

Introduction to machine learning

Frederique Bone

Introductory Data Science for Innovation (995N1) – Week 10, December 2021

Outline

Outline

- Lecture's objectives
- What is machine learning?
- The different types of machine learning
- Working with supervised machine learning
- Working with unsupervised machine learning
- Ethical considerations of machine learning

Lecture's objectives

Lecture's objectives

After having learned about getting data, processing it, working with text (text mining) and understanding data visualisation, this lecture is going to introduce machine learning. You may already have encountered some of the machine learning algorithms we are going to see today.

This lecture aims first at giving you an overview of what machine learning is, giving you a few definitions, what it is generally used for, and in broad lines what are the different types of machine learning that you may encounter.

Second we will delve into a few examples of common algorithms used in machine learning and run a few practical examples in R. We will also discuss some ethical concerns that you should keep in mind when using or implementing such a system.

What is machine learning?

What is machine learning?

Machine learning gives computers the ability to acquire their own knowledge, by extracting patterns from data, without the need to give explicit rules / hard-coded knowledge (Ian Goodfellow, Yoshua Bengio and Courville 2016). The computer will learn from experience.

Machine learning uses computer algorithms to solve problems through the building of models and in turn allows it also to take decisions.

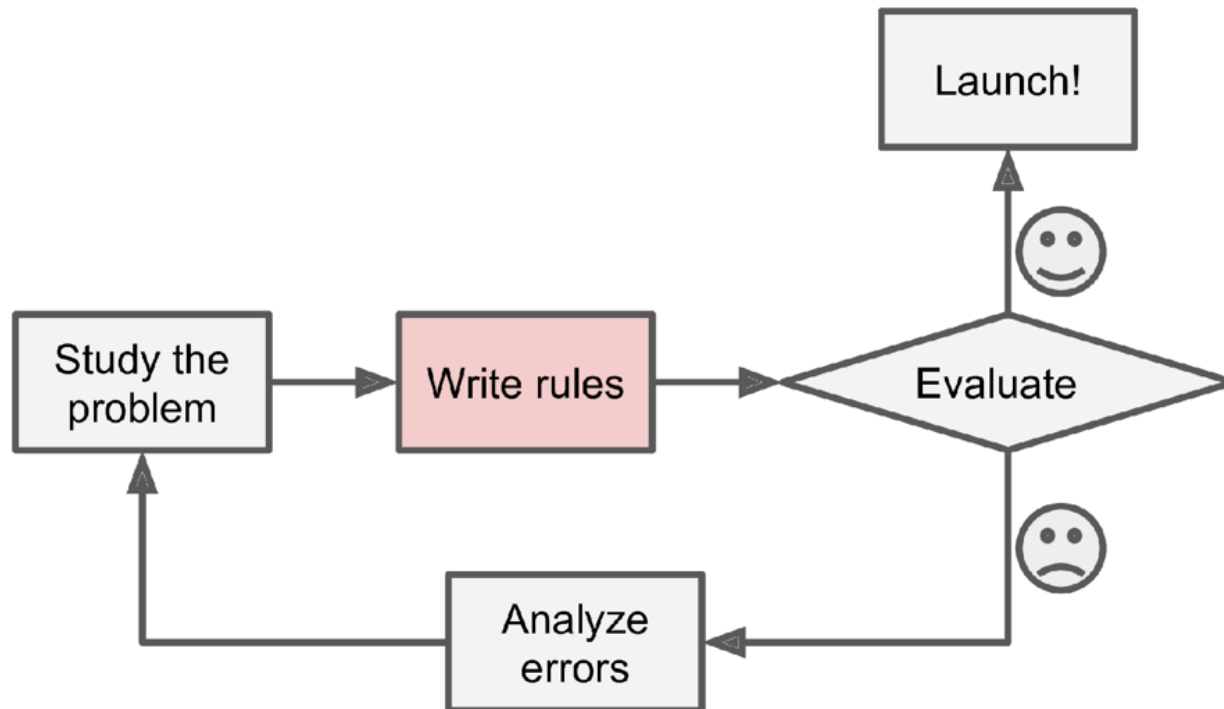
What is machine learning?

“Every algorithm has an input and an output: the data goes into the computer, the algorithm does what it will with it, and out comes the result. **Machine learning** turns this around: in goes the data and the desired result and out comes the algorithm that turns one into the other. Learning algorithms -also known as learners- are algorithms that make other algorithms.”

The master algorithm - Source : Domingos (2015)

What is machine learning?

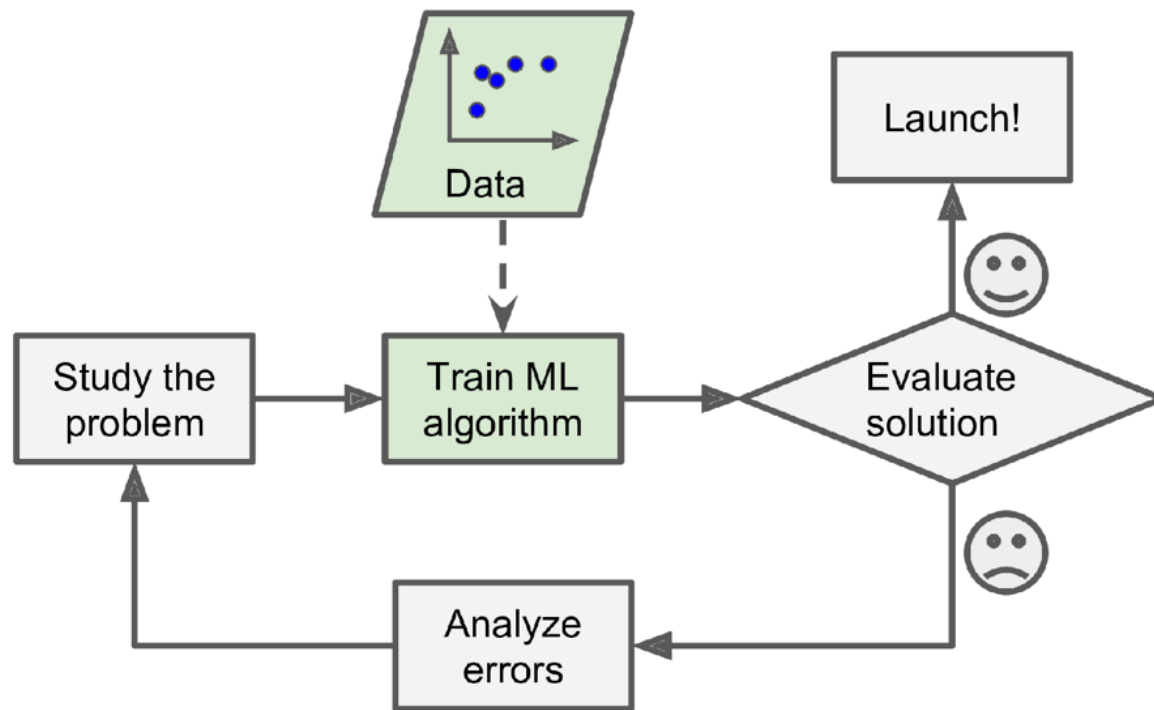
Before machine learning, programmers had to instruct rules so that the computer could follow a step by step guide on how to work with data.



Symbolic approach to programming - Source: Géron (2019)

What is machine learning?

Machine learning involves building models from data. The advance here, is that the model properties may not be known by the programmer, but that the algorithm will make a model using the properties of the data.



Solve problems with learning algorithms Source: (Géron 2019)

Types of machine learning

Types of Machine learning

How can the algorithm learn from data?

Here are the most common types of machine learning techniques:

- Supervised learning
- Unsupervised learning
- Semi-supervised learning
- Reinforcement learning
- Active learning

There are many ways in which a model can be built using data.

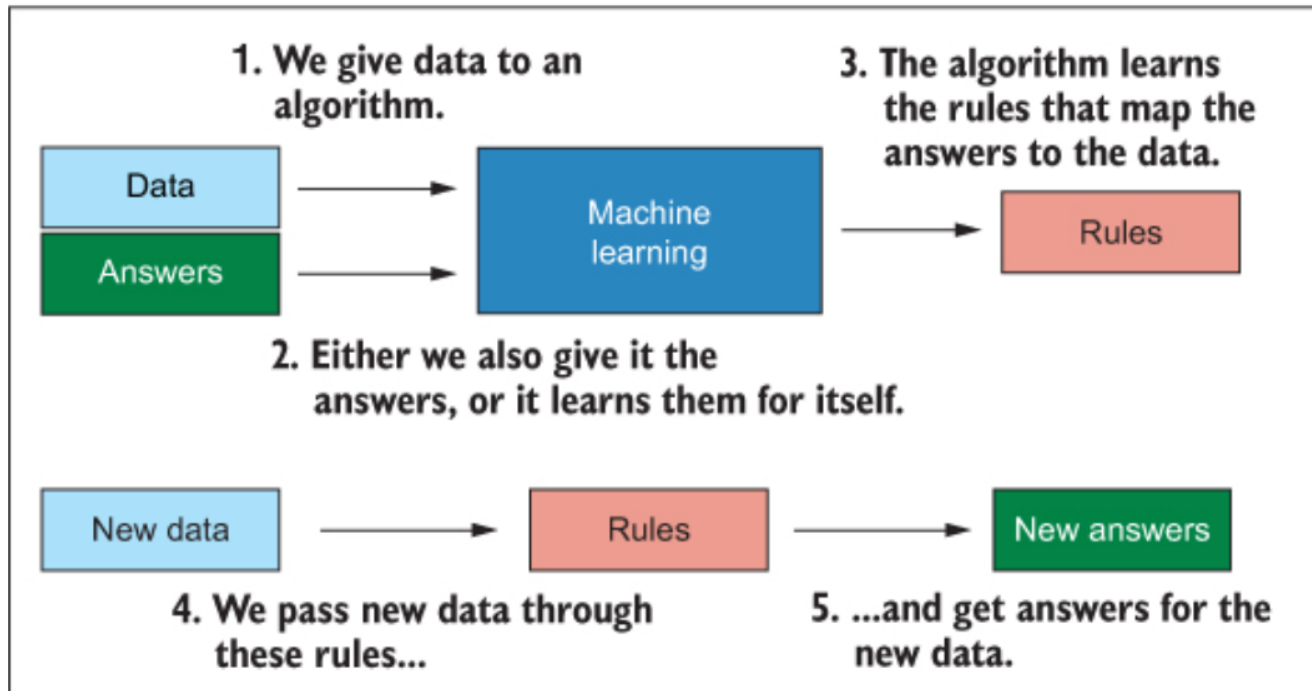
Types of Machine learning

How can the algorithm learn from data? Here are the most common types of machine learning techniques:

- **Supervised learning**
- **Unsupervised learning**
- Semi-supervised learning
- Reinforcement learning
- Active learning

First, we will explore in more details what is meant by supervised and unsupervised learning.

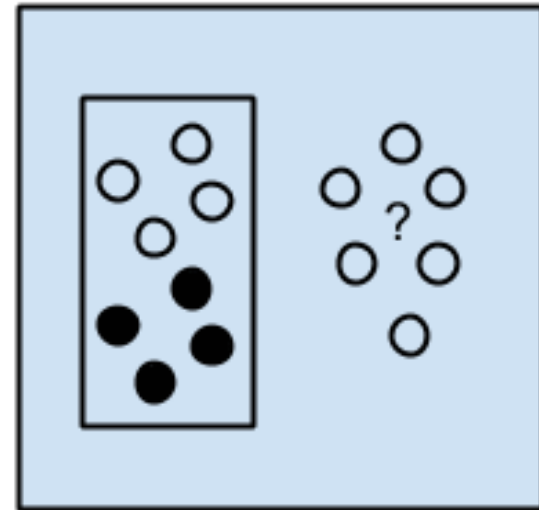
Supervised learning



Solve problems with learning algorithms - Source: Rhys (2020)

Supervised learning

Supervised learning: the machine builds a model from *labelled* examples. This enables the prediction of features for other unlabelled examples. This type of machine learning is used mainly to sort examples in different categories (classification purposes), or predict a target (numeric value) based on a set of features.

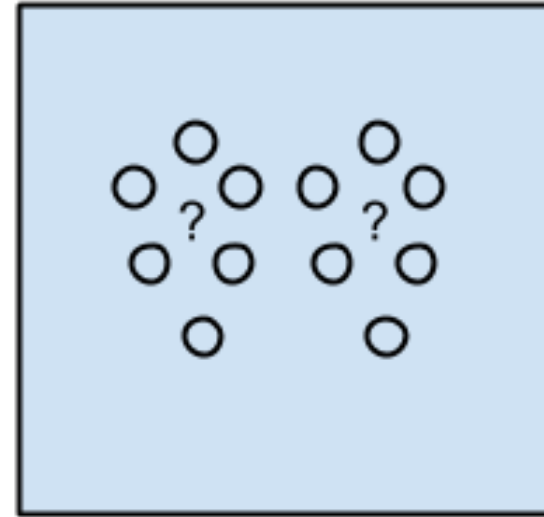


Supervised Learning
Algorithms

Unsupervised learning

Unsupervised learning: This learning is done without labelled examples; the machine learns from the known features of the data. The algorithm try to either group together examples which are similar (i.e. clustering), or understand differentiating features of the data, or reduce the number of variables.

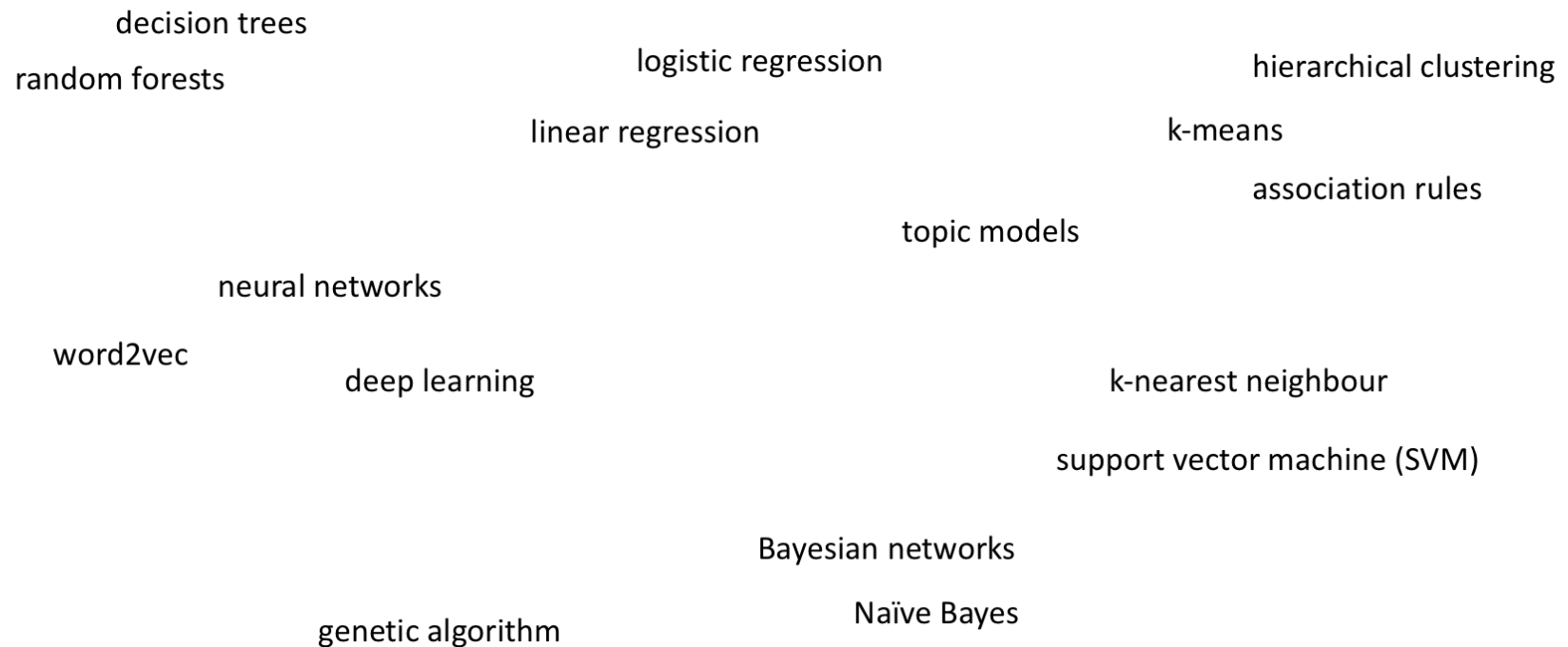
After the ML sorted the data, some labelling may still be required, but these are done post-learning.



Unsupervised Learning
Algorithms

Types of Machine learning

Here are a few algorithms that you may recognise:

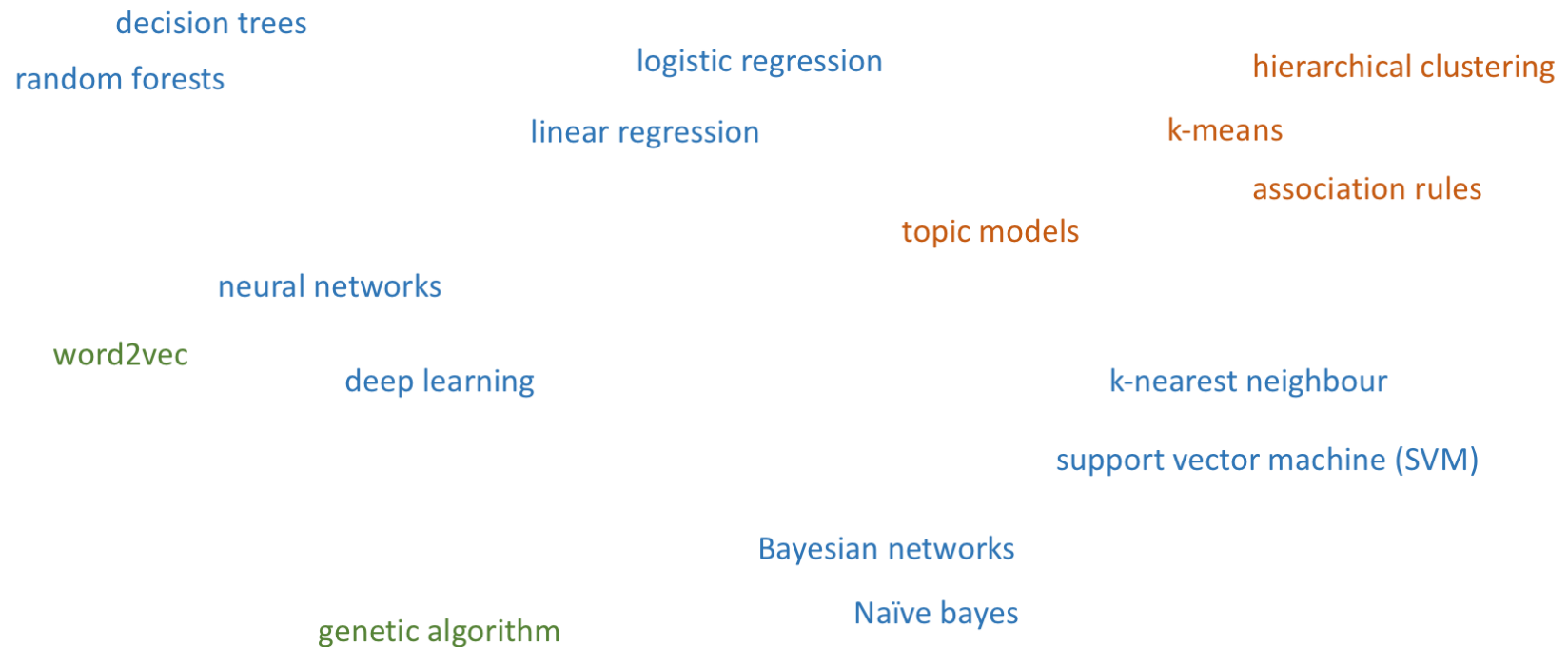


A collection of machine learning algorithm names scattered across the slide. The algorithms listed are: decision trees, random forests, logistic regression, linear regression, hierarchical clustering, k-means, association rules, topic models, neural networks, word2vec, deep learning, k-nearest neighbour, support vector machine (SVM), Bayesian networks, Naïve Bayes, and genetic algorithm.

decision trees
random forests
logistic regression
linear regression
hierarchical clustering
k-means
association rules
topic models
neural networks
word2vec
deep learning
k-nearest neighbour
support vector machine (SVM)
Bayesian networks
Naïve Bayes
genetic algorithm

Types of Machine learning

Classifying algorithms - Blue:supervised, Orange:unsupervised, Green:debatable



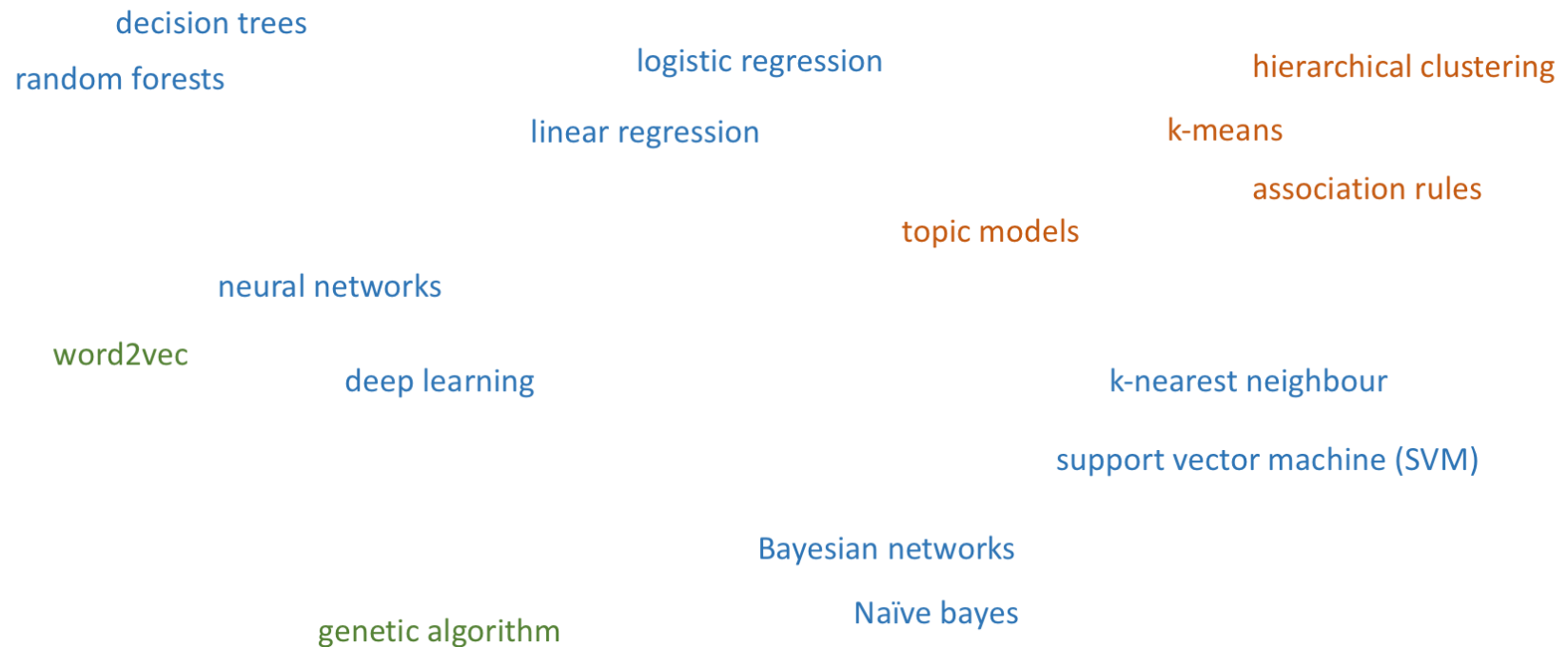
Types of Machine learning

How can the algorithm learn from data?

- Supervised learning
- Unsupervised learning
- **Semi-supervised learning** : Use a mix of supervised and unsupervised learning
- **Active learning**: The machine learning chooses the examples to be labelled
- **Reinforcement learning**: The machine has to maximise a reward given for good action (e.g. alphago)

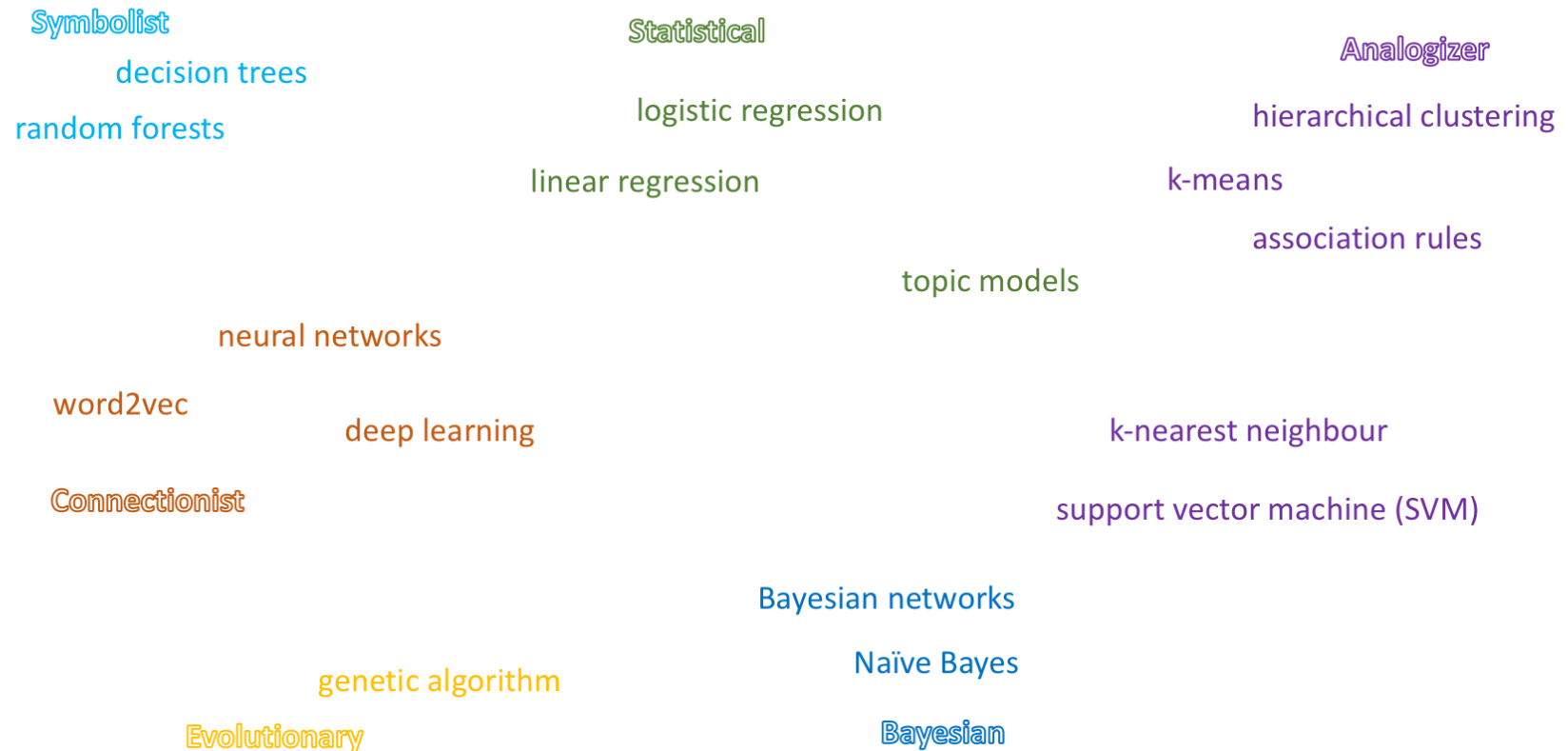
Types of Machine learning

You can classify these algorithms by training type:



Types of Machine learning

Or for instance by method's type, inspired by Domingos (2015):



Supervised and Unsupervised Learning

In the next part of the lecture, we are going to explore in more details what supervised and unsupervised learning entails. While going through some general principles for these two types of machine learning, we will delve into examples used in innovation studies.

We will see some practical examples using in R using text data.

Supervised Learning

General principles

Supervised learning is used when the analyst has a clear idea of the target to be reached. The target is instructed to the machine using data, where one variable acts as the target.

The target can either be a classification, where there can be two (binary) or more (factors) classes; or a regression problem, where the target is a number, and the algorithm will learn by looking how far off it is from the target.

The model will learn from each of the examples given; and to be performant you will need to give it a variety of cases.

The data given to the algorithm is split into two sets, a training set which will be used to build the model, and a test set to evaluate how good the model constructed actually is.

General principles

In **supervised learning**, there are five main steps to follow:

- Prepare the data to the appropriate format
- Split the data into two sets: one to build the model (training data), one to check the performance of the model (testing data)
- Build the model with the training data
- Check how good the model is, using the test data
- Resampling can be done to improve the accuracy of the algorithm

Practicing with R

In this week's exercise we will go more in depth into supervised learning. We will start from an example code of a supervised learning algorithm, which we will try to improve upon.

We will iteratively alter parameters, such as adjusting the training data, changing the algorithm to understand how the performance of the model is changing. Do open your workbook now, and we will start to work through the workbook together.

R packages

There are two main packages for machine learning in R.

These are the **caret**, the **mlr** and more recently the **tidymodels** packages.

For our examples of supervised learning, we are going to use latest version of the mlr (i.e. the mlr3) package, as it is simple to use and has a large range of models.

Practicing supervised learning (1)

To exemplify the process of supervised machine learning, we are going to use some data from the Cancerscreen project. The problem consists of checking whether a given paper is about diagnostics or not.

To classify the data, we used the MeshTerm classification, where we identify terms which are sepcific to diagnostics.

Ultimately the model would be used on data retrieved from bibliometric datasets, such as the Web of Science and Scopus and does not benefit from a classification.

Practicing supervised learning (2)

1. Explore data

```
Medline <- readRDS("Diag_full.rds")
```

```
Medline <- Medline %>%
```

```
  select(-diag_type, -case)%>%
```

```
  unique()
```

```
summary(Medline)
```

```
##          PMID          T_A          diagnosis
## Min.      :22143403 Length:26751      N:11636
## 1st Qu.:25630780   Class :character Y:15115
## Median :25937533   Mode  :character
## Mean      :25928773
## 3rd Qu.:26268906
## Max.      :28590602
```

We have three columns:

PMID : Pubmed ID for each of the document

T_A: text - combination of the title and abstract *diagnosis*: is the paper about diagnosis

Practicing supervised learning (3)

Let's have a look at the data first:

```
Medline <- as_tibble(Medline)
glimpse(Medline)
```

```
## Rows: 26,751
## Columns: 3
## $ PMID      <int> 22143403, 22246415, 22467405, 22548776, 23188650, 23188879, ...
## $ T_A       <chr> "Bayesian analysis on meta-analysis of case-control studies ..."
## $ diagnosis <fct> N, N, Y, N, N, N, N, N, Y, N, Y, Y, Y, Y, Y, N, N, Y, N, Y, ...
```

For supervised machine learning we will need examples, a target and features. The dataset is organised into observations (by articles represented by PMIDs), we have a target which is the variable diagnosis, but how do we transform the text into features?

Practicing supervised learning (4)

We need first to normalise the text data...

```
library(tidytext)
library(tm)
library(SnowballC)
Stop_w<- c(stopwords("en"), "introduction", "conclusion","objective", "aim", "methods", "result")
Medline$T_A <- removeNumbers(Medline$T_A)
Medline$T_A <- tolower(Medline$T_A)
Medline$T_A <- str_replace_all(Medline$T_A, "[^[:alnum:]]", " ")
Medline$T_A <- removeWords(Medline$T_A, Stop_w)
Medline$T_A <- gsub('\\b\\w{1,2}\\b', '', Medline$T_A) # remove words of two letters or under
```

Practicing supervised learning (5)

Then we are transforming the data into a vector of features using tf-idf:

```
# reduce the size of the data to seed up the routine
```

```
Tidy_Med <- Medline [1:3000,]
```

```
# extract token and use stemming.
```

```
Tidy_Med <- Medline %>%
```

```
  unnest_tokens(word, T_A) %>%
```

```
  mutate(word = wordStem(word, language="english"))
```

```
# compute word counts
```

```
Tidy_Med <- Tidy_Med %>%
```

```
  count(PMID, word, sort=T)
```

```
# compute tf_idf
```

```
Tidy_Med <- Tidy_Med %>%
```

```
  bind_tf_idf(word, PMID, n)
```


Practicing supervised learning (6)

Then we are grouping words together, the words will become the features:

```
# 4. Get the word as features in columns
```

```
# save the diagnostics features separately
```

```
diag <- Medline %>%  
  select(PMID, diagnosis)%>%  
  unique()
```

```
# Create a list of word to keep which are higher than a certain threshold
```

```
word <- Tidy_Med %>%  
  group_by(word) %>%  
  summarize(n_doc = n())%>%  
  filter(n_doc>39)
```

In order to reduce the number of features we will only select words which are present in 2% of the documents (i.e. 40 documents).

We also saved the target in a separate document to not loose it.

Practicing supervised learning (7)

Finally, the word features will become columns (to be considered as variables):

```
# only select the words over the threshold
Tidy_Med <- inner_join(Tidy_Med, word)%>%
  select(-n_doc)

#make words into columns/features
Tidy_Med <- Tidy_Med %>%
  select(-c(n, tf, idf))%>%
  pivot_wider(names_from = word,
              values_from = tf_idf,
              values_fill=list(tf_idf = 0))

# add back the diagnosis column
Tidy_Med <- inner_join(diag, Tidy_Med)
```

Practicing supervised learning (8)

We can see now that we have each word being one column, which represent the features and tf-idf score as value. We will use these words as features to train the model...

```
head(Tidy_Med, n=6)
```

```
## # A tibble: 6 × 3,979
##       PMID diagnosis  data faecal  akt  imag  cell  pet  dose degree  alk
##       <int> <fct>      <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 25315432 N         0.0200      0      0      0 0.0102      0      0      0      0
## 2 25572476 N          0          0      0      0 0.0623      0      0      0      0
## 3 25590835 Y          0          0      0      0 0          0      0      0      0
## 4 25898986 N          0          0      0      0 0.0340      0      0      0      0
## 5 25830422 Y          0          0      0      0 0          0      0      0      0
## 6 26390591 N          0          0      0      0 0          0      0      0      0
## # ... with 3,968 more variables: solid <dbl>, load <dbl>, signific <dbl>,
## # year <dbl>, hpv <dbl>, patient <dbl>, mir <dbl>, plan <dbl>,
## # metabolit <dbl>, stat <dbl>, stent <dbl>, coupl <dbl>, valu <dbl>,
## # case <dbl>, biopsi <dbl>, group <dbl>, uptak <dbl>, adenoma <dbl>,
## # experi <dbl>, type <dbl>, weight <dbl>, recommend <dbl>, ros <dbl>,
## # biomark <dbl>, contour <dbl>, polyp <dbl>, screen <dbl>, tissu <dbl>,
## # diseas <dbl>, stage <dbl>, mirna <dbl>, intervent <dbl>, fdg <dbl>, ...
```

Practicing supervised learning (9)

To avoid problems with headers we are going to rename them, before inputting the features into the machine learning algorithm, but keeping the columns names of the PMID and the diagnosis.

```
# 5. Tidy up columns
```

```
Tidy_up <- Tidy_Med
```

```
names(Tidy_up)[3:ncol(Tidy_up)]<- as.character(seq(1, ncol(Tidy_up)-2, by=1))
```

```
colnames(Tidy_up) <- make.names(colnames(Tidy_up),unique = T)
```

```
Tidy_up$diagnosis <- as.factor(Tidy_up$diagnosis)
```

Practicing supervised learning (10)

Let's start to set up our learner:

```
library(mlr3)
library(mlr3learners)

# set up the classifier task using the right data and target
task = TaskClassif$new(id="PMID", backend=Tidy_up, target="diagnosis")

# define the algorithm to be used
learner = lrn("classif.rpart")
```

In *task* we define a new task, which is a classification task. If you want to target a number you would use *TaskRegr*.

Then you setup your learner, in which you decide the type of algorithm you would like to use.

Here we use a classification tree, but more are available.

Practicing supervised learning (11)

Let's set our learner with part of the data:

```
# 7. Train the model
```

```
# train a model of this learner for a subset of the task
```

```
learner$train(task, row_ids = 1:300)
```

```
# this is what the decision tree looks like
```

```
learner$model
```

```
## n= 300
```

```
##
```

```
## node), split, n, loss, yval, (yprob)
```

```
##      * denotes terminal node
```

```
##
```

```
## 1) root 300 142 Y (0.4733333 0.5266667)
```

```
##      2) X15< 0.01194675 207 84 N (0.5942029 0.4057971)
```

```
##      4) X36< 0.04466631 195 72 N (0.6307692 0.3692308)
```

```
##      8) X1697< 0.00562791 185 62 N (0.6648649 0.3351351)
```

```
##     16) X1097< 0.01387616 168 49 N (0.7083333 0.2916667)
```

```
##     32) X220< 0.0176174 160 42 N (0.7375000 0.2625000)
```

```
##     64) X1789< 0.007216191 150 34 N (0.7733333 0.2266667)
```

```
##    128) X970< 0.01890428 143 28 N (0.8041958 0.1958042) *
```

```
##    129) X970>=0.01890428 7 1 Y (0.1428571 0.8571429) *
```

```
##    65) X1789>=0.007216191 10 2 Y (0.2000000 0.8000000) *
```

```
##    33) X220>=0.0176174 8 1 Y (0.1250000 0.8750000) *
```

```
##    17) X1097>=0.01387616 17 1 Y (0.2252041 0.7747959) *
```

Practicing supervised learning (12)

Let's use this model to predict the diagnosis label of our test data:

8. Use the test set to see how the model is classifying the test data

```
predictions = learner$predict(task, row_ids = 301:350)
```

look at the first 10 rows of the predictions

```
head(as.data.table(predictions), n=10)
```

##	row_ids	truth	response
## 1:	301	Y	N
## 2:	302	N	N
## 3:	303	Y	N
## 4:	304	N	N
## 5:	305	Y	N
## 6:	306	Y	Y
## 7:	307	N	N
## 8:	308	Y	N
## 9:	309	Y	N
## 10:	310	Y	Y

Practicing supervised learning (13)

How good is our model?

```
# 9. accuracy of our model on the test set  
predictions$score(msr("classif.acc"))
```

```
## classif.acc  
##           0.64
```

```
# 10. look at the confusion matrix  
predictions$confusion
```

```
##           truth  
## response  N   Y  
##           N 14 15  
##           Y  3 18
```


Supervised learning

While the **accuracy** may give an indication about how good your model is, there are situations in which it may not be enough, the false positive (type I error), or the false negative (type II error) can be a good complement to it.

If you have an unbalanced dataset (90% of diagnostic publications), your classifier can guess diagnostic for all the papers, and it would be 90% accurate.

In some cases you would rather have a model which would minimise the false negative compared to the true positive. If you do some covid testing for choosing who should self isolate, you would rather make an error where you diagnose someone with covid, who is not really sick and ask them to stay at home. There may be cases in which you may want a less accurate model, but minimise one of the error type.

For more information on how to make more indicators with the confusion matrix, you can look at measures such as sensitivity, specificity, precision, recall ... (look at Han, Kamber, and Pei (2012), pp. 366-370)

Supervised learning - practice

Let's try to improve upon the model I have just started. Let's start working with the first exercise.

Unsupervised Learning

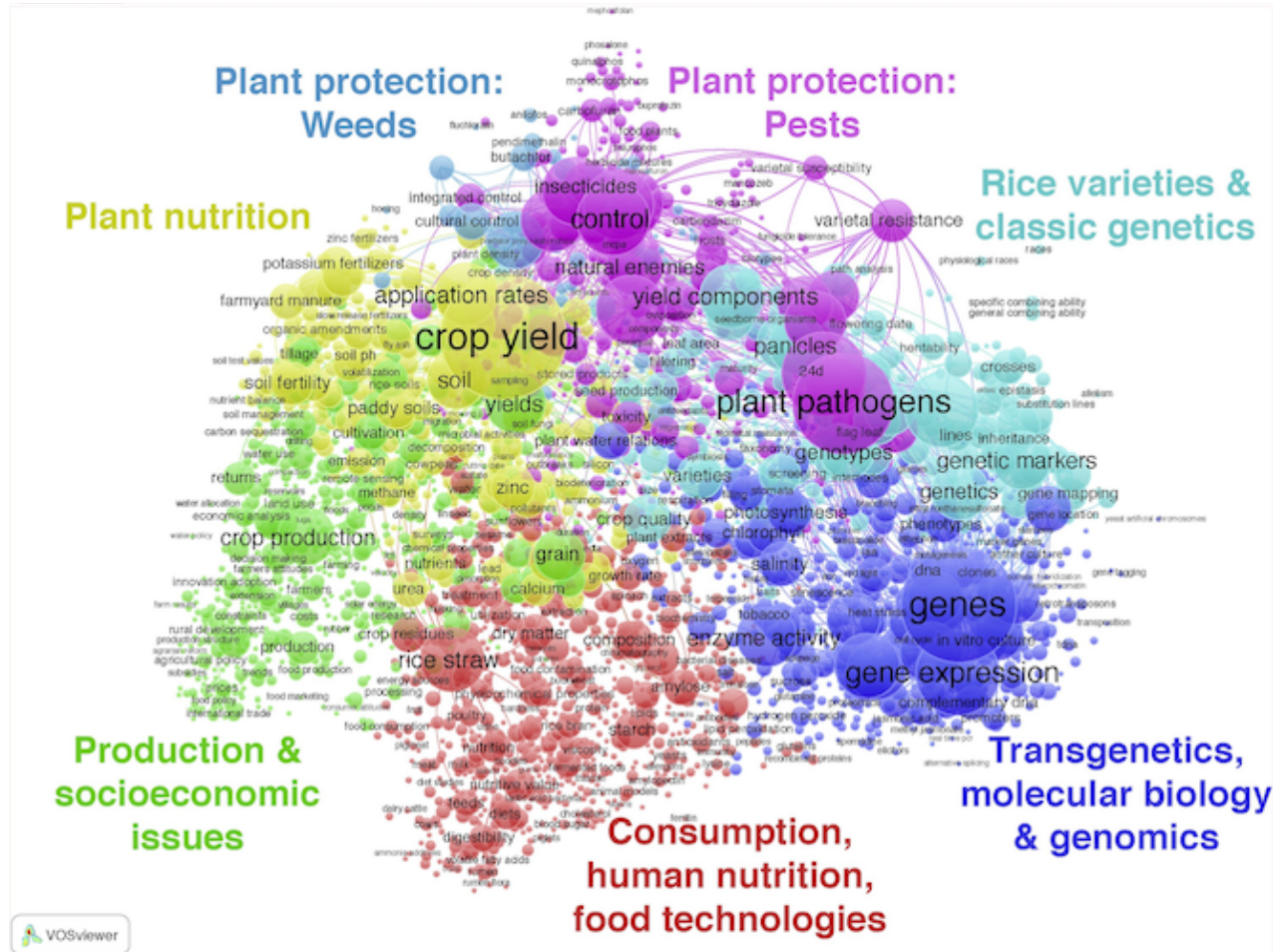
General principles

Unsupervised learning is used for more exploratory purposes, looking at features in the data, and group them into classes.

There is no given target, the algorithm finds similarity in the observations or underlying relevant features. The analyst may need to input some parameters, such as the number of clusters to be found, and usually needs to identify/label clusters after the algorithm has done the grouping.

You have already seen some examples of unsupervised learning in previous classes, such as topic models, or co-word analysis.

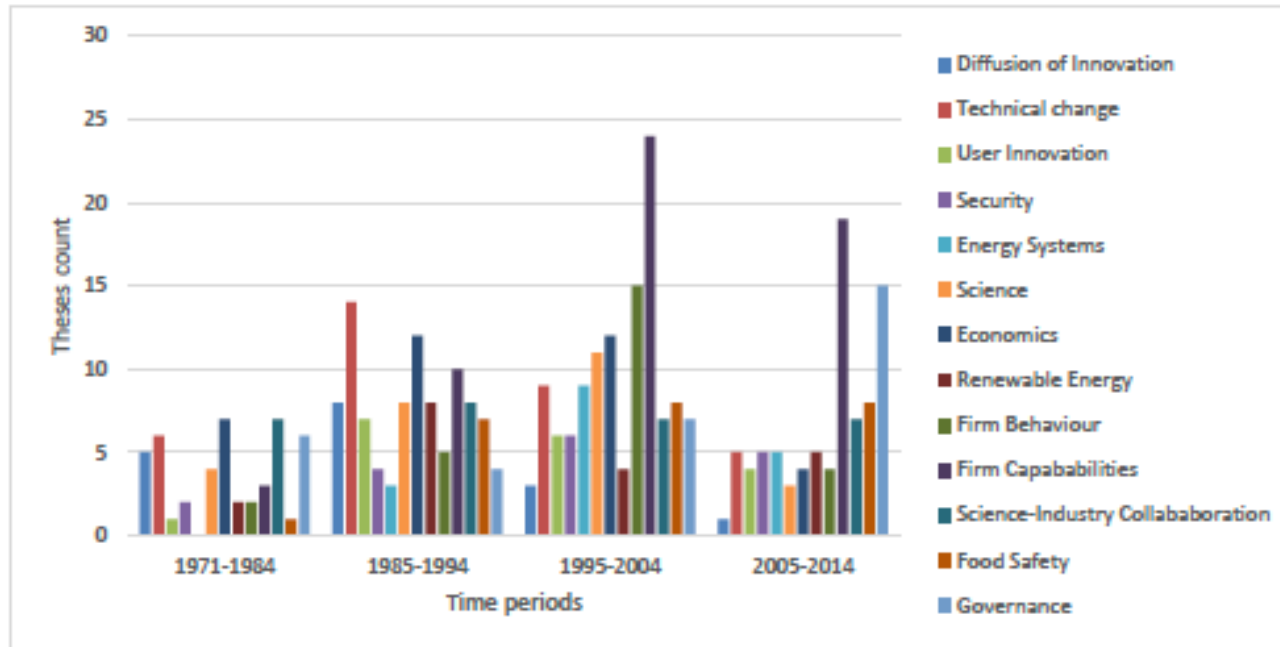
Supervised learning (co-word analysis)



Source: Ciarli and Ràfols (2019)

Examples of supervised learning (topic models)

Figure 7: Evolution of topics studied at SPRU over different time periods.



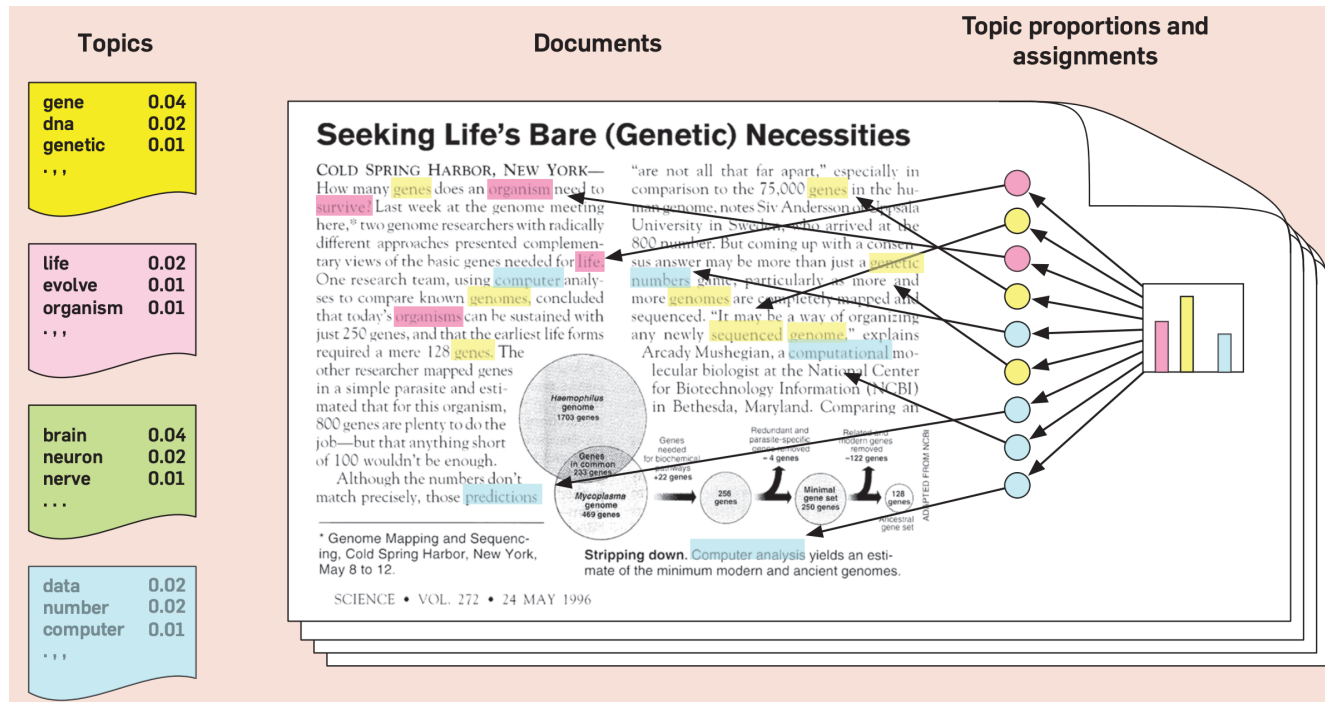
Source: A bibliometric Perspective on SPRU Research Activity: The PhD Data (Lang and Pujols 2016)

Using topic models

Topic models build on the assumption that there are multiple topics within each document; it builds on the assumption about statistical distribution of word /frequency within and between documents.

There are several algorithms to build topic models, LDA (Latent Dirichlet allocation), LSA or LSI (Latent Semantic Analysis), NMF (Non-Negative matrix factorisation) which are the most popular ones. They normally take normalized and vectorized text document as an input (e.g. Tf-Idf form), though you still have to enter parameters such as the number of topics to be computed from the variety of documents in use.

Using topic models



Source: Blei (2012) (blue = data analysis, pink = evolutionary biology, yellow = genetics)

Word embedding models

Word embedding models have become increasingly popular for text analysis. Its main use is to understand the relationship between words in a corpus. After training the model, you can explore words that are the most similar to a given word (i.e. synonyms), antonyms and much more.

Word embeddings became popular after the implementation of Mikolov et al. (2013) called *Word2Vec*. Together with *GloVe* developed by (Pennington, Socher, and Manning 2014), these are the two most popular word embedding algorithms used nowadays.

One of the main novelty of the algorithm is that it uses surrounding words of each token as labels to train the model. It is therefore ambiguous as to whether it is a unsupervised or supervised model.

Practicing GloVe with an example (1)

Using text data from SPRU publications (SPRU history project) gathered from the Web of Science, Scopus and SRO, I have pre-processed the corpus to integrate the title and abstract of each observation in one field, used lower cases, remove special character and punctuation, and merged the results in a dataset. Let's load the dataset and run a GloVe model on it using the *text2vec* package.

```
rm(list=ls())  
library(text2vec)  
library(tidyverse)
```

```
spru <- read_csv("SPRU_publi.csv")  
head(spru)
```

```
## # A tibble: 6 × 1  
##   text  
##   <chr>  
## 1 episodic innovation r d strategies for project based environments many busine...  
## 2 innovation in low tech industries na  
## 3 exploring the capital goods economy complex product systems in the uk over th...  
## 4 governance and co ordination of distributed innovation processes patterns of ...  
## 5 what do we know about innovation in this editorial article we introduce this ...  
## 6 world sustainable development outlook west meet east sharing the past and cur...
```

Practicing GloVe with an example (2)

First, we need to transform the text into tokens. Then, we are going to make a vocabulary using all the tokens that has been identified from the corpus. We will also remove the rarer terms, as these won't have enough training data to be useful for the model.

```
# Create iterator over tokens
tokens <- space_tokenizer(spru$text)
# Create vocabulary. Terms will be unigrams (simple words).
it = itoken(tokens, progressbar = FALSE)
vocab <- create_vocabulary(it)
# Remove tokens used less than 5 times
vocab <- prune_vocabulary(vocab, term_count_min = 5L)
```

Practicing GloVe with an example (3)

In the next step, we first vectorise the vocabulary, which then will be used to construct the term-co-occurrence matrix (tcm), using the skip-gram model. For this model each word is represented using the five words before and the five words after each given token.

```
# Use our filtered vocabulary  
vectorizer <- vocab_vectorizer(vocab)  
# use window of 5 for context words  
tcm <- create_tcm(it, vectorizer, skip_grams_window = 5L)
```

Practicing GloVe with an example (4)

Let's create a model based on 50 features and use 10 iterations to improve the model. Note that the model is based on two matrices, the main matrix and the context matrix.

```
glove = GlobalVectors$new(rank = 50, x_max = 10)
wv_main = glove$fit_transform(tcm, n_iter = 10)
```

```
## INFO [19:35:12.382] epoch 1, loss 0.2569
## INFO [19:35:13.576] epoch 2, loss 0.1713
## INFO [19:35:15.471] epoch 3, loss 0.1447
## INFO [19:35:16.713] epoch 4, loss 0.1290
## INFO [19:35:17.687] epoch 5, loss 0.1179
## INFO [19:35:18.643] epoch 6, loss 0.1096
## INFO [19:35:19.505] epoch 7, loss 0.1030
## INFO [19:35:20.310] epoch 8, loss 0.0977
## INFO [19:35:21.246] epoch 9, loss 0.0934
## INFO [19:35:22.219] epoch 10, loss 0.0897
```

```
wv_context = glove$components
word_vectors = wv_main + t(wv_context)
```

rank: number of features in the model x_max: maximum number of co-occurrences to use in the matrix n_iter: number of iteration to optimise the model

Practicing GloVe with an example (4)

Let's try our model out on two words and what words would be the most similar to them. Let's take the word *firm*:

```
firm = word_vectors["firm", , drop = FALSE]
cos_sim = sim2(x = word_vectors, y = firm, method = "cosine", norm = "l2")
head(sort(cos_sim[,1], decreasing = TRUE), 5)
```

```
##          firm      growth performance      specific      level
##  1.0000000  0.6151884  0.6128337  0.6075509  0.5843449
```

Other techniques

Clustering techniques

Clustering techniques are more traditional unsupervised learning techniques (for any types of problems), a little less used for text mining than the above.

Clustering is the process of partitioning a set of observations into subsets (i.e. classes or clusters). In each cluster, the observations are similar to one another, while dissimilar to observation in other clusters. (Han, Kamber, and Pei 2012, 444)

To do clustering analysis, you need to evaluate how similar observations are in your data, and then having an algorithm (i.e. learner), which will group them into cluster; different learners may bring different clustering. To prepare data for the learner you may want to run similarity measures on the observations (e.g. cosine similarity).

Clustering techniques

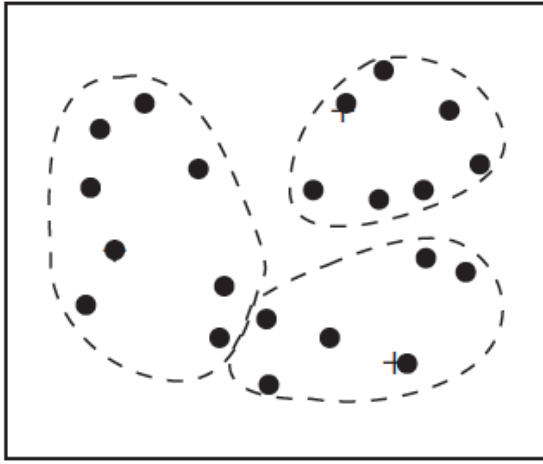
There are three main types of clustering:

Method	General Characteristics
Partitioning methods	<ul style="list-style-type: none">– Find mutually exclusive clusters of spherical shape– Distance-based– May use mean or medoid (etc.) to represent cluster center– Effective for small- to medium-size data sets
Hierarchical methods	<ul style="list-style-type: none">– Clustering is a hierarchical decomposition (i.e., multiple levels)– Cannot correct erroneous merges or splits– May incorporate other techniques like microclustering or consider object “linkages”
Density-based methods	<ul style="list-style-type: none">– Can find arbitrarily shaped clusters– Clusters are dense regions of objects in space that are separated by low-density regions– Cluster density: Each point must have a minimum number of points within its “neighborhood”– May filter out outliers

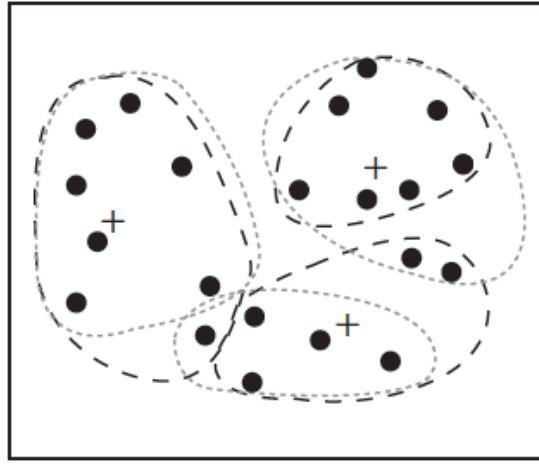
Source: Han, Kamber, and Pei (2012)

Clustering techniques

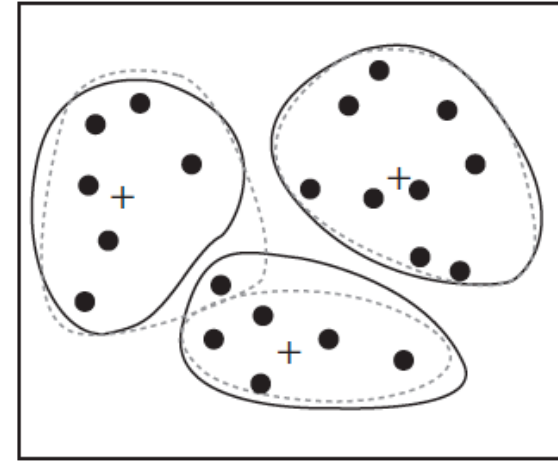
For the partitioning method, a popular example is *k-Means*.



(a) Initial clustering



(b) Iterate

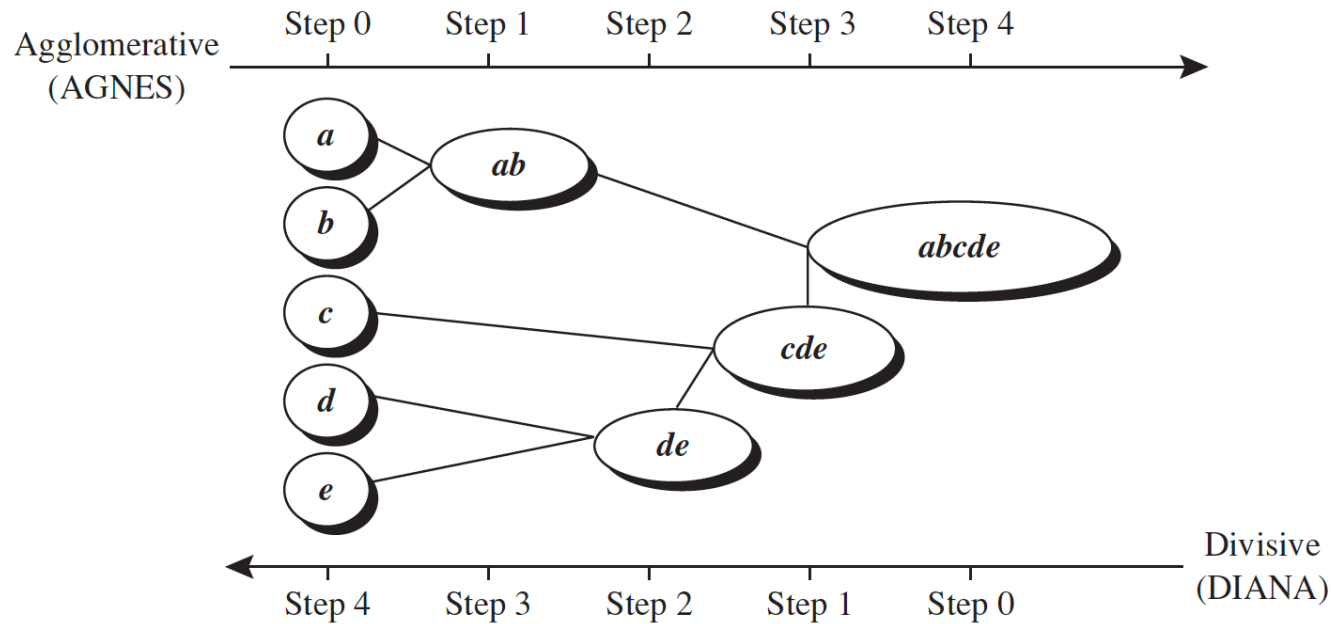


(c) Final clustering

Source: Han, Kamber, and Pei (2012)

Clustering techniques

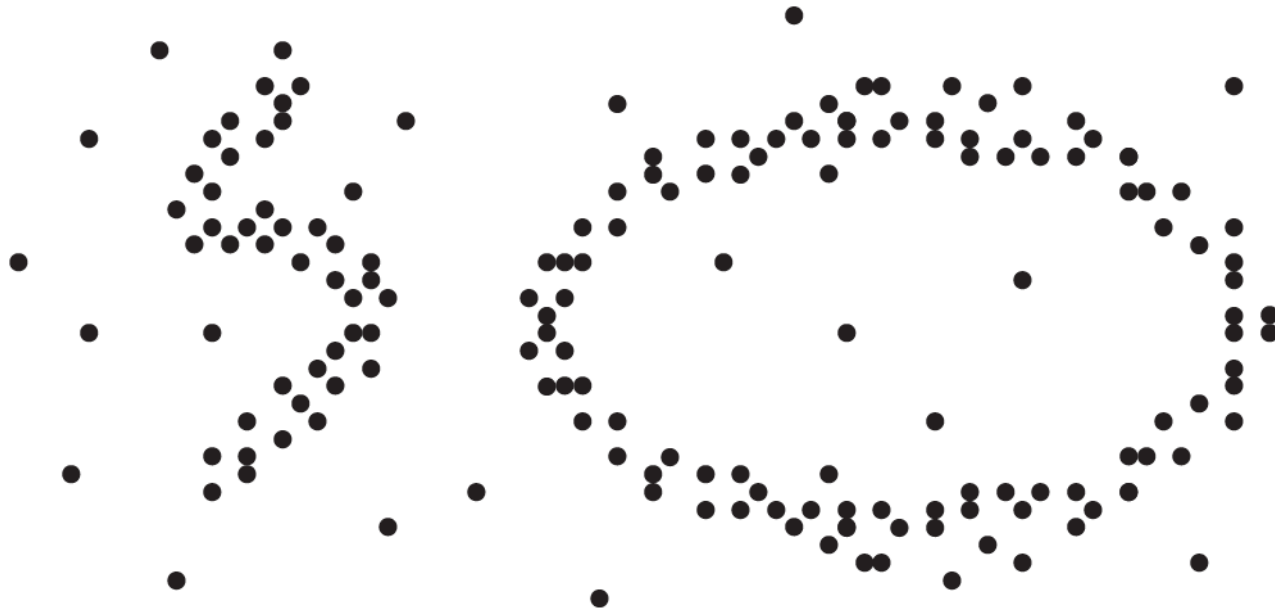
For the hierarchical method, a popular examples are *AGglomerative NESTing* (AGNES) or *Divisive ANALysis* (DIANA).



Source: Han, Kamber, and Pei (2012)

Clustering techniques

For the **density based method**, a popular examples are *Density-Based Spatial Clustering of Applications with Noise* (DBSCAN).



Source: Han, Kamber, and Pei (2012)

Unsupervised learning

There are many other methods within clustering techniques to identify clusters using statistical, grid based methods, and many more algorithms.

The above just provides you with an overview of what is feasible.

Ethical considerations

Ethical considerations

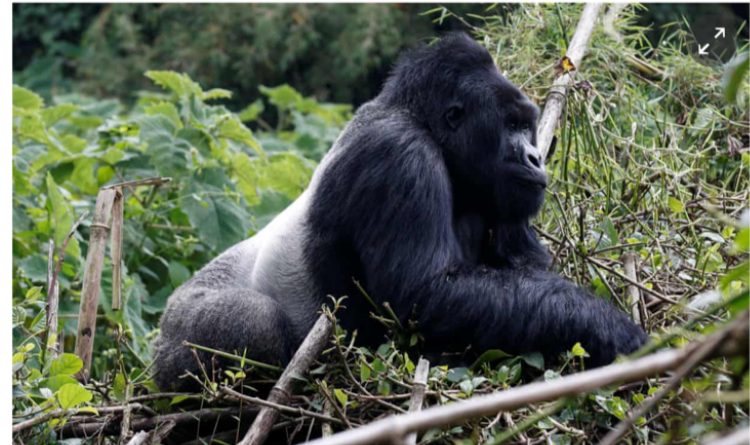
Training data

The model that you developed is as good as your training data. The training data can overrepresent or underrepresent certain categories / characteristics. Thus the resulting model when implemented may not work well for these underrepresented examples.

One example was the automatic labelling of objects in pictures.

Google's solution to accidental algorithmic racism: ban gorillas

Google's 'immediate action' over AI labelling of black people as gorillas was simply to block the word, along with chimpanzee and monkey, reports suggest



▲ A silverback high mountain gorilla, which you'll no longer be able to label satisfactorily on Google Photos.
Photograph: Thomas Mukoya/Reuters

Ethical considerations

Training data

You may include sensitive or data about protected characteristics of individuals. Even if you take care of not including these, other characteristics may be highly correlated with protected characteristics and become discriminatory.

Machine learning may also be rolled out on a variety of uses over time; so the data used to build a model may be fit for purpose at the time of development, but may not be representative of new cases that you encounter and want to generalise from.

Ethical considerations

Accountability

Using machine learning to make decision may raise the question of accountability and transparency, how did we reach the decision, what particularly lead to the decision or the outcome?

Some popular machine learning algorithms, such as neural networks / deep learning, don't have a transparent model and hence it may be difficult to understand 'why' it has recommended a specific decision.

Questions

References

References

Blei, David M. 2012. "Probabilistic topic models." In *Communications of the Acm.* <https://doi.org/10.1145/2133806.2133826>.

Ciarli, Tommaso, and Ismael Ràfols. 2019. "The relation between research priorities and societal demands: The case of rice." *Research Policy* 48 (4): 949–67. <https://doi.org/https://doi.org/10.1016/j.respol.2018.10.027>.

Domingos, Pedro. 2015. "The Master Algorithm." In *Basic Books*.

Géron, Aurelien. 2019. *Hands-on machine learning with Scikit-Learn and TensorFlow: concepts, tools, and techniques to build intelligent systems*.

Han, Jiawei., Micheline. Kamber, and Jian Pei. 2012. *Data mining : concepts and techniques*. Elsevier/Morgan Kaufmann.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. 1-2. MIT Press. <https://doi.org/10.1007/s10710-017-9314-z>.

Lang, Frédérique, and Jane Pujols. 2016. "A SPRU history based on bibliometric analysis of the studies of SPRU PhD students." Brighton: SPRU, Business School, University of Sussex.

Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Efficient estimation of word representations in vector space." In *1st International Conference on Learning Representations, Iclr 2013 - Workshop Track Proceedings*. International Conference on Learning Representations, ICLR.

Pennington, Jeffrey, Richard Socher, and Christopher D Manning. 2014. "GloVe: Global Vectors for Word Representation." In, 1532–43. <http://nlp>.

Rhys, Hefin Ioan. 2020. *MACHINE LEARNING WITH R, TIDYVERSE, AND MLR*. O'REILLY MEDIA.