Introduction to machine learning

Andrew Parnell
andrew.parnell@mu.ie

I-Form Advanced Manufacturing Research Centre

https://andrewcparnell.github.io/intro_to_ml

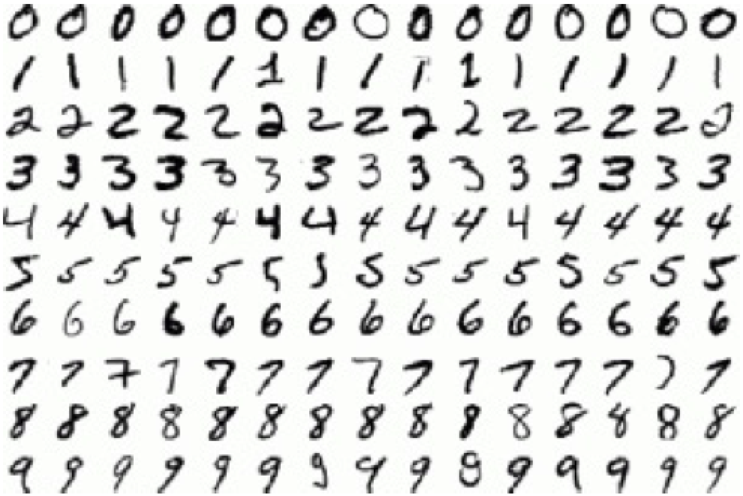
Outline of today

- Machine learning jargon
- How to do machine learning
- Description of machine learning black boxes
- Image classification example
- Assessing performance
- Introduction to Rstudio
- Longer code examples

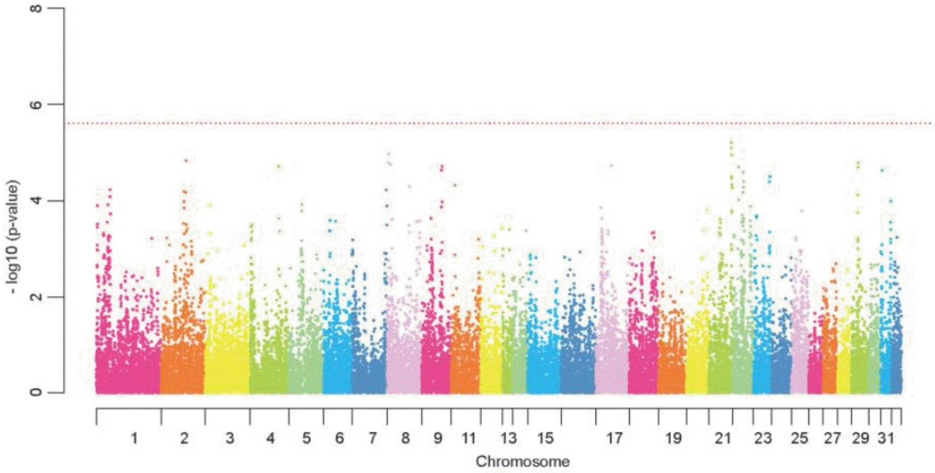
What is ML and AI?

- ML and AI are methods to make a computer take input data and make predictions.
- In the recent past, a huge explosion has led to the Big Data revolution:
  - Volume
  - Variety
  - Veracity
  - Velocity

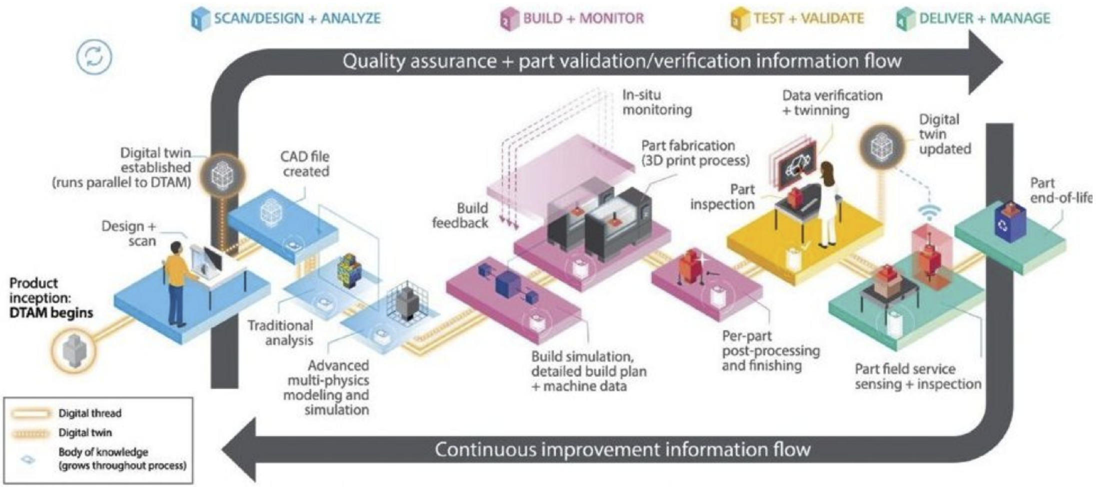
Example 1: digit recognition



Example 2: genomics



Example 3: manufacturing


SCAN/DESIGN + ANALYZE
BUILD + MONITOR
TEST + VALIDATE
DELIVER + MANAGE
Quality assurance + part validation/verification information flow
In-situ monitoring
Data verification + twinning
Digital twin updated
Digital twin established (runs parallel to DTAM)
CAD file created
Part fabrication (3D print process)
Part inspection
Build feedback
Part end-of-life
Design + scan
Product inception: DTAM begins
Traditional analysis
Advanced multi-physics modeling and simulation
Build simulation, detailed build plan + machine data
Per-part post-processing and finishing
Part field service sensing + inspection
Continuous improvement information flow
Digital thread
Digital twin
Body of knowledge (grows throughout process)

The goal

$$\text{Features} \rightarrow \text{Black box} \rightarrow \text{Target}$$

- Target: the value to be predicted
- Features: the values used to predict from
- Training data: The rectangular set of features and target values upon which to fit the . . .
- Black box (Machine learning model): the algorithm by which predictions are made
- Test data: Another rectangular data set with known target and feature values on which to evaluate performance

The number of features is usually called p and the number of samples n

Traditional statistics vs ML

- Traditional statistical analysis has focused on linear regression models and their extensions
- Often this ignores interactions and struggles in very large data sets
- People often base their results on p-values
- ML creates non-linear models with automatic interaction detection and works well on big data
- However you lose interpretation and the focus is often solely on prediction (rather than causation)
- ML models often ignore the experimental design
- ML models often do not provide uncertainty in predictions

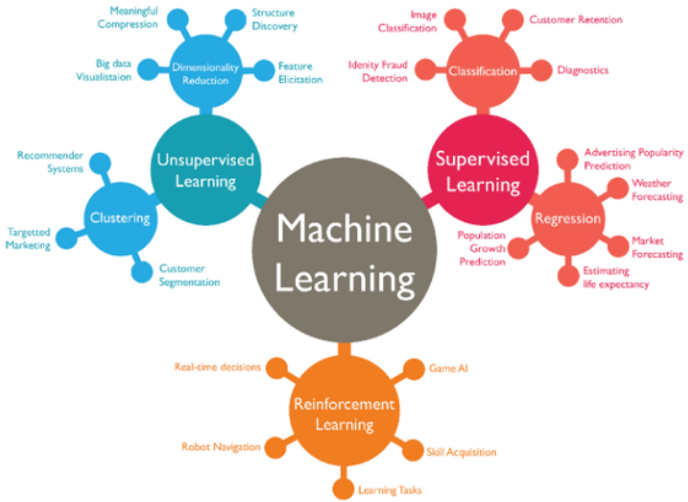
Example code

```r
library(mlr)
# Make a task
task_1 = makeRegrTask(data = my_data_set,
                      target = 'y')
# Create training and test sets
test_set = sample(n, size = n/4)
train_set = (1:n)[-test_set]
# Set up a leaner
learner_1 = makeLearner('regr.randomForest')
# Train it on the training data
train_1 = train(learner = learner_1, task = task_1,
                subset = train_set)
# Predict on the test set data
pred_1 = predict(object = train_1, task = task_1, subset = test_set)
# Check performance
performance(pred_1)
```

Reminder of terminology

- A task is the data set, the features, and the target variable
- A training and test set are just the rows of the data that we will train and test our model on
- A learner is a black box that aims to predict the target from the features
- The predictions on the test set can be compared to the true target values to judge the performance of the learner

Types of Machine learning



Measuring performance

- If the data set is big enough (i.e. n large), split into a training and test set
- Fit the machine learning model on the training set
- Evaluate performance on the test set
- The training and the test set must be as separate as possible

A proper validation will use a separately collected test set

Common pitfalls

- You get to choose the targets and the features; you need to choose these well
- You must get the validation right
- If the data contain serious biases they will be reflected in the model
- If you have a small data set (e.g. $n < 10,000$), sophisticated ML will not help much compared to simple models

Big data: fat vs tall

- Lots of manufacturing data is fat data, i.e. there are more features than cases ($p \gg n$)
- This causes problems for some traditional statistical models but ML can still work well here
- However, identification of effects gets harder as does validation
- Most of the successes of ML have come from tall data where there are many more cases than features ($n \gg p$)

What are the black boxes?

If we write the target as y and the features as X all these can be written as rectangular data structures (like an Excel spreadsheet)

Our job is to find f in:

$$y = f(X) + e$$

by making the e error values as small as possible

- f is known as a learner. There are lots of different types: linear regression, neural networks, random forests, support vector machines, . . .

1. Linear regression

The simplest possible useful model is a linear regression model:

$$y = a + b_1x_1 + b_2x_2 + b_3x_3 + \ldots + e$$

so $f(X) = a + b_1x_1 + b_2x_2 + b_3x_3 + \ldots$

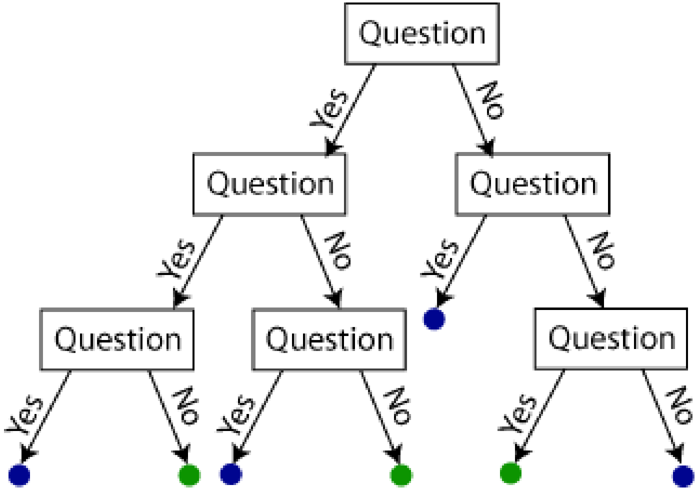
- We usually minimise the least squares function:

$$\sum (y - f(X))^2$$

  This gives us the best values of b_1, b_2, etc

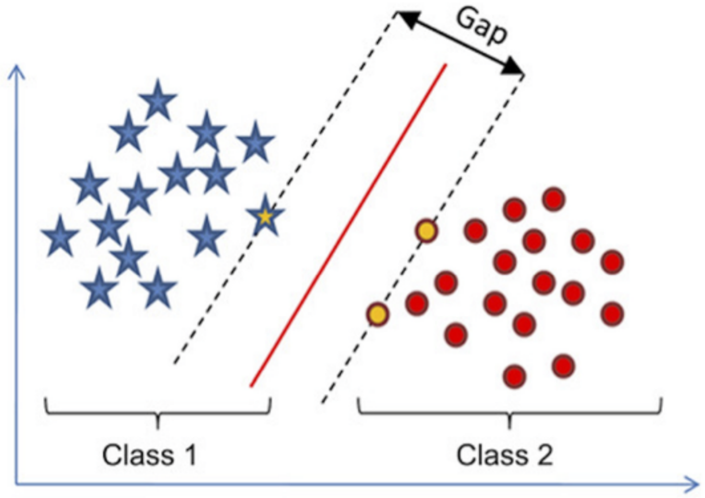
2. Random forests

Change the function f to be a set of *decision trees*



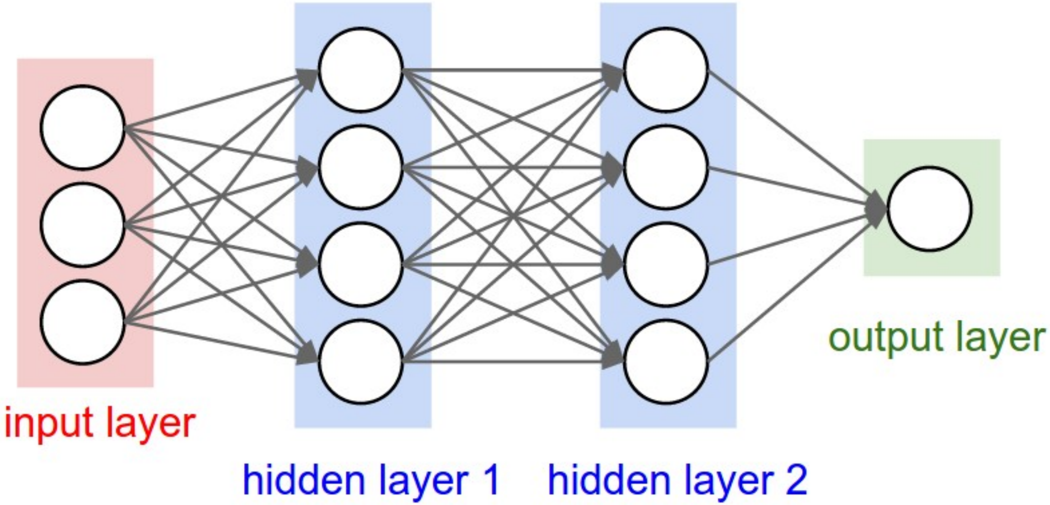
3. Support vector machines

Use only those data points that affect the final prediction:

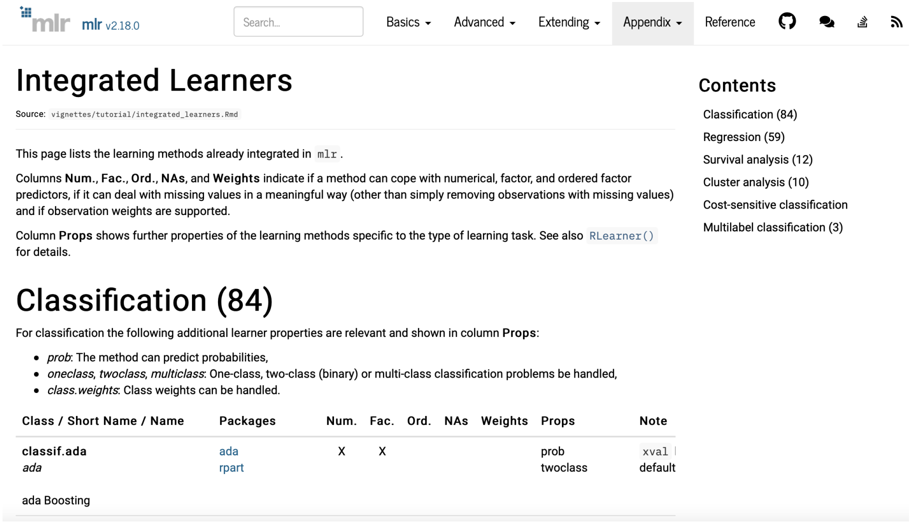


4. Neural networks

Make the regression weight (b) depend on other regression weights



5. Other black boxes



Integrated Learners

Source: `vignettes/tutorial/integrated_learners.Rmd`

This page lists the learning methods already integrated in `mlr`.

Columns **Num.**, **Fac.**, **Ord.**, **NAs**, and **Weights** indicate if a method can cope with numerical, factor, and ordered factor predictors, if it can deal with missing values in a meaningful way (other than simply removing observations with missing values) and if observation weights are supported.

Column **Props** shows further properties of the learning methods specific to the type of learning task. See also `RLearner()` for details.

Classification (84)

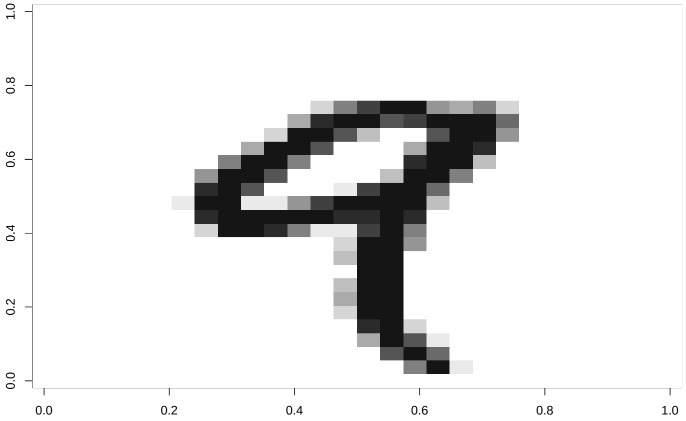
For classification the following additional learner properties are relevant and shown in column **Props**:

- *prob*: The method can predict probabilities,
- *oneclass*, *twoclass*, *multiclass*: One-class, two-class (binary) or multi-class classification problems be handled,
- *class.weights*: Class weights can be handled.

Class / Short Name / Name	Packages	Num.	Fac.	Ord.	NAs	Weights	Props	Note
classif.ada *ada* ada Boosting	ada rpart	X	X				prob twoclass	`xval` default

20 / 28

Example: the digits data



What does the full data look like?

```r
dim(mnist)
```

```
## [1] 70000   785
```

```r
mnist[1:6, c(405:410, 785)] # Mostly white!
```

```
##   pix405 pix406 pix407 pix408 pix409 pix410 digit
## 1      0     81    240    253    253    119     5
## 2      0      0      0      0      0      0     0
## 3    144    150    241    243    234    179     4
## 4     80    240    251    193     23      0     1
## 5    197    241    253    252    251     77     9
## 6    179    253    253    255    253    253     2
```

Fitting a random forest classifier

```r
# Make a task and set up training/test sets
task_mn = makeClassifTask(data = mnist, target = 'digit')
train_set = 1:60000
test_set = 60001:70000
# Set up a leaner
learner_rf = makeLearner('classif.h2o.randomForest',
                        predict.type = 'prob')
# Train it on the training data
train_rf = train(learner = learner_rf, task = task_mn, subset = train_set)
pred_rf = predict(object = train_rf, task = task_mn, subset = test_set)
```

```r
# Check performance
print(performance(pred_rf))
```

```
##   mmce
## 0.0317
```

How well did it go?

```
##      class
##          0    1    2    3    4    5    6    7    8    9
##    0   970    0    0    0    0    1    4    2    3    0
##    1     0 1124    2    2    0    1    4    0    1    1
##    2     7    0  992    7    2    0    1    8   14    1
##    3     0    0    7  972    0    5    1   11   12    2
##    4     1    0    2    0  951    0    6    0    2   20
##    5     3    0    1   11    0  862    7    1    3    4
##    6     4    3    0    0    3    5  936    0    7    0
##    7     1    6   19    2    2    0    0  983    1   14
##    8     3    1    7   10    2    4    3    5  934    5
##    9     8    6    2   14    8    1    1    6    3  960
```

What about other ML black boxes?

It is now easy to train other models in the same way

```
learner_dl = makeLearner('classif.h2o.deeplearning',
                         predict.type = 'prob')
# Train it on the training data
train_dl = train(learner = learner_dl,
                 task = task_mn,
                 subset = train_set)
# Predict on the test set data
pred_dl = predict(object = train_dl,
                  task = task_mn,
                  subset = test_set)
```

```
print(performance(pred_dl))
```

```
##   mmce
## 0.0254
```

Assessing performance of ML models

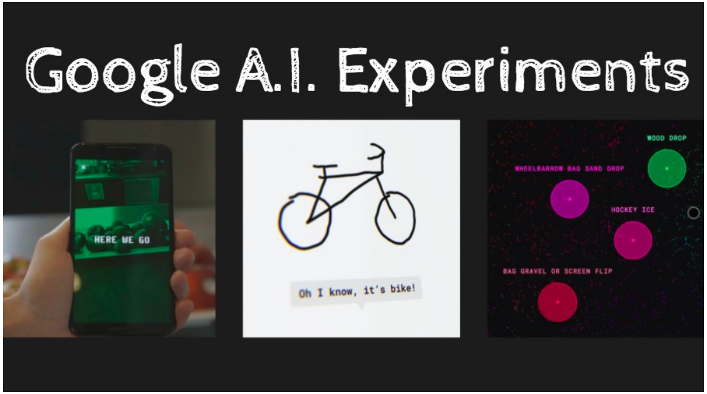
- A good model should have predicted values close to the true values, or predicted classes close to the true classes. Some models will also produce uncertainties (e.g. intervals) or probabilities which can help determine the quality of the fit
- The usual ones people use are the root mean squared error (for regression problems), the misclassification rate (for classification problems)
- Other more advanced ones for classifiers include the receiver operator characteristic curve and precision-recall curve. These work on the probabilities of each observation being in a particular class

Key challenges for ML in Additive Manufacturing

- How do we merge multiple different types of data sets together
- How do we deal with missing data?
- How do we quantify uncertainty in very large data sets?
- How do we deal with streaming data?
- How do we incorporate human/sensor feedback?
- How do we deal with multivariate targets?

Lastly, why not have a play with ML yourself?

`https://aiexperiments.withgoogle.com/quick-draw`



If you want to run the code for any of the analysis in the presentation visit https://goo.gl/kG4CM5