Lecture 8

- Ensemble methods
- Artificial Neural Networks

Presenter: Dr Heshan Kumarage

Week-by-week outline

Week Starting	Seminar	Topic	App Ses	A1	A2	Q/P	A3	Due Date
2/3/2026	1	Introduction to Data Science, R, review of basic statistics	-					
9/3/2026	2	Data visualisation	S1					
16/3/2026	3	Data manipulation	S2					
23/3/2026	4	Regression modelling	S3					
30/3/2026	5	Clustering	S4					
6/4/2026	-	Mid-semester Break						
13/4/2026	6	Classification using decision trees	S5					17/4/2026
20/4/2026	7	Improving and evaluating classifiers. Naïve Bayes classification	S6					
27/4/2026	8	Ensemble methods, Artificial Neural Networks	S7					
4/5/2026	9	Network analysis	S8					
11/5/2026	10	Introduction to text analysis	S9					15/5/2026
18/5/2026	11	Text analysis applications	Quiz/Prac					22/5/2026
25/5/2026	12	Text Network Analysis, Review of the unit, Assignment 3	S10,11,12					
1/6/2026		SWOT VAC	-					
8/6/2026		EXAM PERIOD	-					12/6/2026

Assignment 2

Your task	• The objective of this assignment is to gain familiarity with classification models using R. • This is an individual assignment.
Value	• This assignment is worth **20%** of your total marks for the unit. • It has 40 marks in total.
Suggested Length	• 8 – 10 A4 pages, approximately 1,000 words (for your report) + extra pages as appendix for your R script. • Font size 11 or 12pt, single spacing.
Due Date	**11.55pm Friday 15th May 2026**
Submission	• Submit a single PDF file and single video presentation file on Moodle. • Use the naming convention: *FirstnameSecondnameID.{pdf, mp4, mov etc.}* • Turnitin will be used for similarity checking of all written submissions.
Generative AI Use	• Statement on Generative AI required by the university: In this assessment, you can use generative artificial intelligence (AI) in order to search for R functions and examples to perform tasks that you specify only. Any use of generative AI must be appropriately acknowledged (see Learn HQ).
Late Penalties	• 5% (2 mark) deduction per calendar day for up to one week. • Submissions more than 7 calendar days after the due date will receive a mark of zero (0) and no assessment feedback will be provided.

Assignment 2

Instructions and data

The objective of this assignment is to gain familiarity with classification models using R. You will be using a modified version of the World Values Survey (WVS) data that was used for Assignment 1. We now want to create Machine Learning models to predict the level of confidence in different social organizations based on participant responses to other selected attributes. For Assignment 2, confidence in individual social organisations has been re-coded as binary class variables. You can read more on the World Values Survey here: https://www.worldvaluessurvey.org/WVSContents.jsp

There are two options for compiling your written report:
(1) You can create your report using any word processor with your R code pasted in as machine-readable text as an appendix, and save as a pdf, or
(2) As an R Markdown document that contains the R code with the discussion/text interleaved. Render this as an HTML file and save as a pdf.

Your video report should be less than 100MB in size. You may need to reduce the resolution of your original recording to achieve this. Use a standard file format such as .mp4, or mov for submission.

Assignment 2

Creating your data set

Clear your workspace, set the number of significant digits to a sensible value. Download "**WVSBinaryExtract.csv**" and create your individual data using the following code. Your individual data will have Country, Wave, **three** different binary class variables (confidence in social organisations) and **30** coded participant responses (attributes), with **20,000** observations.

```
rm(list = ls())
set.seed(XXXXXXXX) # Your Student ID is the random seed
WD = read.csv("WVSBinaryExtract.csv")
selected_cols = c(sample(3:49, 30),sample(50:63, 3))
WD = WD[c(1:2, selected_cols)]
WD = WD[sample(nrow(WD), 20000, replace = FALSE),]
```

Use your individual data created above to answer all questions.

Assignment 2

Questions (18 Marks)

1. Explore the data: What is the proportion of "**High**" (Confidence = 1), to "**Low**" (Confidence = 0) for your three binary class variables? Obtain descriptions of the other predictor attributes. Is there anything noteworthy in the data? Are there any attributes you need to consider omitting from your analysis? **(2 Marks)**

2. Document any pre-processing required to make the data set suitable for the model fitting that follows. **(1 Mark)**

3. Divide your data into a 70% training and 30% test set by adapting the following code (written for the iris data). Use your student ID as the random seed.

```
set.seed(XXXXXXXX) #Student ID as random seed
train.row = sample(1:nrow(iris), 0.7*nrow(iris))
iris.train = iris[train.row,]
iris.test = iris[-train.row,]
```

Assignment 2

4. Using your training data and each of the techniques below, implement models to separately classify **each** of your three class variables. For this question you may use the R functions at their default settings if suitable. Note: Do not use any class variables as predictor attributes. **(5 Marks)**

- Decision Tree
- Naïve Bayes
- Bagging
- Boosting
- Random Forest

Assignment 2

5. Using the test data, classify each of the test cases as "High" (Confidence = 1) or "Low" (Confidence = 0) for each of your three class variables using each of the models implemented in Question 4. Create confusion matrices and report the accuracy, precision, recall, and F1-score of each model in classifying each class variable. Summarise your results in **three tables**, one for each class variable. **(3 Marks)**

6. Using the test data, calculate the confidence (probability) of predicting "High" (Confidence = 1) or "Low" (Confidence = 0) using each classifier, and construct ROC curves for each of your three class variables. Make a separate plot for each class variable with all the curves for that class on the same axis. Use a different colour for each classifier. This will give you **three plots** with **five curves** in each. Calculate and report the AUC for each classifier over each class variable. **(4 Marks)**

7. Compare the results in Questions 5 and 6 in relation to the performance of all classifiers across your three class variables. From the five classifiers, is there a single "best" classifier for predicting each of the classes? Is there a "worst" **ensemble** classifier overall? Do the "best" or "worst" classifiers vary between your three class variables? Explain your reasoning. **(3 Marks)**

Assignment 2

Investigative Tasks (18 Marks)

8. Examining each of the **ensemble** models for each of your three class variables, determine the most important attributes in predicting confidence. Summarise your results in **three** tables for each class variable. Which attributes are important overall for predicting confidence in each class variable? Which attributes could be omitted from the data with very little effect on performance? Give reasons. **(5 Marks)**

9. Looking at the relative strengths and weaknesses of each model, and consulting relevant references, can you explain any difference in performance between the models across each of the class variables? That is, why do some of these models perform better than others on the type of data to be fitted? **(2 Marks)**

10. Improve the worst performing **ensemble** model from Question 7. You may do this by adjusting parameters, attribute selection, cross-validation of the basic ensemble model and/or any other applicable process. Show that your new model is better using F1-scores, ROC curves and AUC. Describe how you created your improved model. What factors were important in your decision? State why you chose the attributes you used. **(4 Marks)**

Assignment 2

11. Using the insights from your analysis so far, implement an Artificial Neural Network classifier to compare predictability over time for the country with the greatest number of observations in your data. Using the training data for that country, fit an Artificial Neural Network model to classify confidence in **one** of your three class variables. Describe the attributes used and data pre-processing for implementation. **(3 Marks)**

12. Using the Artificial Neural Network model from Question 11, classify test instances of the selected country separately for the two **Waves** having the greatest number of observations in your test data. How does the model performance change over the two waves? Use F1-scores, ROC curves and AUC to compare. **(4 Marks)**

Assignment 2

Report and Video Presentation (4 Marks)

Write a brief report (suggested length 8 – 10 pages) summarizing your results. Include your R script, copied and pasted as machine readable text, as an appendix. Use commenting in your R script, where appropriate, to help a reader understand your code. Alternatively combine working, comments and reporting in R Markdown. **(2 Marks)**

Record a short presentation using your smart phone, Zoom, or similar method. Your presentation should be approximately 5 minutes in length and summarise your main findings, as well as describing how you conducted your research, and any assumptions made. Pay particular emphasis to your results for the investigative tasks. **(Submission Hurdle and 2 Marks)**

Ensemble Methods

In this section we'll cover:

- Ensemble methods for classification
- Bagging
- Boosting
- Random forests

COVID-19 Patient Health Prediction Using Boosted Random Forest Algorithm

Celestine Iwendi [1], Ali Kashif Bashir [2], Atharva Peshkar [3], R. Sujatha [4], Jyotir Moy Chatterjee [5], Swetha Pasupuleti [6], Rishita Mishra [7], Sofia Pillai [8] and Ohyun Jo [9*]*

Random forest model boosted by the AdaBoost algorithm. The model uses the COVID-19 patient's geographical, travel, health, and demographic data to predict the severity of the case and the possible outcome, recovery, or death. The model has an accuracy of 94% and a F1 Score of 0.86 on the dataset used. The data analysis reveals a positive correlation between patients' gender and deaths, and also indicates that the majority of patients are aged between 20 and 70 years.

https://www.ncbi.nlm.nih.gov/

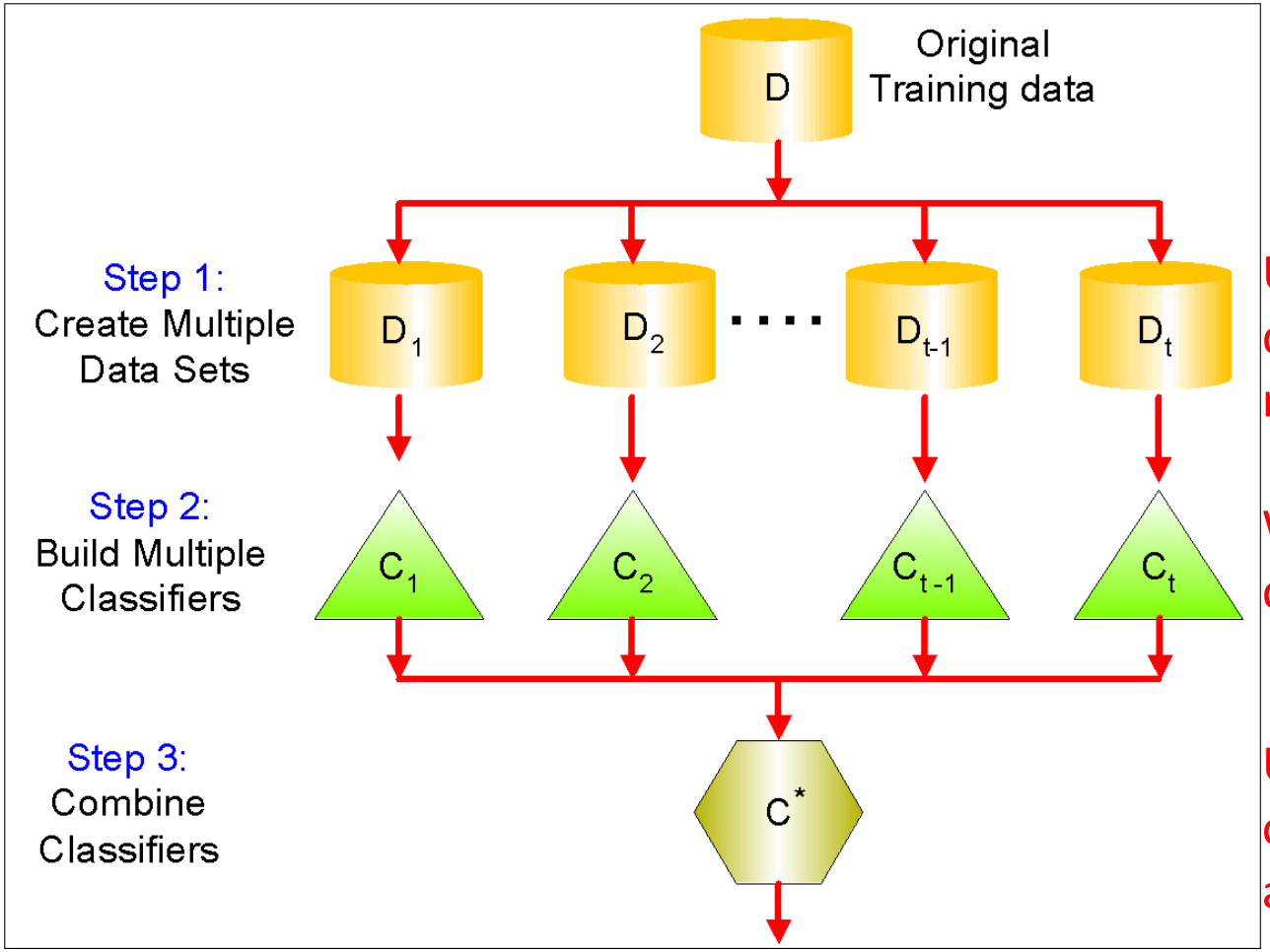
Ensemble methods for classification

To improve classification accuracy:

- Build a collection of 'experts' by constructing a set of classifiers from the training data:
- They could be the same or different types of classifiers.
- Combine the results of each classifier.

The enables the creation of a better classifier from a collection of weaker classifiers.

Ensemble methods: general idea



Using whole data set or resampling

We will use decision trees

Using votes or weighted average

Ensemble methods for classification

Work best when:

- The individual classifiers are moderately (> 50%) accurate.
- Individual classifiers are created independently.
- Pooling the results of each classifier reduces the variance of the overall classification.
- Decision trees work well as the individual classifiers.
- Disadvantage: model produced is not as easy to interpret as a single tree.

Bagging (Bootstrap aggregation)

Bagging algorithm:

- Make multiple replicates of the original data by sampling with replacement from the training set. Replicates are the same size as original.
- Construct a single classifier for each replicate.
- Combine the classifiers by taking a majority vote to produce the final decision.

Bagging (Bootstrap aggregation)

- Bootstrapping: resampling the original data set to produce multiple synthetic data sets.
- Sampling is uniform, each bootstrap replicate may have multiple instances of the original data points and contains approximately 63% of the original data set.
- Build a classifier on each of the replicates and combine results by voting. Multiple classifiers reduce the (high) variance of individual decision trees, (recall: sample variance is $\frac{\sigma}{\sqrt{n}}$).

When to use bagging

- Useful when there is noise in the data.
- Useful for unstable classifiers – that is, small changes in the training data cause large changes in the classifier. Unstable classifiers include decision trees, neural networks, linear regression.
- Not recommended for stable classifiers such as K Nearest Neighbours, Naïve Bayes.

Bagging in R

Example with package “adabag” and Iris data. Package has numerous dependencies. You may also need to install “rpart”.

- Getting started and making training and test set:

```
> install.packages("adabag")
> library(adabag)
> library(rpart)
> data(iris)
```

Bagging in R

Create a stratified sample: randomly choose 25 of each species of iris.

```
> sub <- c(sample(1:50, 25), sample(51:100, 25),
  sample(101:150, 25)) #(row numbers of training data)
```

The variable "sub" is a vector of row numbers (indices) for the training set.

Iris[sub,] is training set, iris[-sub,] is the test set.

Bagging in R

Fitting the model (number of trees is 10)

```
> ibag <- bagging(Species ~ ., data=iris[sub,], mfinal=10)
```

Test the model by making prediction from test set:

```
> ibpred <- predict.bagging(ibag, newdata=iris[-sub,])
> table(observed = iris[-sub,5], predicted = ibpred$class)
```

	predicted		
observed	setosa	versicolor	virginica
setosa	25	0	0
versicolor	0	23	2
virginica	0	1	24

Bagging in R

Check the fitted model for more details on:

```
> names(ibag)
  "formula"      "trees"       "votes"
  "prob"         "class"       "samples"
  "importance"   "terms"       "call"
```

To see all the elements of the model listed type:

```
> ibag
```

Bagging in R

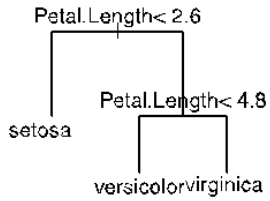
From the help file:

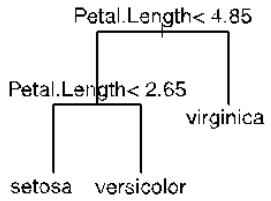
`formula`	the formula used.
`trees`	the trees grown along the iterations.
`votes`	a matrix describing, for each observation, the number of trees that assigned it to each class.
`prob`	a matrix describing, for each observation, the posterior probability or degree of support of each class. These probabilities are calculated using the proportion of votes in the final ensemble.
`class`	the class predicted by the ensemble classifier.
`samples`	the bootstrap samples used along the iterations.
`importance`	returns the relative importance of each variable in the classification task. This measure takes into account the gain of the Gini index given by a variable in each tree.

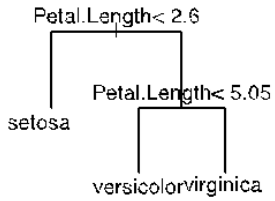
Bagging in R

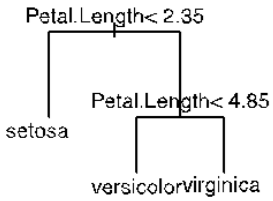
To see the trees, adapt code for Tree 1.

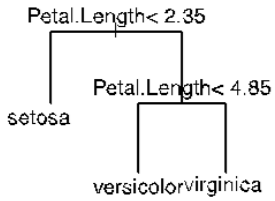
> plot(ibag$trees[[1]]); text(ibag$trees[[1]], pretty = 0)

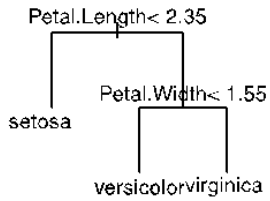


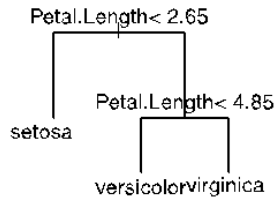


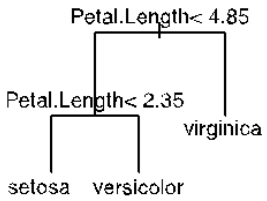


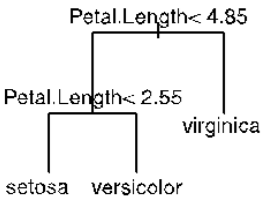


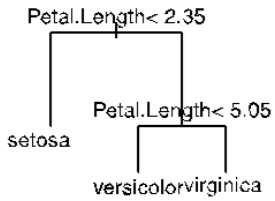












Bagging in R

To see the number of votes for each sample in the test set:

```
> ibag$votes
      [,1] [,2] [,3]
...
[28,]    0   10    0
[29,]    0    7    3
[30,]    0    8    2
[31,]    0   10    0
[32,]    0   10    0
...
```

Sample, votes for: Setosa, Versicolor, Virginica

Bagging in R

Votes have a direct translation to the confidence of each prediction:

```
> ibag$prob
          [,1] [,2] [,3]
...
[28,]  0.0  1.0  0.0
[29,]  0.0  0.7  0.3
[30,]  0.0  0.8  0.2
[31,]  0.0  1.0  0.0
[32,]  0.0  1.0  0.0
...
```

Bagging in R

Variable importance tells the relative contribution of each variable to the classification:

```
> ibag$importance
  Petal.Length Petal.Width Sepal.Length Sepal.Width
  83.34        16.66          0.00         0.00
```

References: Bagging

Alfaro, Gámez and García, adabag: An R Package for Classification with Boosting and Bagging., Journal of Statistical Software, 54(2) 2013

- Read: Abstract, Introduction, Section 3. Functions for extended example on Bagging with R and analysis of output.

James et al., An Introduction to Statistical Learning with Applications in R. Springer. Chapter 8.

Wikipedia: Bagging

Boosting

- Multiple trees are grown slowly, using incremental improvement.
- Original data set is used for all trees. Training examples are weighted, with a higher weight given to hard to classify examples at each step.
- Classification is by a weighted sum from each classifier, with more accurate classifiers having a greater weight.

Boosting

Boosting Algorithm

For each tree:

- Assign equal weights to each point in training set; fit basic tree.
- Repeat *n* iterations: Update weights of misclassified items and normalise; update tree, building on current tree.
- Output the final classifier as weighted sum of votes from each tree.

Boosting: pros and cons

- Enables improved classification of imbalanced data sets (where most instances are of one class).
- Tends to achieve better accuracy than bagging but can lead to over fitting if the number of trees is too great.
- There are numerous Boosting algorithms, we will use Adaptive Boosting, AdaBoost.

Boosting in R

Example with package "adabag" and Iris data.

- As for Bagging:

```
> library(rpart)
> data(iris)
> sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
> iboost <- boosting(Species ~ ., data=iris[sub,], mfinal=10)
> ibpred <- predict.boosting(iboost, newdata=iris[-sub,])
```

Boosting in R

Example with package “adabag” and Iris data. :

> ibpred$prob

> table(observed = iris[-sub,5], predicted = ibpred$class)

	predicted		
observed	setosa	versicolor	virginica
setosa	25	0	0
versicolor	0	22	3
virginica	0	1	24

Boosting in R

Check the fitted model for more details on:

> names(iboost)

```
"formula"     "trees"       "weights"
"votes"       "prob"        "class"
"importance"  "terms"       "call"
```

> For example:

> iboost$weights

```
2.1520 1.6392 1.1197 2.3634 1.2963 0.6482
 1.0740 2.3231 0.6339 1.3406
```

Trees make a weighted contribution to prediction

Boosting in R

For example, the number of votes for each sample in the test set, taking into account the weighting given to different trees:

```
> iboost$votes
          [,1] [,2] [,3]
...
[24,] 14.590  0.000  0.0000
[25,] 14.590  0.000  0.0000
[26,]  0.000 12.837  1.7536
[27,]  1.341 13.250  0.0000
[28,]  0.000 14.590  0.0000...
```

Boosting in R

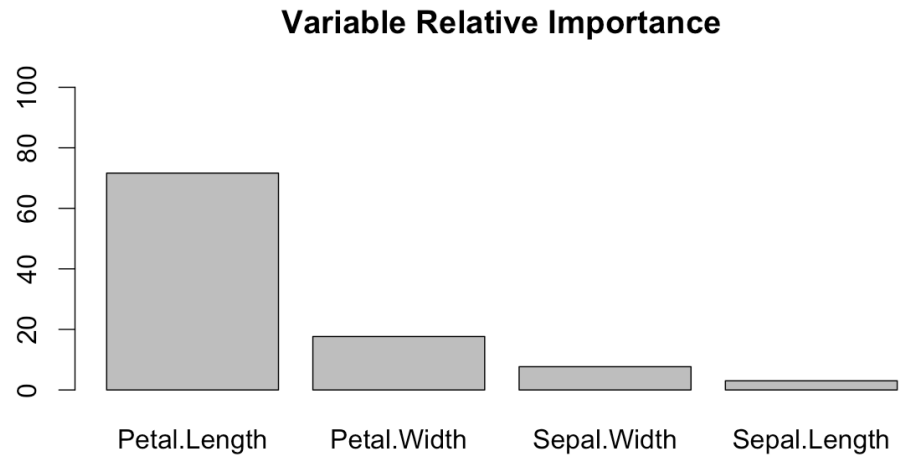
Observing the proportional contribution of each variable to information gain:

```
> iboost$importance
Petal.Length  Petal.Width Sepal.Length  Sepal.Width
       71.632       17.656        3.032        7.681
```

Boosting in R

Information gain, as a plot:

```
> # adapted from Alfaro et al.
> barplot(iboost$importance[order(iboost$importance, decreasing = TRUE)], ylim = c(0, 100), main = "Variable Relative Importance")
```



References: Boosting

Alfaro, Gámez and García, adabag: An R Package for Classification with Boosting and Bagging., Journal of Statistical Software, 54(2) 2013

- Read: Abstract, Introduction, Section 3. Functions for extended example on Boosting with R and analysis of output.

James et al., An Introduction to Statistical Learning with Applications in R. Springer. Chapter 8.

Wikipedia: Boosting (machine learning)

Random Forest

A refinement of bagged decision trees, specifically designed for decision trees.

Random Forest Algorithm

- Create multiple data sets from the original training set using subsets of data points and subsets of attributes.
- Build a decision tree classifier for each data set.
- Combine the classifiers by taking a majority vote to produce the final decision.

Advantages of Random Forests

More accurate than individual trees on large data sets

- No need to prune
- Not sensitive to outliers
- Over fitting is not a problem

Difference between Bagging and Random Forests:

- Bagging varies sample data and the number of trees.
- Random forests vary the attributes used to build tree as well as the sample data and the number of trees.

Random Forest in R

Example with package “randomForest” and Iris data:

```
> install.packages("randomForest")
> library(randomForest)
> sub <- c(sample(1:50, 25), sample(51:100, 25), sample(101:150, 25))
> iris.rf <- randomForest(Species ~ ., data=iris[sub,])
> iris.pred <- predict(iris.rf, iris[-sub,])
```

Random Forest in R

Confusion matrix

> table(observed = iris[-sub, "Species"], predicted = iris.pred)

	predicted		
observed	setosa	versicolor	virginica
setosa	25	0	0
versicolor	0	23	2
virginica	0	1	24

Random Forest in R

Prediction confidence levels

> niris.pred <- predict(iris.rf, iris[-sub,], type="prob")

```
     setosa versicolor virginica
...
45    1.000      0.000     0.000
53    0.000      0.754     0.246
56    0.000      1.000     0.000
57    0.000      0.898     0.102
58    0.036      0.802     0.162
60    0.026      0.858     0.116
61    0.034      0.856     0.110
...
```

Random Forests in R

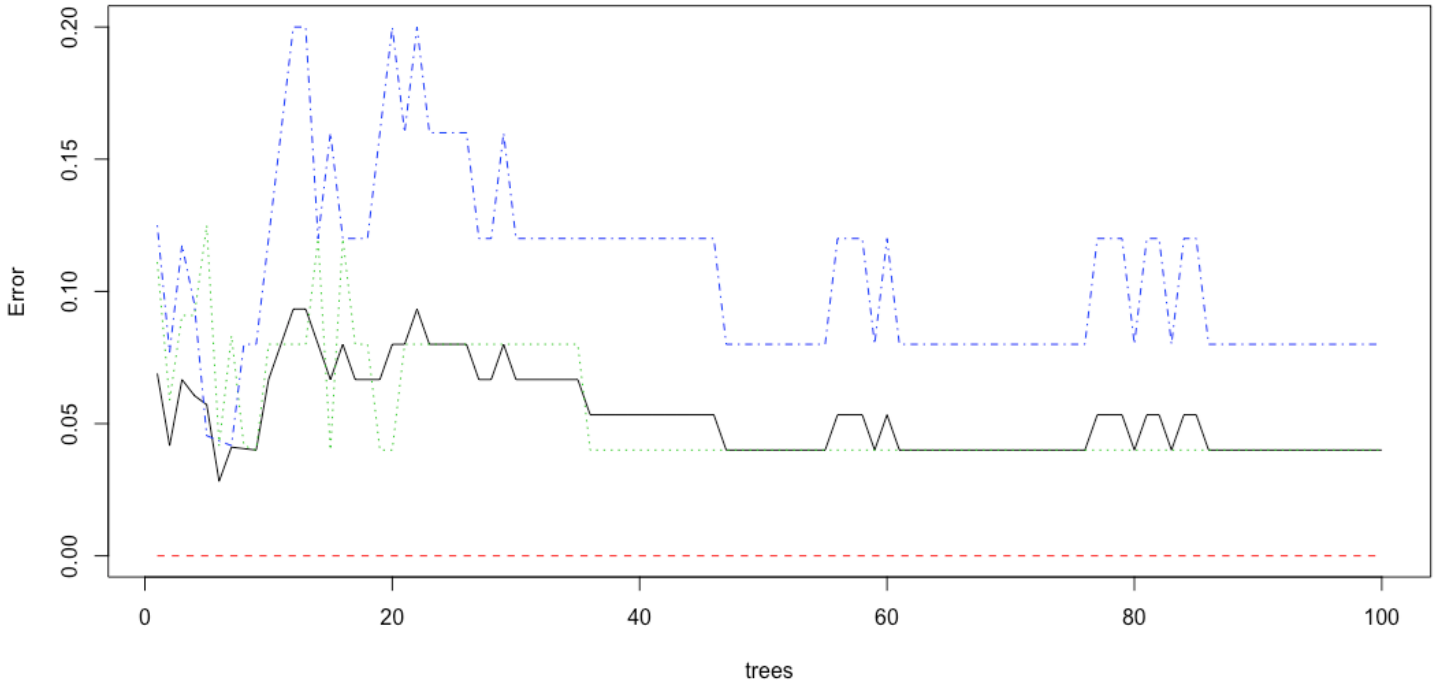
To see the elements of the model

```
> names(iris.rf)
 "call"            "type"
 "predicted"       "err.rate"
 "confusion"       "votes"
 "oob.times"       "classes"
 "importance"      "importanceSD"
 "localImportance" "proximity"
 "ntree"           "mtry"
 "forest"          "y"
 "test"            "inbag"
 "terms"...
```

Random Forest in R

How does number of trees affect error (MSE)?

> plot(randomForest(Species ~ ., data=iris[sub,], keep.forest=FALSE, ntree=100))



References: Random Forest

Breiman, Random Forests. Machine Learning 45, 5 – 32, 2001.

- Abstract and Introduction (paper is very technical)

James et al., An Introduction to Statistical Learning with Applications in R. Springer. Chapter 8.

Wikipedia: Random forest

Random Forests (Breiman and Cutler)

Cross Validation

All these models can be further improved by cross validation. Investigating this is up to you. From the package manuals:

> bagging.cv(formula, data, v = 10, mfinal = 100, control)

> boosting.cv(formula, data, v = 10, boos = TRUE, mfinal = 100, coeflearn = "Breiman", control)

> rfcv(trainx, trainy, cv.fold=5, scale="log", step=0.5, mtry=function(p) max(1, floor(sqrt(p))), recursive=FALSE, ...)

Concluding remarks – Ensemble methods

- This lecture has only provided a very basic introduction to classification using ensemble methods.

- In R, these models will work ‘out of the box’ but it is expected that you will experiment with each method in applied sessions to get a better feel of the factors and parameter settings that affect model performance.

Concluding remarks – Ensemble methods

- See: James et al., Chapter 8, and Alfaro et al., for descriptions of the algorithm for each model, and parameters that can be adjusted.
- Note that these models are being continually developed. There are, for example, several competing boosting algorithms, and Light Gradient Boosting is currently popular.

Artificial Neural Networks

In this section we'll cover:

- Artificial Neural Networks
- Artificial neurons
- Neural network architecture
- Training ANNs
- Neural Networks in R

Artificial Neural Networks

https://brainsciences.org

Artificial Neural Networks, ANNs

- Are computer models of neural behaviour in the (human) brain.
- Are applicable to wide range of problems
- Have the ability to 'learn' by weighting the contribution of each neuron to a decision (output).
- They are accurate, can handle redundant attributes and noisy data.
- Large ANNs give rise to 'deep learning'.

Deep learning

Go, a complex game popular in Asia, has frustrated the efforts of artificial-intelligence researchers for decades.

ARTIFICIAL INTELLIGENCE

Google masters Go

Deep-learning software excels at complex ancient board game.

https://www.nature.com/news/google-ai-algorithm-masters-ancient-game-of-go-1.19234

Deep learning

In China, Japan and South Korea, Go is hugely popular and is even played by celebrity professionals.

But the game has long interested AI researchers because of its complexity. The rules are relatively simple: the goal is to gain the most territory by placing and capturing black and white stones on a 19 × 19 grid.

But the average 150-move game contains more possible board configurations — 10^{170} — than there are atoms in the Universe, so it can't be solved by algorithms that search exhaustively for the best move. ...

Deep learning

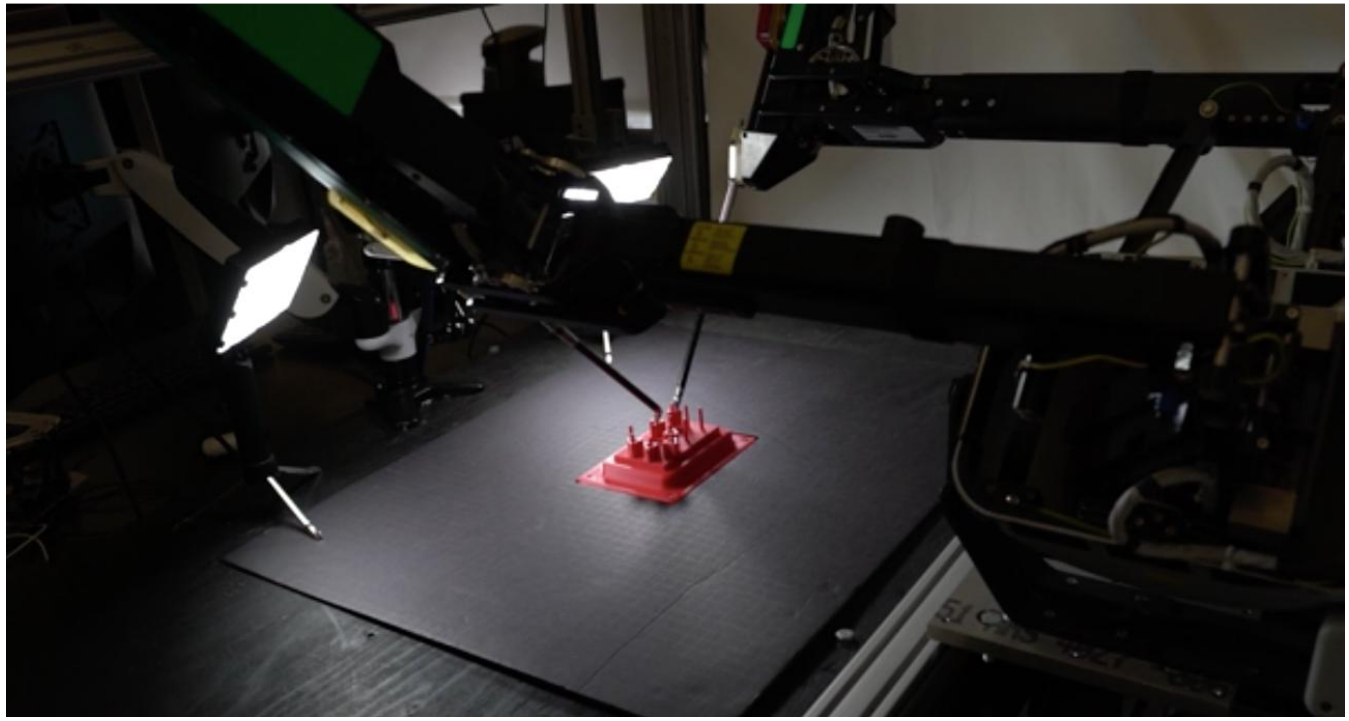
...

To interpret Go boards and to learn the best possible moves, the AlphaGo program applied deep learning in neural networks — brain-inspired programs in which connections between layers of simulated neurons are strengthened through examples and experience.

It first studied 30 million positions from expert games, gleaning abstract information on the state of play from board data, much as other programmes categorize images from pixels ...

The Robot Surgeon Will See You Now

Real scalpels, artificial intelligence — what could go wrong?



https://www.nytimes.com/2021/04/30/technology/robot-surgery-surgeon.html

The future: robotic surgery

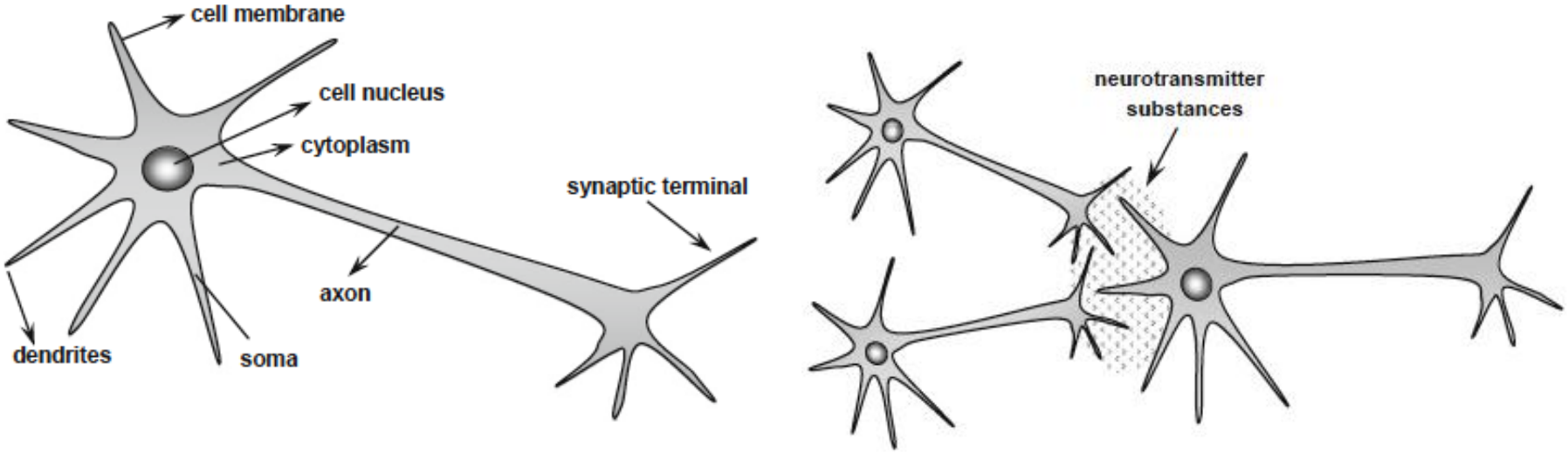
... The change is driven by what are called neural networks, mathematical systems that can learn skills by analyzing vast amounts of data. By analyzing thousands of cat photos, for instance, a neural network can learn to recognize a cat. In much the same way, a neural network can learn from images captured by surgical robots.

Surgical robots are equipped with cameras that record three-dimensional video of each operation. The video streams into a viewfinder that surgeons peer into while guiding the operation, watching from the robot's point of view.

https://www.nytimes.com/2021/04/30/technology/robot-surgery-surgeon.html

Biological Neurons

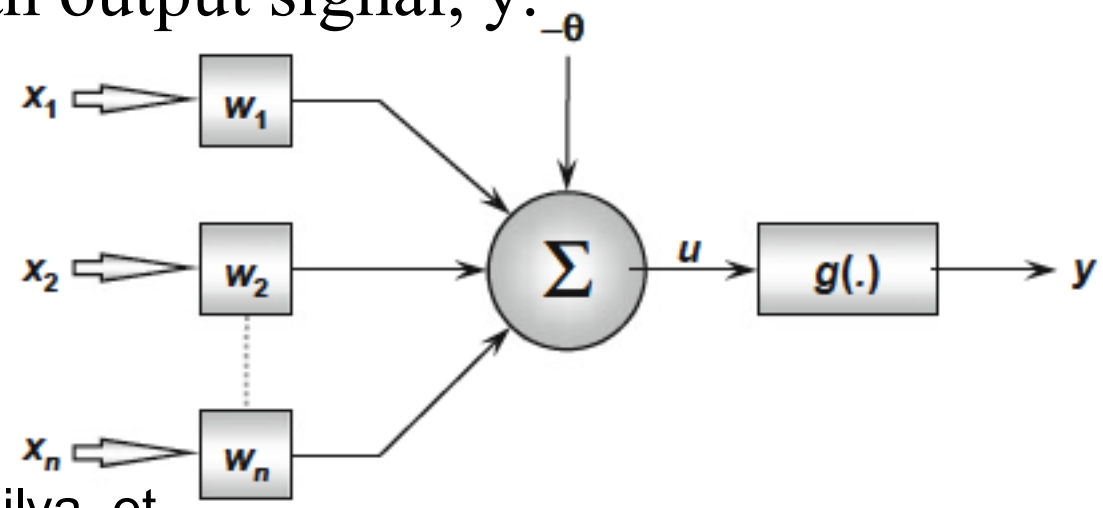
- Cell body processes information from dendrites. Producing an activation potential along axon.
- This triggers a synapse: the transfer of electric impulse from axon to dendrites of neighbouring cells.



Source: da Silva, et al.

Artificial Neurons

- Artificial neurons aggregate weighted (W_1 etc.) input signals, X_1 etc., (and activation threshold bias, -θ) to create an activation voltage.
- This is transmitted via an activation function g(.) as an output signal, y.

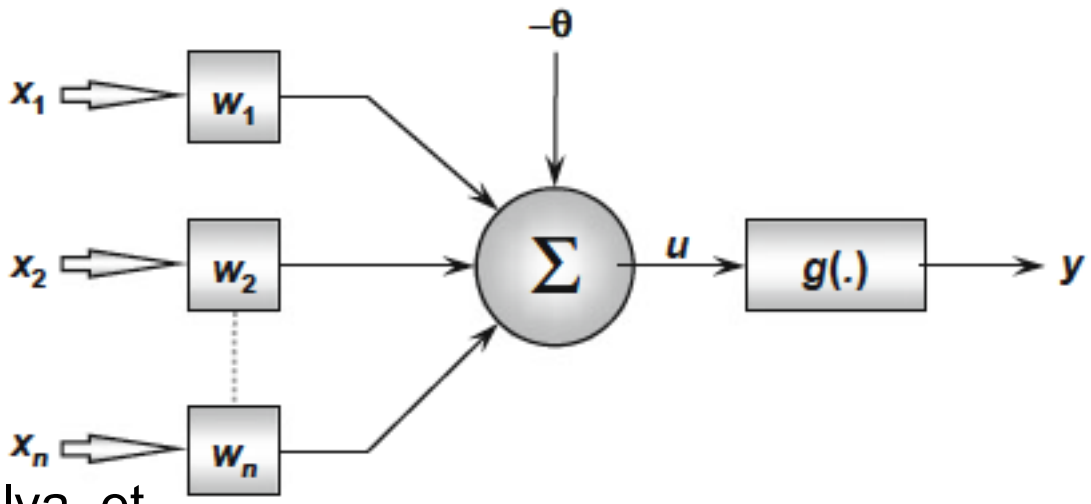


Source: da Silva, et al.

Artificial Neurons

- Under the McCullogh and Pitts model, the output by the artificial neuron can be modelled as:

$$u = \sum_{i=1}^{n} w_i \cdot x_i - \theta \qquad y = g(u)$$



Source: da Silva, et al.

Artificial Neurons

The operation of the artificial neuron can be summarised by the following steps:

- Input is via a set of variables X_1, ... etc.
- These are multiplied by a synaptic weight W_1 ... etc.
- Activation potential is the weighted sum of inputs, with bias subtracted.
- An activation function interprets the activation potential and limits the output of the neuron.

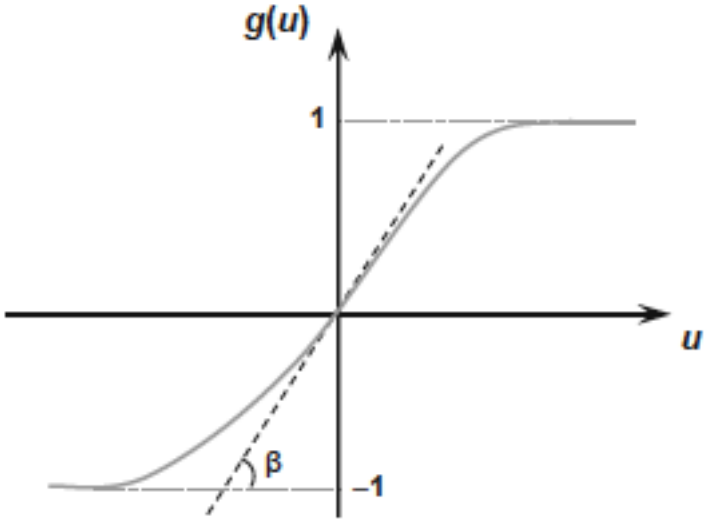
Activation functions

Partially differentiable:

- Step functions (heaviside),
- Ramp functions.

Fully differentiable (smooth)

- Logistic,
- Hyperbolic Tangent (Tanh),
- Gaussian.



Source: da Silva, et al.

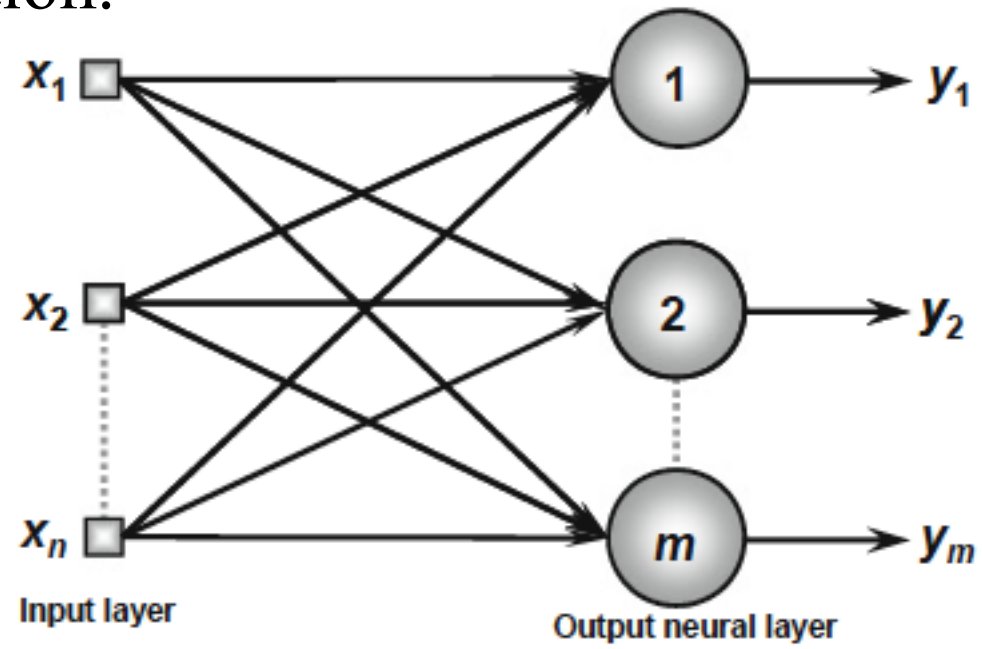
Network architectures

The structure of the ANN determines the:

- Number of inputs the model can accept
- Number of outputs produced
- The hidden layers determine the complexity of interactions that can be modelled.
- Feedback enables ‘learning’.

Single layer feedforward ANN

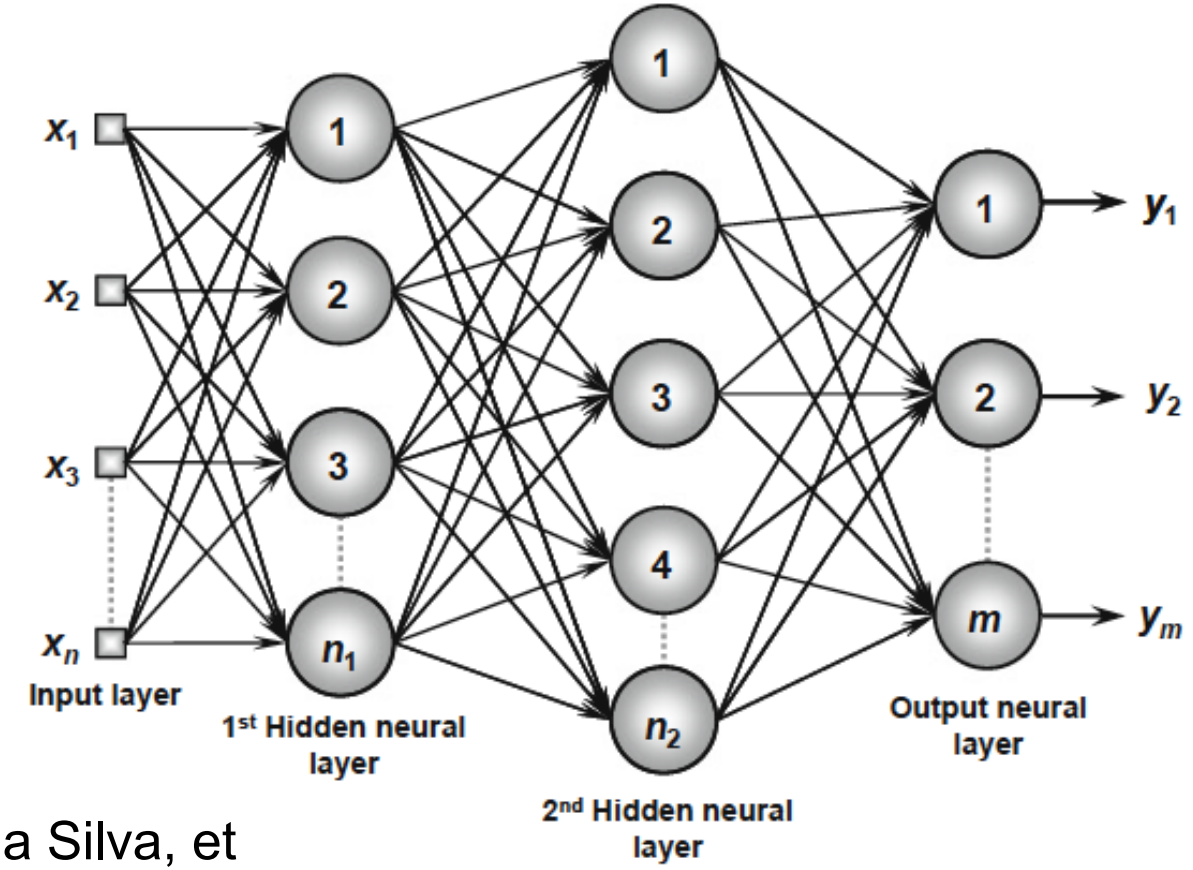
- *n* inputs, *m* outputs, information flow only in one direction.



Source: da Silva, et al.

Multiple layer feedforward ANN

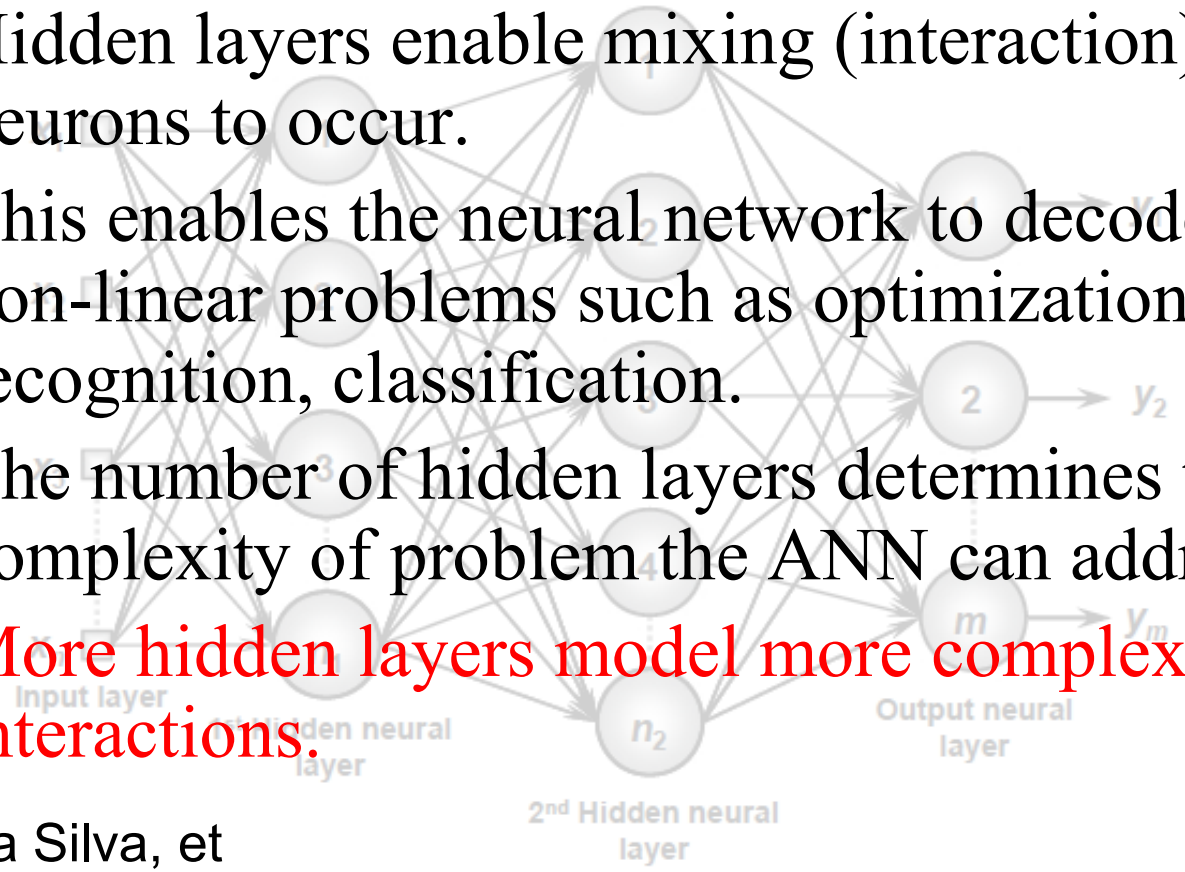
- *n* inputs, *m* outputs, two hidden layers.



Source: da Silva, et al.

Multiple layer feedforward ANN

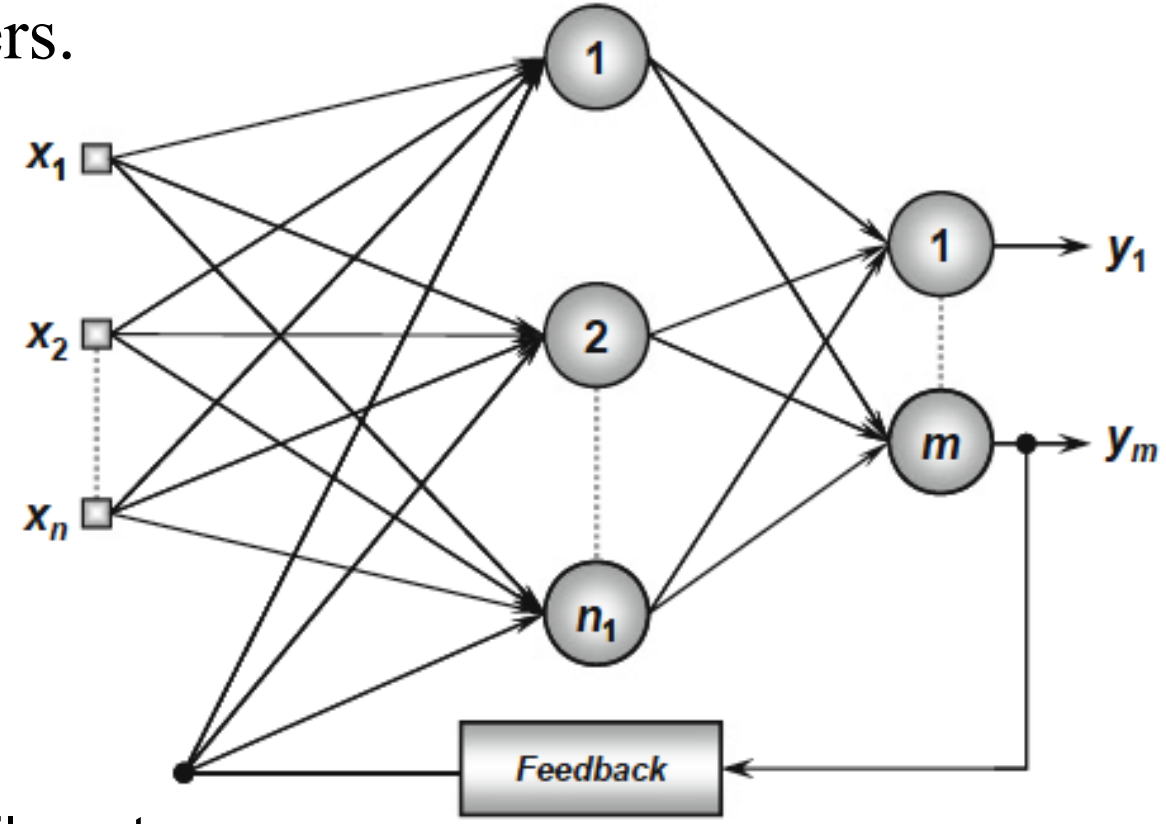
- *n* inputs, *m* outputs, two hidden layers.
- Hidden layers enable mixing (interaction) between neurons to occur.
- This enables the neural network to decode complex, non-linear problems such as optimization, pattern recognition, classification.
- The number of hidden layers determines the complexity of problem the ANN can address.
- More hidden layers model more complex interactions.



Source: da Silva, et al.

Recurrent, or feedback architecture

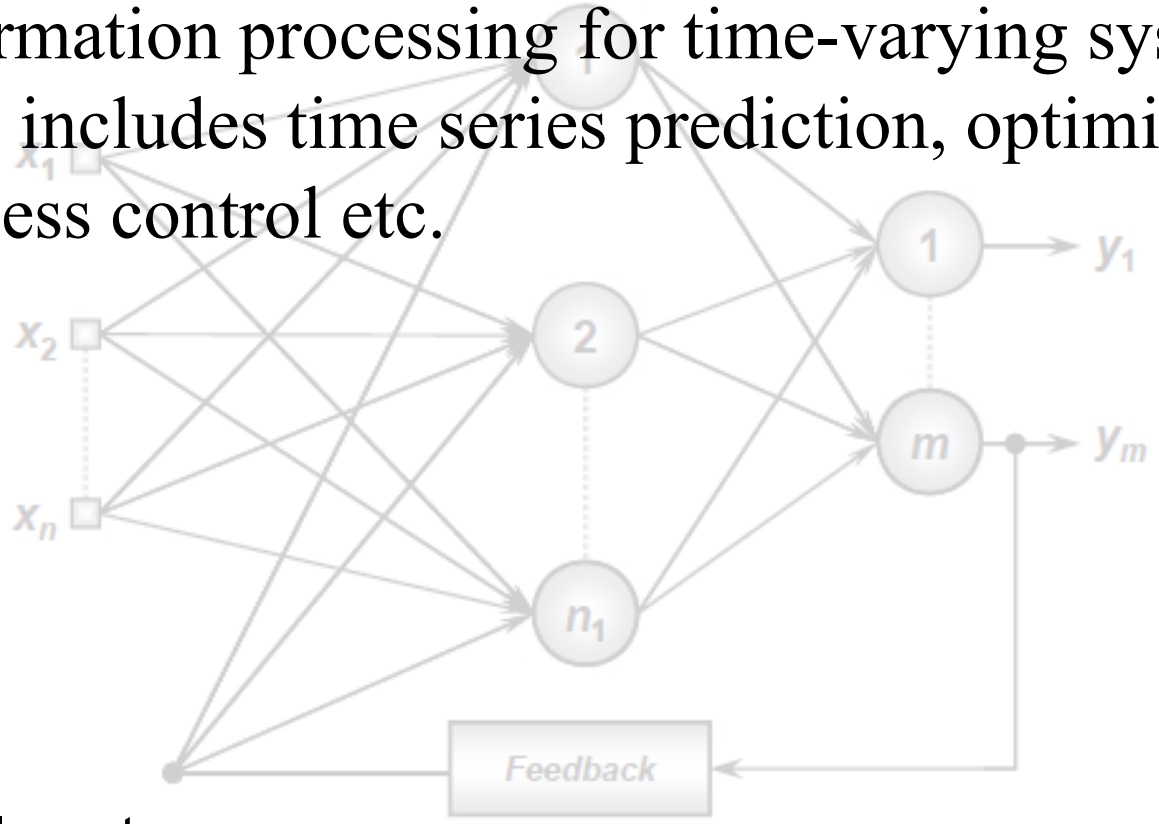
- Outputs of the neurons become inputs for earlier layers.



Source: da Silva, et al.

Recurrent, or feedback architecture

- Feedback architectures enable dynamic information processing for time-varying systems. This includes time series prediction, optimisation, process control etc.



Source: da Silva, et al.

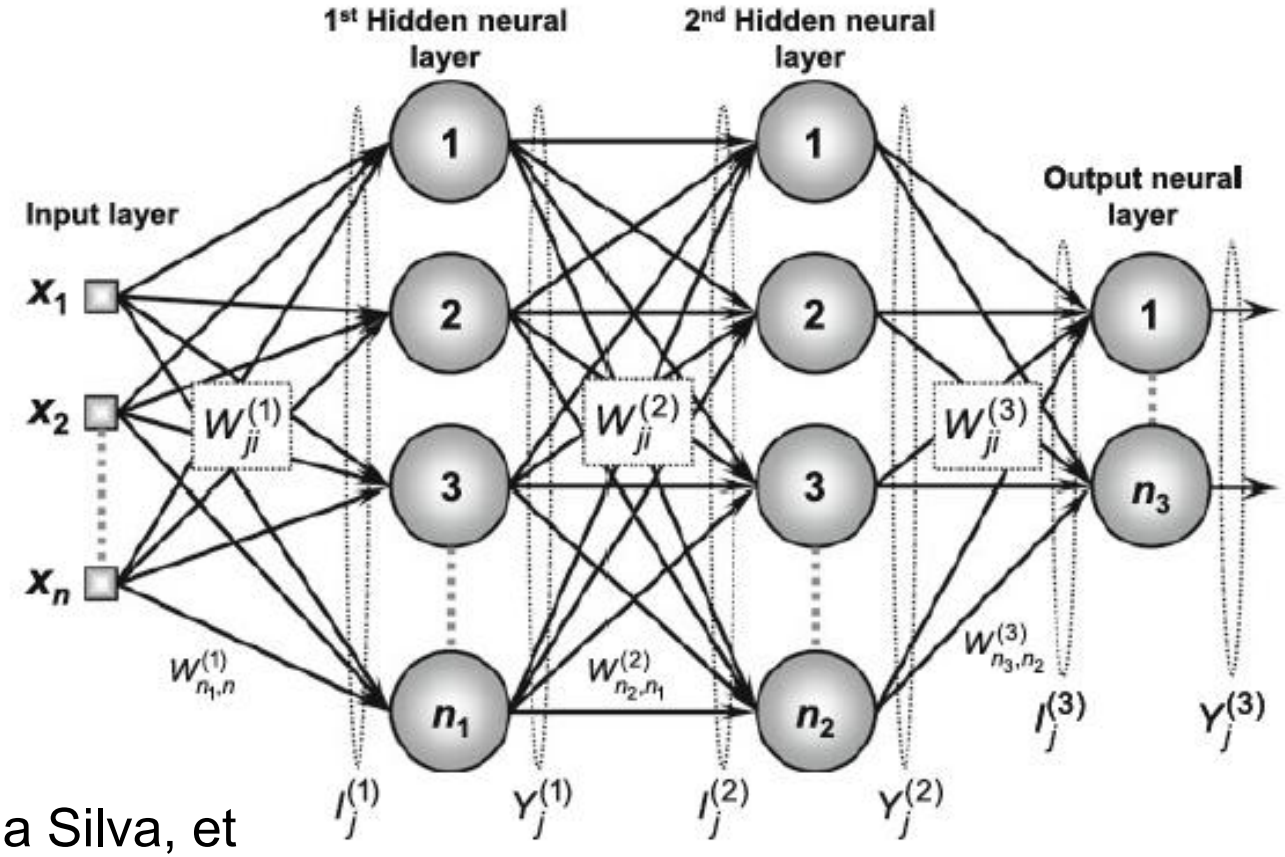
Training ANNs

Training the ANN

- Requires tuning (adjusting) the weights of each synapse and and thresholds of each neuron to produce output results close to those of the training set.
- For *supervised learning*, the procedure is an iterative optimisation to reduce the error between known and predicted outputs.
- For *unsupervised learning* the optimisation is more to produce clusters of similar subsets of the data.

Training ANNs

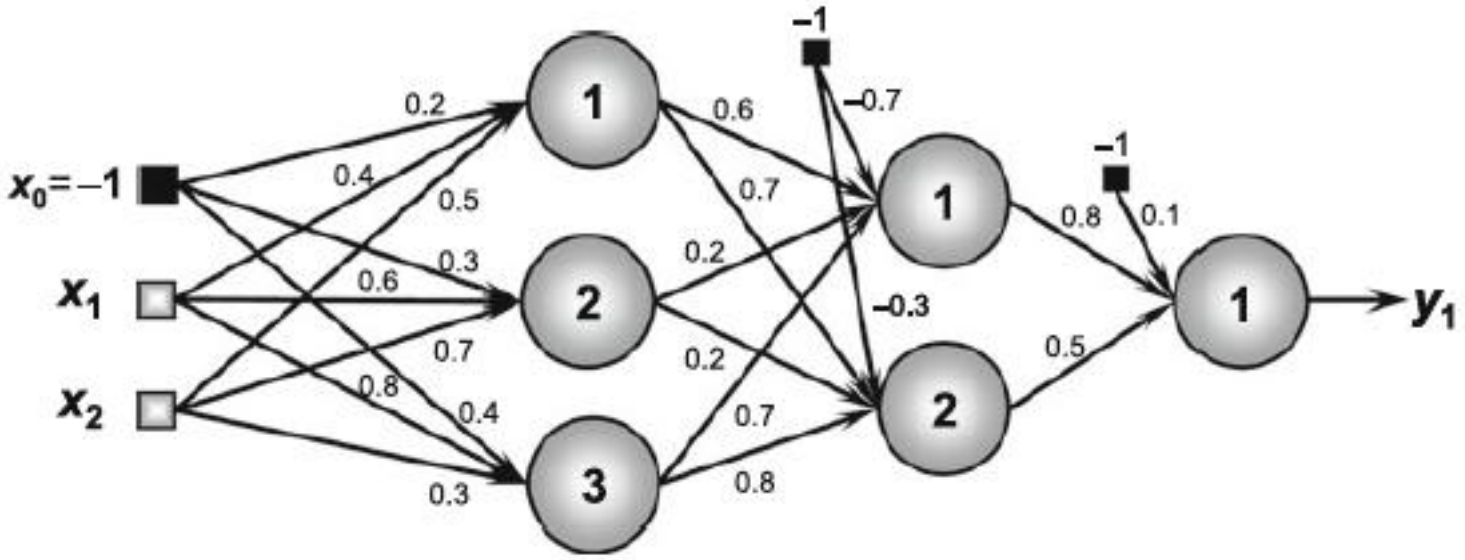
- Multi-layer ANN showing weights at each synapse.



Source: da Silva, et al.

Training ANNs

- Example of a 'trained' ANN showing weights at each synapse and evaluation of input ($X_0 = -1$)



Source: da Silva, et al.

Setting up ANNs

Pre-processing

- One input neuron for each input variable,
- One output neuron for each output class,
- Inputs can only be numerical (R will accept binary TRUE, FALSE),
- Data should be normalised,
- Categorical data needs to be converted to binary columns as indicator variables (one hot encoding),
- No missing values.

ANNs in R

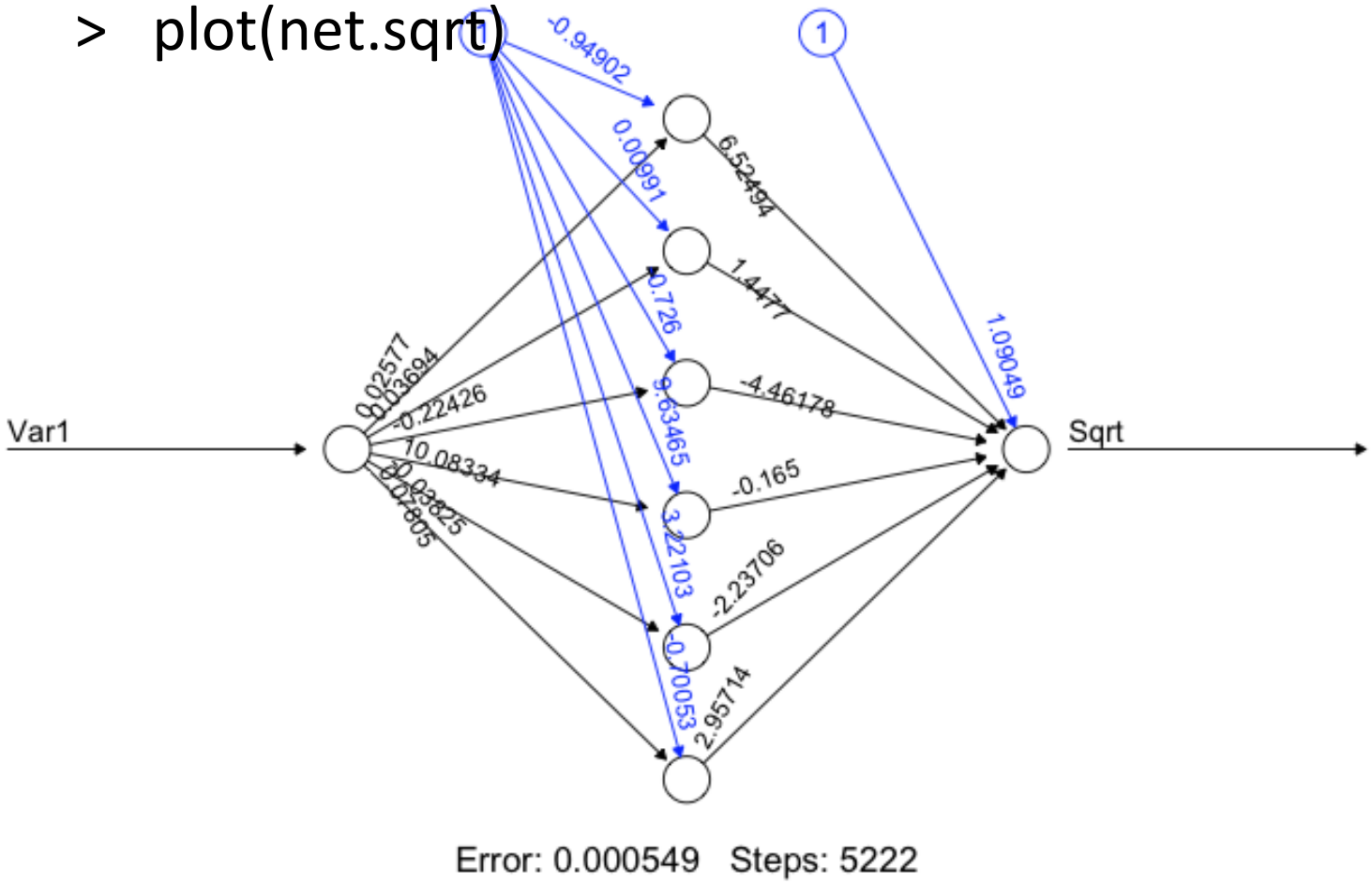
Using the “neuralnet” package. This first example adapted from package documentation.

- This fits a neural network to the square root function using one input and one output neuron.

```
> install.packages("neuralnet")
> library(neuralnet)
> Var1 <- runif(50, 0, 100) #50 rand numbers (0 – 100)
> sqrt.data <- data.frame(Var1, Sqrt=sqrt(Var1))
> net.sqrt <- neuralnet(Sqrt~Var1, sqrt.data, hidden=6,
  threshold=0.01)
```

ANNs in R

> plot(net.sqrt)



ANNs in R

```
> # To test model, first make a list of first 10 squares
> squareslist = c(1:10)
> squareslist = squareslist^2
> squareslist = as.data.frame(squareslist)
> # now compute the square roots using neural net
> compute(net.sqrt, squareslist)
```

ANNs in R

```
> compute(net.sqrt, squareslist)
             [,1]
 [1,] 1.126534
 [2,] 1.990702
 [3,] 3.013558
 [4,] 3.985291
 [5,] 5.000434
 [6,] 6.005939
 [7,] 6.997956
 [8,] 7.997764
 [9,] 9.007573
[10,] 9.948745
```

ANNs in R

To predict multiple (n>2) classes, multiple output nodes are required.

Classifying the Iris data.

- 3 output classes requires 3 output nodes

```
> # adapted from https://www.packtpub.com/books/content/training-and-visualizing-neural-network-r
> library(neuralnet)
```

ANNs in R

- Create training and test sets
  - > set.seed(9999)
  - > ind <- sample(2, nrow(iris), replace = TRUE, prob=c(0.8, 0.2))
  - > iris.train = iris[ind == 1,]; iris.test = iris[!ind == 1,]
  - > # make indicators
  - > iris.train$setosa = iris.train$Species == "setosa”
  - > iris.train$virginica = iris.train$Species == "virginica”
  - > iris.train$versicolor = iris.train$Species == "versicolor”
  - > # could make indicators using “model.matrix” function

ANNs in R

- iris.train dataframe showing response indicators. These replace the "Species" column for output.

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species	setosa	virginica	versicolor
4.9	3	1.4	0.2	setosa	TRUE	FALSE	FALSE
4.6	3.1	1.5	0.2	setosa	TRUE	FALSE	FALSE
5	3.6	1.4	0.2	setosa	TRUE	FALSE	FALSE
4.6	3.4	1.4	0.3	setosa	TRUE	FALSE	FALSE
4.9	3.1	1.5	0.1	setosa	TRUE	FALSE	FALSE
4.8	3.4	1.6	0.2	setosa	TRUE	FALSE	FALSE
4.8	3	1.4	0.1	setosa	TRUE	FALSE	FALSE
...	...	...	...	...	...	...	...

ANNs in R

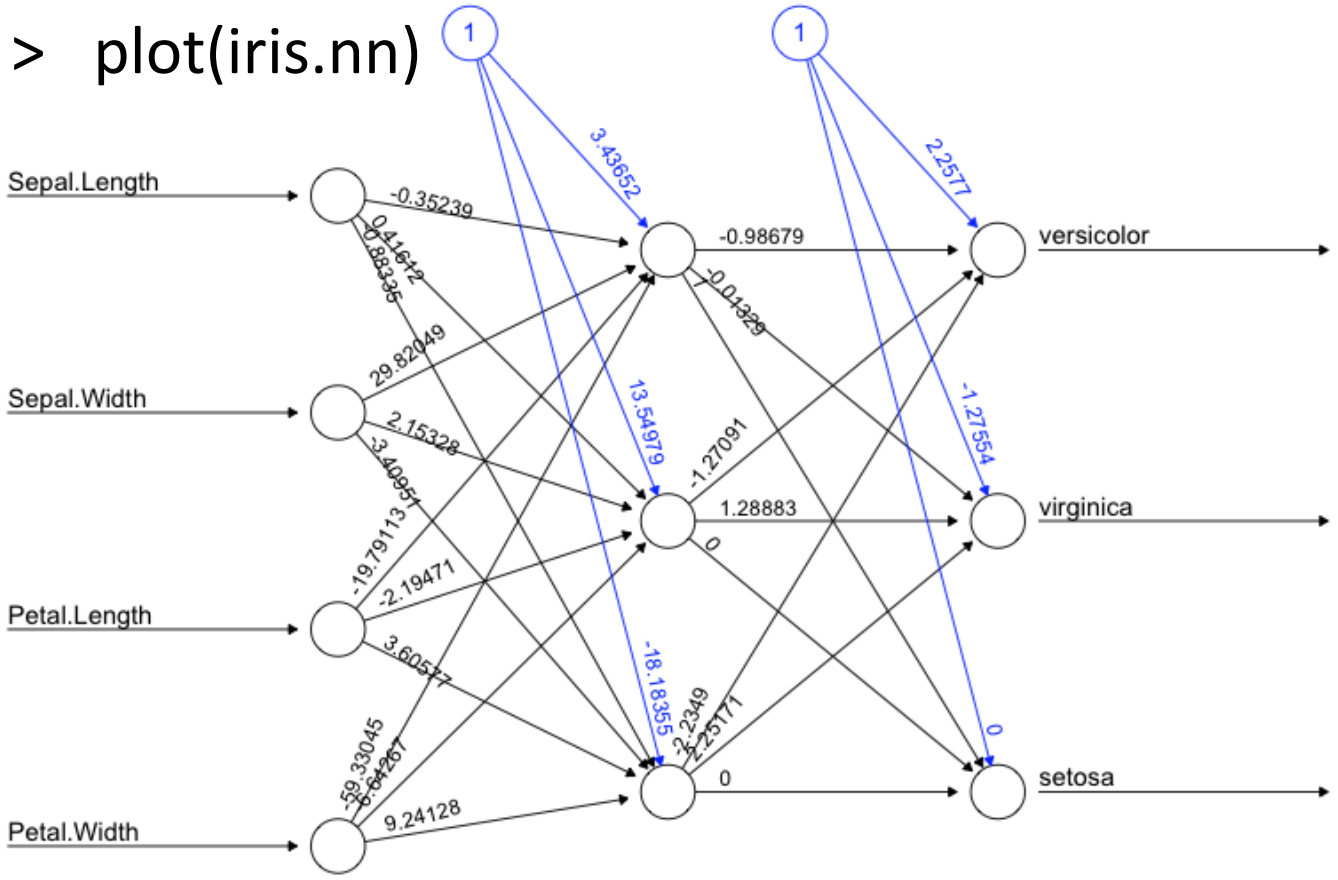
- Fit the model and have a look at the weights

```
> # fit model
> # note all terms need to be written out in the formula
> iris.nn = neuralnet(versicolor + virginica + setosa~ Sepal.Length + Sepal.Width + Petal.Length + Petal.Width, iris.train, hidden=3)
> # plot the neural network
> plot(iris.nn, rep="best")
> # print the weights
> iris.nn$result.matrix
```

All input and output columns need to be specified

ANNs in R

> plot(iris.nn)



Error: 1.892863 Steps: 29368

ANNs in R

```
> # print the weights
> iris.nn$result.matrix
                                        1
error                        1.892862505118327 2
reached.threshold            0.009791143110048 0
steps                    29368.000000000000000 0
Intercept.to.1layhid1        3.436523587527636 5
Sepal.Length.to.1layhid1    -0.352390283720568 3
Sepal.Width.to.1layhid1     29.820487806421038 4
Petal.Length.to.1layhid1   -19.791126958423113 5
Petal.Width.to.1layhid1    -59.330453022500250 9
...
```

ANNs in R

- Make predictions using test data

```
> iris.pred = compute(iris.nn, iris.test[,1:4])
> # round the predictions to 0 or 1
> iris.predr = round(iris.pred$net.result,0)
> # make data frame of A, B, C, classified 0 or 1
> iris.predrdf = as.data.frame(as.table(iris.predr))
> # remove rows classified 0 - leave only classified 1
> iris.predrdfs = iris.predrdf[!iris.predrdf$Freq==0,]
```

ANNs in R

- Raw output from the model:
  - > iris.nn$net.result

```
                 [,1]          [,2]          [,3]
1   -5.039633e-07  1.065400e-06  1.000000e+00
3   -4.969796e-07  1.058318e-06  1.000000e+00
6   -4.891154e-07  1.050343e-06  1.000000e+00
8   -4.986969e-07  1.060059e-06  1.000000e+00
9   -4.705725e-07  1.031538e-06  1.000000e+00
...
```

  - > The following steps convert the output into a usable form...

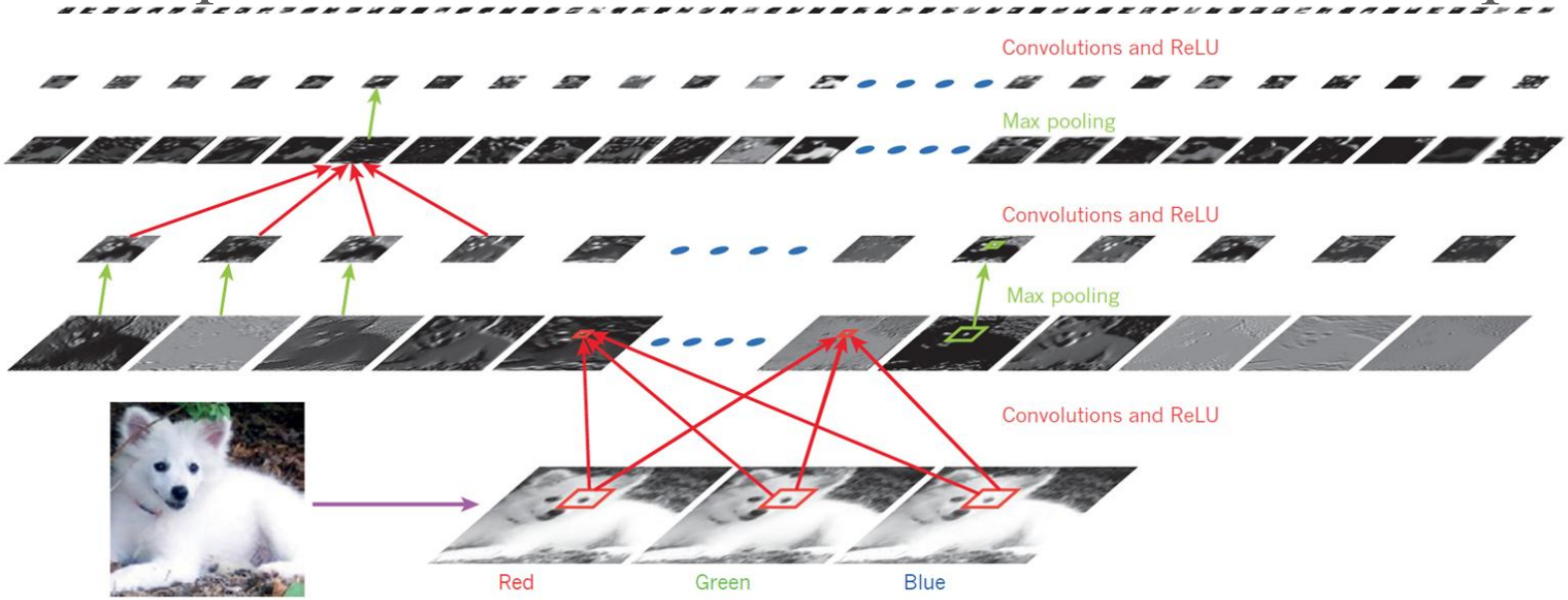
ANNs in R

- Simplify predicted values and plot confusion matrix.
  - > iris.predrdfs$Freq = NULL
  - > colnames(iris.predrdfs) = c("Obs", "Species")
  - > iris.predrdfs = iris.predrdfs[order(iris.predrdfs$Obs),]
  - > table(observed = iris.test$Species, predicted = iris.predrdfs$Species)

```
                predicted
observed        A  B  C
  setosa        0  0 16
  versicolor    8  0  0
  virginica     0 10  0
```

Further reading: Deep Learning

- This paper shows how Convolutional Neural Networks CNNs are being used for complex tasks such as image recognition…
- https://www.nature.com/articles/nature14539.pdf



Concluding thoughts

This lecture has just scratched the surface of ensemble methods and artificial neural networks.

The rest is up to you...

References: ensemble methods

- R packages: adabag, randomForest
- Package manuals. Many examples drawn from these.
- https://cran.r-project.org/web/packages/adabag/adabag.pdf
- https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
- Alfaro, Gámez and García, adabag: An R Package for Classification with Boosting and Bagging., Journal of Statistical Software, 54(2) 2013
- James et al., An Introduction to Statistical Learning with Applications in R, 2nd Ed. Springer. Chapter 8.

References: ANNs

- R package: neuralnet
- https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf
- Günther and Fritsch, neuralnet: Training of Neural Networks, The R Journal Vol. 2/1, June 2010.
- Ivan Nunes da Silva, I. N., Spatti, D. H., Flauzino, R. A., Bartocci Liboni, L. H., and dos Reis Alves, S. F. (2017) Artificial Neural Networks: A Practical Course, Springer, Switzerland. (access online via Monash Library)
- The first few chapters used to prepare today's lecture.

Notes on the presentation

This presentation contains slides created to accompany: *Introduction to Data Mining*, Tan, Steinbach, Kumar. Pearson Education Inc., 2006.

This presentation includes material originally created by Dr. Sue Bedingfield.