

Unsupervised Learning

Questions:

1. How is unsupervised learning related to the statistical clustering problem?

Unsupervised machine learning is a type of machine learning algorithms used to draw inferences from datasets consisting of unlabeled input data without corresponding output data. The goal of unsupervised learning is to find the underlying structure (clustering) or distribution (density estimation) in the data. There are two types of problems that are commonly solved using unsupervised learning – clustering and association. Clustering aims to discover inherent groupings in the data, when association tries to discover rules that describe large portions of the data (ex., customers who buy X also tend to buy Y).

So, clustering is considered to be the most important task of unsupervised learning. In its broad sense, clustering is a process of organizing objects into groups whose members are similar in some way. So, a cluster is a collection of objects that are simultaneously similar between themselves and dissimilar to objects from other clusters. As a result, clustering learning algorithms measure/minimize internal (within the cluster) distances between the datapoints and measure/maximize external (intra-cluster) distances.

There are many different clustering techniques that can be divided into three groups:

- Hierarchical algorithms find successive clusters using previously established clusters using agglomerative (“bottom-up”) or divisive (“top-down”) approach.
- Partitional algorithms determine all clusters at the same time. This group include centroid (ex., K-means), model based, graph theoretic and spectral algorithms.
- Bayesian algorithms try to generate a posterior distribution over the collection of all partitions of the data. Bayesian group includes decision based and non-parametric algorithms.

2. What packages (in R, Python...) perform unsupervised learning?

There are several packages that facilitate unsupervised machine learning in R. Some of them attempt to include more diverse set of tools, and some of them are focused on a particular algorithm of a data type. Below are just a few examples:

- **cluster** package includes several methods for cluster analysis;
- **fpc** package another wrapper package including various methods for clustering among others fixed point clustering, linear regression clustering, clustering by merging Gaussian mixture components, etc.) and cluster validation;
- **HDclust** – clustering of high dimensional data;
- **Pvclust** – hierarchical clustering with p-values via multiscale bootstrap resampling;
- **mclust** – Gaussian mixture modelling for model based clustering, classification, and density estimation;
- **TSclust** - a set of measures of dissimilarity between time series to perform time series clustering;

Unsupervised Learning

- **clustMD** – model based clustering of mixed data using a parsimonious mixture of latent Gaussian variable models;
- **Clusteval** – a suite of tools to evaluate clustering algorithms, clustering and individual clusters;
- **clusterGenomics** – an example of an application-specific package – used for identifying clusters in genomic data using recursive partitioning.

This list is by no means all-inclusive as there are multiple other packages that can be used for unsupervised learning in R.

3. What measures of quality of the learning algorithm might you expect to see?

Theoretically, the main measure of quality of a learning algorithm is its accuracy of predicted results. In practice however, in case of unsupervised machine learning we are often trying to infer a function that describes the structure of unlabeled data that has not been classified or categorized yet. Often, clustering and other unsupervised learning is performed as part of exploratory data analysis when researchers are not sure yet what exactly they are looking for, and do not have any clear ideas of what they expect to find. So, we do not have a “reference level” to which to compare the algorithm’s results, therefore no straightforward way to evaluate the accuracy of the structure produced by the algorithm. One possible way to find if results are meaningful is to use expert opinion (external evaluation of the results), or, if feasible, to define an objective function on clustering (internal evaluation). It is also helpful to compare results of multiple runs of the same algorithm using different parameters (ex., number of classes), or results of applying different algorithms.

Since in our K-means clustering exercise we used the iris dataset that contains labels (species), then in this particular case we are able to evaluate accuracy of the algorithm by comparing clustering results to the actual species.

```
> #Compare the clusters with the species and plot results  
> table(km$cluster, iris$Species)
```

	setosa	versicolor	virginica
1	0	2	36
2	0	48	14
3	50	0	0

The above table shows that two of virginica samples were incorrectly put in to a cluster for versicolor, and 14 of versicolor samples were incorrectly clustered with virginica species. All samples for setosa were classified correctly. So, overall there were 124 correct classifications out of 150 samples, which constitutes 82.67% accuracy. However, for virginica accuracy was 96.0% and for versicolor only 72.0%

After working through the k-means clustering exercise using the Iris dataset, I was wondering about the comparative accuracy of supervised and non-supervised learning models. In my supervised learning assignment, I used Naïve Bayes algorithm to classify text messages into spam and ham category, and it turned out to be highly accurate. So, I decided to use a k-means algorithm on the same dataset for

Unsupervised Learning

comparison reasons. Since K-means algorithm is not commonly used for building spam filters, I was not able to find any guidelines/tutorials for this case. So, I adjusted my code from the previous exercise to accommodate a different algorithm. For comparison reasons I followed the steps from my previous assignments as close as possible.

First, as in previous assignment, I prepared the environment and loaded the libraries Using the following code:

```
> # Unsupervised Learning
>
> rm(list=ls()) #Clear the environment
> setwd("YOUR_PATH") #Set working directory for the assignment
> getwd() #Check working directory
[1] "YOUR_PATH"
> #####
> #####Using Supervised Learning for Spam Detection#####
> #####Based on A Gentle Introduction to Data Classification with R#####
> #####https://blog.paperspace.com/intro-to-datascience/#####
> #####
>
>
> ###Load packages
> library(quanteda)
```

For comparison reasons I used quanteda library again to construct a corpus object and perform text manipulations.

I loaded the data into a table:

```
> ###Load SMS SPAM COLLECTION
> ###https://archive.ics.uci.edu/ml/datasets/SMS+Spam+Collection
>
> #Load Data into a table
> raw.data <- read.table("SMSSpamCollection", header = FALSE, sep="\t", quote="", stringsAsFactors = FALSE)
```

And checked to make sure that the data was loaded correctly:

```
> #Add column names
> names(raw.data) <- c("Label", "Text")
>
> #Check the structure and content of the data table
> summary(raw.data)
  Label      Text
Length:5574 Length:5574
Class :character Class :character
Mode :character  Mode :character
> table(raw.data$Label)
```

```
ham spam
4827  747
```

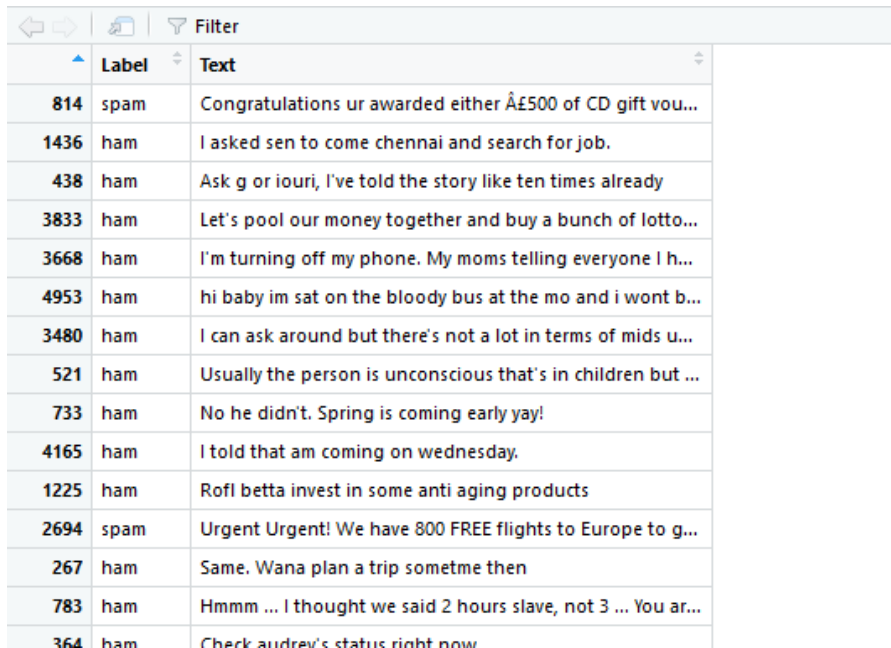
The initial data set contains 5574 messages (4827 ham messages and 747 spam messages).

Unsupervised Learning

As in previous exercise, I used `sample()` command to randomize the raw data and checked the results using `View()`:

```
> #Randomizse data using sample() command
> set.seed(1912)
> raw.data <- raw.data[sample(nrow(raw.data)),]
> View(raw.data) #Check data
```

The screen shot of the output is below:



	Label	Text
814	spam	Congratulations ur awarded either Â£500 of CD gift vou...
1436	ham	I asked sen to come chennai and search for job.
438	ham	Ask g or iouri, I've told the story like ten times already
3833	ham	Let's pool our money together and buy a bunch of lotto...
3668	ham	I'm turning off my phone. My moms telling everyone I h...
4953	ham	hi baby im sat on the bloody bus at the mo and i wont b...
3480	ham	I can ask around but there's not a lot in terms of mids u...
521	ham	Usually the person is unconscious that's in children but ...
733	ham	No he didn't. Spring is coming early yay!
4165	ham	I told that am coming on wednesday.
1225	ham	Rofl betta invest in some anti aging products
2694	spam	Urgent Urgent! We have 800 FREE flights to Europe to g...
267	ham	Same. Wana plan a trip sometme then
783	ham	Hmmm ... I thought we said 2 hours slave, not 3 ... You ar...
364	ham	Check audrey's status right now

Then, I constructed the corpus object using the text field from the raw data:

```
> ###Construct corpus object
> sms.corpus <- corpus(raw.data$Text) #Construct corpus using text field
>
```

I only minimally pre-processed the text (convert to lower case), keeping all characteristic marks of text messages (abundant use of punctuation etc.), used `dfm()` command to load data into a document term frequency matrix:

```
> ####Pre-processing sms data, wieght word counts
> sms.dfm <- dfm(sms.corpus, tolower = TRUE) #convert to lower case
> sms.dfm <-dfm_trim(sms.dfm, min_docfreq = 3)
> sms.dfm <- dfm_weight(sms.dfm)
```

Unsupervised Learning

Since I was not training the algorithm on any labelled data, there was no need to split the dataset into two groups (training and testing), so I proceeded to fitting the model and output results using the following code:

```
> sms.cluster <- kmeans(sms.dfm, 2) # Chose 2 clusters( for spam and ham)
> sms.cluster$size # Output the number of objects in each cluster
[1] 857 4717
```

I used 2 for number of clusters, as I was interested in spam/ham classification.

```
> summary(raw.data)
  Label      Text
Length:5574 Length:5574
Class :character Class :character
Mode :character  Mode :character
> table(raw.data$Label)

ham spam
4827 747
>
> #Randomize data using sample() command
> set.seed(1912)
> raw.data <- raw.data[sample(nrow(raw.data)),]
> view(raw.data) #check data
> ###Construct corpus object
> sms.corpus <- corpus(raw.data$Text) #Construct corpus using text field
>
> ####Pre-processing sms data, wieght word counts
> sms.dfm <- dfm(sms.corpus, tolower = TRUE) #convert to lower case
> sms.dfm <- dfm_trim(sms.dfm, min_docfreq = 3)
> sms.dfm <- dfm_weight(sms.dfm)
> ###Perform K-means clustering
>
> sms.cluster <- kmeans(sms.dfm, 2) # Chose 2 clusters( for spam and ham)
> sms.cluster$size # Coutput the number of objects in each cluster
[1] 857 4717
\
```

Conclusions:

- The above results demonstrate that kmeans() was able to subdivide the dataset into two clusters containing 857 and 4717 text messages. The data set actually contained 747 spam and 4827 ham messages, for the algorithm considerably overestimated the number of spam messages. Since in real life situations there are no “spam” and “ham” labels available before hand, I did make detailed comparisons for each groups, and just concluded the k-means algorithm yielded less accurate results than the Naïve Bayes algorithm.
- I was experimenting with the R package which is new for me (quanteda), and it is possible that I have not used all its features to the maximum, which might have influenced my analysis results.