

Do not distribute course material

You may not and may not allow others to reproduce or distribute lecture notes and course materials publicly whether or not a fee is charged.

<https://mpatacchiola.github.io/blog/2020/07/31/gaussian-mixture-models.html>

Chapter 9 in Hands-On Machine Learning with Scikit-Learn, Keras & TensorFlow

<http://cs229.stanford.edu/notes2020spring/cs229-notes7a.pdf>

<http://cs229.stanford.edu/notes2020spring/cs229-notes7b.pdf>

pages 179-181 in <http://ciml.info/>

Clustering, K-Means and EM

INTRODUCTION TO MACHINE LEARNING

PROF. LINDA SELLIE

THANKS TO PROF RANGAN FOR SOME OF THE SLIDES

Outline

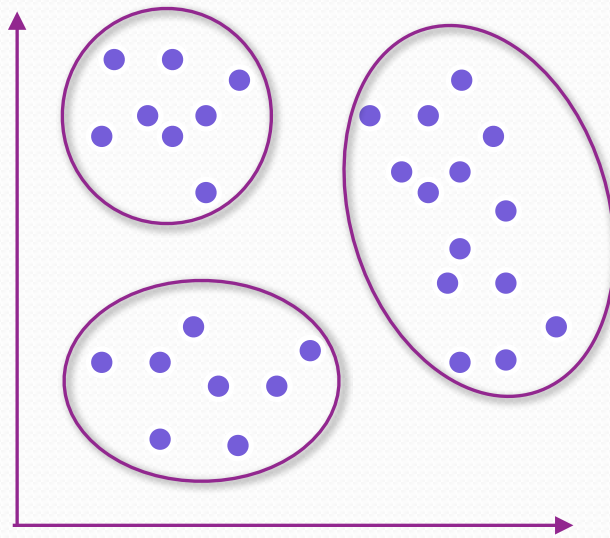
 Motivating Examples: Document clustering, image segmentation, image compression

- ❑ K-means
- ❑ K++-means (how to initialize the parameters before starting the algorithm)
- ❑ (On our own) K-means for document clustering

Unsupervised Machine Learning

$$\{(\mathbf{x}^{(1)}, \cancel{y^{(1)}}), (\mathbf{x}^{(2)}, \cancel{y^{(2)}}), \dots, (\mathbf{x}^{(N)}, \cancel{y^{(N)}})\}$$

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$$

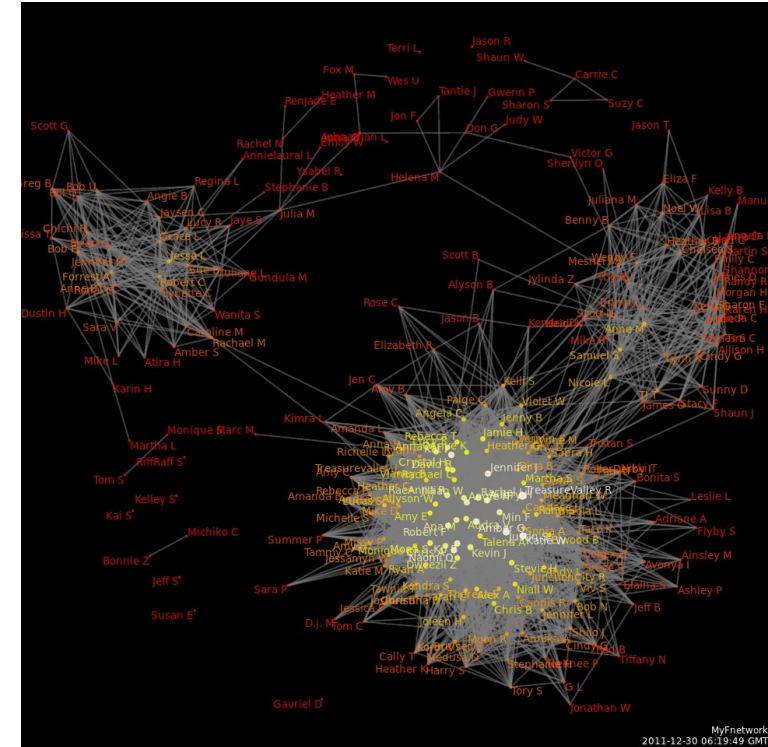
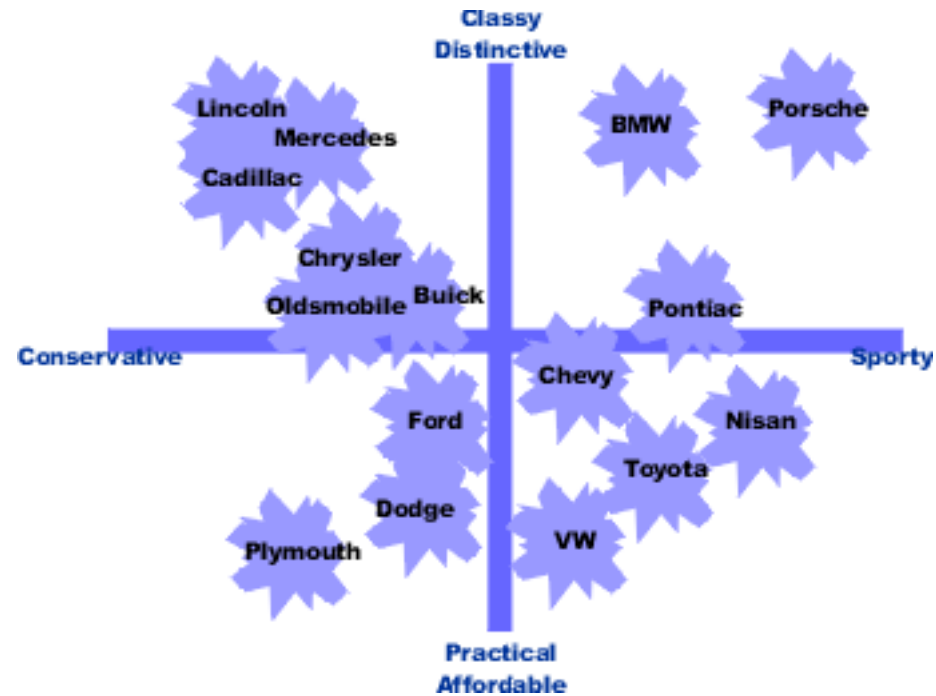


Some clustering applications

- Customer segmentation based on their purchases and activities. Allows targeted marketing for different clusters
- Dimensionality reduction: If there are k cluster each example will have k new features. Each feature is a measure of how well the example fits into a cluster
- Impute missing values
- Anomaly detection (aka outlier detection)
- Semi-supervised learning (you receive a small amount of labeled data). Label the unlabeled data in the cluster according to the labeled data
- Search engines
- Segmentation

Clustering

By Kencf0618 [CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/>) or CC BY-SA 3.0 (<https://creativecommons.org/licenses/by-sa/3.0/>)], from Wikimedia Commons



https://en.wikipedia.org/wiki/Market_segmentation

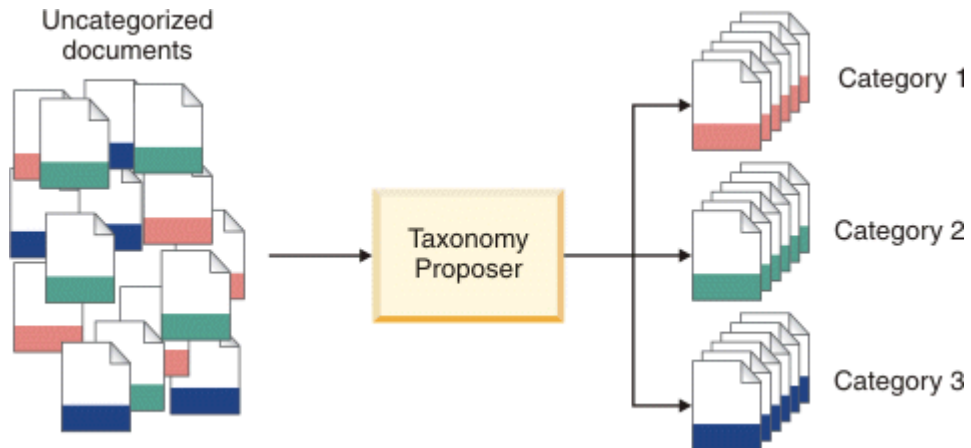
Document Clustering

IBM

IBM Knowledge Center

[Content Classification](#) > [Content Classification 8.8.0](#) > [Configuring](#) > [Cat](#)
Using the Taxonomy Proposer to discover new categories

Using the Taxonomy Proposer to discover new categories



- ❑ Data mining

- ❑ Often have huge numbers of documents

- ❑ How can we organize this?

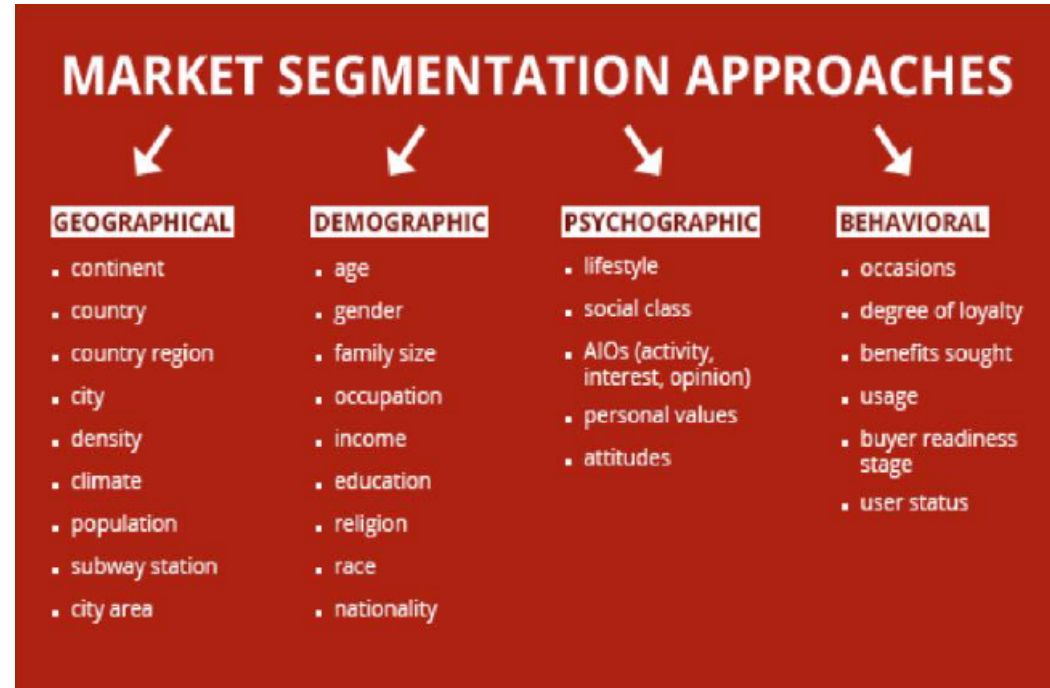
- ❑ Key idea: documents are often in clusters

- ❑ Can we detect these clusters?

- ❑ Can be a lucrative service
 - See IBM service to left

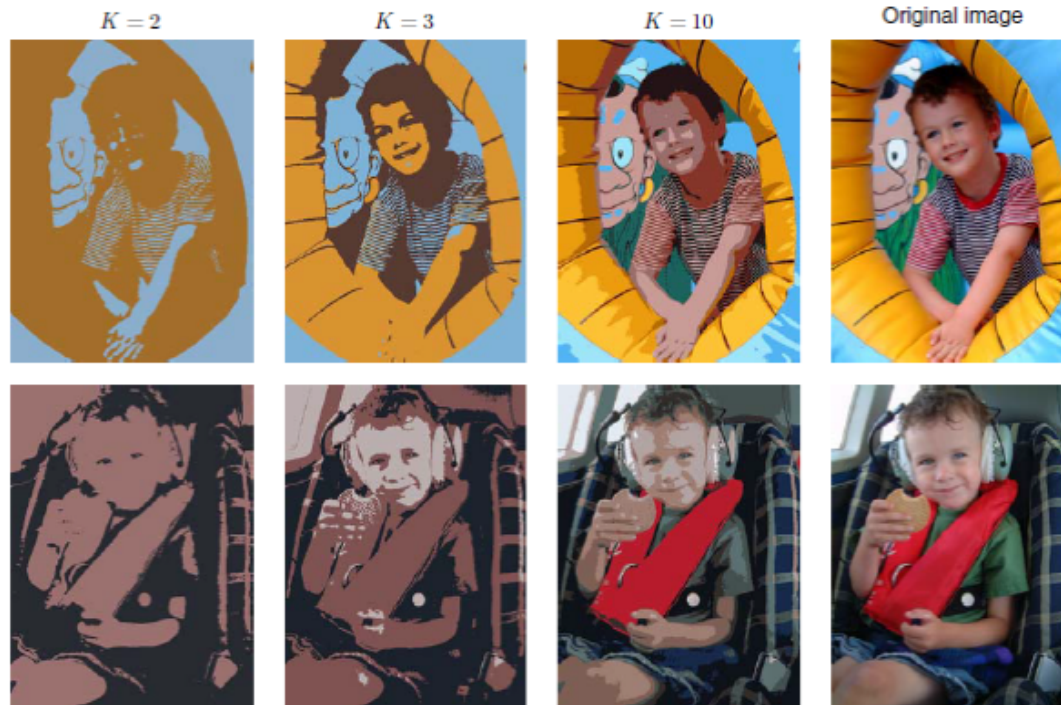
Clustering

- ❑ Clustering has many applications
 - Any time you want to segment data
 - Uncovering latent discrete variables
- ❑ Examples:
 - Segmenting sections of an image
 - Segmenting customers in market data



From: Market segmentation possibilities in the tourism market context of South Africa

Image Segmentation



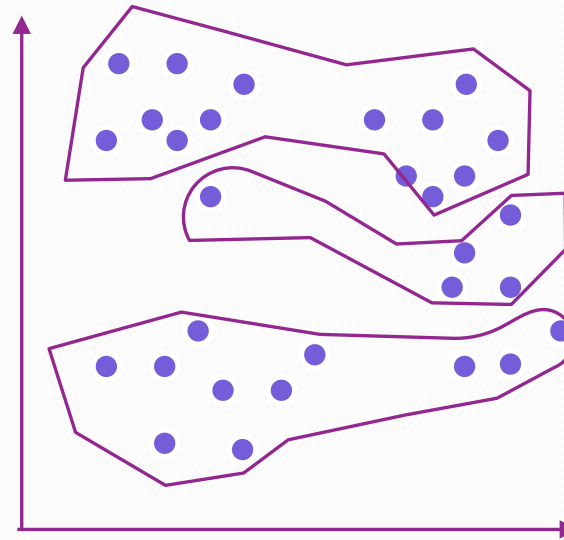
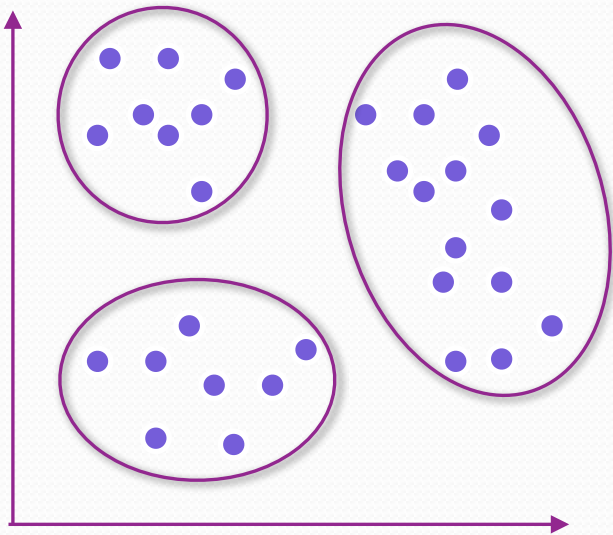
- ❑ Also from Bishop.
- ❑ Use K-means on the RGB values (dimension = 3)

How can we find clusters in the data?

What makes a “good” cluster?

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$$

$$c^{(1)}, c^{(2)}, \dots, c^{(N)} \quad 1 \leq c^{(i)} \leq K$$

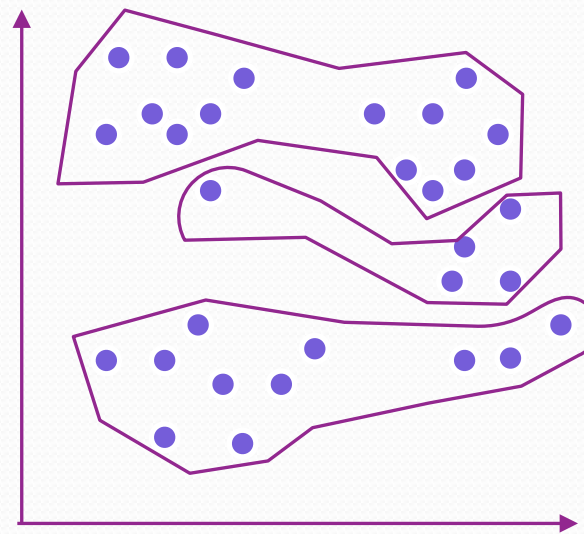
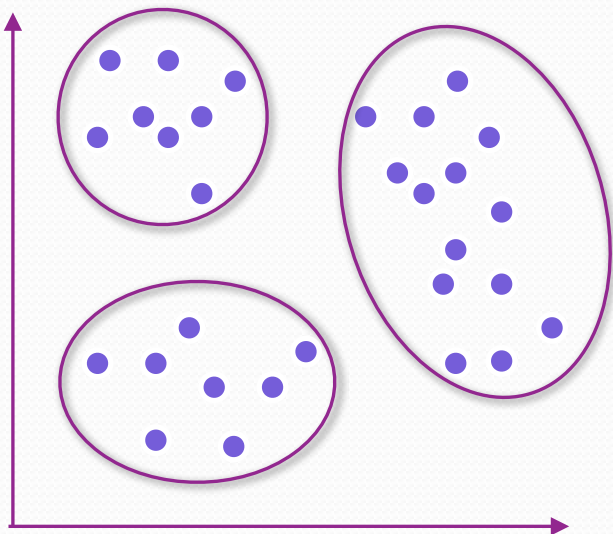


What makes one clustering assignment better than another?

“Goodness” Metric

$$\{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \dots, \mathbf{x}^{(N)}\}$$

$$c^{(1)}, c^{(2)}, \dots, c^{(N)} \quad 1 \leq c^{(i)} \leq K$$

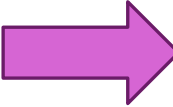


$$\sum_{\mathbf{x} \in \text{cluster 1}} \|\mathbf{x} - \mu_1\|_2^2 + \sum_{\mathbf{x} \in \text{cluster 2}} \|\mathbf{x} - \mu_2\|_2^2 + \sum_{\mathbf{x} \in \text{cluster 3}} \|\mathbf{x} - \mu_3\|_2^2$$

$$= \sum_{i=1}^3 \sum_{\mathbf{x} \in \text{cluster } i} \|\mathbf{x} - \mu_i\|_2^2$$

$$= \sum_{j=1}^N \|\mathbf{x}^{(j)} - \mu_{c^{(j)}}\|_2^2$$

Outline

- ❑ Motivating Examples: Document clustering, image segmentation, image compression
- ❑ K-means
- ❑ K++-means (how to initialize the parameters before starting the algorithm)
- ❑ (On our own) K-means for document clustering

Clustering - grouping items

- ❑ Clustering is a classic unsupervised learning task. There are many algorithms for clustering high-dimensional data
- ❑ One clustering method is K-means clustering. It finds a predetermined (K) number of clusters in an unlabeled dataset
- ❑ K-means clustering assigns each example examples one of K clusters, S_1, S_2, \dots, S_K so as to minimize the sum of squared distance of each point to its cluster center:

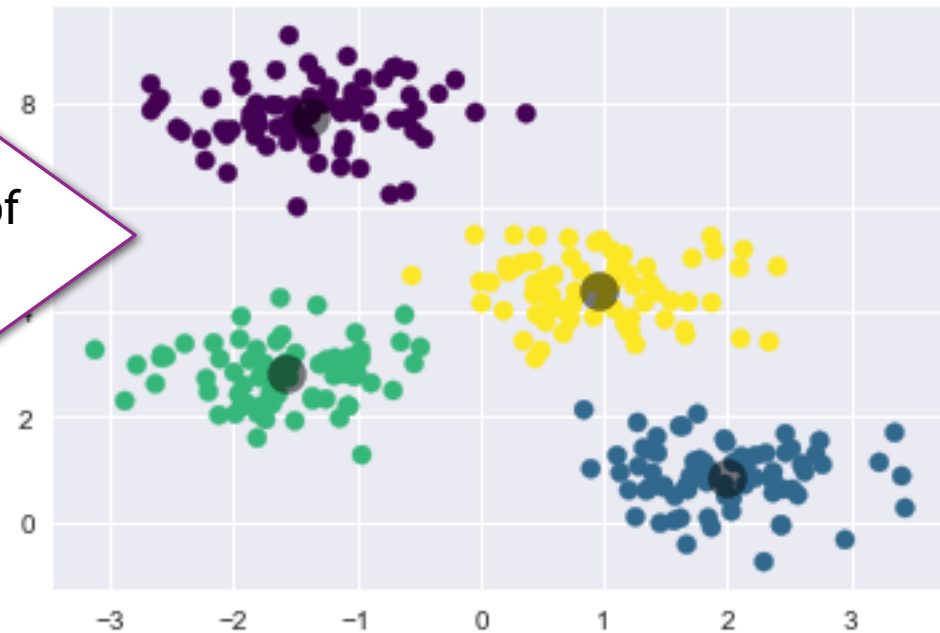
$$J(c, \mu) = \sum_{i=1}^N \left\| \mathbf{x}^{(i)} - \mu_{c^{(i)}} \right\|^2$$

NP hard to solve this problem!

There is an exponential number of ways to assign points to clusters

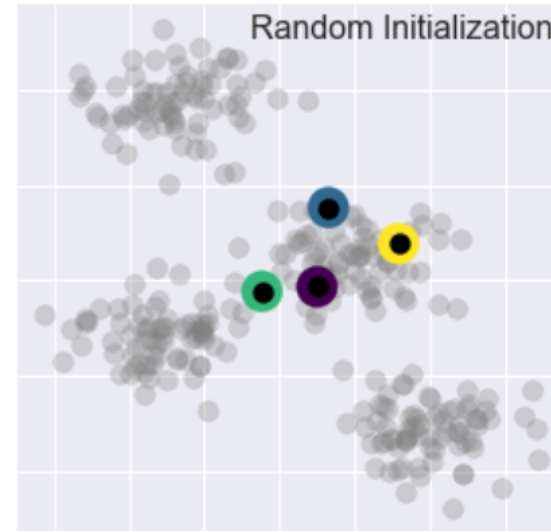
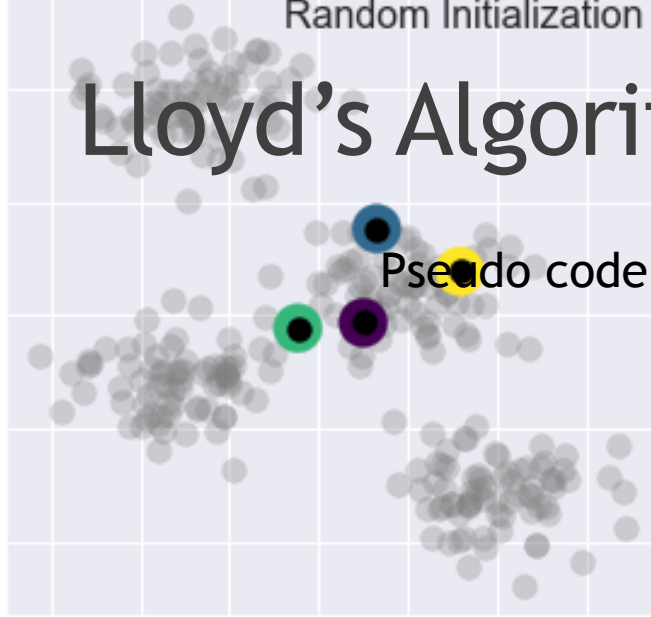
where μ_j is the center of cluster j (i.e. the *mean* of its cluster)
 $c^{(i)}$ is the cluster $\mathbf{x}^{(i)}$ belongs to

- ❑ Goal is to have examples in the same cluster to be “close” to each other



Lloyd's Algorithm (Stuart Lloyd, 1957)

Pseudo code from CS229 Lecture notes



1. Initialize cluster *centroids* $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^d$ randomly
2. Repeat until convergence:
For every i , set

$$c^{(i)} := \arg \min_j \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2$$

For every $j \in \{1, \dots, K\}$, set

$$\mu_j := \frac{\sum_{i=1}^N 1\{c^{(i)} = j\} \mathbf{x}^{(i)}}{\sum_{i=1}^N 1\{c^{(i)} = j\}}$$

}

Update cluster membership of every example. Every example belongs to the cluster it is closest to.

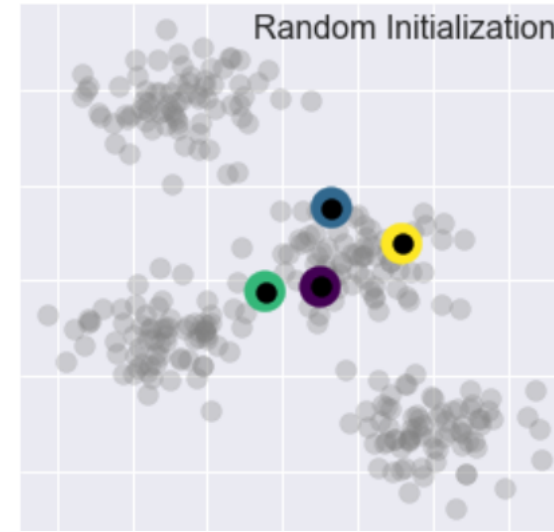
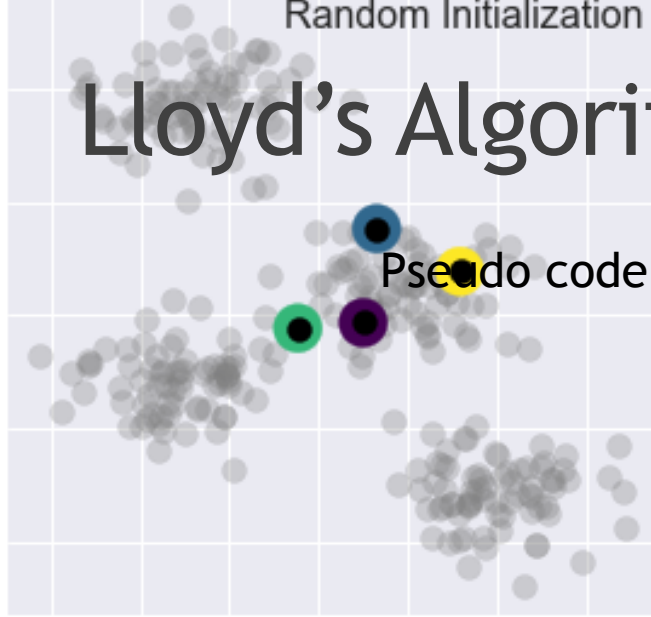
Suppose $\mathbf{x}^{(2)}, \mathbf{x}^{(9)}, \mathbf{x}^{(21)}$ were assigned to cluster 1 then $\mu_1 = (\mathbf{x}^{(2)} + \mathbf{x}^{(9)} + \mathbf{x}^{(21)})/3$

Definition:

$$1\{c^{(i)} = j\} = \begin{cases} 1 & c^{(i)} = j \\ 0 & c^{(i)} \neq j \end{cases}$$

Lloyd's Algorithm (Stuart Lloyd, 1957)

Pseudo code from CS229 Lecture notes



1. Initialize cluster *centroids* $\mu_1, \mu_2, \dots, \mu_K \in \mathbb{R}^d$ randomly
2. Repeat until convergence:
For every i , set

$$c^{(i)} := \arg \min_j \|x^{(i)} - \mu_j\|^2$$

For every $j \in \{1, \dots, K\}$, set

$$\mu_j := \frac{\sum_{i=1}^N 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^N 1\{c^{(i)} = j\}}$$

}

Update cluster membership of every example. Every example belongs to the cluster it is closest to.

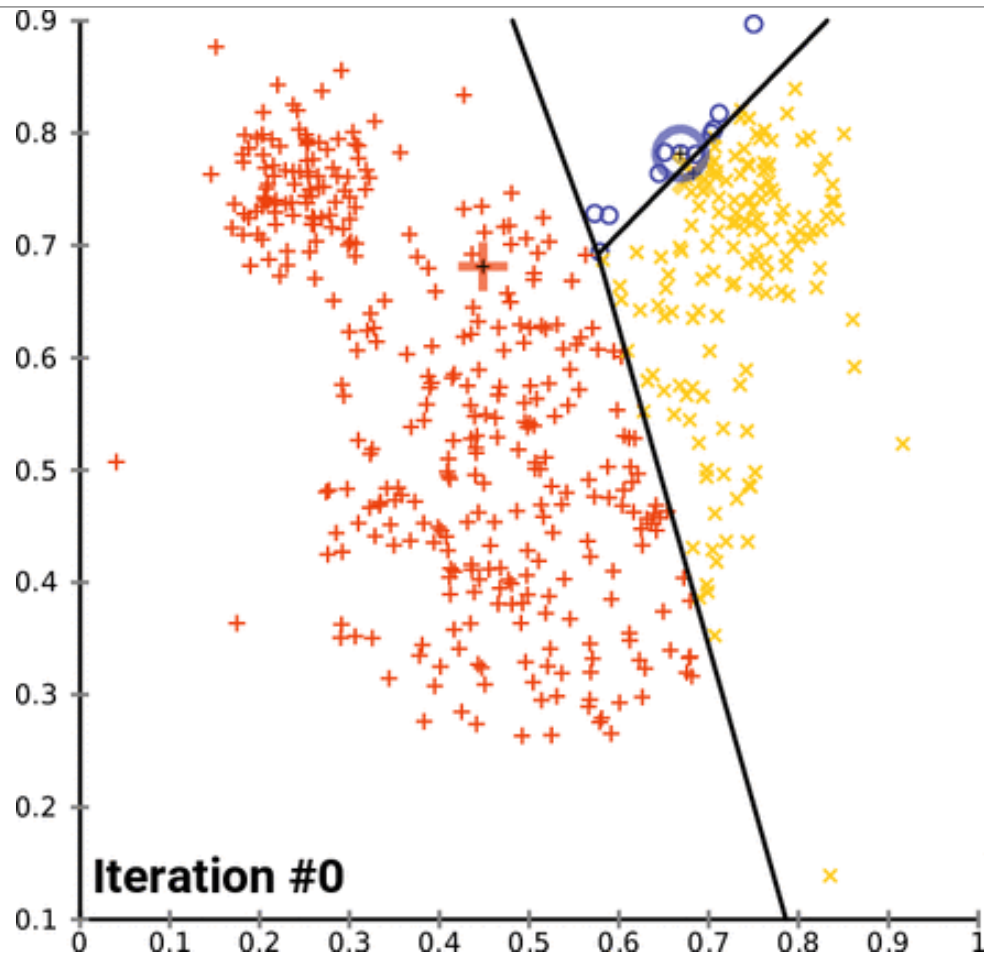
Update *centroid* of each cluster to be the *average(mean)* of examples assigned to cluster j

Definition:

$$1\{c^{(i)} = j\} = \begin{cases} 1 & c^{(i)} = j \\ 0 & c^{(i)} \neq j \end{cases}$$

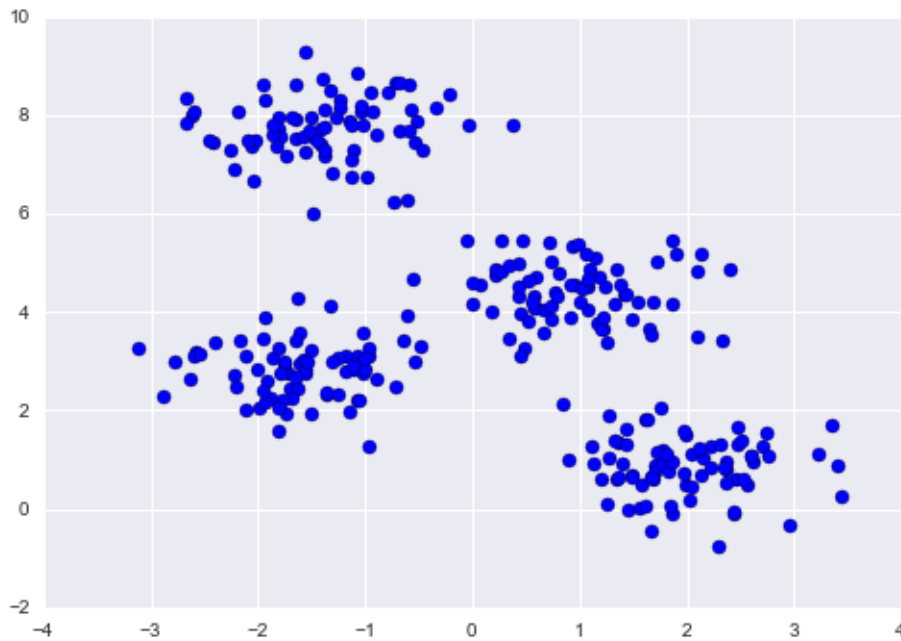
K-Means illustrated

By Chire [GFDL (<http://www.gnu.org/copyleft/fdl.html>) or CC BY-SA 4.0 (<https://creativecommons.org/licenses/by-sa/4.0>)], from Wikimedia Commons



Uh Oh...

- ❑ The K-means clustering algorithm is guaranteed to improve the result on each step...and converge - but not to a globally optimal solution
- ❑ However K-means is not guaranteed to find a global minimum - only a local minimum
- ❑ Finding the global minimum K-means error is NP-hard...
- ❑ People run the algorithm with many initial configurations and keep the one that performs best



K-Means Converges

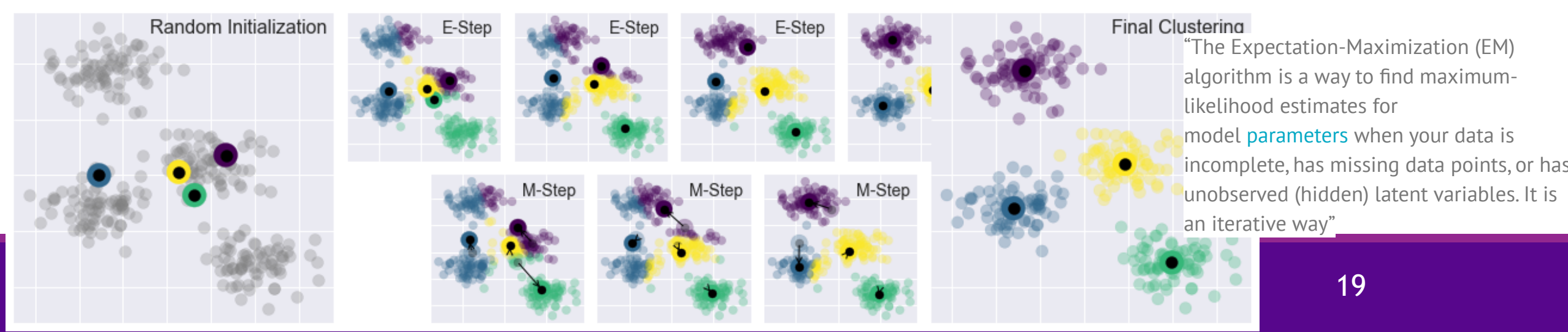
□ The algorithm converges to a partition that is “locally optimal.”

- Given the cluster centers μ_j , we cannot find a better assignment of the examples to clusters
- Given the cluster assignments ($c^{(i)}$ for all $i \in 1 \dots N$), we cannot find better centers

□ The K-means algorithm is a variant of the E-M algorithm.

- The E step (Expectation step) involves updating our expectation of what cluster each example belongs to.
- The M step (Maximization step) involves maximizing the best location of the cluster centers.

□ The algorithm works by minimizing a complex error function by separating the data into two steps:
If one step is known it is easy to optimize the other step

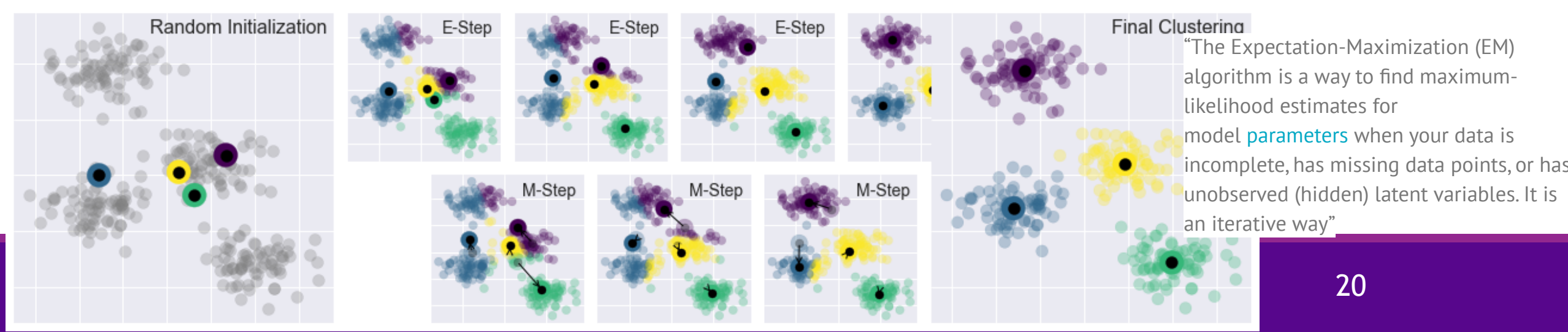


K-Means Converges

- ❑ The algorithm converges to a partition that is “locally optimal”
 - Given the cluster centers μ_j , we cannot find a better assignment of points to be its center?
 - Given the cluster assignments $(c^{(i)})$ for all $i \in 1 \dots N$, we cannot find a better location for the cluster center?
- ❑ The K-means algorithm is a variant of the E-M algorithm.
 - The E step (Expectation step) involves updating our expectation of which cluster each point belongs to.
 - The M step (Maximization step) involves maximizing the likelihood of the data given the current cluster centers.
- ❑ The algorithm works by minimizing a complex error function by separating the data into two steps: If one step is known it is easy to optimize the other step

Given an assignment of points to clusters, could we find a better cluster center than taking the average of the points in a cluster to be its center?

- A) Yes
- B) No
- C) Maybe



The following two slides were not covered in class. Optional material.

Centroid is Minimizer $\mu_j = \frac{1}{|S_j|} \sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)}$

We show that our choice of μ_j is better than any other point \mathbf{p} .

To show this we need to prove that:

$$\sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \frac{1}{|S_j|} \sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)} \right\|^2 \leq \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mathbf{p} \right\|^2$$

Proof:

$$\sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mathbf{p} \right\|^2 = \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \underbrace{\mathbf{x}^{(i)} - \mu_j}_{\mathbf{a}} + \underbrace{\mu_j - \mathbf{p}}_{\mathbf{b}} \right\|^2$$

Adding $0 = -\mu_j + \mu_j$

Here \mathbf{a}, \mathbf{b} are vectors. Notice that:
 $\|\mathbf{a} + \mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 + 2\mathbf{a}^T \mathbf{b}$

$$= \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2 + \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mu_j - \mathbf{p} \right\|^2 + 2 \sum_{\mathbf{x}^{(i)} \in S_j} (\mathbf{x}^{(i)} - \mu_j)^T (\mu_j - \mathbf{p})$$

$$= \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2 + \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mu_j - \mathbf{p} \right\|^2$$

$$\geq \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2$$

We can move $(\mu_j - \mathbf{p})$ in front of the sum: $2(\mu_j - \mathbf{p})^T \sum_{\mathbf{x}^{(i)} \in S_j} (\mathbf{x}^{(i)} - \mu_j)$

We can rewrite this as:

$$= 2(\mu_j - \mathbf{p})^T \left(\left(\sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)} \right) - |S_j| \mu_j \right)$$

Now notice that: $|S_j| \mu_j = \sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)}$

Thus $\left(\sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)} \right) - |S_j| \mu_j = 0$

Centroid is Minimizer $\mu_j = \frac{1}{|S_j|} \sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)}$

We show that our choice of μ_j is better than any other point \mathbf{p} .

To show this we need to prove that:

$$\sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2 \leq \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mathbf{p} \right\|^2$$

Proof:

$$\sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mathbf{p} \right\|^2 = \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \underbrace{\mathbf{x}^{(i)} - \mu_j}_{\mathbf{a}} + \underbrace{\mu_j - \mathbf{p}}_{\mathbf{b}} \right\|^2$$

Adding $0 = -\mu_j + \mu_j$

Here \mathbf{a}, \mathbf{b} are vectors. Notice that:
 $\|\mathbf{a} + \mathbf{b}\|^2 = \|\mathbf{a}\|^2 + \|\mathbf{b}\|^2 + 2\mathbf{a}^T \mathbf{b}$

$$= \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2 + \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mu_j - \mathbf{p} \right\|^2 + 2 \sum_{\mathbf{x}^{(i)} \in S_j} (\mathbf{x}^{(i)} - \mu_j)^T (\mu_j - \mathbf{p})$$

$$= \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2 + \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mu_j - \mathbf{p} \right\|^2$$

$$\geq \sum_{\mathbf{x}^{(i)} \in S_j} \left\| \mathbf{x}^{(i)} - \mu_j \right\|^2$$

We can move $(\mu_j - \mathbf{p})$ in front of the sum: $2(\mu_j - \mathbf{p})^T \sum_{\mathbf{x}^{(i)} \in S_j} (\mathbf{x}^{(i)} - \mu_j)$

We can rewrite this as:

$$= 2(\mu_j - \mathbf{p})^T \left(\left(\sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)} \right) - |S_j| \mu_j \right)$$

Now notice that: $|S_j| \mu_j = \sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)}$

Thus $\left(\sum_{\mathbf{x}^{(i)} \in S_j} \mathbf{x}^{(i)} \right) - |S_j| \mu_j = 0$

Proof of convergence (to a local min)

Theorem (K - Means Convergence Theorem)

We update μ and we update c . For each update we show that they never increase the value of

$$J(c, \mu) = \sum_{i=1}^N \left\| \mathbf{x}^{(i)} - \mu_{c^{(i)}} \right\|^2$$

There are only a finite number of values that can be assigned to μ and c . (μ is the mean of a subset of the examples and $c \in \{1, 2, \dots, K\}$).

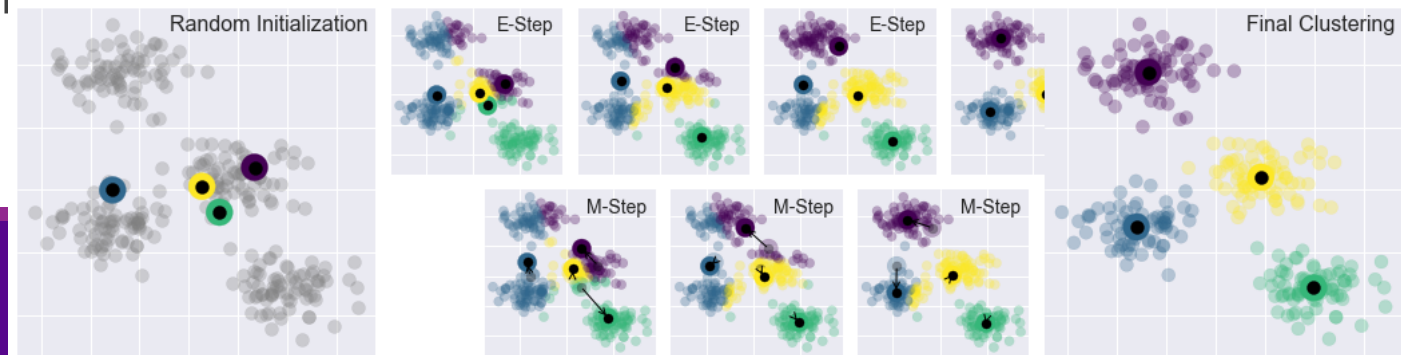
We also know that $J(c, \mu) \geq 0$.

Thus $J(c, \mu)$ can only decrease a finite number of times. When it stops decreasing the algorithm has converged (to a local minimum)

When we update $c^{(i)}$, it must be that $\left\| \mathbf{x}^{(i)} - \mu_{c^{(i \text{ new})}} \right\|^2 \leq \left\| \mathbf{x}^{(i)} - \mu_{c^{(i)}} \right\|^2$

When we update μ_j as the mean of the points which are in this cluster - it directly minimizes $\sum_{c^{(i)}=j} (\mathbf{x}^{(i)} - \mu_j)^2$

Thus every iteration decreases the cost function



Since it is possible to converge to a local minimum instead of a global minimum, you should run the algorithm 10 times and choose the clustering with the lowest $J(c, \mu)$

K-Means

□ A simple implementation in Python from

<https://jakevdp.github.io/PythonDataScienceHandbook/05.11-k-means.html>

```
from sklearn.metrics import pairwise_distances_argmin

def find_clusters(X, n_clusters, rseed=2):
    # 1. Randomly choose clusters
    rng = np.random.RandomState(rseed)
    i = rng.permutation(X.shape[0])[:n_clusters]
    centers = X[i]

    while True:
        # 2a. Assign labels based on closest center
        labels = pairwise_distances_argmin(X, centers)

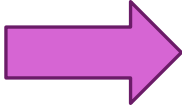
        # 2b. Find new centers from means of points
        new_centers = np.array([X[labels == i].mean(0)
                                for i in range(n_clusters)])

        # 2c. Check for convergence
        if np.all(centers == new_centers):
            break
        centers = new_centers

    return centers, labels

centers, labels = find_clusters(X, 4)
plt.scatter(X[:, 0], X[:, 1], c=labels,
            s=50, cmap='viridis');
```

Outline

- ❑ Motivating Examples: Document clustering, image segmentation, image compression
- ❑ K-means
-  ❑ K++-means (how to initialize the parameters before starting the algorithm)
- ❑ (On our own) K-means for document clustering

How to choose the initial values...

- ❑ The big concern is poor initialization at the start of the algorithm.
- ❑ One heuristic (we will refine it on the next slide) is to use the *furthest-first* algorithm

1. Pick a random example j and set $\mu_1 = \mathbf{x}^{(j)}$

2. For $k'' = 2..K$:

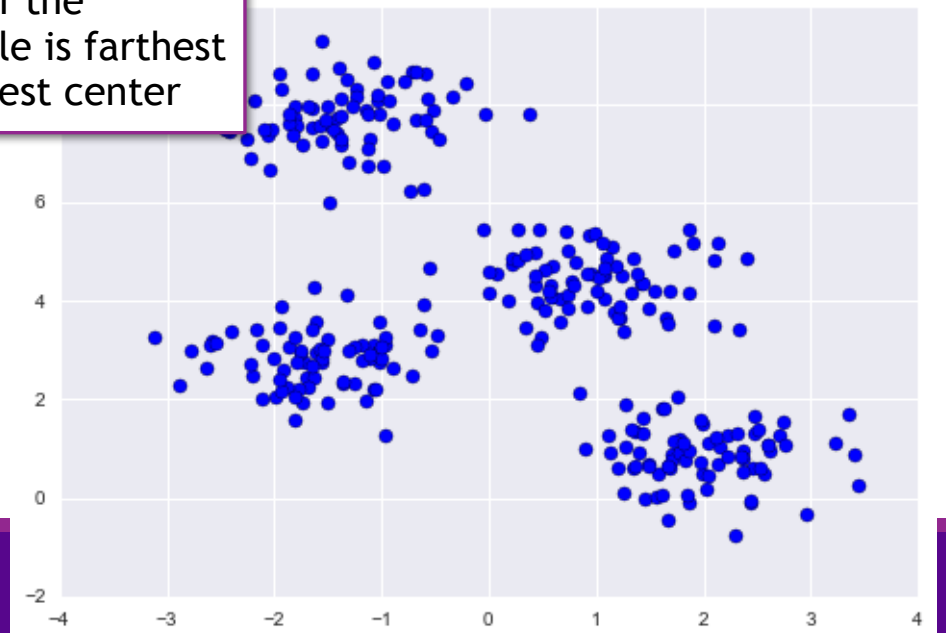
Find the example j that is as far as possible from all previously selected means; namely:

$$j = \arg \max_j \min_{k' < k''} \left\| \mathbf{x}^{(j)} - \mu_{k'} \right\|^2$$

and set $\mu_{k''} = \mathbf{x}^{(j)}$

Find index of the training example is farthest from its closest center

The problem is that this algorithm is sensitive to outliers.



K-means++ algorithm

It can be proven that the expected value of the $J(c, \mu)$ when running K-means++ is never more than $O(\log K)$ times optimal $J(c, \mu)$

□ Instead of choosing the furthest example from your existing clusters, choose the next center randomly with probability proportional to its distance squared

□ Algorithm k-means++

$\mu_1 = \mathbf{x}^{(j)}$ for j chosen uniformly at random // randomly initialize first point

for $k''=2$ to K do

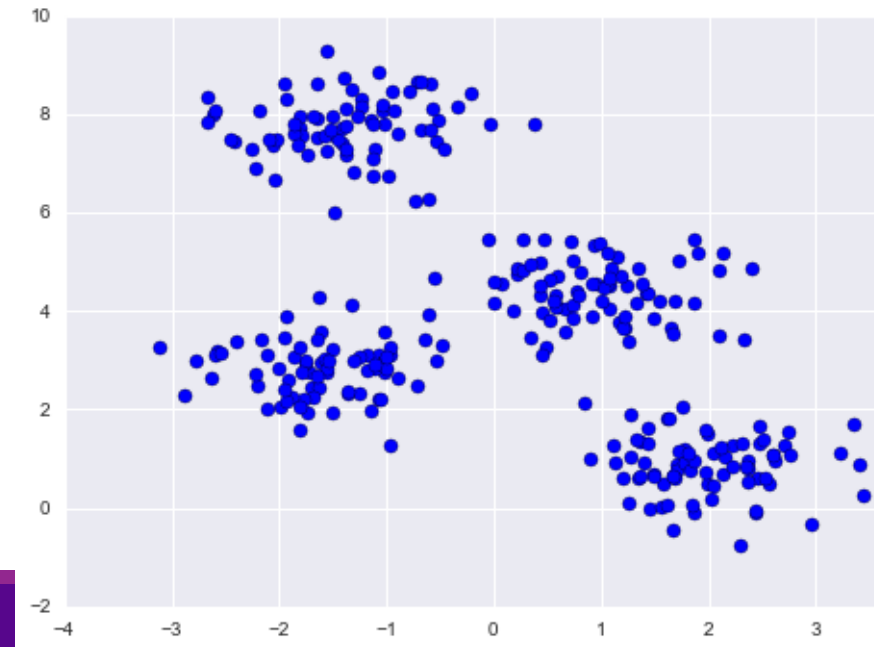
$$d_j = \min_{k' < k''} \left\| \mathbf{x}^{(j)} - \boldsymbol{\mu}_{k'} \right\|, \forall j \quad // \text{compute distances}$$

$$p_j = \frac{d_j^2}{\sum_{i=1}^m d_i^2}, \forall j \quad // \text{normalize to probability distribution}$$


j = random chosen with probability p_j

$$\mu_{k''} = \mathbf{x}^{(j)}$$

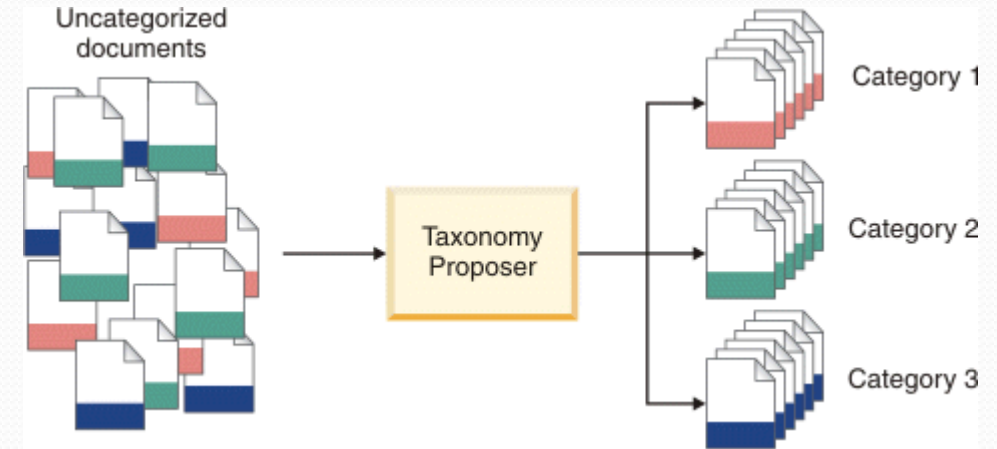
run k-means using μ as initial centers



Outline

- ❑ Motivating Examples: Document clustering, image segmentation, image compression
- ❑ K-means
- ❑ K++-means (how to initialize the parameters before starting the algorithm)
-  ❑ (On our own) K-means for document clustering

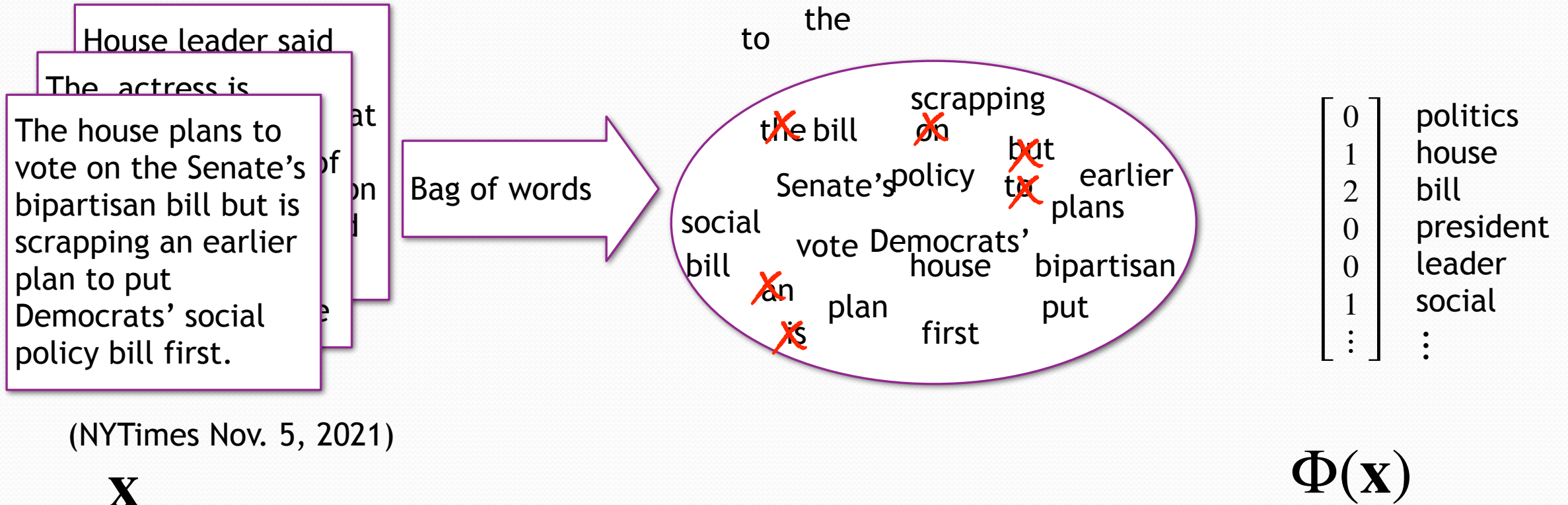
Feature Extraction:



If we want to use K-means into cluster documents, we first need to convert text into a set of numerical values.

How can we do this?

Documents as feature vectors



Transform the feature vectors to emphasize more “relevant” words

Turning text into a feature vector

Document 1

The quick brown
fox jumped over
the lazy dog's
back.

Document 2

Now is the time
for all good men
to come to the
aid of their party.

- ❑ Document is natively text
- ❑ Must represent as a numeric vector
- ❑ Represent by word counts
 - Enumerate all words
 - Each document is count of frequencies
- ❑ Stopwords

Discussion Questions

- ❑ Is the absolute number of times a word appears the correct metric?
- ❑ What about the length of the document?
- ❑ What about the frequency of the word?
- ❑ What words “matter”?

the, for, a, in

convolutional, gradient

- ❑ Perhaps:
 - if a word appears frequently, it is important (give it a high score)
 - If a word appears in many documents, it is not important (give it a low score)

Ideas:

$$TF_{\text{"this"}, d_1} = \frac{1}{5} = 0.2$$

$$TF_{\text{"this"}, d_2} = \frac{1}{7} \approx 0.14$$

$$IDF_{\text{"this"}} = \log\left(\frac{2}{2}\right) = 0$$

$$TF_{\text{"example"}, d_1} = \frac{0}{5} = 0$$

$$TF_{\text{"example"}, d_2} = \frac{3}{7} \approx 0.429$$

$$IDF_{\text{"example"}} = \log\left(\frac{2}{1}\right) = 1$$

Example modified from <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

❑ How can we categorize how important a word is in a document?

❑ Perhaps:

- if a word appears frequently, it is important (give it a high score) convolutional, gradient
- except if the word appears in many documents, it is not important (give it a low score)

❑ Steps:

- Count the frequency of every word in the document

Term frequency

$$TF_{i,n} = \frac{\text{num times word } i \text{ in doc } n}{\text{total num words in doc } n}$$

- Determine how much information a word provides: Inverse Document Frequency (IDF)

The more common a word is
the lower its IDF score

Inverse doc frequency

$$IDF_i = \log\left[\frac{\text{Total num docs in corpus}}{\text{Num docs with word } i}\right]$$

Document 1

Term	Term Count
this	1
Is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
Is	1
another	2
example	3

Ideas:

$$TF_{\text{"this"}, d_1} = \frac{1}{5} = 0.2$$

$$TF_{\text{"this"}, d_2} = \frac{1}{7} \approx 0.14$$

$$IDF_{\text{"this"}} = \log\left(\frac{2}{2}\right) = 0$$

$$TF_{\text{"example"}, d_1} = \frac{0}{5} = 0$$

$$TF_{\text{"example"}, d_2} = \frac{3}{7} \approx 0.429$$

$$IDF_{\text{"example"}} = \log\left(\frac{2}{1}\right) = 1$$

Frequency is relative to the size of the document

Example modified from

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

How can we categorize how important a word is in a document?

Perhaps:

- if a word appears frequently, it is important (give it a high score)
- except if the word appears in many documents, it is not important (give it a low score)

Steps:

- Count the frequency of every word in the document

$$TF_{i,n} = \frac{\text{num times word } i \text{ in doc } n}{\text{total num words in doc } n}$$

- Determine how much information a word provides: Inverse Document Frequency (IDF)

The more common a word is the lower its IDF score

$$IDF_i = \log\left[\frac{\text{Total num docs in corpus}}{\text{Num docs with word } i}\right]$$

Document 1

Term	Term Count
this	1
Is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
Is	1
another	2
example	3



NYU

TANDON SCHOOL OF ENGINEERING

<https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

Term Frequency - Inverse Document Frequency

Use TF-IDF weight for vectors:

$$X[n, i] = TF_{i,n} \times IDF_i$$

Document weight vector

Term frequency

Inverse doc frequency

$$TF_{i,n} = \frac{\text{num times word } i \text{ in doc } n}{\text{total num words in doc } n}$$

$$IDF_i = \log \left[\frac{\text{Total num docs in corpus}}{\text{Num docs with word } i} \right]$$

$$TF_{\text{"this"}, d_1} = \frac{1}{5} = 0.2$$

$$IDF_{\text{"this"}} = \log \left(\frac{2}{2} \right) = 0$$

$$TF_{\text{"example"}, d_1} = \frac{0}{5} = 0$$

$$IDF_{\text{"example"}} = \log \left(\frac{2}{1} \right) = 1$$

$$TF_{\text{"this"}, d_2} = \frac{1}{7} \approx 0.14$$

$$TF_{\text{"example"}, d_2} = \frac{3}{7} \approx 0.429$$

Example modified from <https://en.wikipedia.org/wiki/Tf%E2%80%93idf>

$$TF\text{-}IDF_{\text{"this"}, d_1} = 0.2 \times 0 = 0$$

$$TF\text{-}IDF_{\text{"example"}, d_1, D} = 0. \times 1 = 0$$

$$TF\text{-}IDF_{\text{"this"}, d_2} = 0.14 \times 0 = 0$$

$$TF\text{-}IDF_{\text{"example"}, d_2} = 0.429 \times 1 = 0.429$$

Term Frequency - Inverse Document Frequency

Document 1

Term	Term Count
this	1
is	1
a	2
sample	1

Document 2

Term	Term Count
this	1
is	1
another	2
example	3

Term	TF-IDF Example 1	TF-IDF Example 2
this	0	0
is	0	0
a	0.4	0
sample	0.2	0
example	0	0.429
another	0	0.286

$$\text{TF-IDF}_{\text{"is"}, d_1} = 0.2 \times 0 = 0$$

$$\text{TF-IDF}_{\text{"is"}, d_2} = 0.143 \times 0 = 0$$

$$\text{TF-IDF}_{\text{"a"}, d_1} = 0.4 \times 1 = 0.4$$

$$\text{TF-IDF}_{\text{"a"}, d_2} = 0 \times 1 = 0$$

$$\text{TF-IDF}_{\text{"sample"}, d_1} = 0.2 \times 1 = 0.2$$

$$\text{TF-IDF}_{\text{"sample"}, d_2} = 0 \times 1 = 0$$

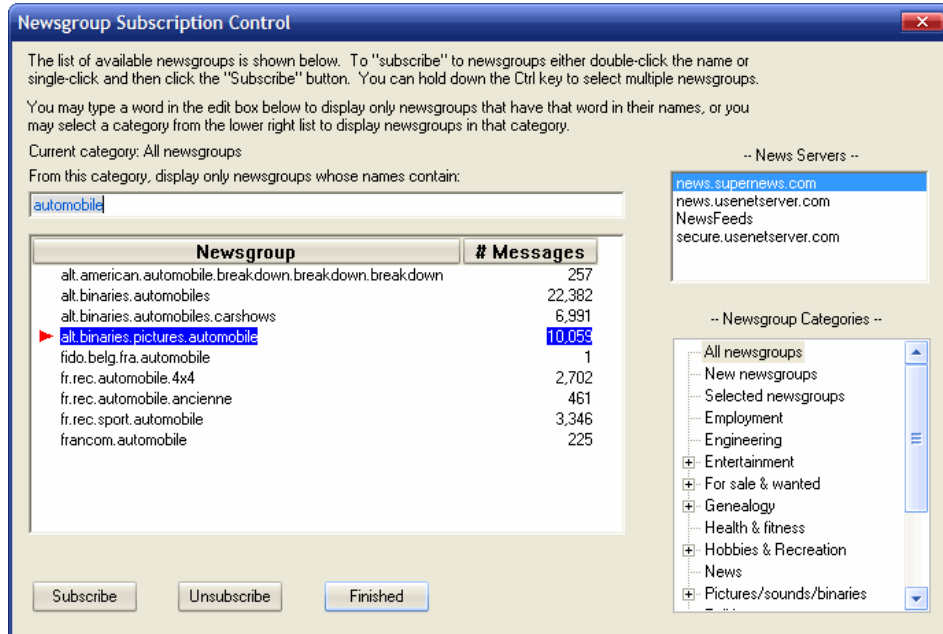
$$\text{TF-IDF}_{\text{"another"}, d_1} = 0.286 \times 1 = 0.286$$

$$\text{TF-IDF}_{\text{"another"}, d_2} = 0 \times 1 = 0$$

This is a very small example. Obviously we would usually compute the TF-IDF for a large collection of documents

On your own, you can look at Prof Rangan's document clustering example

UseNet Newsgroups



- ❑ Began in late 1970s
- ❑ Discussion groups for various topics
 - Started on early university networks
 - Migrated to Internet
 - Peaked in 1990s
- ❑ Useful for studying clustering
 - Simple documents
 - “ground truth”: Docs have categories

“The 20 Newsgroups data set is a collection of approximately 20,000 newsgroup documents, partitioned (nearly) evenly across 20 different newsgroups.”

Loading the Data

- ❑ See Prof. Rangan's full clustering code https://github.com/sdrangan/introml/blob/master/unit12_cluster/demo1_doc_cluster.ipynb
- ❑ Taken from http://scikit-learn.org/stable/auto_examples/text/document_clustering.html
- ❑ Newsgroups built into sklearn

```
categories = [  
    'alt.atheism',  
    'talk.religion.misc',  
    'comp.graphics',  
    'sci.space',  
]  
  
# Uncomment the following to do the analysis on all the categories  
#categories = None  
  
print("Loading 20 newsgroups dataset for categories:")  
print(categories)  
  
dataset = fetch_20newsgroups(subset='all', categories=categories,  
                             shuffle=True, random_state=42)
```

```
Loading 20 newsgroups dataset for categories:  
['alt.atheism', 'talk.religion.misc', 'comp.graphics', 'sci.space']
```

A Typical Newsgroup Post

❑ Data for the posts are in:

- Dataset.data
- Dataset.labels
- Dataset.target_names

Post from comp.graphics

From: richter@fossi.hab-weimar.de (Axel Richter)
Subject: True Color Display in POV
Keywords: POV, Raytracing
Nntp-Posting-Host: fossi.hab-weimar.de
Organization: Hochschule fuer Architektur und Bauwesen Weimar, Germany
Lines: 6

Hallo POV-Renderers !
I've got a BocaX3 Card. Now I try to get POV displaying True Colors
while rendering. I've tried most of the options and UNIVESA-Driver
but what happens isn't correct.
Can anybody help me ?

Computing TF-IDF in Python

- ❑ Can compute the TF-IDF using sklearn functions

```
vectorizer = TfidfVectorizer(  
    max_df=0.5, # max doc freq (as a fraction) of any word to include in the vocabulary  
    min_df=2,   # min doc freq (as doc counts) of any word to include in the vocabulary  
    max_features=10000, # max number of words in the vocabulary  
    stop_words='english', # remove English stopwords  
    use_idf=True ) # use IDF scores
```

```
print("Extracting features from the training dataset using a sparse vectorizer")  
t0 = time()  
X = vectorizer.fit_transform(dataset.data)  
print("done in %fs" % (time() - t0))  
print("n_samples: %d, n_features: %d" % X.shape)
```

```
Extracting features from the training dataset using a sparse vectorizer  
done in 1.329500s  
n_samples: 3387, n_features: 10000
```

Typical TF-IDF scores

weimar	0.565396
pov	0.518174
renderers	0.183033
univesa	0.178595
und	0.174842
fuer	0.171591
true	0.159214
raytracing	0.150534
displaying	0.140240
ve	0.139752
options	0.134309
rendering	0.133027
driver	0.129544
happens	0.122540
colors	0.119138
card	0.113776
display	0.108457
germany	0.108231
tried	0.106282
color	0.103717
anybody	0.100397
correct	0.100234
isn	0.084694
got	0.081865
keywords	0.081865
try	0.080601
help	0.078058
nntp	0.044277
host	0.043985
posting	0.042608



Plotting the Results

- Most important words in each cluster
 - Highest weights in cluster centers

Cluster 0: graphics image thanks university files file 3d help gif ac

Cluster 1: space com nasa article gov university posting like just host

Cluster 2: god com sandvik people keith jesus say don morality sgi

Cluster 3: henry access toronto digex pat alaska zoo spencer net space

The original categories: comp.graphics, sci.space, talk.religion.misc, alt.atheism,

Confusion Matrix

- ❑ Estimated clusters vs. true categories
- ❑ Can you see where it got confused?

		Predicted			
True	alt.atheism	[0.02664797	0.5559633	0.6512605]
	comp.graphics	[0.67461431	0.00458716	0.00947867]
	sci.space	[0.24263675	0.01651376	0.98420221]
	talk.religion.misc	[0.05610098	0.42293578	0.00631912]

```
dataset.target_names
```

```
['alt.atheism', 'comp.graphics', 'sci.space', 'talk.religion.misc']
```

An Example “Wrong” cluster

Actual newsgroup: talk.religion.misc

Most common newsgroup in cluster: sci.space

From: dickene@access.digex.com (Dick Eney)

Subject: Re: Swastika (was: Hitler - pagan or Christian?)

Organization: Express Access Online Communications, Greenbelt, MD USA

Lines: 15

NNTP-Posting-Host: access.digex.net

The observation that the Tree of Life would rotate clockwise in the northern hemisphere and counterclockwise in the southern probably doesn't give enough consideration to the feebleness of the Coriolis force compared to, say, the phototropism of vegetation. A much more likely explanation is the classic one: that the clockwise swastika is the Sun-wheel, because the sun progresses across the sky that way. (Although that's not the historical way it happened; clocks were first made as little imitation images of the sun moving thru the heavens. So it's more valid to talk of the clock going sunwise, but do the engineers listen to me? Of course not.) Anyway, there is still much uncertainty about whether the anti-swastika goes counter-sunwise because that represents Evil, or because it is the Sun's twin-opposite, the Moonwheel. The use of anti-Sun to represent Evil may be because humans are so strongly visually-oriented, but I'm not going to try to settle THAT one just now.

-- Diccon Frankborn (dickene@access.digex.com)