the faces 4, 5, and 6. Because the faces on the opposite sides of a die sum to 7, if x adds extra weight to face 6, then 1, which is a "neptune" connection is the most likely. The more weight added, the more likely "neptune" becomes.

Probabilistic Structure Revisited (continued)

This dice mechanism is actually an accurate representation of the probability model for a classification problem. What it does is define three conditional probabilities:

$$P(C = \text{“neptune”}\,|X = x)$$
$$P(C = \text{“normal”}\,|X = x)$$
$$P(C = \text{“smurf”}\,|X = x)$$

The first is the probability of the connection C being “neptune” if features X are known to take on the value x. The second and third are the values for “normal” and “smurf” respectively.

Logistic Regression

Logistic regression models these conditional probabilities directly. It does so by modeling the logarithm of the pairwise odds.

Odds are the ratio of two probabilities. For instance, the probability of a UK woman dying between the ages of 35 to 65 is 0.09, for an ex-smoker it is 0.10, and for a current smoker 0.16 (Callum, C. (1998). The UK smoking epidemic: deaths in 1995. London: Health Education Authority.) Thus, the odds of a UK woman dying of smoking are 0.16/0.09=1.78, nearly 2 to 1.

The logarithm of the ratio of two probabilities is called a logit. Hence the name, logit regression or logistic regression.

Logistic regression models the logit as a linear function of the features.

Logistic Regression (continued)

For our application the logistic model is

$$\begin{aligned}
\log\left(\frac{P(C = \text{“neptune”}\,|\,X = x)}{P(C = \text{“smurf”}\,|\,X = x)}\right) &= \beta_{01} + \beta_{11}(\text{catch}) + \beta_{21}(\text{same.srv.rate}) \\
\log\left(\frac{P(C = \text{“normal”}\,|\,X = x)}{P(C = \text{“smurf”}\,|\,X = x)}\right) &= \beta_{02} + \beta_{12}(\text{catch}) + \beta_{22}(\text{same.srv.rate}) \\
P(C = \text{“smurf”}\,|\,X = x) &= 1 - P(C = \text{“neptune”}\,|\,X = x) \\
&\quad - P(C = \text{“normal”}\,|\,X = x)
\end{aligned}$$

Here $P(C = \text{“smurf”}\,|\,X = x)$ is chosen as the denominator, but exactly the same classifications will result regardless of which category is chosen as the denominator. Only two ratios need to be modeled; the third can be gotten by subtraction because probabilities add to one.

Probabilistic Structure Revisited (continued)

Given a set of observed features x^o, the optimal classification strategy is to compute

$$P(C = \text{“neptune”}\,|X = x^o)$$
$$P(C = \text{“normal”}\,|X = x^o)$$
$$P(C = \text{“smurf”}\,|X = x^o)$$

and classify the connection according to whichever is largest. That is, if $P(C = \text{“normal”}\,|X = x^o)$ is the largest of the three, then for that x^o the connection is predicted to be “normal”.

Logistic Regression (continued)

The boundary between "normal" and "smurf" will be where

$$\frac{P(C = \text{"neptune"}\,|X = x)}{P(C = \text{"smurf"}\,|X = x)} = 1$$

which is where

$$\log\left(\frac{P(C = \text{"neptune"}\,|X = x)}{P(C = \text{"smurf"}\,|X = x)}\right) = 0$$

which is where

$$0 = \beta_{01} + \beta_{11}(\text{catch}) + \beta_{21}(\text{same.srv.rate})$$

which means that the boundaries for logistic regression, like those for linear discriminant analysis, are linear.

Logistic Regression (continued)

Hastie, Tibshirani, and Friedman (2001) state that in their experience, there is hardly any difference between the classification regions chosen by linear discriminant analysis and logistic regression.

This is just as well because, other than the binary case, where there are only two categories, it is hard to find code implementing the method. There is no code in R that works for our application, which has three categories.

Classification: Synthesis

- Linear discriminant analysis is the best of the probabilistic classification tools.

- With derived features, linear discriminant analysis is flexible.