FIT3152 Data analytics – Lecture 9

Ensemble Methods

Artificial Neural Networks

References to this lecture

- R packages: adabag, randomForest, neuralnet
- Reference manual for each package. Many class examples drawn from these.
- Alfaro, Gámez and García, adabag: An R Package for Classification with Boosting and Bagging., Journal of Statistical Software, 54(2) 2013
- Breiman, Random Forests. Machine Learning 45, 5 – 32, 2001.
- James et al., An Introduction to Statistical Learning with Applications in R. Springer. Chapter 8.
- neuralnet: Training of Neural Networks, Günther and Fritsch, The R Journal Vol. 2/1, June 2010.
- Wikipedia pages (see links in notes).

References to this lecture

- R packages: adabag, randomForest, neuralnet
- 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
- Reference manual for each package. Many class examples drawn from these.
- https://cran.r-project.org/web/packages/adabag/adabag.pdf
- https://cran.r-project.org/web/packages/randomForest/randomForest.pdf
- https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

Week-by-week

Week Starting	Lecture	Topic	Tutorial	A1	A2
2/3/21	1	Intro to Data Science, review of basic statistics using R	...		
9/3/21	2	Exploring data using graphics in R	T1		
16/3/21	3	Data manipulation in R	T2	Released	
23/3/21	4	Data Science methodologies, dirty/clean/tidy data, data manipulation	T3		
30/3/21	5	Network analysis	T4		
6/4/21		Mid-semester Break			
13/4/21	6	Regression modelling	T5		
20/4/21	7	Classification using decision trees	T6	Submitted	
27/4/21	8	Naïve Bayes, evaluating classifiers	T7		Released
4/5/21	9	Ensemble methods, artificial neural networks	T8		
11/5/21	10	Clustering	T9		
18/5/21	11	Text analysis	T10		Submitted
25/5/21	12	Review of course, Exam preparation	T11		

SETU

Student Evaluation of Teaching and Units (SETU) has opened for Semester 1.

- All students are encouraged to participate. Your feedback is very important.
- You will see a block in Moodle linking you to the survey.
- There are 100, $50 vouchers for students to win.
- The University will email students a weekly reminder with links to the survey.

Assignment 2

The objective of this assignment is to gain familiarity with classification models using R.

- You will be using a modified version some Kaggle competition data, to predict cloud cover in Australia.
- The data contains a number of meteorological observations as attributes, and the class attribute "CloudTomorrow".
- Parts 1 – 7 will be familiar from tutorials.
- Parts 8 – 11 are a bit more challenging and will require some independent learning and initiative.

Assignment 2

FIT3152 Data analytics: Assignment 2

This assignment is worth 20% of your final marks in FIT3152.

Due: Friday 21st May 2021 11:55pm GMT+10

How to submit: Submit your written report as a single pdf with R code pasted in as machine-readable text as an appendix, or as an R Markup document that contains the R code with the discussion/text interleaved. Render this as an HTML file and print off as a pdf and submit. Whichever method you choose, you will submit a single pdf, and your R code will be machine readable text. Use the naming convention: Firstname.Lastname.studentID.pdf. Upload the file to Moodle. Do not zip. Do not submit your data file.

Objective:

The objective of this assignment is to gain familiarity with classification models using R.

You will be using a modified version of the Kaggle competition data: Predict next-day rain in Australia. https://www.kaggle.com/jsphyg/weather-dataset-rattle-package, but predicting whether or not the following day will be cloudy. The data contains a number of meteorological observations as attributes, and the class attribute "CloudTomorrow". Details of the decision attributes follow the assignment description.

You are expected to use R for your analysis, and may use any R package. Clear your workspace, set the number of significant digits to a sensible value, and use 'WAUS' as the default data frame name for the whole data set. Read your data into R using the following code:

Assignment 2

```
rm(list = ls())
WAUS <- read.csv("CloudPredict2021.csv")
L <- as.data.frame(c(1:49))
set.seed(88888888) # Your Student ID is the random seed
L <- L[sample(nrow(L), 10, replace = FALSE),] # sample 10 locations
WAUS <- WAUS[(WAUS$Location %in% L),]
WAUS <- WAUS[sample(nrow(WAUS), 2000, replace = FALSE),] # sample 2000 rows
```

We want to obtain a model that may be used to predict whether it is going to be cloudy tomorrow for 10 locations in Australia.

Assignment questions:

1. Explore the data: What is the proportion of cloudy days to clear days.? Obtain descriptions of the predictor (independent) variables – mean, standard deviations, etc. for real-valued attributes. Is there anything noteworthy in the data? Are there any attributes you need to consider omitting from your analysis? (1 Mark)

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.

Assignment 2

```
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,]
```

4. Implement a classification model using each of the following techniques. For this question you may use each of the R functions at their default settings, or with minor adjustments to set factors etc. (5 Marks)

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

5. Using the test data, classify each of the test cases as 'cloudy tomorrow' or 'not cloudy tomorrow'. Create a confusion matrix and report the accuracy of each model. (1 Mark)

6. Using the test data, calculate the confidence of predicting 'cloudy tomorrow' for each case and construct an ROC curve for each classifier. You should be able to plot all the curves on the same axis. Use a different colour for each classifier. Calculate the AUC for each classifier. (1 Mark)

7. Create a table comparing the results in Parts 5 and 6 for all classifiers. Is there a single "best" classifier? (1 Mark)

8. Examining each of the models, determine the most important variables in predicting whether or not it will rain tomorrow. Which variables could be omitted from the data with very little effect on performance? Give reasons. (2 Marks)

Assignment 2

9. Starting with one or some of the classifiers you created in Part 4, create a classifier that is simple enough for a person to be able to classify whether it will be cloudy or not tomorrow by hand. Describe your model, either with a diagram or written explanation. How well does your model perform, and how does it compare to those in Part 4? What factors were important in your decision and why you chose the attributes you used. (2 Marks)

10. Create the best tree-based classifier you can. You may do this by adjusting the parameters, and/or cross-validation of the basic models in Part 4, or using an alternative tree-based learning algorithm. Show that your model is better than the others using appropriate measures. Describe how you created your improved model, and why you chose that model. What factors were important in your decision and why you chose the attributes you used. (2 Marks)

11. Using the insights from your analysis so far, implement an Artificial Neural Network classifier and report its performance. Comment on attributes used and your data pre-processing required. How does this classifier compare with the others? Can you give any reasons? (2 Marks)

12. Write a brief report (suggested length 6 pages) summarizing your results in parts 1 – 10. 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)

End of semester exam

The end of semester exam:

- Will be online, e-vigilated. The university will advise you of the arrangements for sitting the exam.
- The exam is closed book.
- You may use a calculator: scientific, graphing, CAS are all ok.
- The practice exam has been setup as a mock exam. It is a good indicator of length/complexity. Link: https://student-eassessment.monash.edu/mod/quiz/view.php?id=3863
- Solutions will be released in Week 12.

Quick revision from last week:

Question 1

Based on this confusion matrix, TPR is:

a. $\frac{1}{3}$

b. $\frac{2}{3}$

c. $\frac{0}{3}$

d. $\frac{3}{3}$

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

Question 2

Based on this confusion matrix, FPR is:

a. $\frac{1}{3}$

b. $\frac{2}{3}$

c. $\frac{0}{3}$

d. $\frac{3}{3}$

		PREDICTED CLASS	
		Class=Yes	Class=No
ACTUAL CLASS	Class=Yes	2 (TP)	1 (FN)
	Class=No	3 (FP)	0 (TN)

Question 3

The probability $P(A_3|M)$ =

a. $\frac{2}{5}$

b. $\frac{3}{5}$

c. $\frac{2}{7}$

d. $\frac{5}{7}$

Name	A1: Gives Birth	A2: Can Fly	A3: Live in Water	A4: Has Legs	Class
bat	yes	yes	no	yes	M
cat	yes	no	no	yes	M
dolphin	yes	no	yes	no	M
human	yes	no	no	yes	M
platypus	no	no	no	yes	M
porcupine	yes	no	no	yes	M
whale	yes	no	yes	no	M

Unknown	yes	no	yes	no	?

Question 4

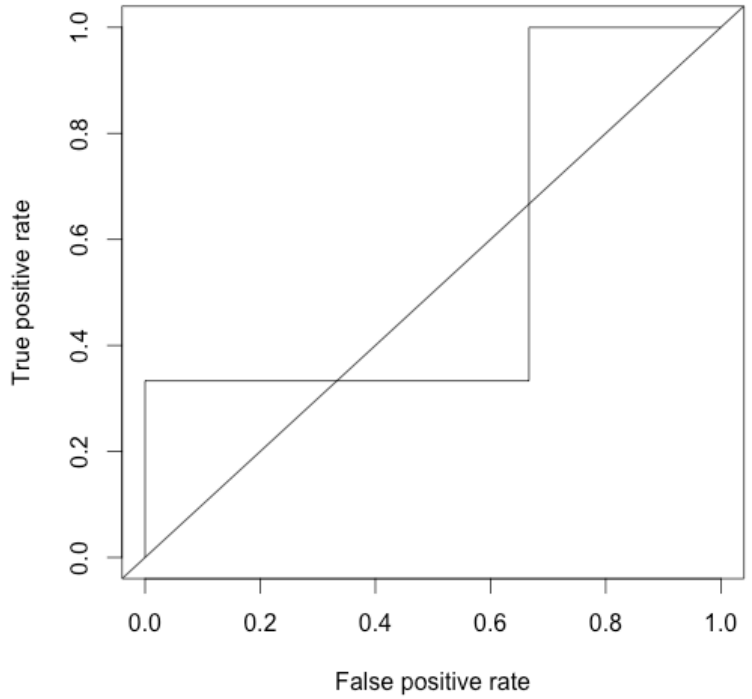
Ignoring the random guess, AUC ≈

a. 100%

b. 90%

c. 65%

d. 50%



Ensemble Methods

Definition

Why they work

Bagging

Boosting

Random Forests

R Examples

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

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*]*

This paper proposes a fine-tuned 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/

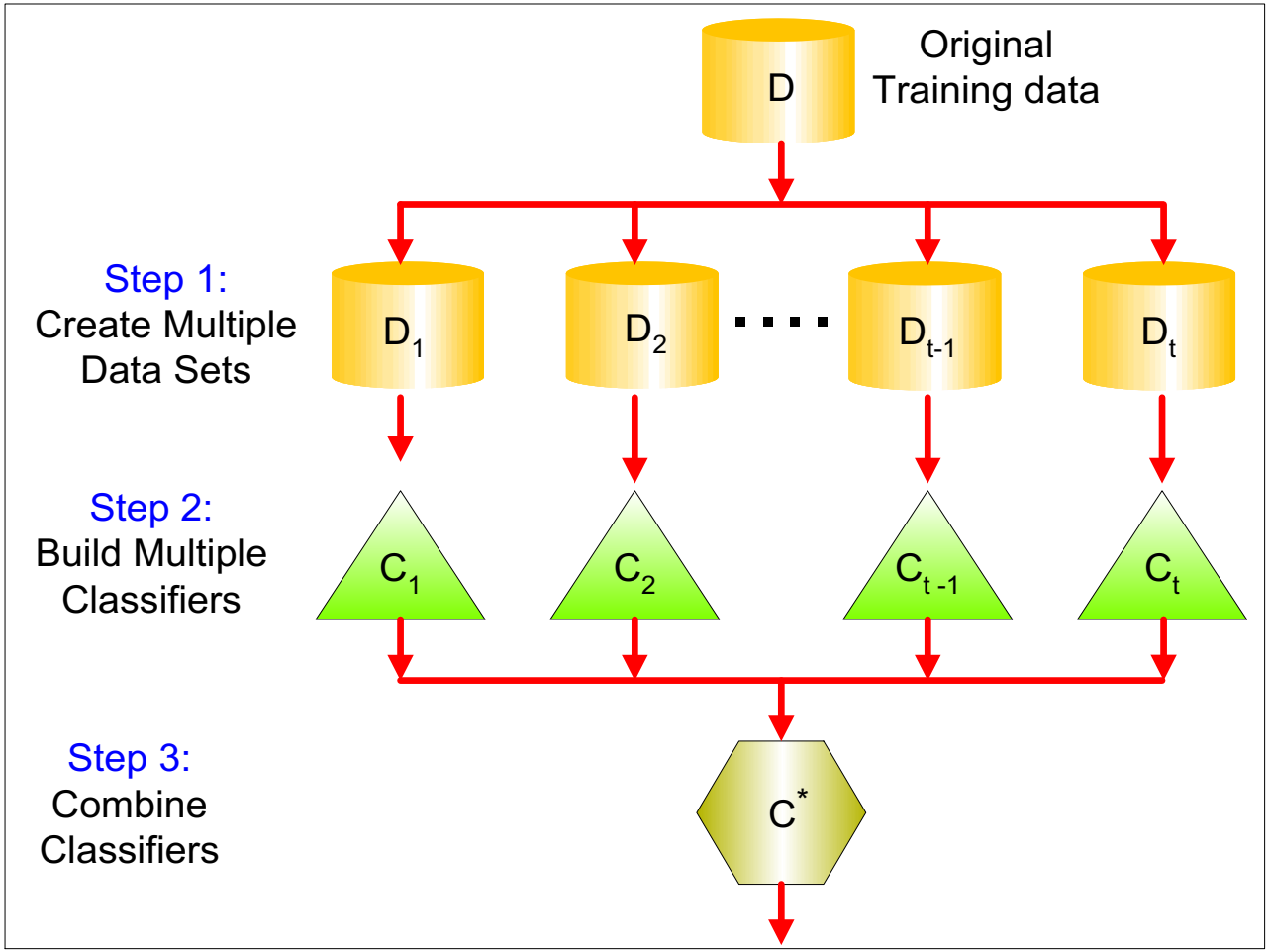
Ensemble methods for classification

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



Ensemble methods for classification

Work best when:

- The individual classifiers are moderately (> 50%) accurate.
- Individual classifiers are created independently.
- Pooling the results of the 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)
```

Testing the model

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

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

? bagging

- Description

  Fits the Bagging algorithm proposed by Breiman in 1996 using classification trees as single classifiers.

- Usage

  bagging(formula, data, mfinal = 100, control,...)

  Arguments

formula	**a formula, as in the lm function.**
data	**a data frame...**
mfinal	**an integer, the number of iterations for which boosting is run or the number of trees to use. Defaults to mfinal=100 iterations.**
control	**options that control details of the rpart algorithm...**

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

For example, 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
...
```

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.

Data set is not bootstrap sampled, but 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

- Description

  Fits the AdaBoost.M1 (Freund and Schapire, 1996) and SAMME (Zhu et al., 2009) algorithms using classification trees as single classifiers.

- Usage

```
boosting(formula, data, boos = TRUE, mfinal = 100,
coeflearn = 'Breiman',   control,...)
Arguments
formula      a formula, as in the lm function.
data         a data frame...
boos         a bootstrap sample is drawn...
Mfinal       iterations for boosting...
coeflearn    learning coefficient...
control      controls rpart...
```

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
```

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

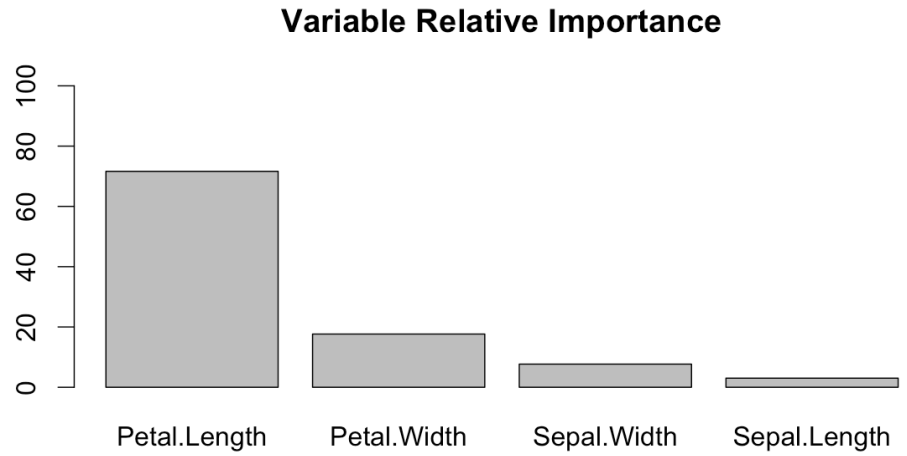
Observing the proportional contribution of each variable to information gain:

```
> 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 Forests

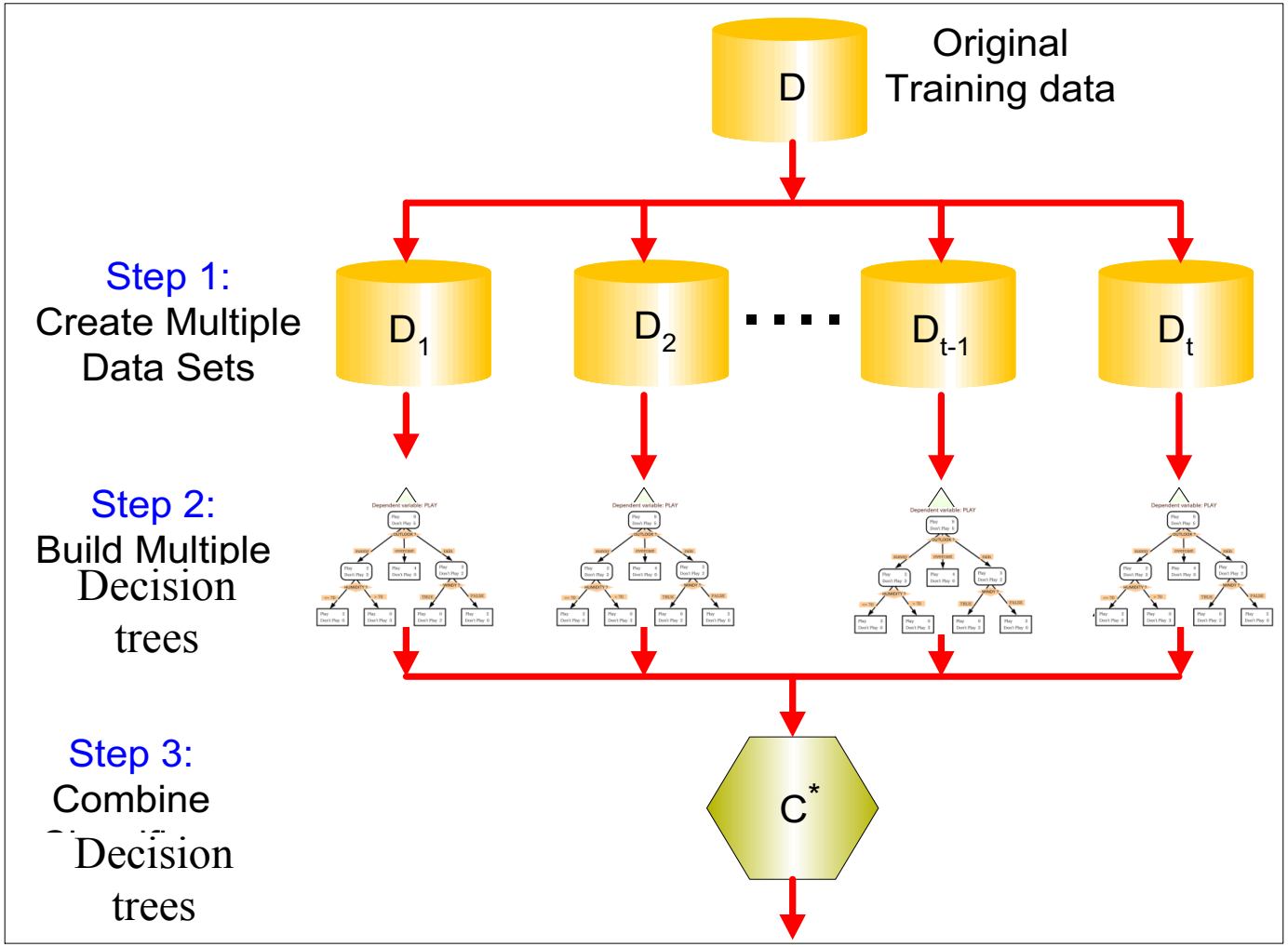
A refinement of bagged decision trees.

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.

General Idea



Advantages of Random Forests

More accurate than individual trees on large data sets

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

Difference between Bagging and Random Forests:

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

Random Forests 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 Forests 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 Forests 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
...
```

? randomForest

- Description

  randomForest implements Breiman's random forest algorithm (based on Breiman and Cutler's original Fortran code) for classification and regression...

- Usage

  randomForest(x, y=NULL, xtest=NULL, ytest=NULL, ntree=500 ...

  Too many parameters to articulate.

Random Forests in R

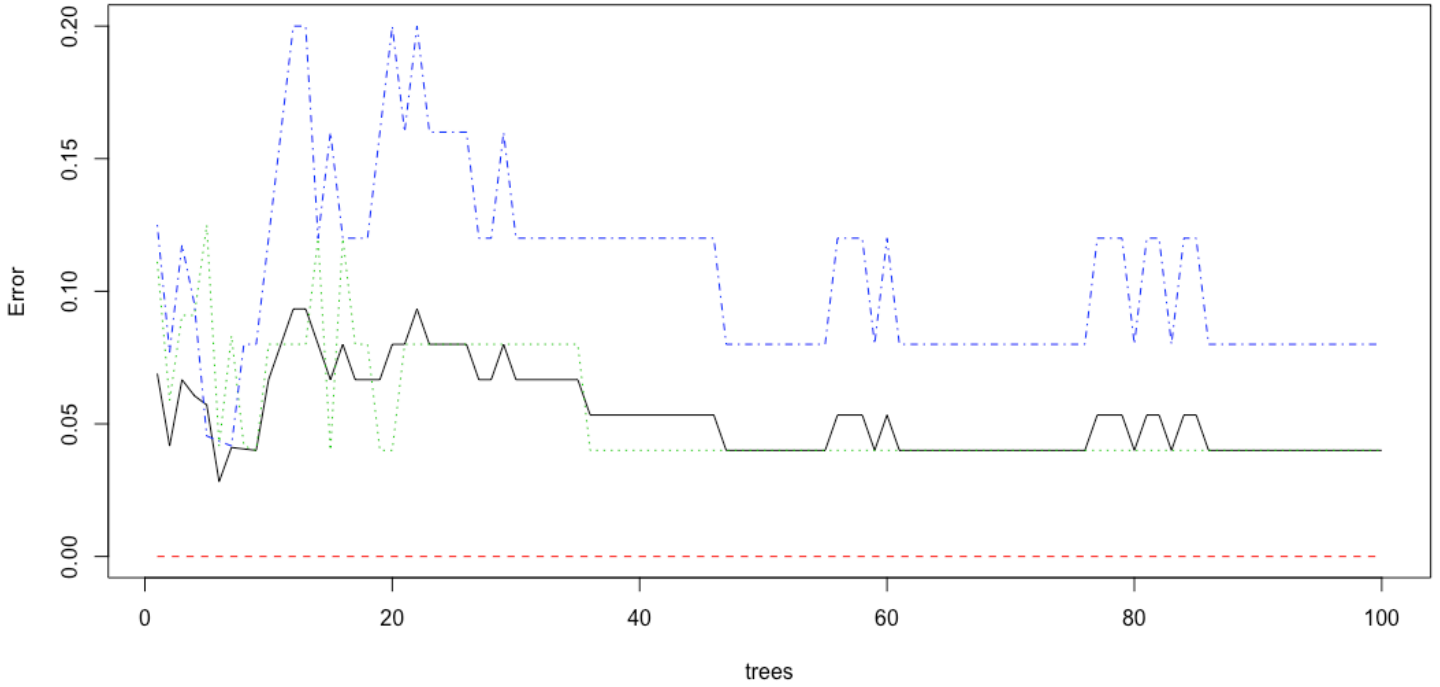
To see the elements of the model

```
> 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 Forests in R

How does number of trees affect error?

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



References: Random Forests

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, but 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

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 the tutorial to get a better feel of the factors and parameter settings that affect model performance.

Concluding remarks

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.

Gradient boosting (see package “xgboost”), is currently popular.

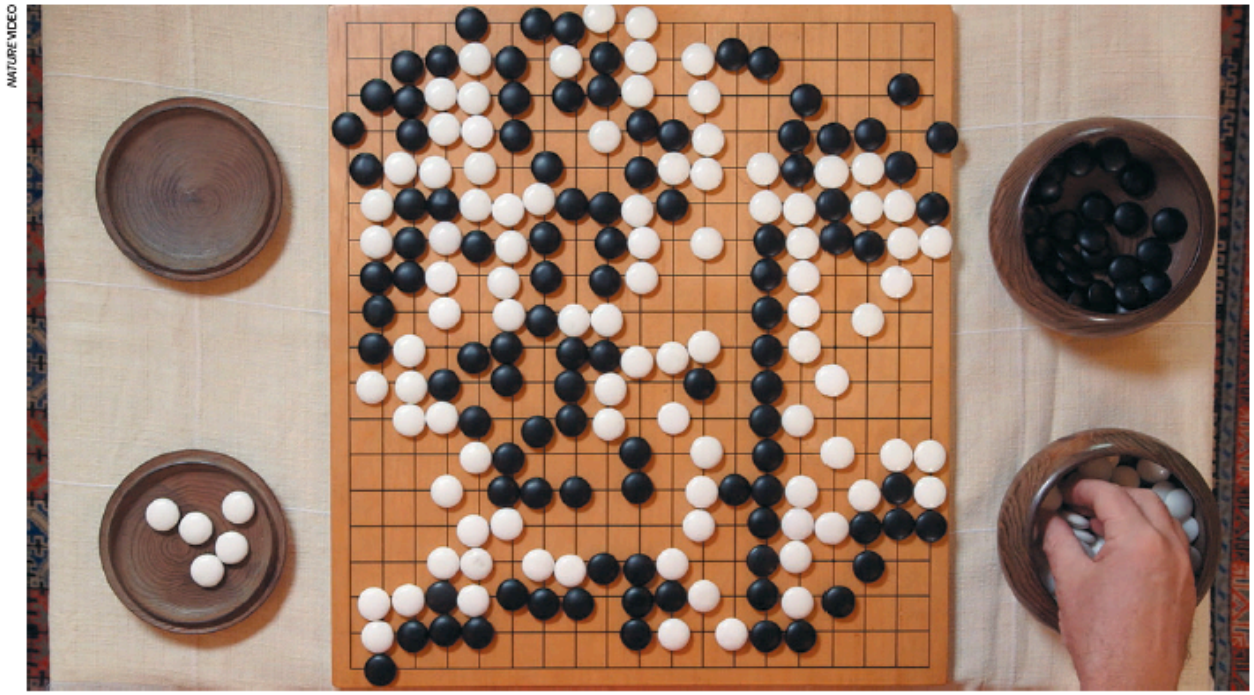
Artificial Neural Networks

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

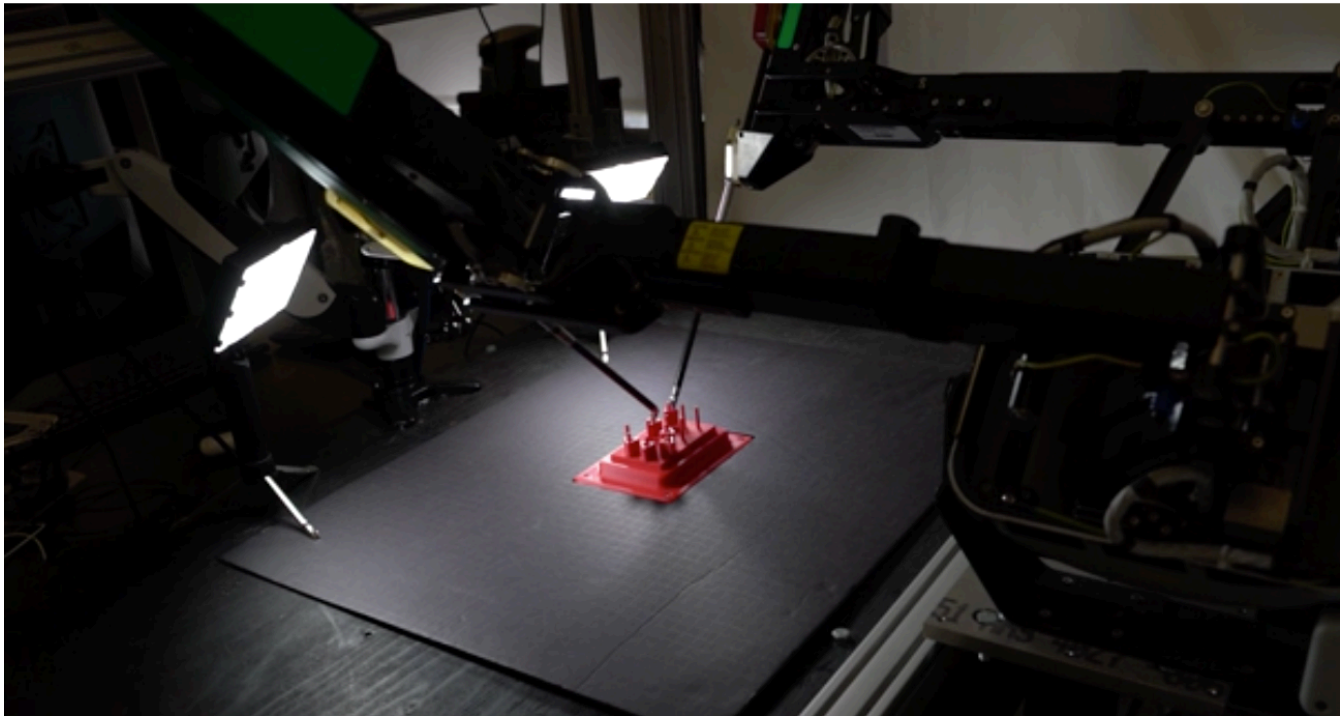
...

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

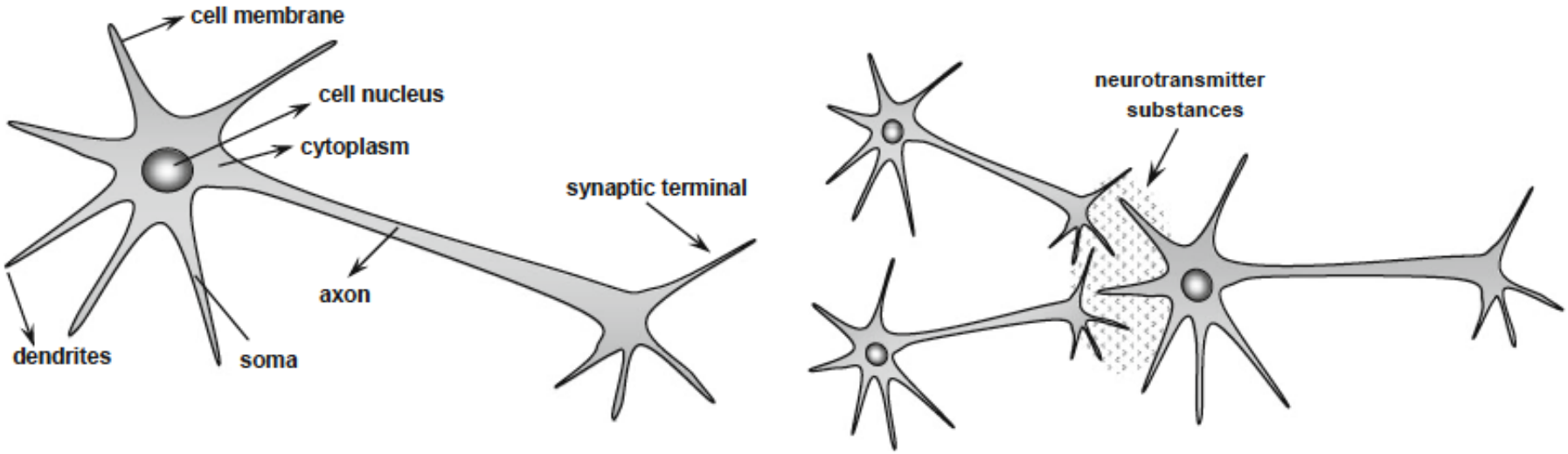
... The change is driven by what are called neural networks, mathematical systems that can learn skills by analyzing vast amounts of data. By analyzing thousands of cat photos, for instance, a neural network can learn to recognize a cat. In much the same way, a neural network can learn from images captured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

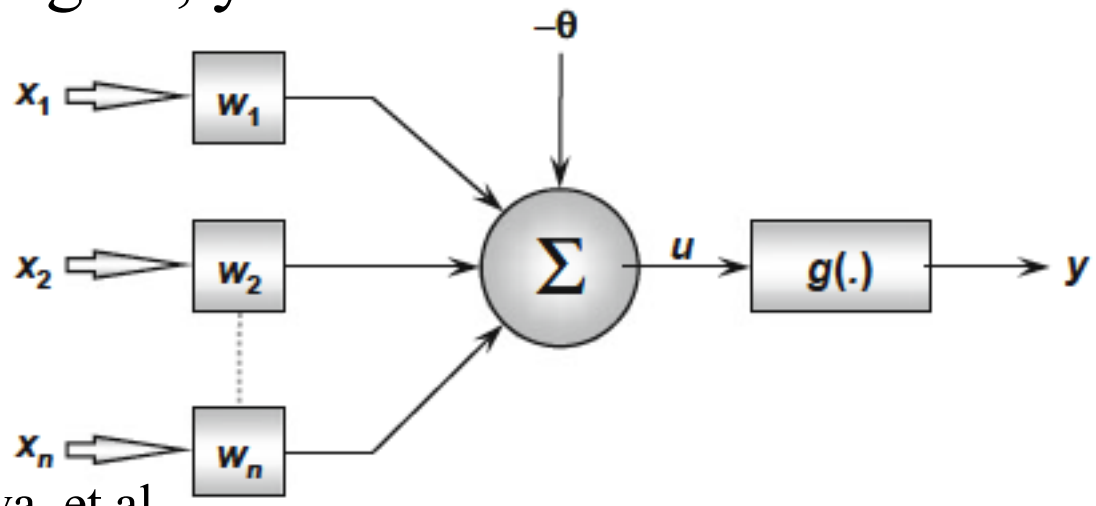
- Cell body processes information from dendrites. Producing an activation potential along axon.
- This triggers a synapse: the transfer of electric impulse from axon to dendrites of neighbouring cells.



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.

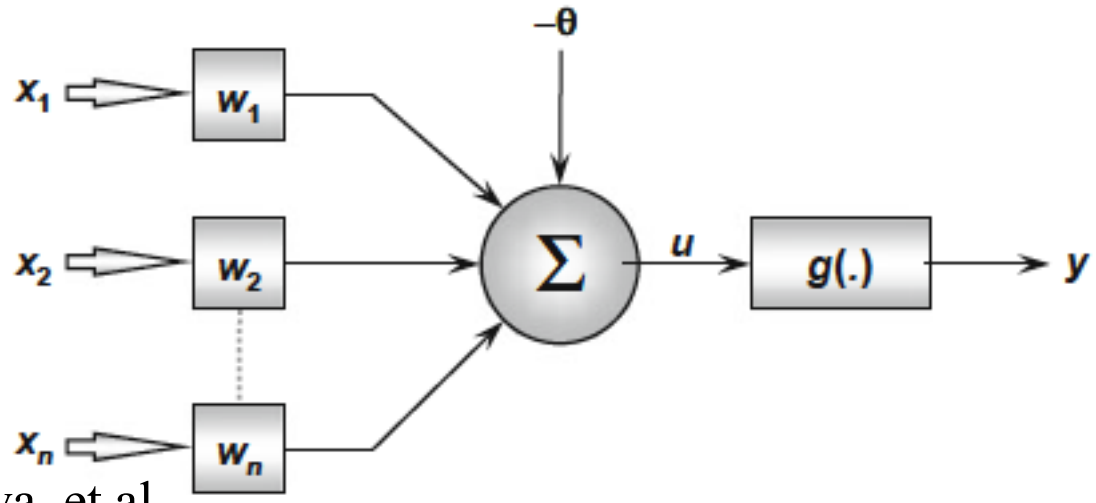


Source: da Silva, et al.

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.

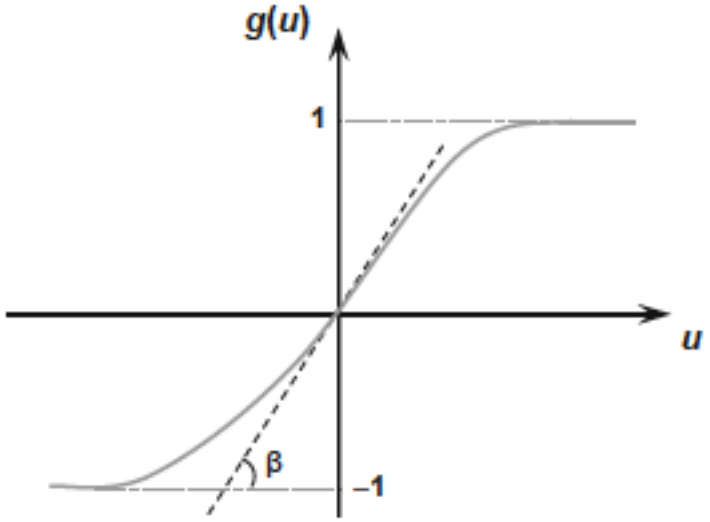
Activation functions

Partially differentiable:

- Step functions (heaviside),
- Ramp functions.

Fully differentiable (smooth):

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



Source: da Silva, et al.

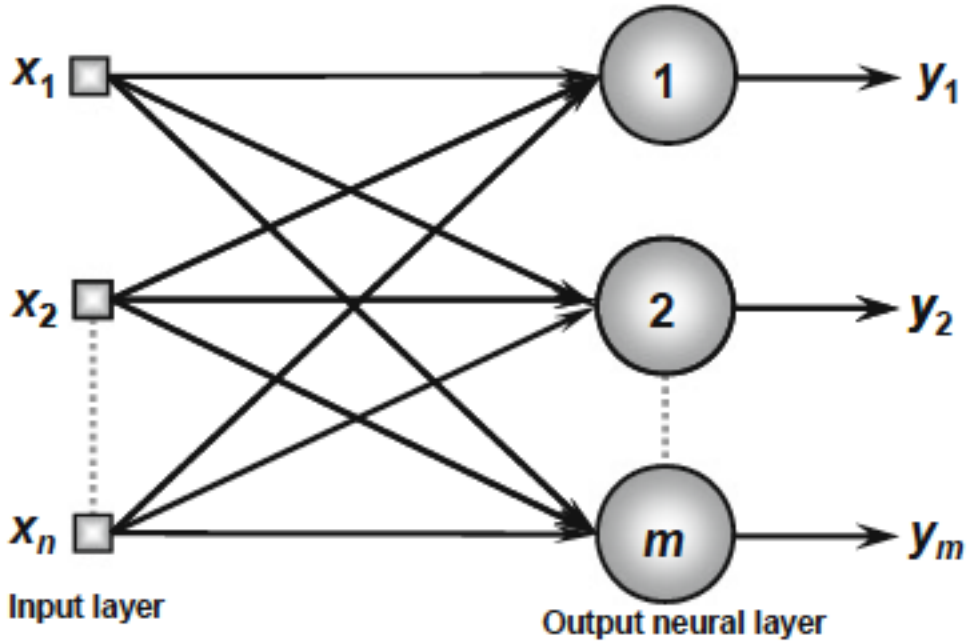
Network architectures

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

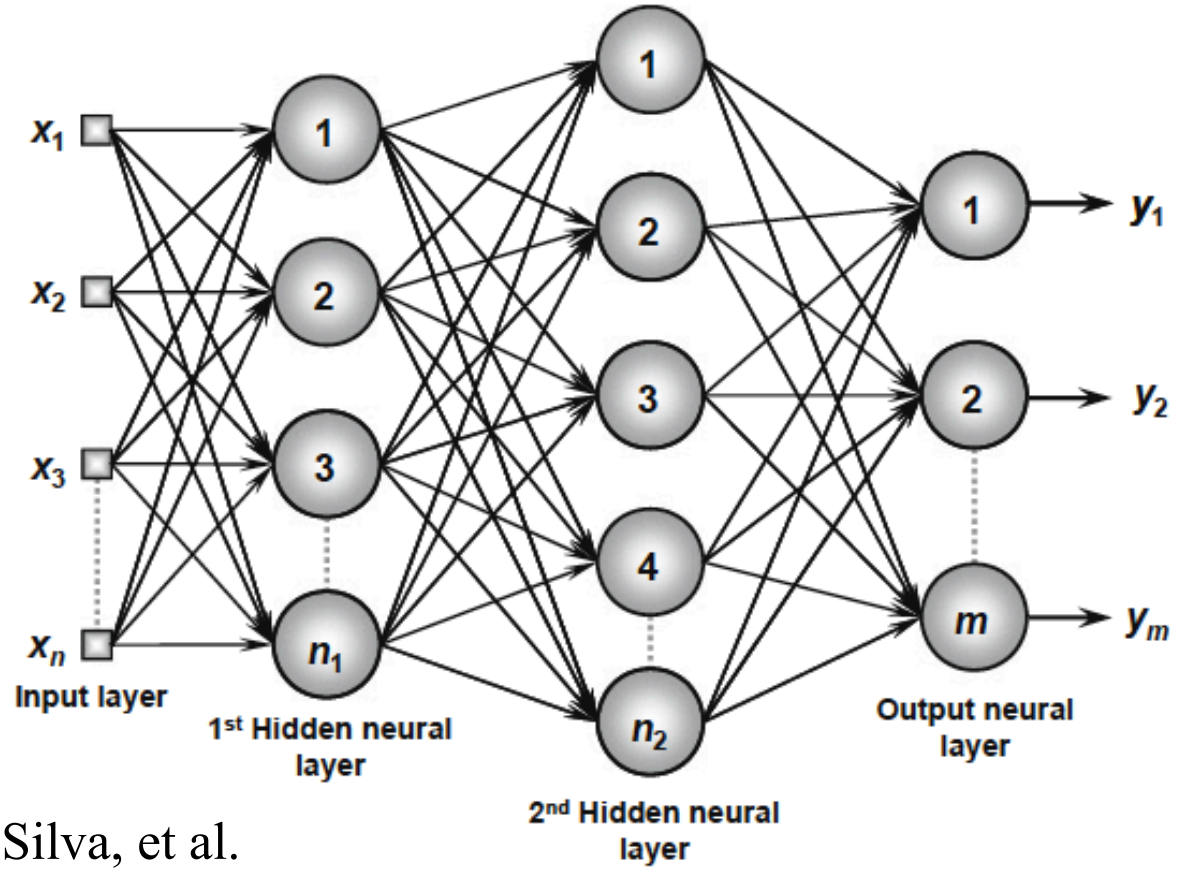
- *n* inputs, *m* outputs, information flow only in one direction.



Source: da Silva, et al.

Multiple layer feedforward ANN

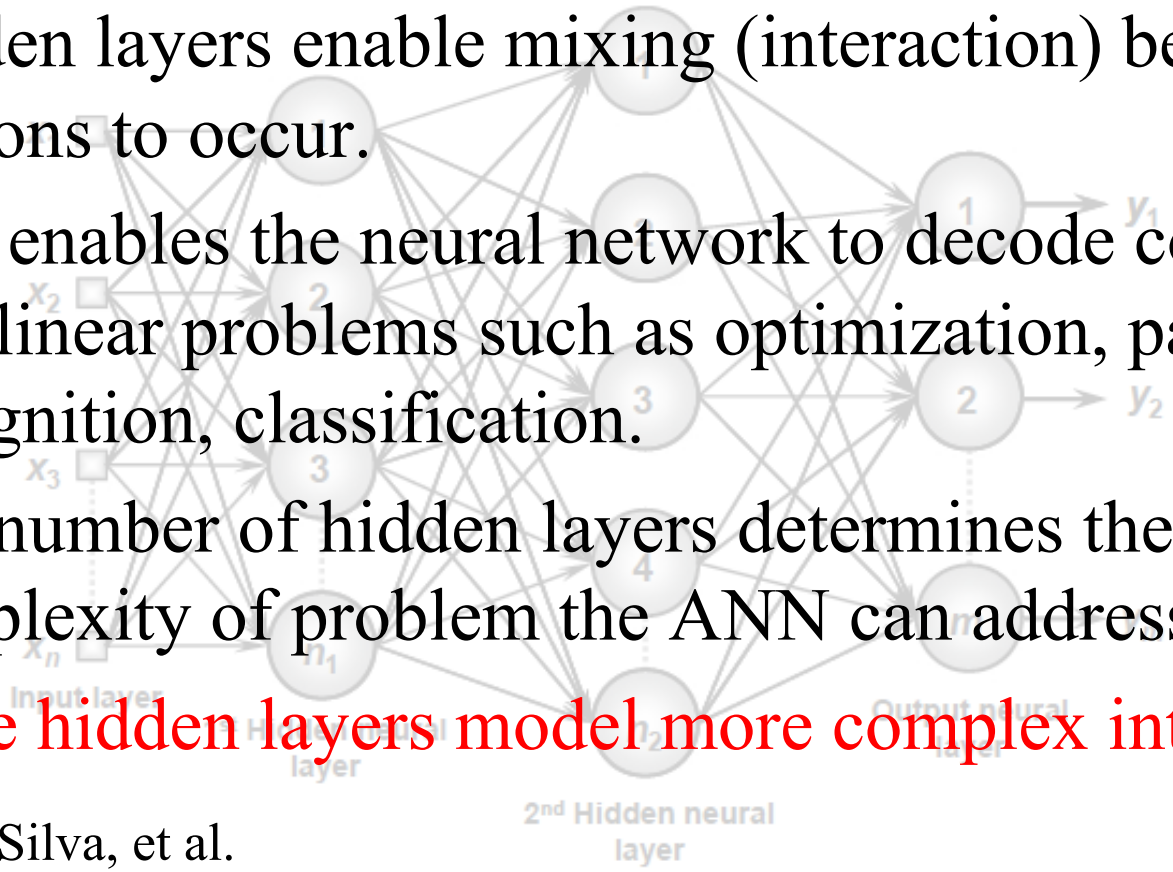
- *n* inputs, *m* outputs, two hidden layers.



Source: da Silva, et al.

Multiple layer feedforward ANN

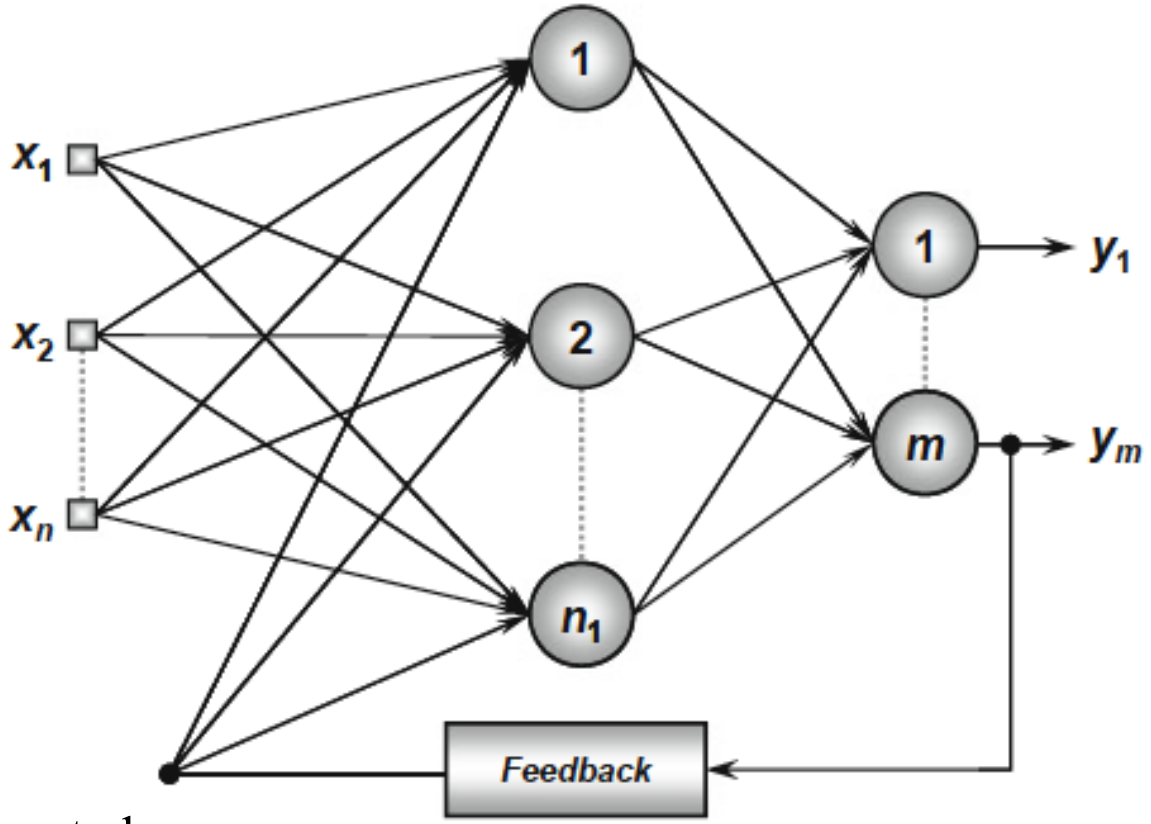
- *n* inputs, *m* outputs, two hidden layers.
- Hidden layers enable mixing (interaction) between neurons to occur.
- This enables the neural network to decode complex, non-linear problems such as optimization, pattern recognition, classification.
- The number of hidden layers determines the complexity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



Source: da Silva, et al.

Recurrent, or feedback architecture

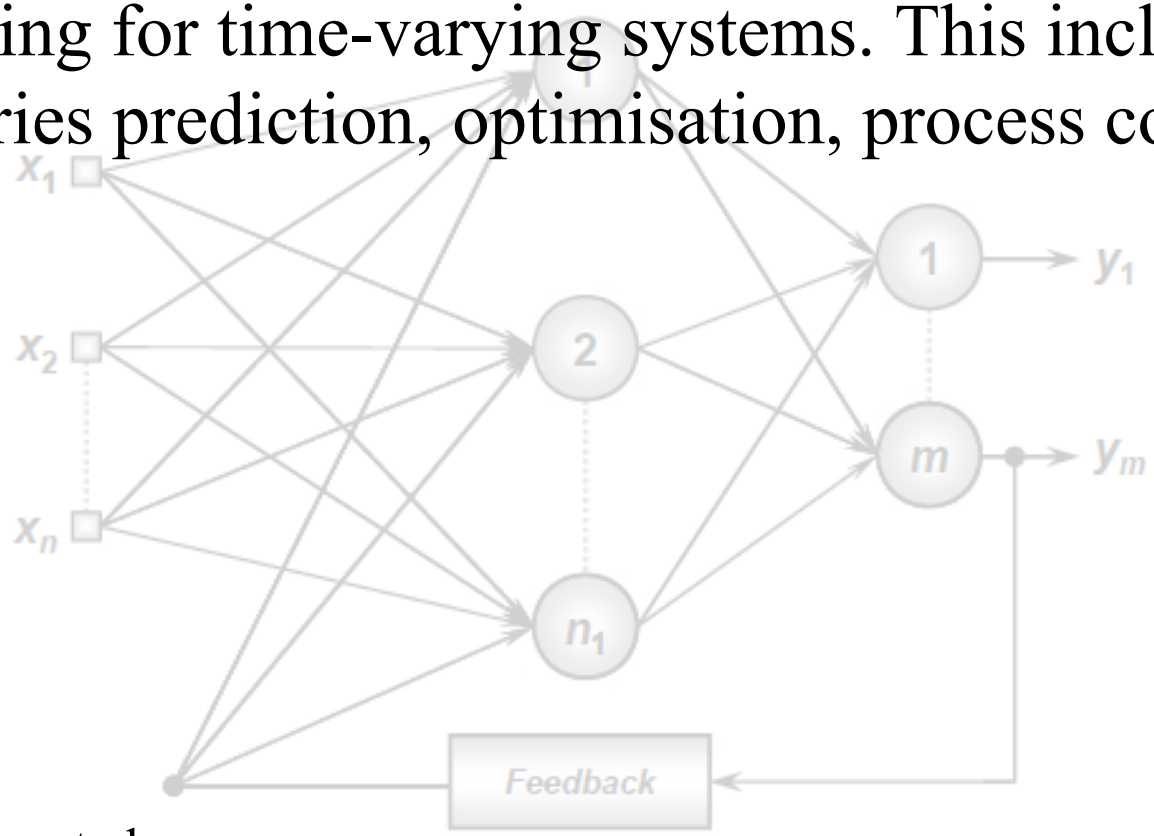
- Outputs of the neurons become inputs for earlier layers.



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.

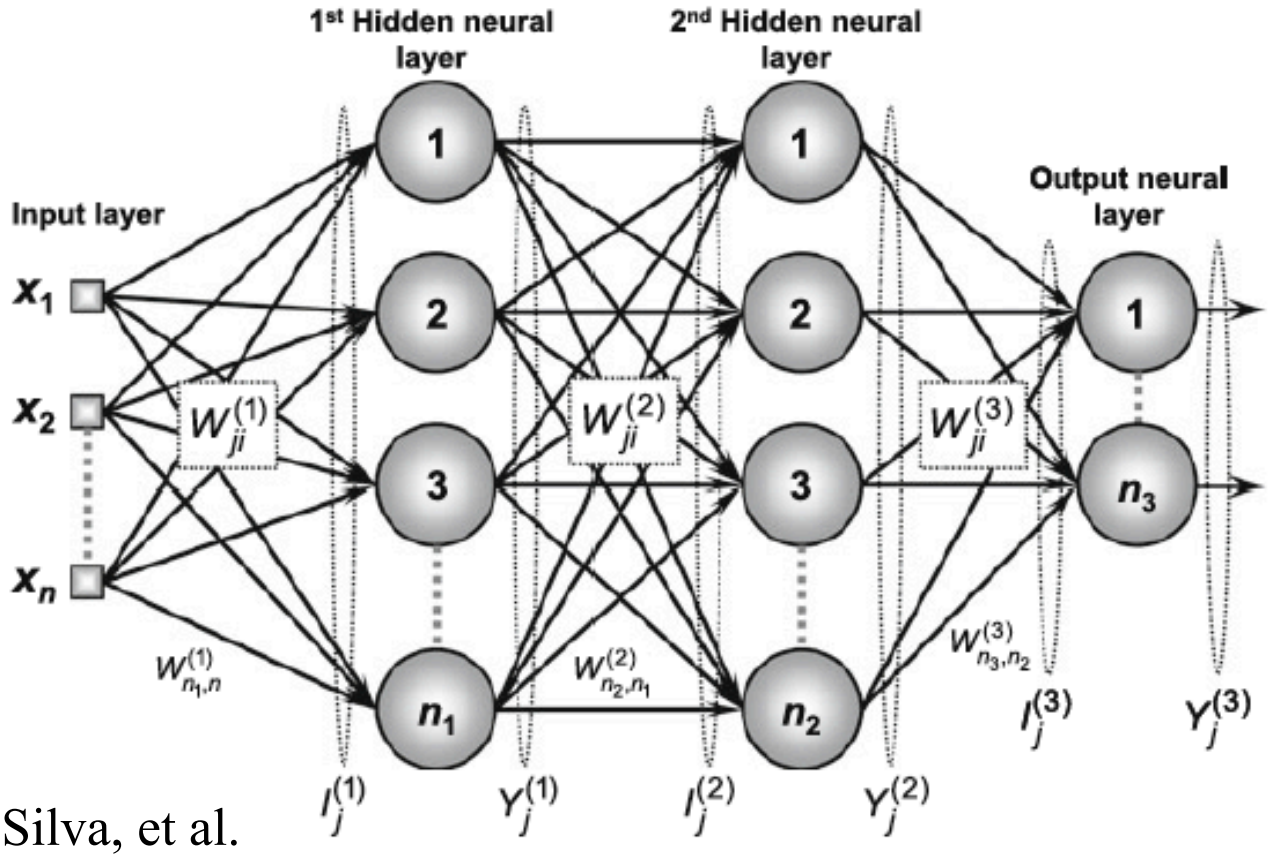
Training ANNs

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

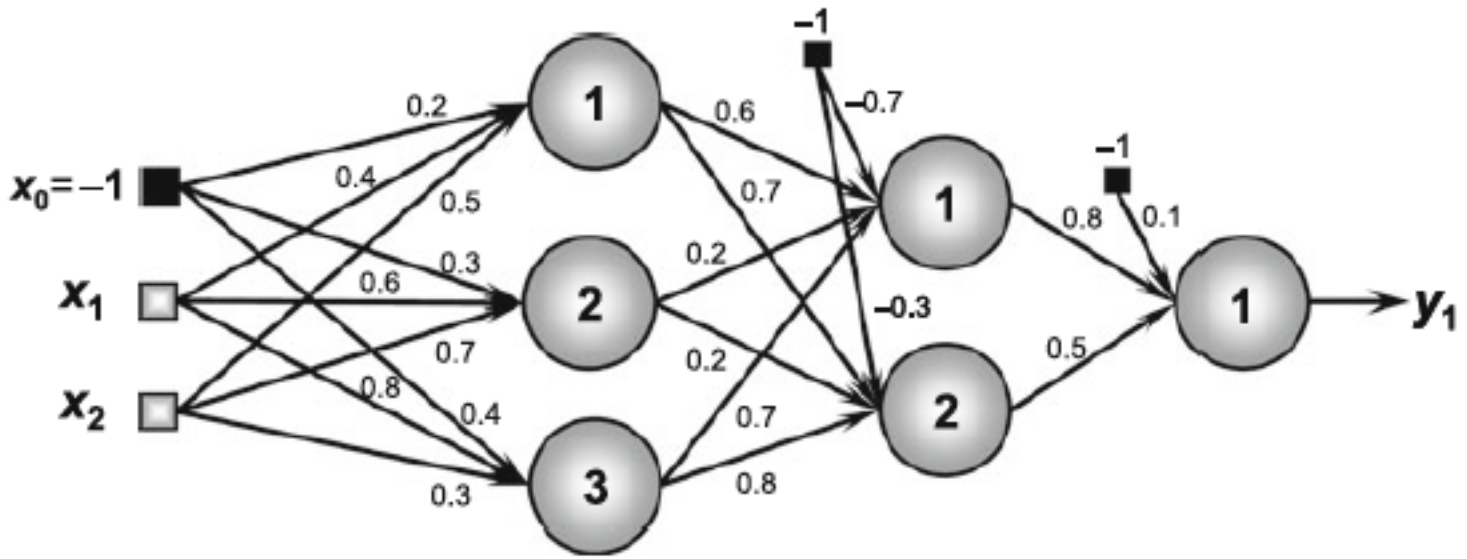
- Multi-layer ANN showing weights at each synapse.



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,
- No missing values.

ANNs in R

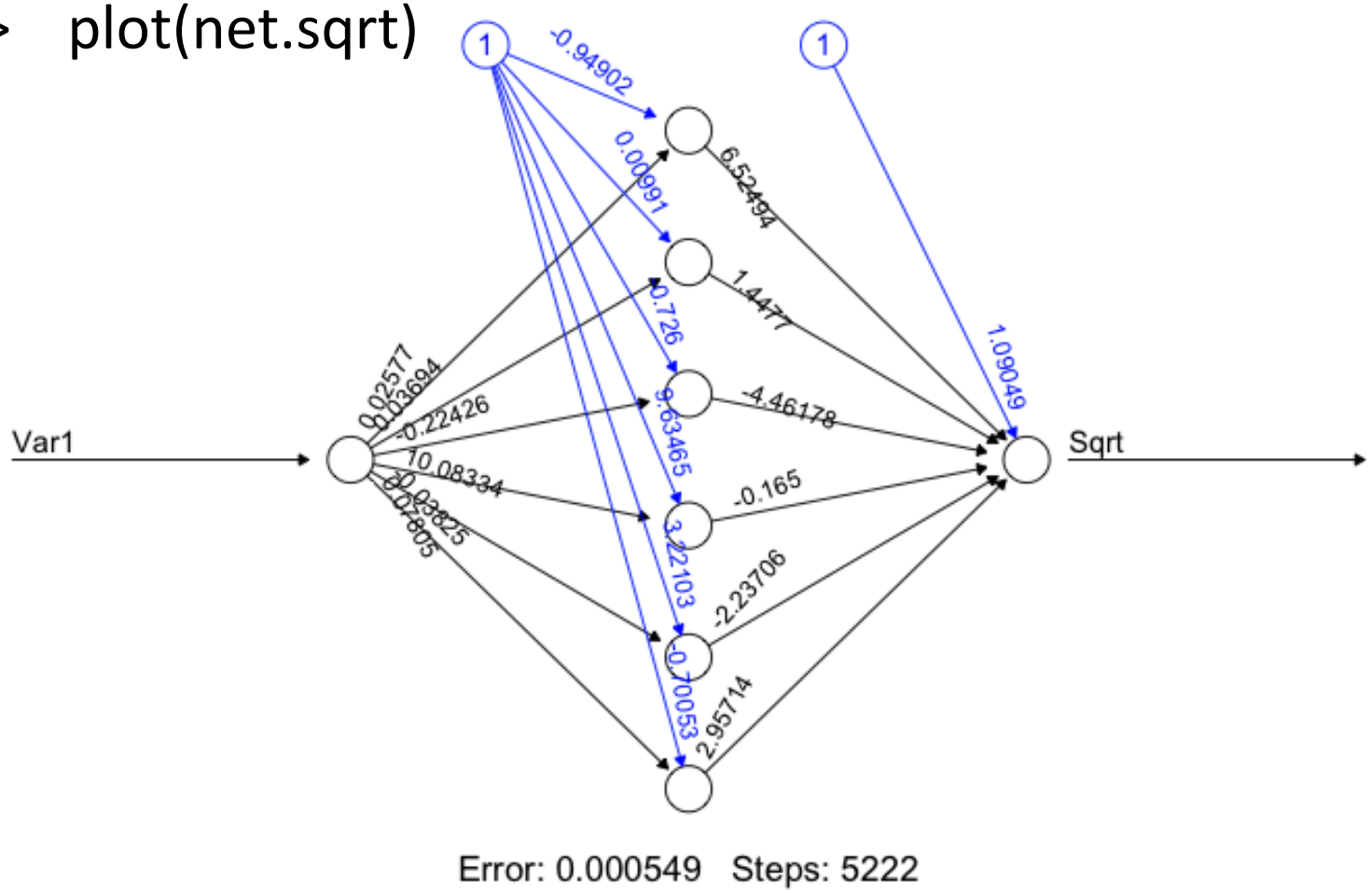
Using the “neuralnet” package. This first example adapted from package documentation.

- 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”

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
...	...	...	...	...	...	...	...

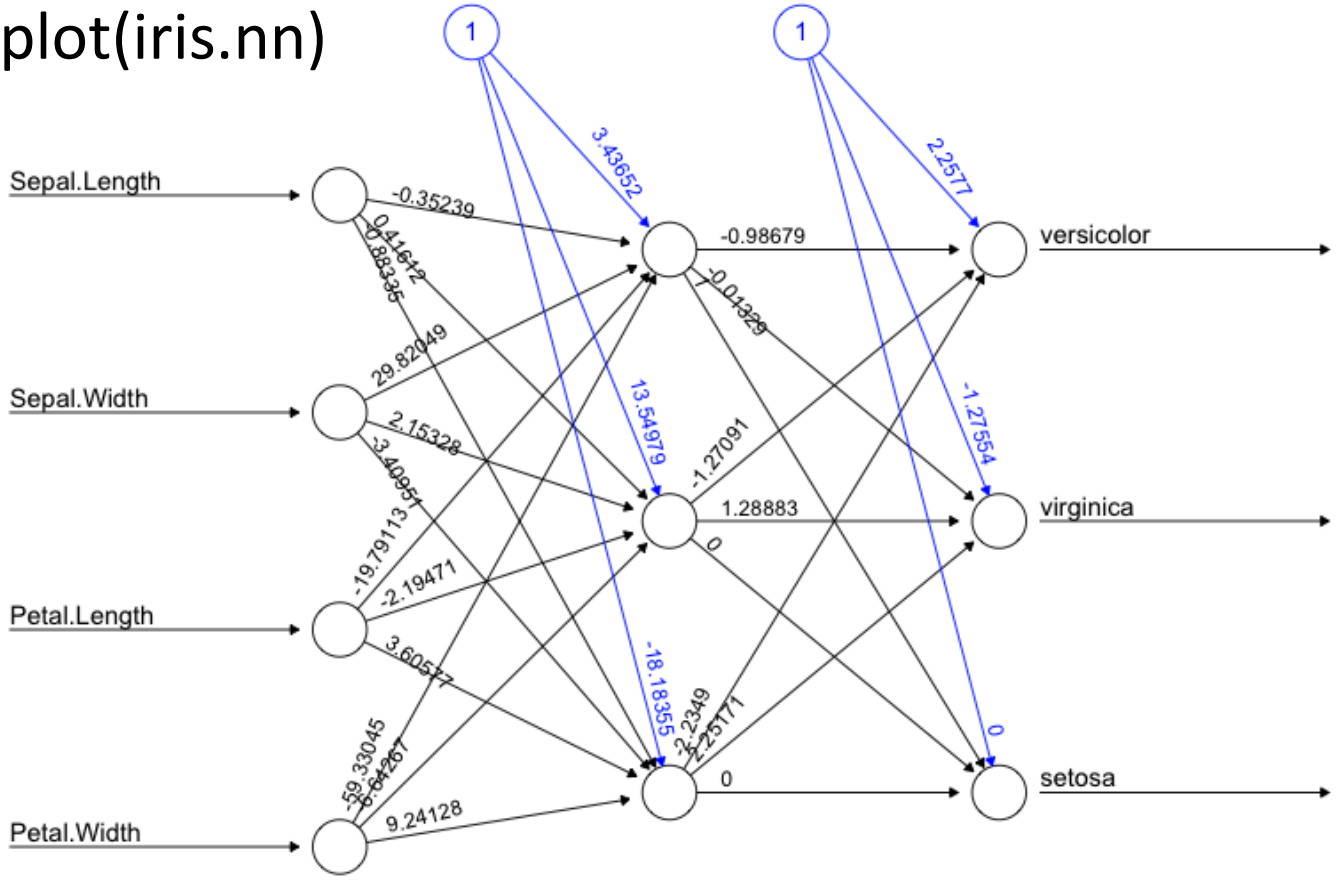
ANNs in R

- 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
```

ANNs in R

> plot(iris.nn)



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:
  - > $net.res

```
                  [,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
```

Reading: Artificial Neural Networks

- 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.
- Günther and Fritsch, neuralnet: Training of Neural Networks, The R Journal Vol. 2/1, June 2010.

Concluding thoughts

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

The rest is up to you...

Revision questions: answers

1. B
2. D
3. C
4. D

Notes on the presentation

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

Parts of this presentation originally created by Dr. Sue Bedingfield, with additions by Rui Jie Chow.