Clustering & Classification using ANNs

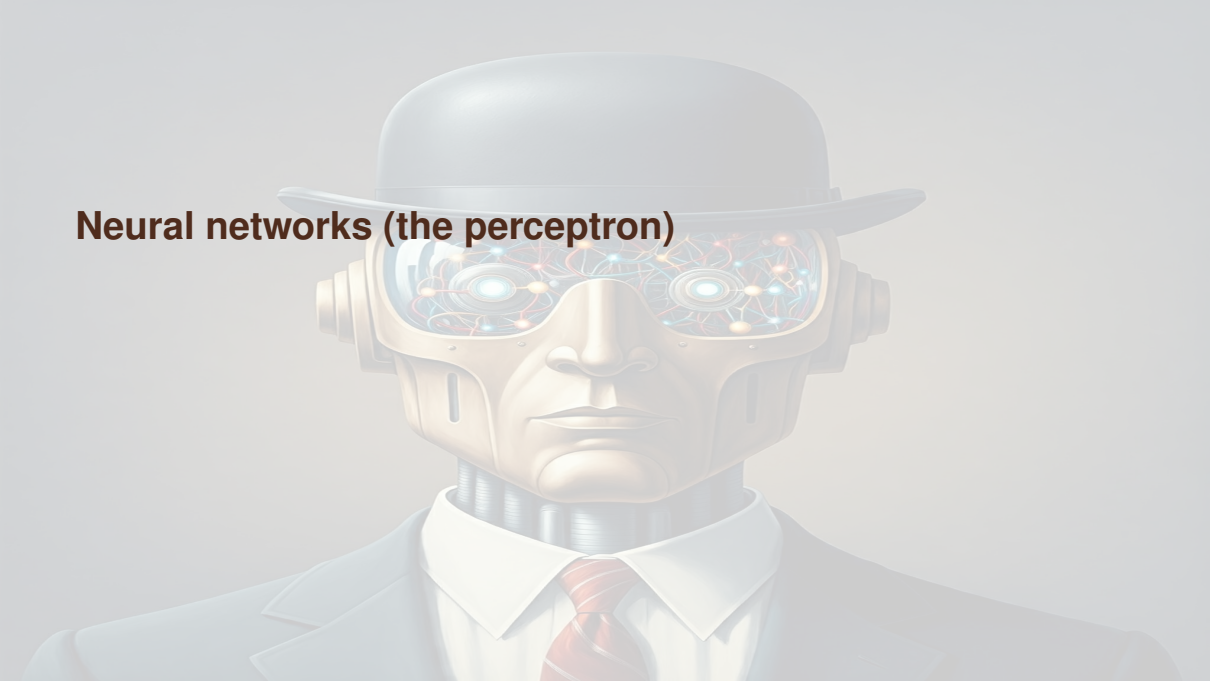
Julien Arino
University of Manitoba
julien.arino@umanitoba.ca

The University of Manitoba campuses are located on original lands of Anishinaabeg, Ininew, Anisininew, Dakota and Dene peoples, and on the National Homeland of the Red River Métis. We respect the Treaties that were made on these territories, we acknowledge the harms and mistakes of the past, and we dedicate ourselves to move forward in partnership with Indigenous communities in a spirit of Reconciliation and collaboration.

Outline

Neural networks (the perceptron)

Neural networks (the perceptron)



Setting Up R

```r
# Install nicely
if (!require("neuralnet")) {
  install.packages("neuralnet")
  library(neuralnet)
}
```

The `neuralnet` package provides:

- Flexible network specification
- Training via backpropagation
- Prediction on new data
- Network visualization

Using `neuralnet` to learn OR

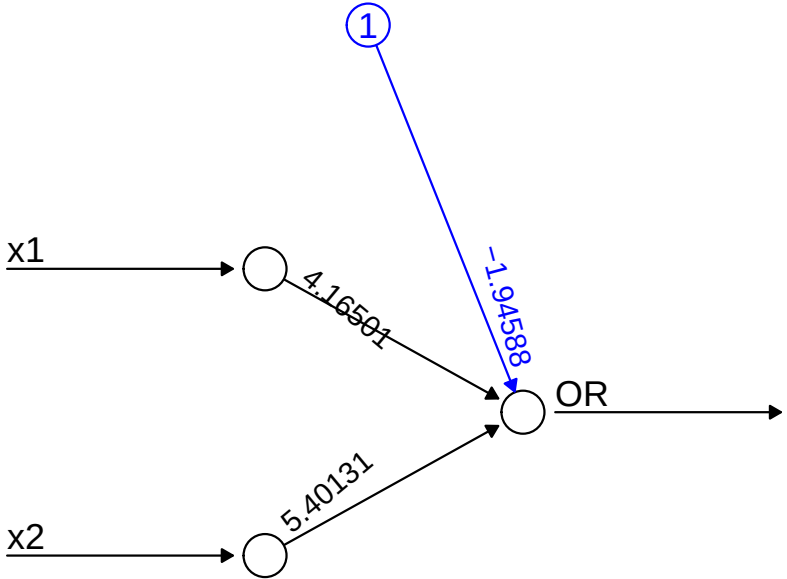
First, create the truth table

```
OR_table = matrix(c(0, 0, 0,
                    1, 0, 1,
                    0, 1, 1,
                    1, 1, 1),
                  nc = 3,
                  byrow = TRUE)
OR_table = as.data.frame(OR_table)
colnames(OR_table) = c("x1", "x2", "OR")
```

Now create and train the NN

The “formula” is to find the OR column using the x1 and x2 columns. We use no hidden layer

```
nn_OR = neuralnet(OR ~ x1 + x2,
                  data = OR_table,
                  act.fct = "logistic",
                  hidden = 0,
                  linear.output = FALSE)
# Plot the result
plot(nn_OR, rep = "best")
```



Error: 0.013088 Steps: 46

Testing the result

```
pred = predict(nn_OR, OR_table)
OR_table$result = pred > 0.5
kable(OR_table, "latex", booktabs = TRUE)
```

x1	x2	OR	result
0	0	0	FALSE
1	0	1	TRUE
0	1	1	TRUE
1	1	1	TRUE

Learning XOR

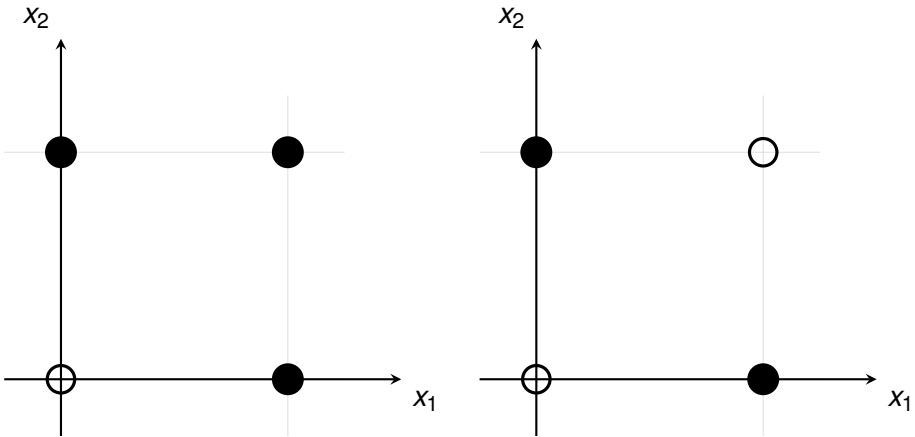
Let us now look at the XOR truth table

$$\begin{array}{cccc} 0 & 0 & \mapsto & 0 \\ 1 & 0 & \mapsto & 1 \\ 0 & 1 & \mapsto & 1 \\ 1 & 1 & \mapsto & 0 \end{array}$$

This problem is not solvable with a simple perceptron of the type we just used, as truth table is not *linearly separable*

Indeed, we would get weights $w_1 > 0$, $w_2 > 0$ to activate when presenting $[1, 0]$ and $[0, 1]$, but would require that the sum of the weights when applied to the input $[1, 1]$, give a negative value.

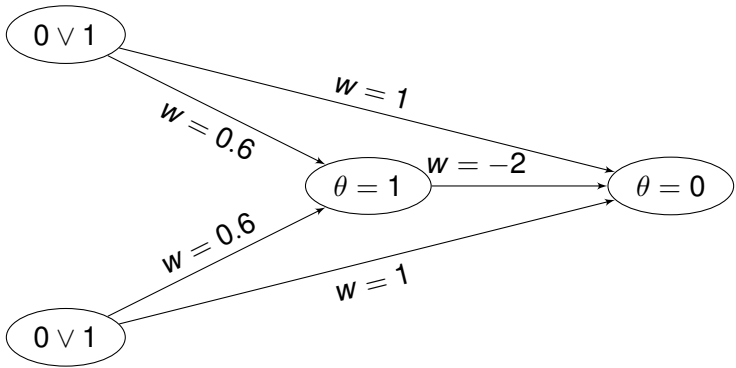
Linear separability and OR and XOR



A single-layer perceptron can only learn linearly separable problems

Adding a hidden layer

It is possible to do XOR, but we need to add a **hidden layer**



Now the XOR truth table

```
XOR_table = matrix(c(0, 0, 0,
                     1, 0, 1,
                     0, 1, 1,
                     1, 1, 0),
                   nc = 3, byrow = TRUE)
XOR_table = as.data.frame(XOR_table)
colnames(XOR_table) = c("x1", "x2", "XOR")
kable(XOR_table, "latex", booktabs = TRUE)
```

x1	x2	XOR
0	0	0
1	0	1
0	1	1
1	1	0

Try to learn it without a hidden layer

```
nn_XOR = neuralnet(XOR ~ x1 + x2,
                   data = XOR_table,
                   act.fct = "logistic",
                   hidden = 0,
                   linear.output = FALSE)
pred = predict(nn_XOR, XOR_table)
XOR_table$result = pred > 0.5
kable(XOR_table, "latex", booktabs = TRUE)
```

x1	x2	XOR	result
0	0	0	TRUE
1	0	1	TRUE
0	1	1	FALSE
1	1	0	FALSE

Now with a hidden layer

```
nn_XOR = neuralnet(XOR ~ x1 + x2,
                   data = XOR_table,
                   act.fct = "tanh",
                   hidden = 1)
pred = predict(nn_XOR, XOR_table)
XOR_table$result = pred > 0.5
kable(XOR_table, "latex", booktabs = TRUE)
```

x1	x2	XOR	result
0	0	0	FALSE
1	0	1	TRUE
0	1	1	TRUE
1	1	0	TRUE

Still hard with one hidden node

Using two hidden nodes

```
set.seed(42)
nn_XOR <- neuralnet(XOR ~ x1 + x2,
                    data = XOR_table,
                    hidden = 2,                    # need >=2 neurons
                    act.fct = "logistic",          # sigmoid
                    linear.output = FALSE,         # classification
                    err.fct = "ce",                # cross-entropy
                    algorithm = "rprop+",          # rprop+ often works well
                    stepmax = 1e6,
                    threshold = 1e-4,
                    rep = 1)
# Use compute() to get predictions
pr <- compute(nn_XOR, XOR_table[, c("x1", "x2")])$net.result
XOR_table$pred_prob <- pr[,1]
XOR_table$result <- (pr > 0.5)
```

Did it work?

```
kable(XOR_table, "latex", booktabs = TRUE)
```

x1	x2	XOR	result	pred_prob
0	0	0	FALSE	0.0000767
1	0	1	TRUE	0.9999371
0	1	1	TRUE	0.9999047
1	1	0	FALSE	0.0000766

Using two hidden nodes (another way)

```
set.seed(42)
nn_XOR_tanh <- neuralnet(XOR ~ x1 + x2,
                         data = XOR_table,
                         hidden = 2,
                         act.fct = function(x) tanh(x),
                         linear.output = FALSE,
                         err.fct = "sse", # SSE is OK with tanh targets
                         algorithm = "rprop+",
                         stepmax = 1e6,
                         threshold = 1e-4)

pr2 <- compute(nn_XOR_tanh, XOR_table[, c("x1", "x2")])$net.result
XOR_table$pred_tanh <- pr2[,1]
XOR_table$result_tanh <- as.integer(pr2 > 0)  # positive -> class 1
```

Did it work?

```
kable(XOR_table, "latex", booktabs = TRUE)
```

x1	x2	XOR	result	pred_prob	pred_tanh	result_tanh
0	0	0	FALSE	0.0000767	0.0000675	1
1	0	1	TRUE	0.9999371	0.9931660	1
0	1	1	TRUE	0.9999047	0.9918250	1
1	1	0	FALSE	0.0000766	0.0000730	1

An example from the `neuralnet` manual – Training vs testing sets

`iris` is a built-in `R` dataset detailing physical characteristics of 150 flowers from 3 iris species

```
train_idx <- sample(nrow(iris), 2/3 * nrow(iris))
iris_train <- iris[train_idx, ]
iris_test <- iris[-train_idx, ]
```

Thus we pick at random 2/3 of the data for training and 1/3 for testing. See some considerations on training, validation and testing on this Wikipedia page

An example from the `neuralnet` manual

```
nn <- neuralnet(Species == "setosa" ~ Petal.Length + Petal.Width,
                iris_train, linear.output = FALSE)
pred <- predict(nn, iris_test)
table(iris_test$Species == "setosa", pred[, 1] > 0.5)

##
##         FALSE TRUE
##   FALSE    31    0
##   TRUE      0   19
```

Another example – multiclass classification

```
nn <- neuralnet((Species == "setosa") +
                  (Species == "versicolor") +
                  (Species == "virginica")
                ~ Petal.Length + Petal.Width,
               iris_train, linear.output = FALSE)
pred <- predict(nn, iris_test)
table(iris_test$Species, apply(pred, 1, which.max))

##
##                1  2  3
##    setosa     19  0  0
##    versicolor  0 14  1
##    virginica   0  2 14
```