MATH5743M: Statistical Learning - Lecture 10

Dr Seppo Virtanen, s.virtanen@leeds.ac.uk

Semester 2: 25 Apr 2022

Outline

- Standard error for linear regression
- Specifying 'own' models
- Example of overfitting
- Example of highly correlated inputs for linear regression
- K-means clustering

Standard errors - where do they come from?

The glm function gives us the standard errors on the parameters, which are a measure of our *uncertainty* about the true parameter values, and used to calculate confidence intervals.

One-dimensional linear regression model:

$$y_i = \beta_0 + \beta_1 x_i + \epsilon, \ \epsilon \sim \mathcal{N}(0, \sigma^2)$$

Lets focus on the parameter β_1. From week 3 we know that the MLE for this parameter is:

$$\hat{\beta}_1 = \frac{\mathrm{COV}(x, y)}{\mathrm{VAR}(x)}$$

Standard error for β_1

The MLE gives us the best single estimate of β_1, but we'd also like to know how much this estimate would vary if we were to 'repeat the experiment', i.e. if we were to select data points with the **same input values**, but which would have different values of y because of the **random variation of the residuals**, ϵ.

Standard error for β_1

Since our model tells us how the value of y_i is determined, we can substitute this back into our estimate:

$$\hat{\beta}_1 = \frac{\text{COV}(x, \beta_0 + \beta_1 x + \epsilon)}{\text{VAR}(x)}$$

Note here that the parameter values on the right hand side are the *true* (and unknown) values. Expanding this out gives:

$$\begin{aligned}
\hat{\beta}_1 &= \frac{\text{COV}(x, \beta_0) + \text{COV}(x, \beta_1 x) + \text{COV}(x, \epsilon)}{\text{VAR}(x)} \\
&= \frac{0 + \beta_1 \text{VAR}(x) + \text{COV}(x, \epsilon)}{\text{VAR}(x)} \\
&= \beta_1 + \frac{\text{COV}(x, \epsilon)}{\text{VAR}(x)}
\end{aligned} \tag{1}$$

Standard error for β_1

Therefore:

$$\hat{\beta}_1 - \beta_1 = \frac{\mathrm{COV}(x, \epsilon)}{\mathrm{VAR}(x)}$$

So the estimate will differ from the true value by some amount that depends on the (sample) covariance of the residuals and the input values, as well as the variance of the input values. What are the expectation and variance of this difference?

Standard error for β_1

We can treat $\mathrm{VAR}(x)$ as a constant because the input values are fixed, only the values of ϵ are random, so we need to focus on $\mathrm{COV}(x, \epsilon)$. First, the expectation: since ϵ and x are independent, this must be zero. That means that the expected value of $\hat{\beta}_1$ is simply β_1, i.e. the estimate is *unbiased* – on average we will get the true value.

Standard error for β_1

What about the variance? First we expand out $\text{COV}(x, \epsilon)$):

$$\begin{aligned}
\text{COV}(x, \epsilon) &= \frac{n}{n-1}\left[\frac{1}{n}\sum_{i=1}^{n} x_i\epsilon_i - \bar{x}\bar{\epsilon}\right] \\
&= \frac{n}{n-1}\left[\frac{1}{n}\sum_{i=1}^{n} x_i\epsilon_i - \bar{x}\frac{1}{n}\sum_{i=1}^{n}\epsilon_i\right] \qquad (2) \\
&= \frac{1}{n-1}\left[\sum_{i=1}^{n}(x_i - \bar{x})\epsilon_i\right]
\end{aligned}$$

Standard error for β_1

Now we find the (non-sample) variance of this quantity by applying the rules for the variance of a sum of independent variables: $\mathrm{VAR}(aX + bY) = a^2\mathrm{VAR}(X) + b^2\mathrm{VAR}(Y)$. Therefore:

$$
\begin{aligned}
\mathrm{VAR}(\mathrm{COV}(x, \epsilon)) &= \mathrm{VAR}\left(\frac{1}{n-1}\left[\sum_{i=1}^{n}(x_i - \bar{x})\epsilon_i\right]\right) \\
&= \frac{1}{(n-1)^2}\mathrm{VAR}(\epsilon)\sum_{i=1}^{n}(x_i - \bar{x})^2 \\
&= \frac{1}{(n-1)}\mathrm{VAR}(\epsilon)\mathrm{VAR}(x) \\
&= \frac{1}{(n-1)}\sigma^2\mathrm{VAR}(x)
\end{aligned}
\tag{3}
$$

Standard error for β_1

Finally we can use this result to calculate the variance of the error $\hat{\beta}_1 - \beta_1$:

$$\begin{aligned} \mathrm{VAR}(\hat{\beta}_1 - \beta_1) &= \mathrm{VAR}\left(\frac{\mathrm{COV}(x, \epsilon)}{\mathrm{VAR}(x)}\right) \\ &= \frac{\mathrm{VAR}\big(\mathrm{COV}(x, \epsilon)\big)}{\mathrm{VAR}(x)^2} \\ &= \frac{1}{(n-1)} \frac{\sigma^2}{\mathrm{VAR}(x)} \\ &= \frac{\sigma^2}{\sum_{i=1}^{n} (x_i - \bar{x})^2} \end{aligned} \tag{4}$$

Roll-your-own models

Consider a situation which is not perfectly addressed by either linear regression or logistic regression, and create a model based on the specific problem, before applying the general principles we have learnt.

Consider a group of anglers (people who fish with a rod and line). Each angler, $i \in 1, \ldots n$ goes down to the same lake and fishes for a different amount of time, t_i. During this time he or she catches a number of fish, $f_i \in 1, 2, \ldots, \infty$. How could we model this?

Roll-your-own models

The first thing to notice is that the amount of fish caught is a numeric value - this is a regression problem. But unlike the linear regression problems we've looked at before, this numeric value will typically only take relatively small integers - catching a fish is a rare event.

You have probably seen before that the standard distribution for modelling the integer number of rare events is the *Poisson distribution*, defined as below for the probability that the number of events, K is equal to some integer k:

$$P(K = k) = \frac{\exp(-\lambda)\lambda^k}{k!} \tag{5}$$

where λ is the mean or expected number of events.

Roll-your-own models

How can we use this to model our anglers' success, and predict the number of fish caught by another angler? Well if we assume that the expected number of fish caught is proportional to the time spent at the lake (i.e. there is a constant probability per unit time of catching a fish), this means that each individual will have a different value for this expectation, λ_i, which is proportional to their time at the lake:

$$\lambda_i = \beta t_i \tag{6}$$

where we have assumed that β is the same for everyone. In other words, the anglers are all equally good, but those who spend longer at the lake will *on average* catch more fish.

Roll-your-own models

With this assumption we can write down a formula for the probability of f_i, the number of fish caught by angler i, using the Poisson distribution formula above and substituting our expression for λ_i:

$$\begin{aligned} P(f_i \mid t_i) &= \frac{\exp(-\lambda_i)\lambda_i^{f_i}}{f_i!} \\ &= \frac{\exp(-\beta t_i)(\beta t_i)^{f_i}}{f_i!} \end{aligned} \tag{7}$$

Roll-your-own models

Note that this model now has one free parameter, β, that closely resembles the coefficients seen in both linear and logistic regression models – it tells us how strongly time at the lake influences the number of fish caught. So how can we estimate this parameter from a data set? By maximising the log-likelihood of course!

Roll-your-own models

So first we need to define this log-likelihood. Assuming that each angler's catch is independent of the others (no crossed lines or fights over fish!), then recall that the log-likelihood is simply the sum of the log-probabilities for each data point:

$$\begin{aligned}
\ln \mathcal{L} &= \sum_{i=1}^{n} \ln P(f_i \mid t_i) \\
&= \sum_{i=1}^{n} \ln \left(\frac{\exp(-\beta t_i)(\beta t_i)^{f_i}}{f_i!} \right) \\
&= \sum_{i=1}^{n} \left(-\beta t_i + f_i \ln(\beta t_i) - \ln(f_i!) \right)
\end{aligned} \tag{8}$$

Now that log-likelihood has some scary looking terms like $\log(f_i!)$, but remember that any terms like this that don't include β won't affect the maximum-likelihood value for β, since they don't change when β is varied.

Roll-your-own models

To get the MLE for β we need to maximise the log-likelihood, which we do the standard way, by differentiating and setting to zero at the MLE estimate $\hat{\beta}$

$$
\begin{aligned}
0 &= \frac{d \ln \mathcal{L}}{d\beta}|_{\hat{\beta}} \\
&= \sum_{i=1}^{n} \left(-t_i + \frac{f_i}{\hat{\beta}} \right) \\
&= \sum_{i=1}^{n} -t_i + \sum_{i=1}^{n} \frac{f_i}{\hat{\beta}}
\end{aligned}
$$

which can be rearranged to give:

$$
\hat{\beta} = \frac{\sum_{i=1}^{n} f_i}{\sum_{i=1}^{n} t_i} \tag{9}
$$

Roll-your-own models

We can make predictions for new anglers, or for test data (for doing model comparison). For example, if another angler is heading down to the lake for 60 minutes, and wants to know the probability that he will catch at least one fish, we can calculate this based on $\hat{\beta} = 0.0234$

$$
\begin{aligned}
P(f \geq 1 \mid \hat{\beta}, t = 60) &= 1 - P(f = 0 \mid \hat{\beta}, t = 60) \\
&= 1 - \frac{\exp(-\hat{\beta} \times 60)(\hat{\beta} \times 60)^0}{0!} \\
&= 1 - \exp(-0.0234 \times 60) \\
&= 1 - 0.245 \\
&= 0.755
\end{aligned} \tag{10}
$$

Roll-your-own models

By combining this with cross-validation and data splitting you can also perform model selection for different models. For example, you might think that another factor influences the catch, such as your experience level (in years of fishing experience, e_i). In that case you might try an alternative model such as:

$$\lambda_i = \beta_1 t_i + \beta_2 e_i,$$

and use cross-validation to test whether this makes better predictions than using time alone.

Modelling principles

1. What is the form of the output data? Classes or numbers? Integers or real numbers? Small or large values?
2. Based on the type of data, what is the best *distribution* to model it? Binomial (for binary classification)? Normal (for real values or large integers)? Poisson (for small integers, rare events)?
3. Using this distribution, how can we define a model that incorporates the input variables as predictors of the distribution parameters?
4. What is the *likelihood*? How can we *maximise it*.
5. How can we compare this model to others?

Modelling principles

Some useful distributions for modelling outputs include:

- Gaussian or normal distribution (linear regression)
- Bernoulli distribution (logistic regression)
- Poisson distribution (see the example above)
- Beta distribution
- Multinomial distribution

Beta distribution

Beta distribution for $x \in (0, 1)$,

$$x \sim \text{Beta}(\alpha, \beta) = \frac{\Gamma(\alpha + \beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha-1}(1-x)^{\beta-1},$$

where $\Gamma(\cdot)$ denotes the Gamma function.

Here, x takes continuous values within an interval with lower and upper values for the bounds.

The parameters of the distribution include continuously-valued shape and rate parameters $\alpha > 0$ and $\beta > 0$, respectively.

The expected value is given by $E[x] = \frac{\alpha}{\alpha+\beta}$

Multinomial distribution

Count data (non-negative integers) over more than two categories: The distribution generalises Bernoulli/Binomial distribution that is ssuitable for two categories.

For

$$\mathbf{x} \sim \text{Multinomial}(n, \mathbf{p}) = \frac{n!}{\prod_{k=1}^{K} x_k!} \prod_{k=1}^{K} p_k^{x_k},$$

where $n = \sum_{k=1}^{K} x_k$, K denotes the number of categories, x_k is the count for the kth category out of K and p_k is the probability for the kth category.

$x_k \in \{0, 1, 2, \ldots, n\}$, for $k \in \{1, \ldots, K\}$ and $p_k \geq 0$, for $k = 1, \ldots, K$, and $\sum_{k=1}^{K} p_k = 1$.

Multinomial distribution

Similarly to a special case of the Binomial distribution with $n = 1$ we get the Bernoulli distribution, for the the multinomial distribution with $n = 1$ we get the categorical distribution that may be used for classification tasks where we have more than two classes. Note here that outputs and parameters are bold-faced as they denote K-dimensional vectors.

The expected value is given by $E[x_k] = np_k$.

Modelling principles for GLM-type models

To construct GLM-type models for different types of outputs and tasks (regression vs classification) we need to connect the outputs y to inputs $x_1, x_2, \ldots, x_P$, that co-occur, using the expected value of the probabity distribution for the outputs. We have already seen:

- linear regression: $E[y] = \beta_0 + \sum_{p=1}^{P} \beta_p x_p$ (identity function, $I(x) = x$)
- logistic regression: $E[y] = \phi(\beta_0 + \sum_{p=1}^{P} \beta_p x_p)$, where $\phi(x) = \frac{1}{1+\exp(-x)}$ denotes the logistic function
- Poisson regression: $E[y] = \exp(\beta_0 + \sum_{p=1}^{P} \beta_p x_p)$

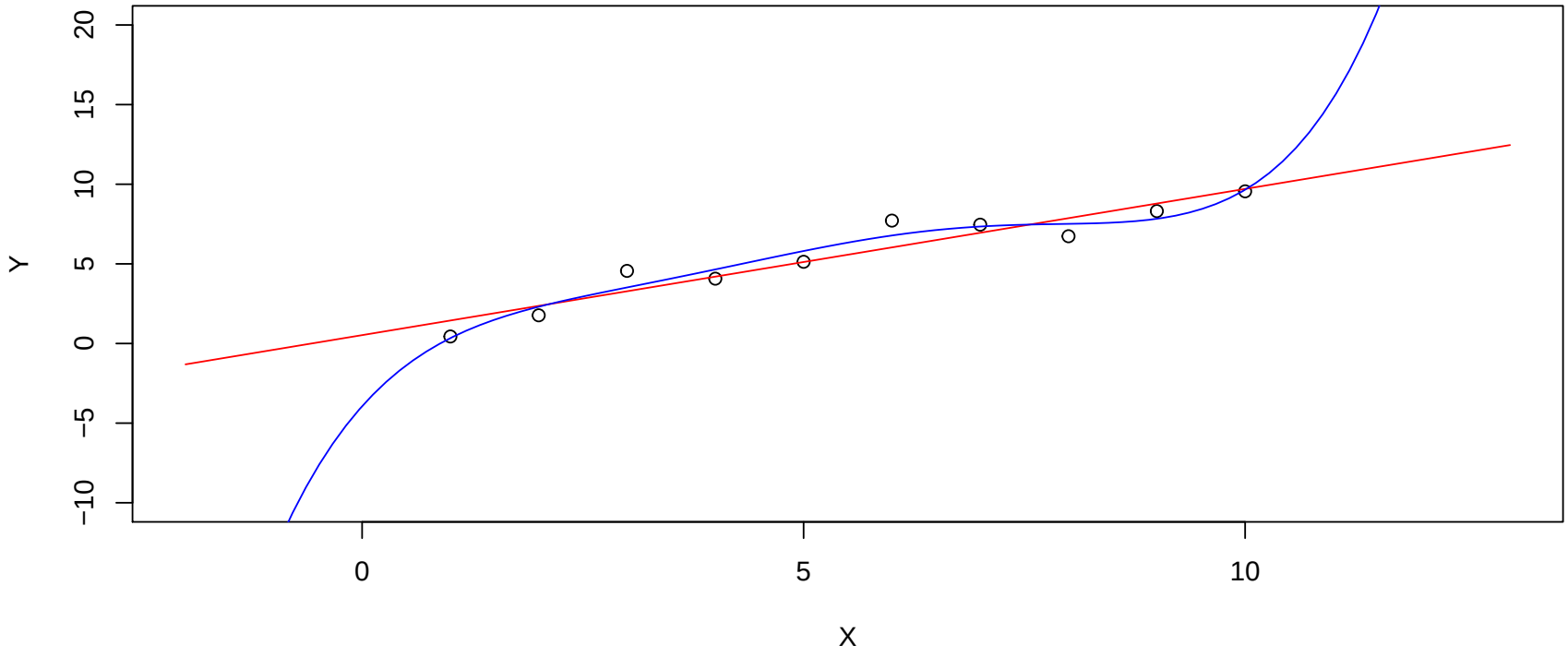
The modelling steps are similar to, though a bit more complicated for, the multinomial and Beta distributions, where we need to express $E[y]$ suitably by taking some function of the linear combination of the inputs to take into account restrictions or constraints of the probabiity distribution of the outputs.

Modelling principles for GLM-type models

After choosing the probability distribution for the outputs, parameterising the model with a suitable function for GLM-type models, we need to write down the joint likelihood function over the observed data points and maximise the log likelihood with respect to the model parameters.

This corresponds to maximum likelihood estimation that can be carried out analytically to obtain a closed form solution or by numerical optimisation when closed form expressions do not exist.

Overfitting demonstration



- We can see clearly here that the polynomial fit in blue is closer on average to the data points than the red linear model.
- But notice how quickly the predictions of the polynomial model move away from the red line when we move outside the range of the original data.

Difficulties and dangers for linear regression

- Linear regression can be applied so easily once you are familiar with the basic procedures in R that you might easily go ahead with an analysis of a data set before thinking about whether such a statistical model is likely to give you sensible results.

Highly correlated inputs

- Consider the following toy problem. I know that ice cream sales tend to be higher on warm, sunny days in summer.
- I'm interested to find out whether people buy more ice creams because of the number of hours of sunlight, or because of the temperature.
- Of course, sunlight and temperature are related, so when there are many hours of sunlight it will also tend to be warmer.

Highly correlated inputs

```
set.seed(55)
sunlight_hours = rnorm(20, 12, 3)
temperature = 10 + sunlight_hours + rnorm(20, 0, 1)
sales = 10 + temperature + rnorm(20, 0, 5)
icecream_data = data.frame(sunlight_hours, temperature,
                           sales)
```

Highly correlated inputs

```
icecream_model = glm(sales ~ sunlight_hours + temperature,
                     data=icecream_data)
summary(icecream_model)
```

```
##
## Call:
## glm(formula = sales ~ sunlight_hours + temperature, data
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -6.4564  -3.1500  -0.3761   3.2535   7.0245
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)    19.41075   12.16233   1.596    0.129
## sunlight_hours  1.10680    1.14160   0.970    0.346
## temperature    -0.08575    1.10523  -0.078    0.939
##
```

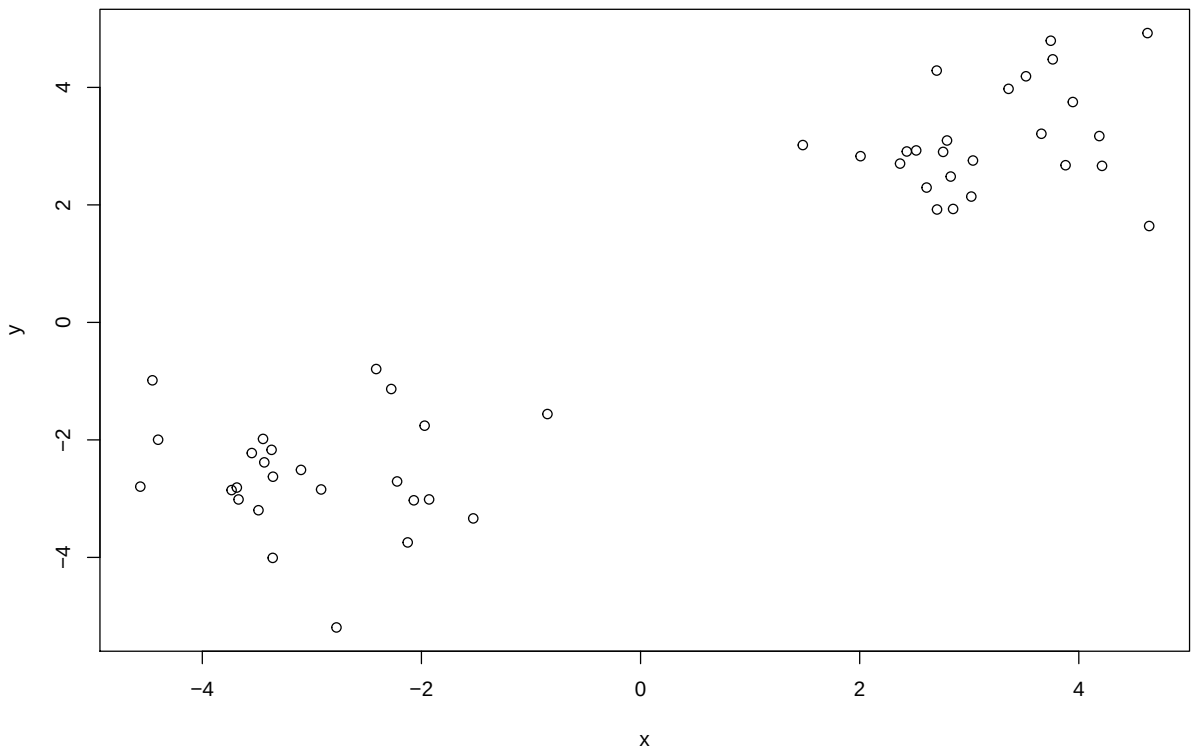
Replicate the process (change the 'seed' argument)

```
##
## Call:
## glm(formula = sales ~ sunlight_hours + temperature, data
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -4.2009  -2.9208  -0.6242   2.2657   7.7058
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)     15.6559     7.8310   1.999   0.0618 .
## sunlight_hours  -0.6087     0.7176  -0.848   0.4081
## temperature      1.0494     0.6578   1.595   0.1290
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1
##
## (Dispersion parameter for gaussian family taken to be 13
##
```

Unsupervised learning; K-means clustering

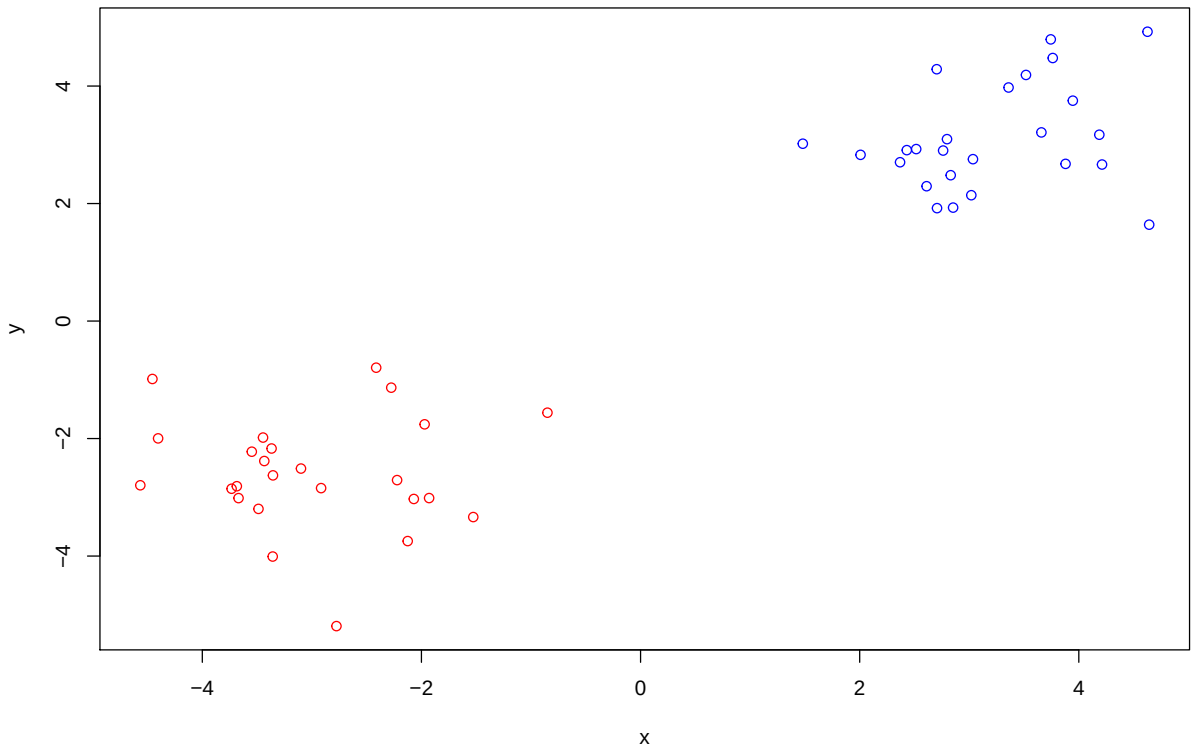
Example: *Clustering*, the partitioning on a number of data points into two or more distinct groups.

```
x1 = cbind(rnorm(25, -3, 1), rnorm(25, -3, 1))
x2 = cbind(rnorm(25, 3, 1), rnorm(25, 3, 1))
D = rbind(x1, x2)
plot(D[, 1], D[, 2], xlab="x", ylab="y")
```



Unsupervised learning; K-means clustering

```
kmeans_object = kmeans(D, 2)
cluster_id = kmeans_object$cluster
cols = c("Red", "Blue")
plot(D[, 1], D[, 2], col=cols[cluster_id],
     xlab="x", ylab="y")
```



K-means clustering

For a set of observations x, the k-means cost function is:

$$\arg\min_{C} \sum_{k=1}^{K} \sum_{x \in C_k} ||x - \mu_k||^2$$

. Here we use set notation C_k, for $k = 1, \ldots, K$ to represent partition of N data points.

Alternatively we may introduce cluster indicators for each data point. The cost function becomes

$$\sum_{n=1}^{N} \sum_{k=1}^{K} z_{n,k} ||x_n - \mu_k||^2,$$

where $z_{n,k}$ takes value 1 to indicate cluster membership for x_n and the remaining values are zero.

K-means clustering

1. initialise cluster centers μ_k, for $k = 1, \ldots, K$
2. allocate each data point to the nearest cluster (compute $z_{n,k}$)
3. update cluster centers μ_k by taking averate over the data points allocated to eacj cluster.
4. repeat steps 2 and 3 until the assignments do not change