Proteomics and Variable Selection

Alex Lewin

With thanks to Paul Kirk for some graphs

Department of Epidemiology and Biostatistics,

School of Public Health, Imperial College

a.m.lewin@imperial.ac.uk

Overview

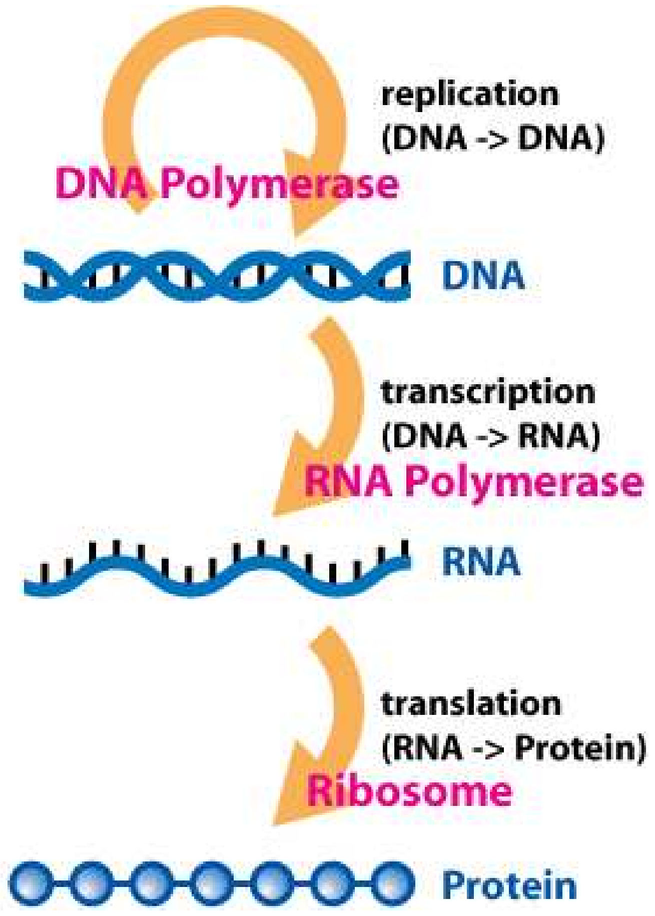
- Introduction to Proteomics and an Application
- Introduction to Classification and Regression
- Linear models
  - Linear regression: least squares
  - Penalised linear regression
  - Classification: logistic regression
- Prediction error and cross-validation
- Other models

Proteomics

Overall aim in molecular biology is the understanding of proteins and their functions in living organisms.

Proteomics is the large-scale study of proteins. We will be discussing the particular problem of inferring which proteins are present in particular groups of samples.

Aim to infer which proteins are important in particular biological processes.



Proteomics - Mass Spectrometry

Proteins ionized by laser, accelerated in electric field.

Measure time-of-flight t to infer mass to charge ratio.

$t^2 \propto m/z$

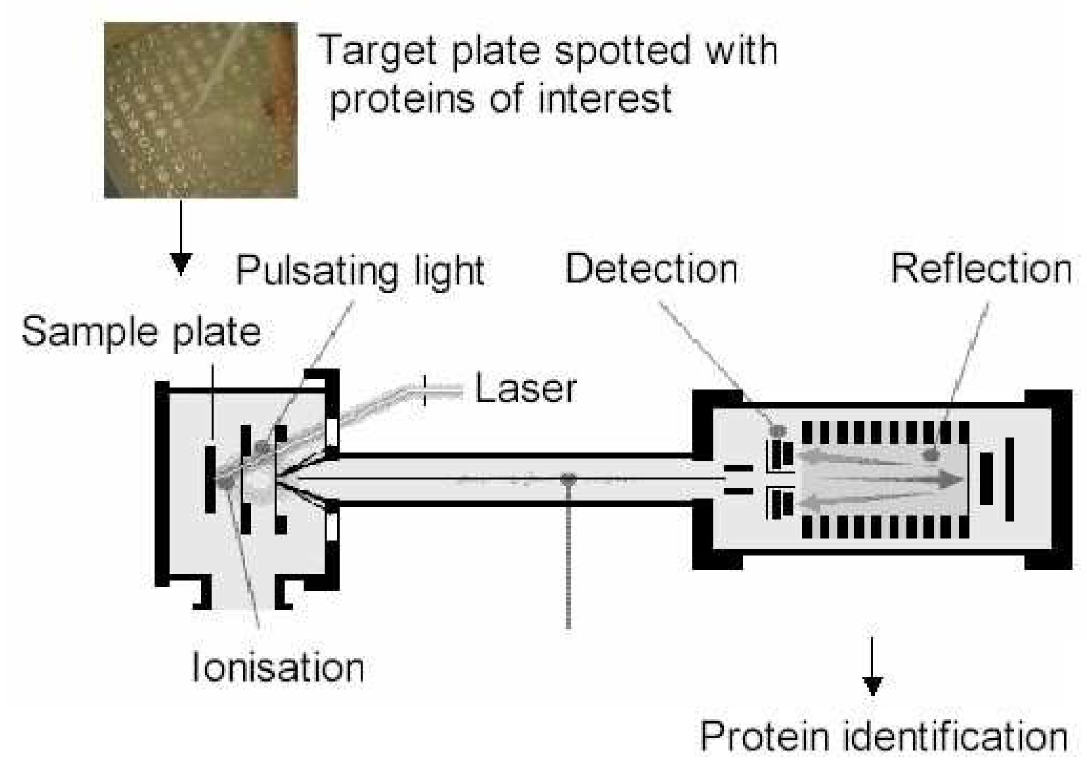


Image from http://biochemistry.wur.nl/Biochem/educatio/images/Maldi.jpg

Proteomics - Mass Spectrometry

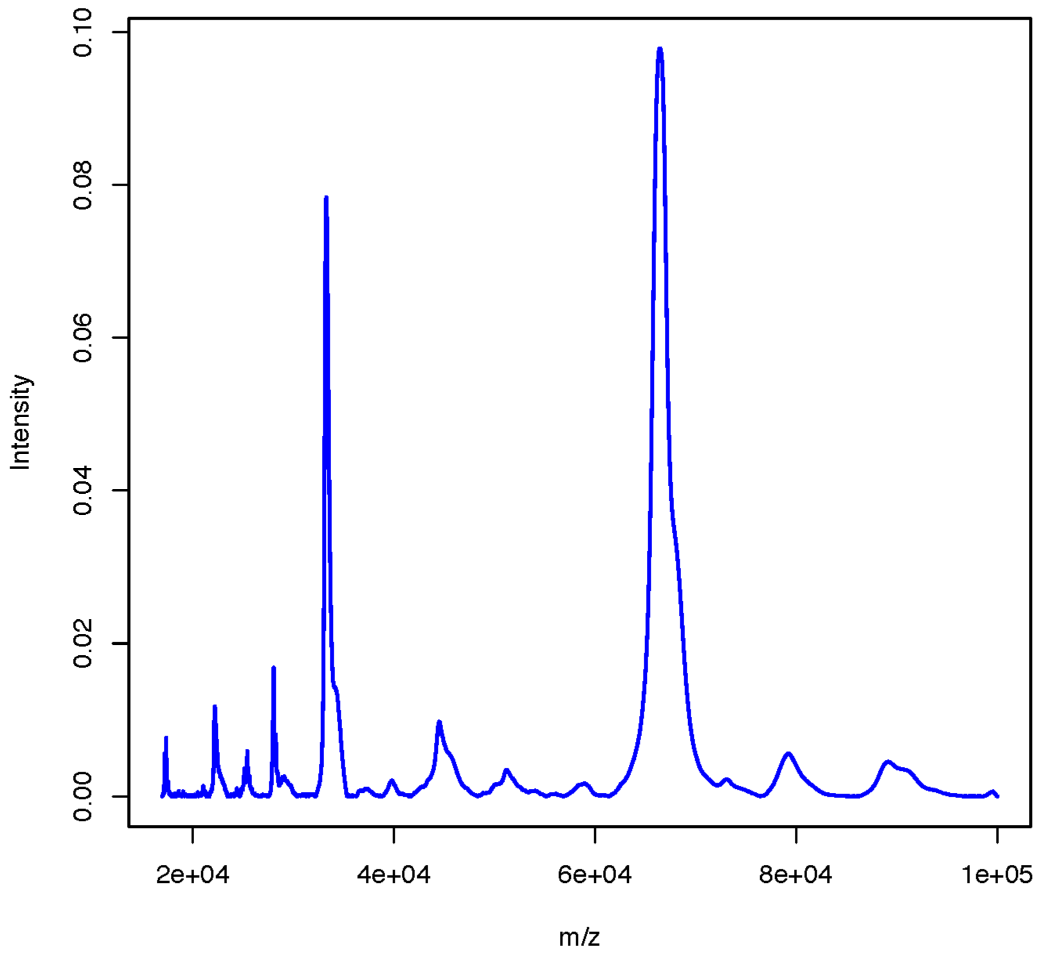
Mass spectrum is the distribution of counts per time interval:

intensity v. m/z

Intensity $\sim$ relative abundance of ions

Protein identified by presence of peaks with m/z corresponding to peptides contained in the protein.

Pre-processing of data: spectra from different samples are aligned; keep only the peaks which appear in more than a certain number of samples.



Application: HAM/TSP vs. AC

Human T-Lymphotropic Virus type 1 (HTLV-1) is a human RNA retrovirus that causes T-cell leukaemia and T-cell lymphoma.

HTLV-1 infection is also the cause of a debilitating disease called HAM/TSP.

The vast majority of infected individuals ($\sim 90\%$) remain lifelong asymptomatic carriers (ACs).

- Blood plasma samples were taken from 34 HAM/TSP individuals and 34 ACs.
- SELDI mass-spectrometry was used to investigate the protein content of these samples.

We seek to identify peaks in the spectra whose heights enable us to distinguish between the 2 classes of individuals.

Application: HAM/TSP vs. AC

Aim is to find a way to predict whether a new patient will go on to develop HAM/TSP or remain AC.

Select small number of proteins to base the classification on.

- Easier to measure smaller number of biomarkers for future diagnosis
- Interpretation of model (want to find proteins which are really involved in the biological process)

Introduction to Classification and Regression

Notation

n = number of individuals
p = number of input variables (here spectrum peaks)

Label individuals by i, where $i = 1, \cdots, n$
Label peaks by j, where $j = 1, \cdots, p$

There are two types of data for each individual:

- The outcome variable y_i is the class of individual i (eg. HAM or AC). This can also be called the ***response***.
- Input variables are those we use to try to make the prediction (in this case peak intensities). Call these x_{ij}.

Input variables can also be called ***covariates***, ***features, predictors*** or simply '***variables***'.

We try to pick a small number of the input variables for the classifier, hence ***variable selection*** or ***feature selection***.

Classification Rules

Now consider a new individual, labelled by $i = 0$.

- Know the peak intensities x_{0j} for the new individual.
- Don't know which group the new individual belongs to (HAM or AC).

Example of a prediction rule could be:

($x_{02} > 0.08$ and $x_{05} > 0.06$) $\Rightarrow y_0^* = HAM$

Here we use a star in y_0^* to indicate that this is a predicted quantity.

Relation to univariate analysis

Consider a problem with 2 classes (eg. two types of leukaemia: ALL and AML often studied with gene expression)

We could consider this as a differential expression problem:

1) Perform a test for each gene to see if its expression differs significantly between the classes. (eg. a modified t-test or permutation test).

2) Account for multiple testing.

This produces a list of genes whose expression differs between classes.

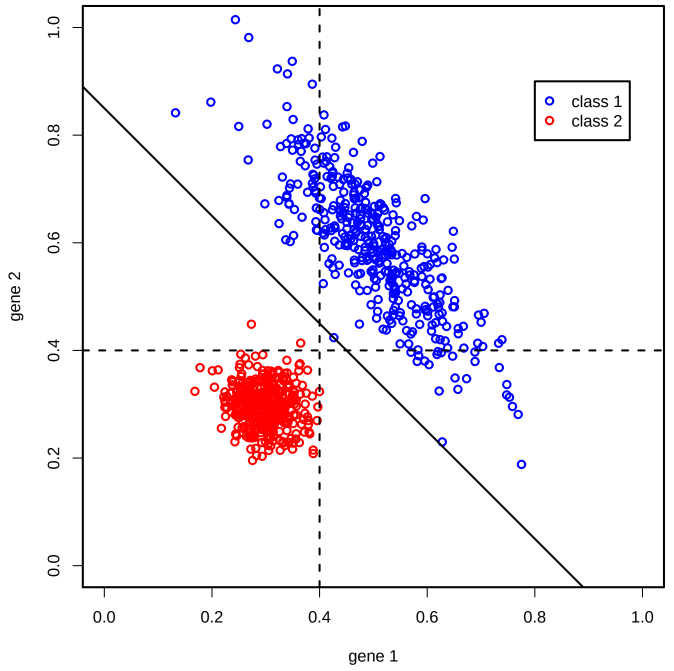
But,

- Treats each gene separately
- Takes no account of correlation between genes.
- Doesn't produce a prediction or classifier.

Relation to univariate analysis

Toy example: 400 individuals in each of 2 classes. Expression for 2 genes.

Neither gene alone can completely separate the 2 classes. But a combination of the two genes can separate all but outlying points.



Classifier using only gene 1 would be something like this:

$x_{01} < 0.4 \Rightarrow y_0^* = 2$

$x_{01} > 0.4 \Rightarrow y_0^* = 1$

But see that many points in class 1 would be mis-classified by this rule.

Same thing happens for a rule using only gene 2.

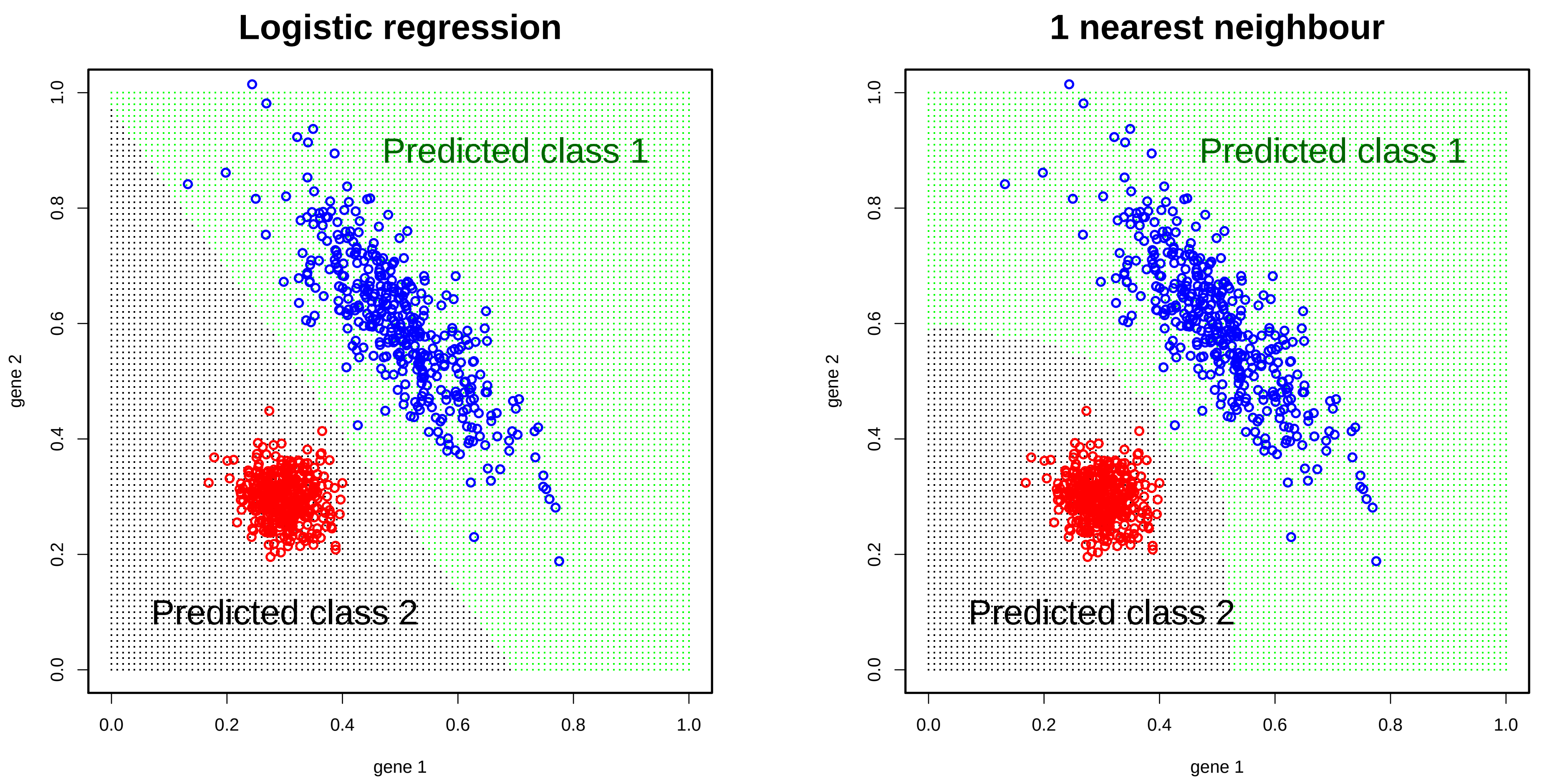
But using both genes, we can find a rule which separates the classes:

$x_{01} + x_{02} < 0.85 \Rightarrow y_0^* = 2$

$x_{01} + x_{02} > 0.85 \Rightarrow y_0^* = 1$

Classification

Decision boundaries don’t have to be linear:



Logistic regression finds a straight line separating classes.

1-nearest neighbour classifies points to the class of the nearest point in data set.

Classification

Choice of shape/smoothness of decision boundary: balance between

- too smooth may be too simple to separate classes well
- too wiggly will be sensitive to individual points in the data set used to find the boundary

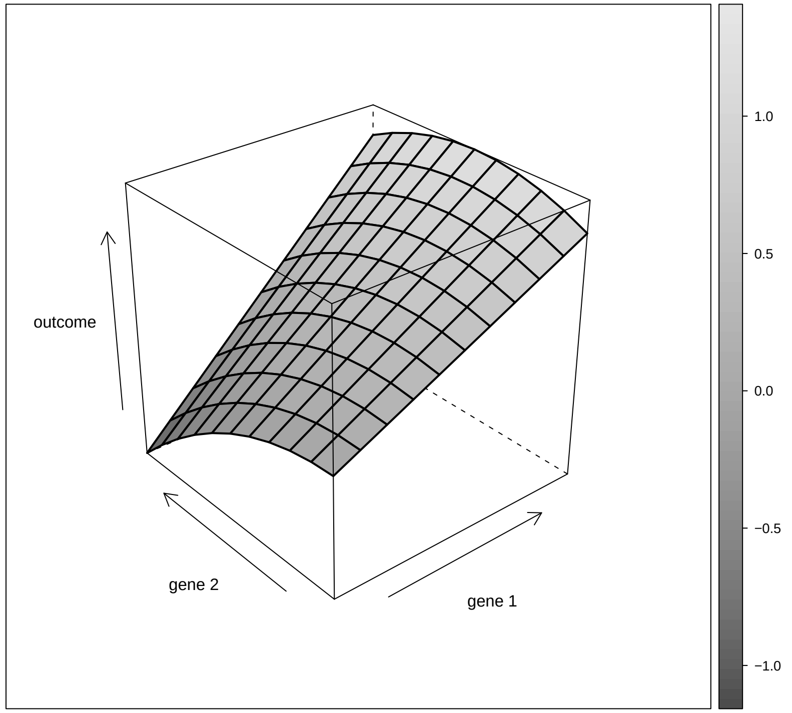
Trade-off between mis-classifying points in this data set and making bad predictions for future points.

Introduction to Regression

If outcome variable is continuous, process is called regression.

For example, in the HAM/TSP v. AC example, the proviral load has been measured for all individuals in the study. May wish to find proteins peaks whose intensity correlates with proviral load.

Find a surface which describes the relationship between the outcome and predictor variables.



Introduction to Regression

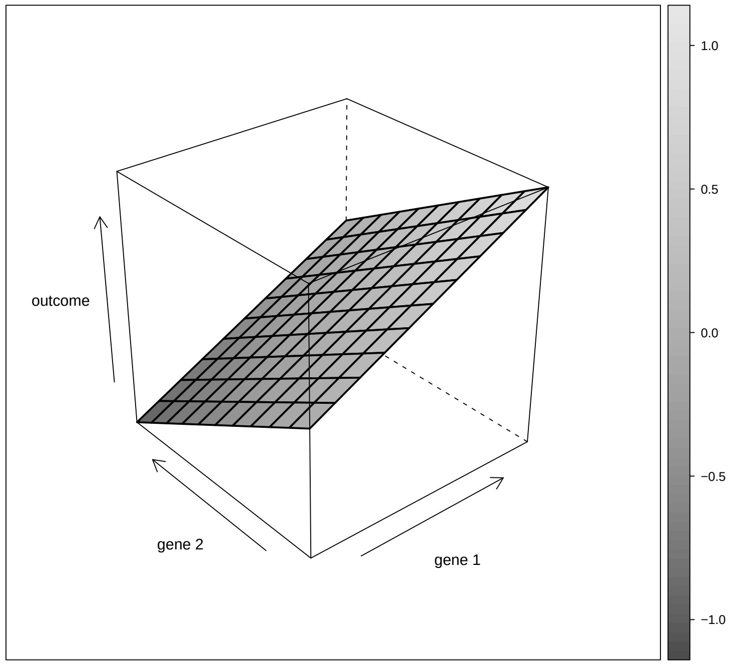
As with classification, there are two types of variable for each individual:

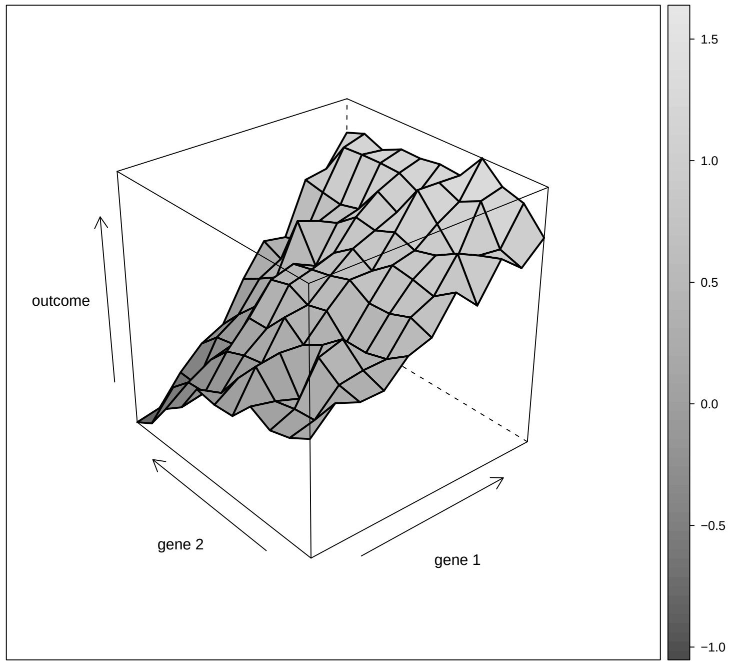
- The outcome variable y_i is now the continuous variable which is to be modelled in terms of the predictors.
- Input variables as before

Here we try to pick a small number of the input variables which explain the variability of the outcome variable.

Introduction to Regression

Here we have to choose how smooth to make the surface.



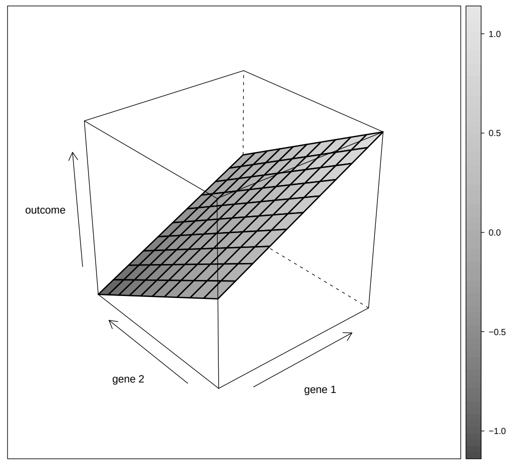


Choice of shape/smoothness of surface: balance between

- too smooth may give high errors on this data set
- too wiggly will be sensitive to individual points in the data set

Trade-off between high errors for points in this data set and high errors for future data.

Regression - linear models



Linear Regression: Ordinary Least Squares

Define random variables Y the outcome variable (here continuous) and X_j the input variable j. These correspond to observed y_i and x_{ij} for individual j.

Simple linear model:

$$y_i = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij} + \epsilon_i$$

β_0 is the intercept term representing the expectation of Y for an individual with all variables X_j zero.

Each variable j has its own regression coefficient β_j: this represents the effect that variable j has on the outcome, when all other variables are held constant.

(Linear model $\rightarrow$ variable has same effect whatever values of other variables.)

Linear Regression: Ordinary Least Squares

ϵ_i is the residual error for individual i, after the effect of variable j has been taken into account.

Fit the model by minimising the sum of squared residuals:

$$RSS = \sum_{i=1}^{n} \epsilon_i^2 = \sum_{i=1}^{n} (y_i - \beta_0 - \sum_{j=1}^{p} \beta_j x_{ij})^2$$

Solution can be written in matrix format.

Straightforward and quick computationally.

Details of matrix solution

Write the model as

$$\boldsymbol{y} = \boldsymbol{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

Here $\boldsymbol{y}$ and $\boldsymbol{\epsilon}$ are the n-vectors of the outcomes and residual errors for all individuals.

$\boldsymbol{\beta}$ is the ($p+1$)-vector of all regression coefficients, including β_0.

$\boldsymbol{X}$ is the $n \times (p+1)$ matrix of input variables for all individuals, where row i is ($1, x_{i1}, ..., x_{ip}$) - the 1 corresponds to β_0.

In matrix format, the solution for $\boldsymbol{\beta}$ minimising the residual sum of squares is

$$\hat{\boldsymbol{\beta}}^{OLS} = (\boldsymbol{X}^T\boldsymbol{X})^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

This involves matrix inversion of a $(p+1) \times (p+1)$ matrix.

Large p, small n problem

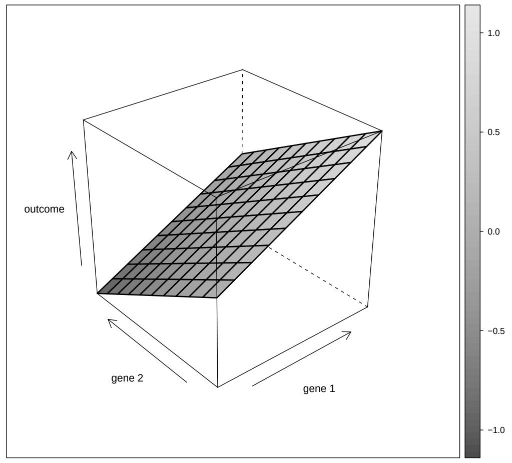
However, in most biomarker selection problems, p is much larger than n. In this case the solution for the β_j is not unique. (There are many different ways to write Y as a linear function of the X_j.)

This is known as the large p, small n problem. Present for all supervised learning models, not only linear models.

Hence we need variable selection. Some possibilities:

- filter the input variables
- subset selection
- penalisation methods shrink the regression coefficients or set some of them to zero.

Penalised linear regression



Penalised Linear Regression: Ridge regression

Penalised methods shrink the regression coefficients by imposing a penalty on their size.

Ridge regression uses the same linear regression model as before, but minimises a different quantity:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda \sum_{j=1}^{p}\beta_j^2$$

Parameter λ controls the amount of shrinkage. Larger λ gives more shrinkage: larger $\lambda \Rightarrow$ second term more important $\Rightarrow \sum_{j=1}^{p}\beta_j^2$ must be smaller.

An equivalent way to think of it is that β_j minimises the residual sum of squares subject to the constraint $\sum_{j=1}^{p}\beta_j^2 \leq s$, where s has a one-to-one correspondence with λ.

Details of matrix solution

The solution in matrix format is

$$\hat{\boldsymbol{\beta}}^{ridge} = (\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I}_p)^{-1}\boldsymbol{X}^T\boldsymbol{y}$$

where $\boldsymbol{I}_p$ is the $p \times p$ identity matrix.

Use centred and scaled variables in $\boldsymbol{X}$ and $\boldsymbol{y}$, and no intercept, so $\boldsymbol{X}$ is $p \times p$.

So the ridge regression solution is also straightforward and reasonably quick computationally.

Penalised Linear Regression: Lasso

In ridge regression, regression coefficients are shrunk towards zero, but are not actually zero. Lasso is a method which produces zeros for most regression coefficients.

Lasso minimises:

$$\sum_{i=1}^{n}(y_i - \beta_0 - \sum_{j=1}^{p}\beta_j x_{ij})^2 + \lambda\sum_{j=1}^{p}|\beta_j|$$

This time β_j minimises the residual sum of squares subject to the constraint $\sum_{j=1}^{p}|\beta_j| \leq t$, where t has a one-to-one correspondence with λ.

This modified penalty produces many zero regression coefficients, so the lasso performs a type of subset selection.

Penalised Linear Regression: Lasso

Lasso tends to pick one of a group of correlated predictors. Ridge gives the whole group smaller coefficients than if only one of the group was available.

The solution to the lasso minimisation is not linear in the $\boldsymbol{y}$, so $\hat{\boldsymbol{\beta}}^{lasso}$ cannot be written down as with ordinary least squares and ridge regression.

Several methods have been proposed for fitting the lasso: quadratic programming, least angle regression and co-ordinate descent. The last two involve step-wise procedures, starting with one predictor in the model and successively adding the next best in when required. They calculate the solutions for all values of λ at once.

Connections between fitting methods

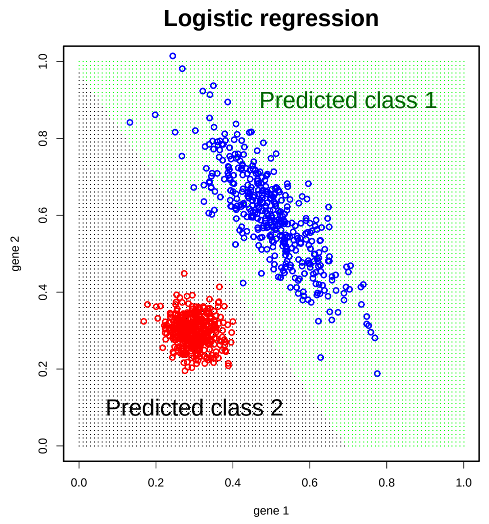
Ordinary least squares for linear regression is equivalent to maximum likelihood (assuming the outcome Y comes from a Normal distribution)

- Log likelihood for the mean of a Normal distribution is the sum of squares.

Penalised methods for linear regression are equivalent to Bayesian methods for estimating the regression coefficients (assuming Y is Normal)

- Sum of squares is the log likelihood term.
- Penalty terms correspond to a prior on the regression coefficients (Normal for ridge regression, Laplace or double exponential for lasso).
- But ridge/lasso find the maximum of the posterior distribution only.

Classification - generalised linear models



Classification: generalised linear models

Suppose there are G classes. Our aim is to model the probabilities of being in each class, given the values of the input variables $\mathbb{P}(\text{ class } g|\boldsymbol{x}_i), g = 1, \cdots G$.

Probabilities sum to one: $\sum_{g=1}^{G} \mathbb{P}(\text{ class } g|\boldsymbol{x}_i) = 1$ so there are $G-1$ quantities to estimate.

Cannot model the probabilities as linear because probabilities must be between 0 and 1. But we can transform:

$$\begin{aligned}
\log\left(\frac{\mathbb{P}(\text{ class } 1|\boldsymbol{x}_i)}{\mathbb{P}(\text{ class } G|\boldsymbol{x}_i)}\right) &= \beta_0^{(1)} + \sum_{j=1}^{p} \beta_j^{(1)} x_{ij} \\
&\vdots \\
\log\left(\frac{\mathbb{P}(\text{ class } G-1|\boldsymbol{x}_i)}{\mathbb{P}(\text{ class } G|\boldsymbol{x}_i)}\right) &= \beta_0^{(G-1)} + \sum_{j=1}^{p} \beta_j^{(G-1)} x_{ij}
\end{aligned}$$

So here we have $G-1$ linear functions (unlike in the continuous outcome case, where there is just one).

Classification: logistic regression

When there are only 2 classes, often label them by 0 and 1.

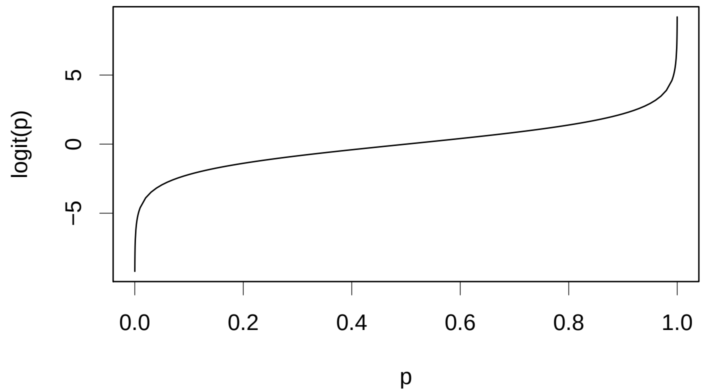
Here we have just one function to model:

$$\log\left(\frac{\mathbb{P}(\text{ class } 1|\boldsymbol{x}_i)}{\mathbb{P}(\text{ class } 0|\boldsymbol{x}_i)}\right) \quad = \quad \log\left(\frac{\mathbb{P}(\text{ class } 1|\boldsymbol{x}_i)}{1-\mathbb{P}(\text{ class } 1|\boldsymbol{x}_i)}\right)$$

It is usual to write $p(\boldsymbol{x}_i) = \mathbb{P}(\text{ class } 1|\boldsymbol{x}_i)$, and then the model is

$$\text{logit}(p(\boldsymbol{x}_i)) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$$

where $\text{logit}(p) = \log(p/(1-p))$. This is called the logistic function - hence this model is called the ***logistic regression model***.



Classification: logistic regression

Usually fit logistic regression models using maximum likelihood. The likelihood function is multinomial (assuming individuals are independent): $\mathcal{L} = \prod_{i=1}^{n} \mathbb{P}(\text{ class } y_i | \boldsymbol{x}_i)$.

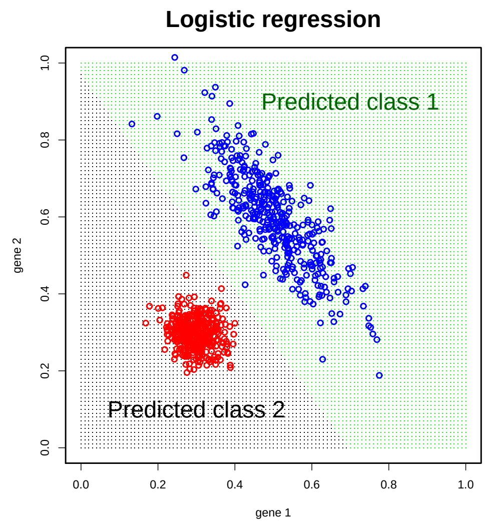
Here y_i is 0 or 1.

$$\begin{aligned}
\log\mathcal{L} &= \sum_{i=1}^{n} \log\mathbb{P}(\text{ class } y_i|\boldsymbol{x}_i) \\
&= \sum_{i=1}^{n} (y_i \log(p(\boldsymbol{x}_i, \boldsymbol{\beta})) + (1-y_i)\log(1-p(\boldsymbol{x}_i, \boldsymbol{\beta}))) \\
&= \sum_{i=1}^{n} \left(y_i \sum_{j=0}^{p} \beta_j x_{ij} - \log(1 + e^{\sum_{j=0}^{p} \beta_j x_{ij}}) \right)
\end{aligned}$$

So now we have the log Likelihood in terms of the observed y_i, $\boldsymbol{x}_i$ and the parameters β_j.

To find $\hat{\beta}_j$, solve $\partial \log\mathcal{L}/\partial\beta_j = 0$. These equations are non-linear in β_j, so an iterative method (iteratively weighted least squares) is used to solve them.

Penalised Classification



Classification: logistic regression

As with linear regression, logistic regression can be penalised, eg. with ridge or lasso penalties.

Maximise the penalised log likelihood:

$$\log\mathcal{L} - \lambda \sum_{j=1}^{p} \beta_j^2$$

or

$$\log\mathcal{L} - \lambda \sum_{j=1}^{p} |\beta_j|$$

Note the minus sign in front of the penalty, since maximising not minimising.

Fitting method for ridge logistic regression is iteratively weighted least squares, for lasso logistic regression the methods are similar to those for lasso linear regression.

Application: HAM/TSP vs. AC

34 HAM/TSP individuals and 34 ACs.

For each person, have 650 peak intensities.

Use logistic regression (classification) with lasso penalty to select small number of peaks:

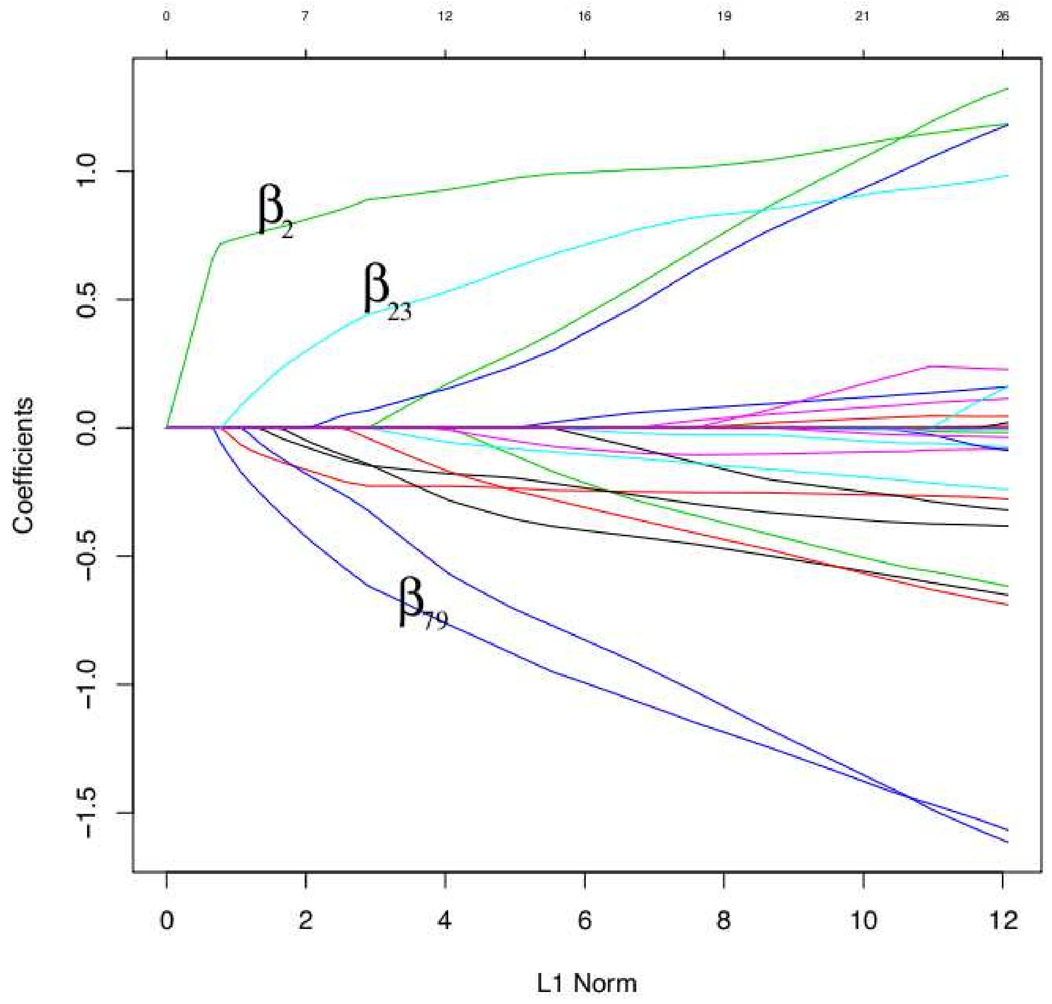
$$\text{logit}(\mathbb{P}(\text{ class HAM} \mid \text{peaks for person } i\ )) = \beta_0 + \sum_{j=1}^{p} \beta_j peak_{ij}$$

(i labels the 68 people, j labels the 650 peaks)

Lasso penalty maximises
$\text{logit}(\mathbb{P}(\text{ class HAM} \mid \text{peaks for person } i\ )) - \lambda \sum_{j=1}^{p} |\beta_j|$.

Regularisation Paths for HAM v. AC

Plot β_j v. λ for each j (one line per variable).

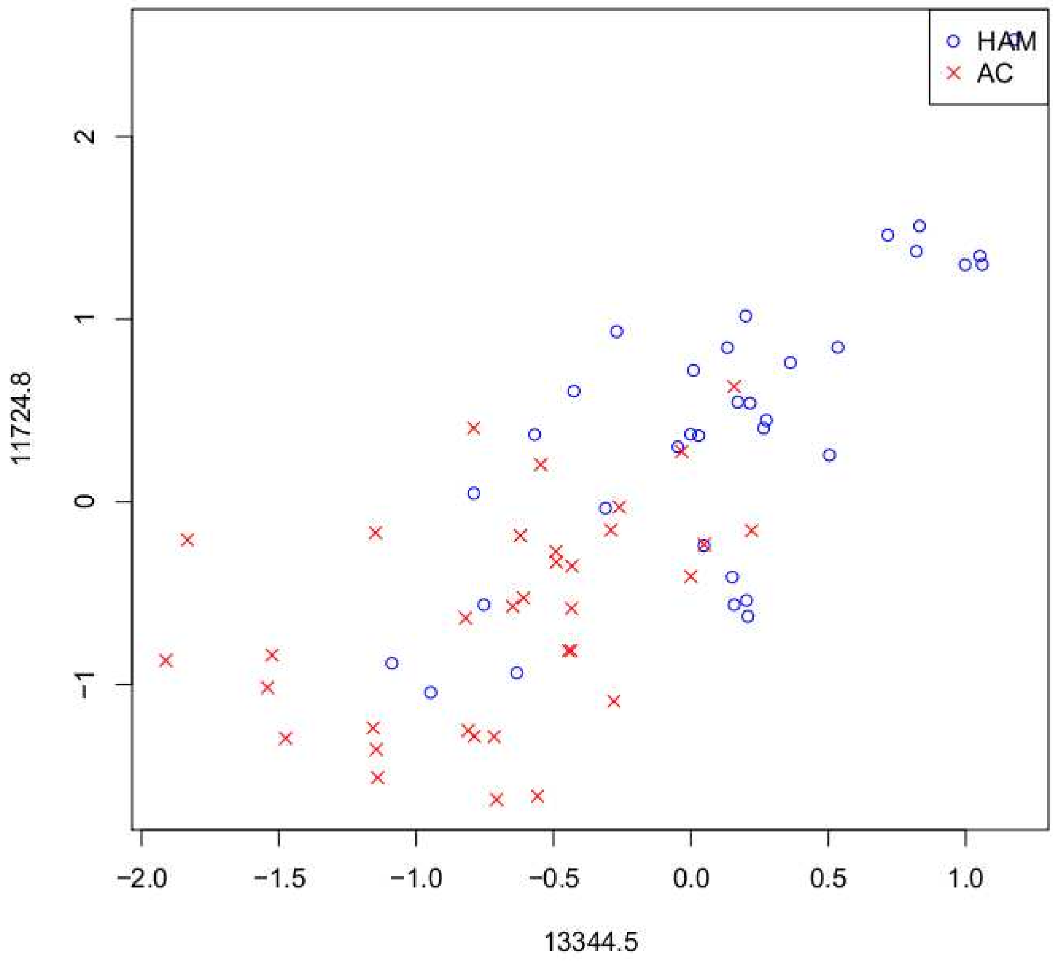


For small L1 norm (large λ), very few coefficients are non-zero. As λ decreases, more variables enter the model.

Prediction error and cross-validation

Separation of Classes (HAM v. AC)

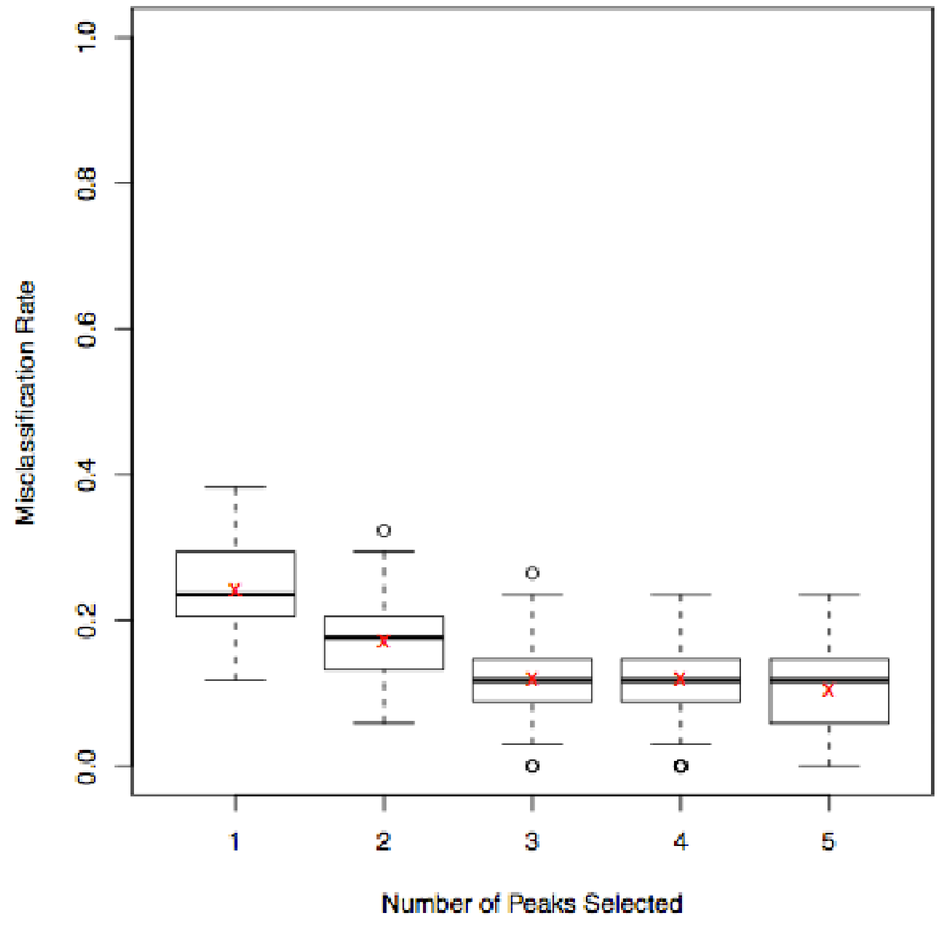
Disease status in terms of the top 2 peaks:



Good but not perfect separation. Can we separate the classes if we include more peaks?

Misclassification Error (HAM v. AC)

Look at numbers of misclassifications for different numbers of peaks.

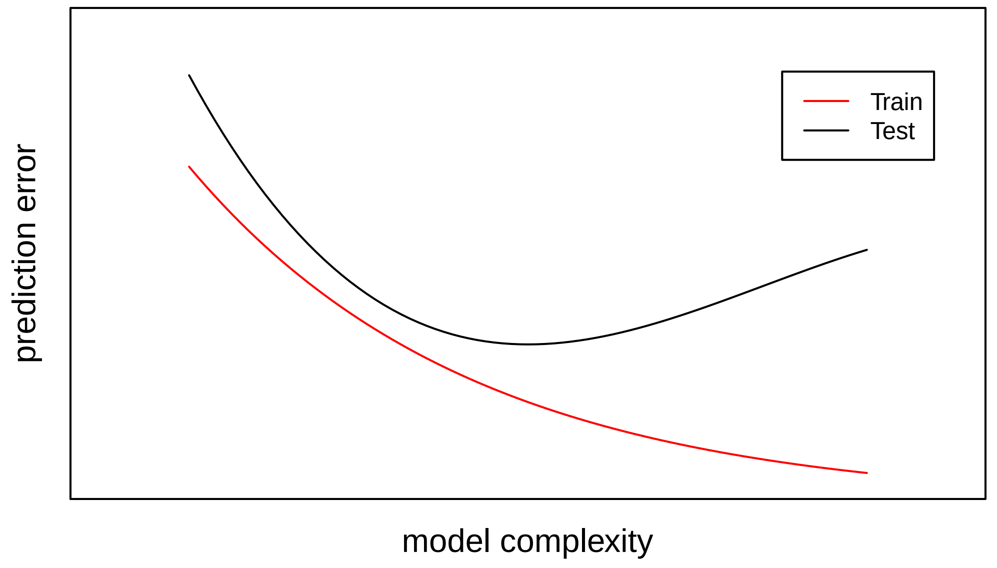


Error rate decreases as we include more peaks - where do we stop?

Use prediction error to choose model complexity

Decreasing $\lambda \Rightarrow$ less penalised $\Rightarrow$ more variables included in model (more terms in $\hat{f}$) $\Rightarrow$ more complex model $\Rightarrow$ better fit to the ***current*** data set

But if the model fits too well to the current data, we may be fitting to small details of the current data set which will not be present in future data.



Cross Validation

We seek a balance between the fit on current data and on future data.

Look at this within the one data set we have, by splitting it into ***training*** data and ***test*** data.

Fit the model to the training data (i.e. find the best fitting $\hat{\beta}^{lasso}$).

Calculate the error in the model using the test data (using fixed $\hat{\beta}^{lasso}$).

Important: this means that different data points are used for fitting the model and estimating the prediction error.

Prediction error for linear regression

Regression models attempt to find a function which describes the behaviour of the outcome Y in terms of the predictors $\{X_1, \cdots X_p\}$.

Write $\boldsymbol{x}_i$ for the vector of predictor variables for individual i. Then we write the function in general as

$$y_i = f(\boldsymbol{x}_i) + \epsilon_i$$

For example, in penalised linear regression, $f(\boldsymbol{x}_i, \boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$

Recall the estimate (best fit) of $\boldsymbol{\beta}$ is denoted by $\hat{\boldsymbol{\beta}}$. The function of $\boldsymbol{x}_i$ using the best fit values of the regression coefficients is $\hat{f}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i, \hat{\boldsymbol{\beta}})$.

Prediction error for linear regression

Prediction error is the error we find when we compare the best fit predictions from the model with the actual observed data.

For continuous outcomes the error usually used is the average squared error between the observed values and the predictions from the model:

$$err = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}(\boldsymbol{x}_i))^2$$

Prediction error for classification

Binary classification models in general can be written as

$$\mathbb{P}(y_i = 1|\boldsymbol{x}_i) = f(\boldsymbol{x}_i)$$

For the generalised linear models we have discussed so far, $f(\boldsymbol{x}_i, \boldsymbol{\beta}) = \beta_0 + \sum_{j=1}^{p} \beta_j x_{ij}$, and as before, the best fit function is that found using the best fit parameters, $\hat{f}(\boldsymbol{x}_i) = f(\boldsymbol{x}_i, \hat{\boldsymbol{\beta}})$.

Make a class prediction for each point:

$$y_i^* = \begin{cases} 1, & \hat{f}(\boldsymbol{x}_i) > 0.5; \\ 0, & \hat{f}(\boldsymbol{x}_i) < 0.5. \end{cases}$$

Prediction error for classification

How do we characterise prediction error (comparing y_i^* with y_i)?

One possibility is simply to count the number of misclassifications:

$$\frac{1}{n}\sum_{i=1}^{n} I[y_i^* \neq y_i] \tag{1}$$

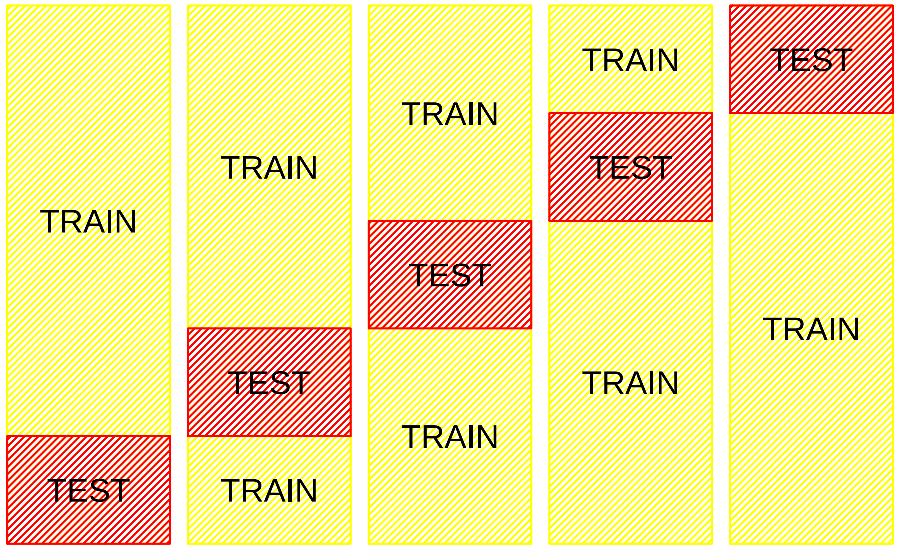
Notation: $I[\ ]$ is an ***indicator function***: it takes value 1 if the expression inside $[\ ]$ is true, 0 otherwise. (So the misclassification error is simply the proportion of misclassifications.)

Cross-validation

We can use ***cross-validation*** to get a better estimate of the prediction error.

Here we average the test error over several splits of the data.

Example: 5−fold cross−validation



With 5 splits, we obtain 5 different best estimates $\hat{f}_k$, $k = 1, \cdots 5$, one on each training set.

If we take the first split, the test error is

$$err_1 = \frac{1}{(n/5)} \sum_{i \in \text{TEST}_1} (y_i - \hat{f}_1(\boldsymbol{x}_i))^2$$

Cross-validation

Each individual appears in 4 training sets, but only 1 test set. We evaluate the prediction error for individual i using $\hat{f}_k$ for the split in which individual i appears in the test set and not the training set.

So if we label the test set for individual i by $k(i)$, we use $\hat{f}_k$ for the prediction error. The overall cross-validation prediction error for the model is then

$$CVerr = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_{k(i)}(\boldsymbol{x}_i))^2$$

It can be seen that this is the average of the test errors of the 5 sets:

$$CVerr = \frac{1}{5}\left(\frac{1}{(n/5)}\sum_{i\in\text{TEST}_1}(y_i - \hat{f}_1(\boldsymbol{x}_i))^2 + \cdots + \frac{1}{(n/5)}\sum_{i\in\text{TEST}_5}(y_i - \hat{f}_5(\boldsymbol{x}_i))^2\right)$$

Cross-validation

Cross-validation error can be used to find the optimal value of λ: calculate the prediction error on test data for a range of λ: the optimal λ minimises the prediction error.

Can also be used to compare models, eg. ridge v. lasso.

The split into groups for cross-validation must be chosen at random, not based on order of individuals in the study. (Picture shown above is just for convenience).

Also possible to average over many more random splits (so each individual appears in more than one test set, though never in the corresponding training set).

HAM v. AC example: Cross-validation outline

Data set contains 34 HAM, 34 AC.

Sample 100 subsets of 17 HAM, 17 AC.

Cross Validation:

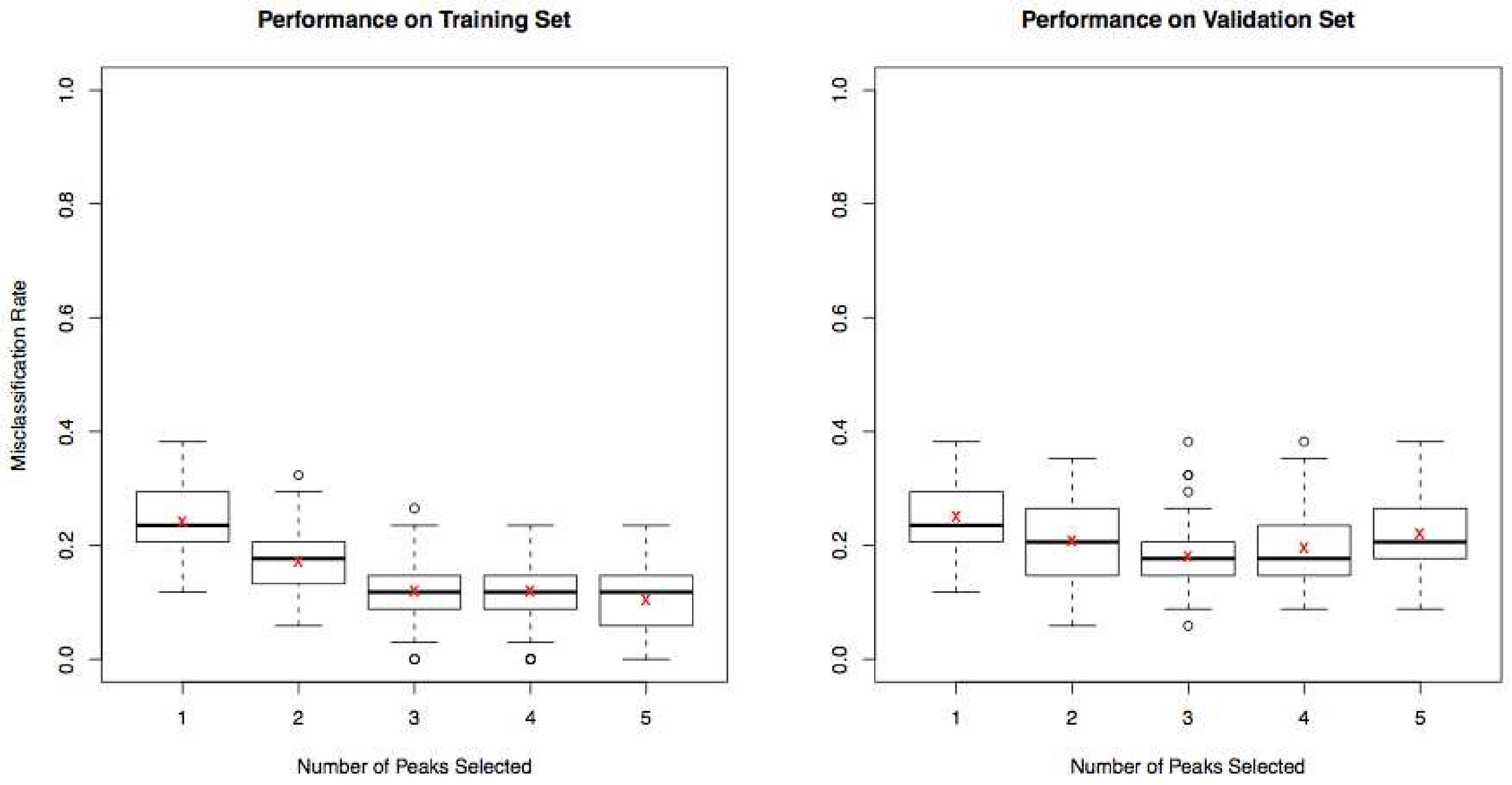
> Set λ.
> Run model 100 times (once for each subset).
> Get 100 test errors (on the left-out individuals).
> Average over the 100 subsets to get $CVerr(\lambda)$.
>
> Repeat for range of λ, find λ_{min} which has the minimum $CVerr(\lambda)$.

Set $\lambda = \lambda_{min}$.

Run model once on whole data set to get final model and peak selection.

Misclassification Error (HAM v. AC)

Look at numbers of misclassifications for different numbers of peaks.



The best validation misclassification rate is 18% errors, achieved using 3 peaks.

Final model (HAM v. AC)

Run lasso on all data, selecting the top 3 peaks.

$$\begin{aligned}\text{logit}(\mathbb{P}(\text{ class HAM} \mid \text{peaks for person } i\ )) &= 0.23 + 2.07 Intensity(peak 11.7) \\ &+ 1.65 Intensity(peak 13.3) \\ &- 1.34 Intensity(peak 2.1)\end{aligned}$$

The cross-validation error of 18% does not correspond exactly to these 3 peaks: it's an average over the results for the 100 subsets.

To find a good estimate of prediction error, obtain a separate validation data set, make class predictions using the above model and compare with the true classes (future work for this application).

Other methods

Some other methods for prediction

Many different methods available for prediction/discrimination:

- Decision trees: fit interaction terms; interpretable.
- Boosted trees: fit an average over many small trees, but iterative method to gradually improve prediction.
- Lots of non-linear methods fitting basis functions: spline functions, wavelets, support vector machines.
- Neural networks fit complex non-linear functions of predictors.
- Local methods: kernel smoothing and nearest neighbours.

Many of these methods do not provide automatic variable selection or ranking.

But active research area.

Bayesian methods

Methods described here are usually fit by optimisation $\rightarrow$ given one data set, find the one best solution.

Then may use bootstrap techniques to get an idea of uncertainty of parameter estimates.

Another approach is to use Bayesian models using MCMC (Markov Chain Monte Carlo) methods. These explore the parameter space more fully, enabling estimation of uncertainty in parameters and predictions.

Obtains more information, coherent inference. But complex computationally.

Also active research area.

Bibliography/Software

<u>Further Reading:</u>

The Elements of Statistical Learning (Hastie, Tibshirani and Friedman), pub. Springer.

<u>Software in R:</u>

glm for linear regression, logistic regression

lm.ridge for ridge regression

glmnet for lasso (linear or logistic regression)

Some of these are standard packages in R, others can be downloaded from http://cran.r-project.org/