Learning out of data collections

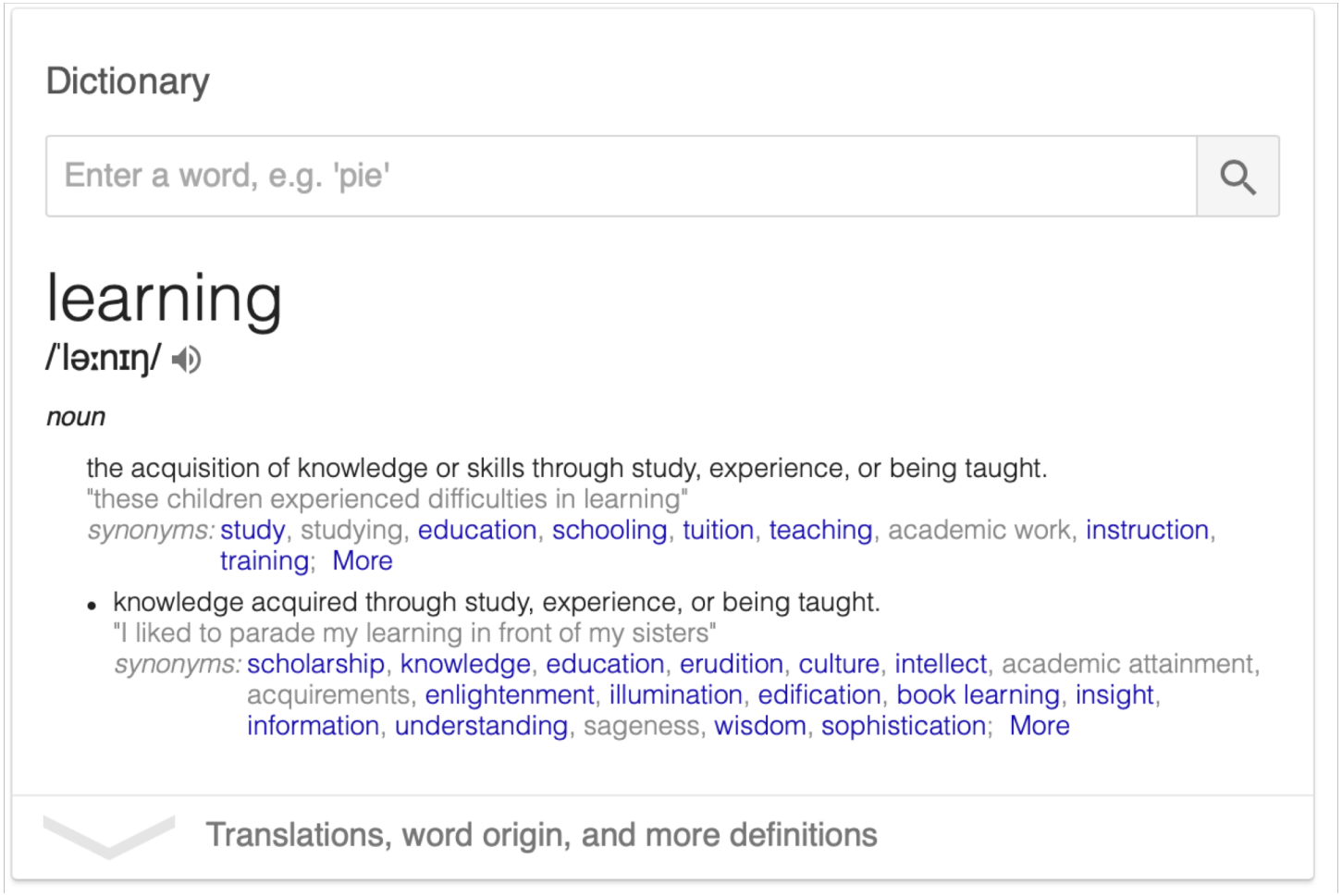
Unsupervised Learning

Genoveva Vargas-Solar
French Council of Scientific Research, LIG
genoveva.vargas@imag.fr

http://vargas-solar.com/data-centric-smart-everything/

* This presentation was created using the content of L. Igual and S. Seguí, Introduction to Data Science: A Python approach to concepts, techniques and applications, Undergraduate topics in Computer Science Series, Springer, 2017

WHAT IS LEARNING?



Dictionary

Enter a word, e.g. 'pie'

learning

/ˈləːnɪŋ/

noun

the acquisition of knowledge or skills through study, experience, or being taught.
"these children experienced difficulties in learning"
synonyms: study, studying, education, schooling, tuition, teaching, academic work, instruction, training; More

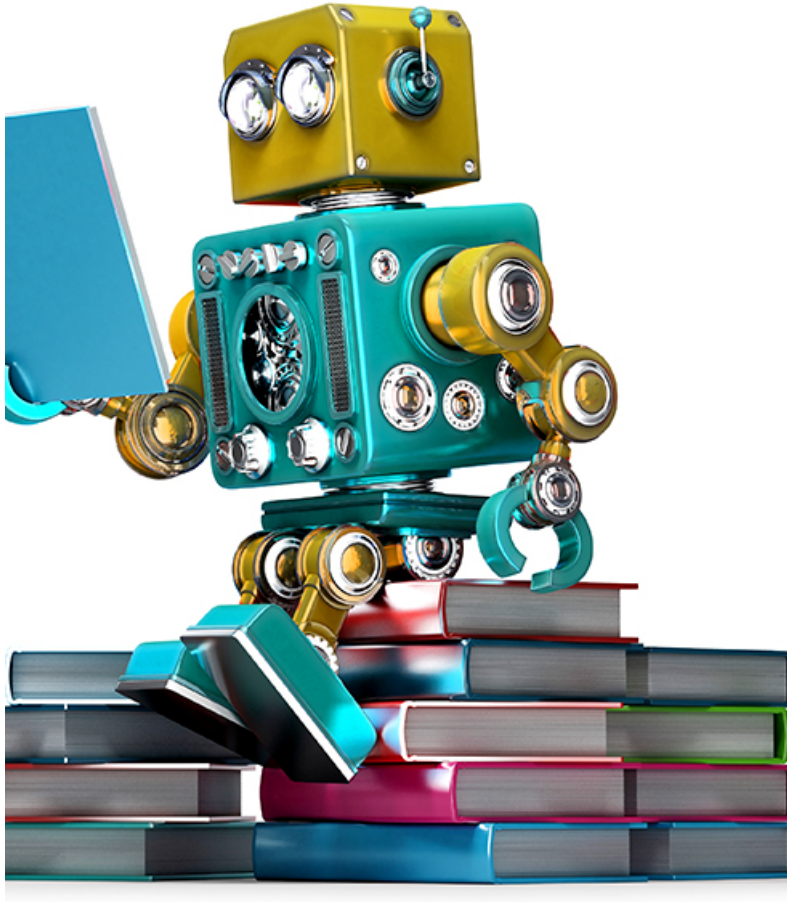
- knowledge acquired through study, experience, or being taught.
  "I liked to parade my learning in front of my sisters"
  synonyms: scholarship, knowledge, education, erudition, culture, intellect, academic attainment, acquirements, enlightenment, illumination, edification, book learning, insight, information, understanding, sageness, wisdom, sophistication; More

Translations, word origin, and more definitions

MACHINE LEARNING

- Coding programs that **automatically adjust their performance** in accordance to their **exposure to information in data**

- Learning achieved via a **parametrized model with tunable parameters** that are automatically **adjusted** according to **different performance criteria**



MACHINE LEARNING

- **Supervised learning**: algorithms that learn from a training set of labeled examples to generalize to the set of all possible inputs
  - Logistic regression, support vector machines, decision trees, random forest, etc

- **Unsupervised learning**: algorithms learn from a training set of unlabeled examples
  - Explore data according to some statistical, geometric or similarity criterion
  - K-means clustering, kernel density estimation

- **Reinforcement learning**: algorithms that learn via reinforcement from criticism that provides information on the quality of a solution, but not on how to improve it.
  - Improved solutions are achieved by iteratively exploring the solution space.

UNSUPERVISED LEARNING

Objective: Find hidden structure in unlabelled data

- Algorithms that learn from a training set of unlabeled or unannotated examples
  - using the features of the inputs to categorize them
  - according to some geometric or statistical criteria

Problems tackled

- **Clustering**: partition the set of examples into groups
- **Dimensionality reduction**: reduce the dimensionality of data
  - Principal Component Analysis, Independent component analysis, non-negative matrix factorization
- **Outlier detection**: find unusual events that distinguish part of the data from the rest according to certain criteria
- **Novelty criteria**: deals with cases wen changes occur in the data

CLUSTERING

- Clustering is a process of grouping similar objects together i.e., to partition unlabeled examples into disjoint subsets of clusters, such that:
  - Examples within a cluster are similar: ***high intraclass similarity***
  - Examples in different clusters are different ***low interclass similarity***

How to measure for similarity/dissimilarity? Two kinds of inputs can be used for grouping:

(a) in *similarity-based clustering*, the input to the algorithm is an $n \times n$ *dissimilarity matrix* or *distance matrix*;

(b) in *feature-based clustering*, the input to the algorithm is an $n \times D$ *feature matrix* or *design matrix*, where
  - n is the number of examples in the dataset
  - D the dimensionality of each sample

ASPECTS TO CONSIDER

- What is a natural grouping among the objects?

  → Need to define the "**groupness**" and the "**similarity/distance**" among data

- How can we group samples?
  - What are the best procedures?
  - Are they efficient? Are they fast? Are they deterministic?

How many clusters should we look for in the data?
- Shall we state this number a priori?
- Should the process be completely data driven or can the user guide the grouping process?
- How can we avoid "trivial" clusters?
- Should we allow final clustering results to have very large or very small clusters?
- Which methods work when the number of samples is large? Which methods work when the number of classes is large?

What constitutes a good grouping?
- What objective measures can be defined to evaluate the quality of the clusters?

DISTANCES

The most widespread distance metric is the *Minkowski distance*:

$$d(a,b) = (\sum_{i=1}^{d} |a_i - b_i|^p)^{1/p}$$

- `d(a, b)` stands for the distance between two elements $a, b \in \mathbb{R}^d$
- `d` is the dimensionality of the data
- `p` is a parameter

The best-known instantiations of this metric are as follows:

- `p = 2`: *Euclidean distance*
- `p = 1`: *Manhattan distance*, and
- `p = inf`: *max-distance* corresponding to the component $|a_i - b_i|$ with the highest value

MEASURING CLUSTERING QUALITY

Two families of techniques:

- those that allow to compare clustering techniques
  → **Rand Index, Homogeneity, Completeness & V-measure Scores**
- those that check on specific properties of the clustering, e.g., "compactness"
  → **Silhouette Score**

Rand index evaluates the similarity between two results of data clustering

Silhouette coefficient: evaluates the compactness of the results of applying a clustering approach

RAND INDEX

Given a set of n elements $S = \{o_1, \ldots, o_n\}$, we can compare two partitions of S :

- $X = \{X_1, \ldots, X_r\}$, a partition of S into r subsets
- $Y = \{Y_1, \ldots, Y_s\}$, a partition of S into s subsets

Consider the following defintions:

- a is the number of pairs of elements in S that are in the **same** subset in **both** X and Y
- b is the number of pairs of elements in S that are in **different** subsets in **both** X and Y
- c is the number of pairs of elements in S that are in the **same** subset in X but in **different** subsets in Y
- d is the number of pairs of elements in S that are in **different** subsets in X , but in the **same** subset in Y

$$R = \frac{a+b}{a+b+c+d},$$

R: [0,1]

ADJUSTED RAND INDEX

Adjusted Rand index, adjusts the Rand index with respect to random grouping of elements

$$AR = \frac{\binom{n}{2}(a+d) - [(a+b)(a+c)+(c+d)(b+d)]}{\binom{n}{2}^2 [(a+b)(a+c)+(c+d)(b+d)]}.$$

HOMOGENEITY

- **Homogeneity** if all of its clusters contain only data points which are members of the same original (single) class.

```
In [1]: print("%.3f" % metrics.homogeneity_score([0, 0, 1, 1],
                                                  [0, 0, 0, 0]))
```

Original

Predicted

4 samples

2 labels: 0, 1

Homogeneity is 0 since the samples in the predicted cluster 0 come from original cluster 0 and cluster 1

COMPLETENESS

Completeness if all the data points that are members of a given class are elements of the same predicted cluster

Original

```
print metrics.completeness_score([0, 0, 1, 1],
                                 [1, 1, 0, 0])
```

Predicted

4 samples

2 labels: 0, 1

Completeness is 1 since

- all the samples from the original cluster with label 0 go into the same predicted cluster with label 1, and
- all the samples from the original cluster with label 1 go into the same predicted cluster with label 0

Both scores have real positive values between 0.0 & 1.0

larger values being desirable

How can we define a measure that takes into account the completeness & the homogeneity?

V-MEASURE

Harmonic mean between the homogeneity and the completeness

$$v = 2 * (\text{homogeneity} * \text{completeness}) / (\text{homogeneity} + \text{completeness}).$$

-0.0 means the clustering is extremely bad

-1.0 indicates a perfect clustering result

-

```
]:  print("%.3f" % metrics.v_measure_score([0, 0, 0, 0],
                                           [0, 1, 2, 3]))
```

If class members are completely split across different clusters, the assignment is totally incomplete

→V-measure is null

```
print("%.3f" % metrics.v_measure_score([0, 0, 1, 1],
                                       [0, 0, 0, 0]))
```

Clusters that include samples from different classes destroy the homogeneity of the labeling

→V-measure is null

SILHOUTTE SCORE

Evaluate the final 'shape' of the clustering result

$$Silhouette(i) = \frac{b - a}{\max(a, b)}.$$

Function of the intracluster distance of a

-sample in the dataset *a* and the nearest cluster distance, *b* for each sample

if Silhouette $s(i)$ is

- close to 0 the sample is on the border of its cluster and the closest one from the rest of the dataset clusters
- A negative value means that the sample is closer to the neighbour cluster

- **The average of the Silhouette coefficients of all samples** of a given cluster defines the "goodness" of the cluster. A high positive value, i.e., close to 1 would mean a compact cluster, and vice versa.

- **The average of the Silhouette coefficients of all clusters** gives idea of the quality of the clustering result

CLUSTERING TAXONOMY

- **Soft partition** algorithms assign a probability of the data belonging to each cluster

- **Hard partition** algorithms, where each data point is assigned precise membership of one cluster

According to the grouping process of the hard partition algorithm, there are two large families of clustering techniques:

- *Partitional algorithms*: start with a random partition and refine it iteratively ("flat" clustering)

- *Hierarchical algorithms*: organize the data into hierarchical structures, data can be agglomerated in a bottom-up direction, or split in a top-down manner

K-MEANS CLUSTERING (1/2)

Hard partition algorithm assigning each data point to a single cluster

- Divides a set of n samples X into k disjoint clusters c_i, $i = 1, \ldots, k$
- each described by the mean μ_i of the samples in the cluster
- Means are called cluster **centroids**

Solves the following minimization problem:

$$\arg\min_c \sum_{j=1}^{k} \sum_{x \in c_j} d(x, \mu_j) = \arg\min_c \sum_{j=1}^{k} \sum_{x \in c_j} ||x - \mu_j||_2^2$$

where c_i is the set of points that belong to cluster i and μ_i is the center of the class c_i.

K-means clustering objective function uses the square of the **Euclidean distance** or **inertia** or **within-cluster sum- of-squares**

K-MEANS CLUSTERING (2/2)

This problem is not trivial to solve (in fact, it is NP-hard problem)

The algorithm only hopes to find the global minimum, but may become stuck at a different solution

Should centroids belong to the original set of points?

$$inertia = \sum_{i=0}^{n} min_{\mu_j \in c}(||x_i - \mu_j||^2)).$$

K-MEANS ALGORITHM

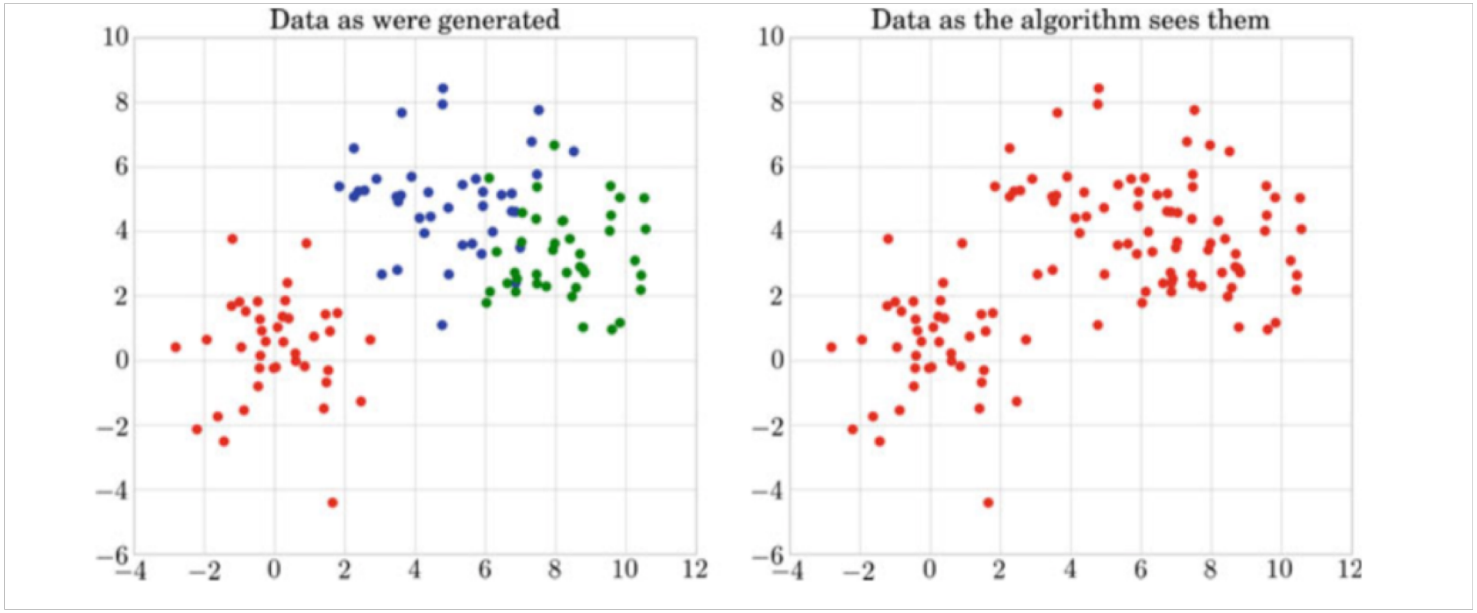
First, decide the number of clusters, k.

1. Initialize (e.g., randomly) the k cluster **centroids**
2. Decide the class memberships of the n data samples by assigning them to the nearest-cluster centroids (e.g., the center of gravity or mean).
3. Re-estimate the k cluster centers, ci, by assuming the memberships found above are correct.
4. If none of the n objects changes its membership from the last iteration, then exit Otherwise go to step 2.

RUNNING THE K-MEANS ALGORITHM (1/3)

First, we will create three sample distributions:

```
MAXN = 40
X = np.concatenate([
    1.25*np.random.randn(MAXN, 2),
    5 + 1.5*np.random.randn(MAXN, 2)])
X = np.concatenate([
    X, [8, 3] + 1.2*np.random.randn(MAXN, 2)])
```



RUNNING THE K-MEANS ALGORITHM (2/3)

Let us assume that we expect to have three clusters ($k = 3$) and apply the K-means command from the Scikit-learn library:

```
from sklearn import cluster

K = 3 # Assuming we have 3 clusters!
clf = cluster.KMeans(init = 'random', n_clusters = K)
clf.fit(X)
```

Each clustering algorithm in Scikit-learn is used as follows.

- First, instantiate an object from the clustering technique
- Then we can use the method `fit` to adjust the learning parameters

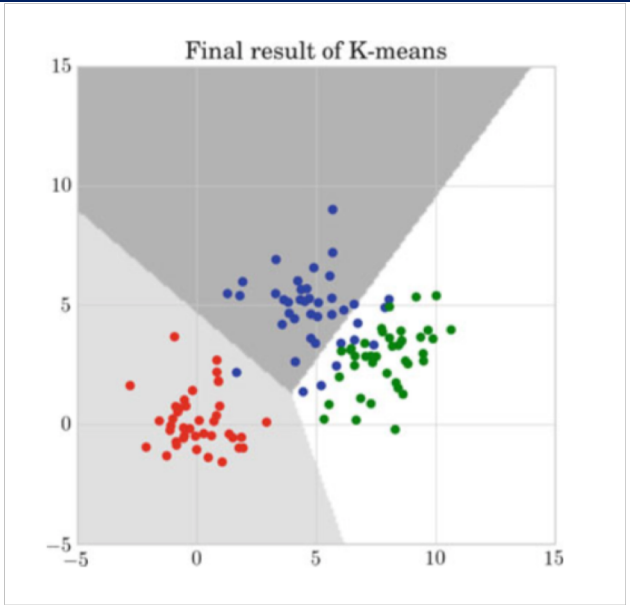
- Also the method `predict` that, given new data, returns the cluster they belong to
- For the class, the labels over the training data can be found in the attribute `labels_` or obtained using the method `predict`

RUNNING THE K-MEANS ALGORITHM (3/3)

How many “mis-clusterings” do we have?

1. Tessellate the space and color all grid points from the same cluster with the same color

2. Overlay the initial sample distributions



- Ideally in each partitioned subspace the sample points are of the same color
- However, the resulting clustering, represented by the color subspace in gray, does not usually coincide exactly with the initial distribution, represented by the color of the data
- For example, if most of the blue points belong to the same cluster, there are a few ones that belong to the space occupied by the green data

EVALUATING K-MEANS ALGORITHM RESULTS (1/2)

When computing the **Adjusted Rand** index, we get:

```
print ('The Adjusted Rand index is: %.2f' %
       metrics.adjusted_rand_score(y.ravel(), clf.labels_))
```

- Taking into account that the Adjusted Rand index belongs to the interval [0, 1]
  - the result of 0.66 in our example means that although most of the clusters were discovered
  - not 100% of them were (see the plot)