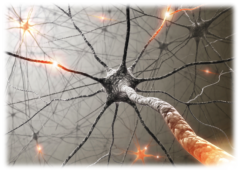
Machine learning: k-means



- In this module, we'll talk about **K-means**, a simple algorithm for clustering, a form of unsupervised learning.

[Brown et al, 1992]

Word clustering

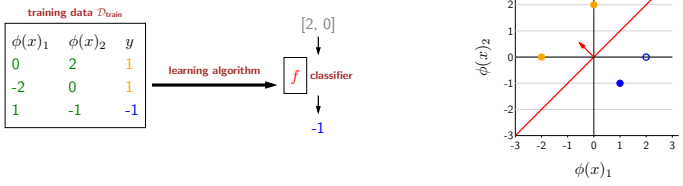
Input: raw text (100 million words of news articles)...

Output:

Cluster 1: Friday Monday Thursday Wednesday Tuesday Saturday Sunday weekends Sundays Saturdays
Cluster 2: June March July April January December October November September August
Cluster 3: water gas coal liquid acid sand carbon steam shale iron
Cluster 4: great big vast sudden mere sheer gigantic lifelong scant colossal
Cluster 5: man woman boy girl lawyer doctor guy farmer teacher citizen
Cluster 6: American Indian European Japanese German African Catholic Israeli Italian Arab
Cluster 7: pressure temperature permeability density porosity stress velocity viscosity gravity tension
Cluster 8: mother wife father son husband brother daughter sister boss uncle
Cluster 9: machine device controller processor CPU printer spindle subsystem compiler plotter
Cluster 10: John George James Bob Robert Paul William Jim David Mike
Cluster 11: anyone someone anybody somebody
Cluster 12: feet miles pounds degrees inches barrels tons acres meters bytes
Cluster 13: director chief professor commissioner commander treasurer founder superintendent dean custodian
Cluster 14: had hadn't hath would've could've should've must've might've
Cluster 15: head body hands eyes voice arm seat eye hair mouth

- Here is a classic example of clustering from the NLP literature, called Brown clustering. This was the unsupervised learning method of choice before word vectors.
- The input to the algorithm is simply raw text, and the output is a clustering of the words.
- The first cluster more or less represents days of the week, the second is months, the third is natural resources, and so on.
- It is important to note that no one told the algorithm what days of the week were or months or family relations. The clustering algorithm discovered this structure automatically.
- On a personal note, Brown clustering was actually my first experience that got me to pursue research in NLP. Seeing the results of unsupervised learning when it works was just magical. And of course today, we're seeing even more strongly the potential of unsupervised learning with neural language models such as BERT and GPT-3.

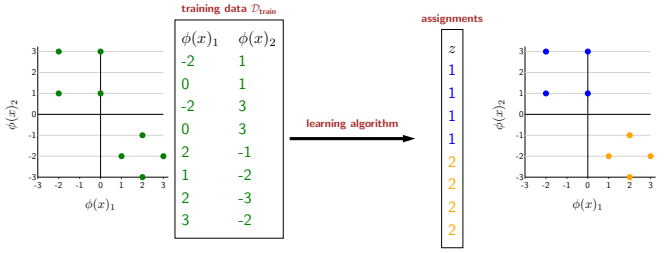
Classification (supervised learning)



Labeled data is expensive to obtain

- I want to contrast unsupervised learning with supervised learning.
- Recall that in classification you're given a set of **labeled** training examples.
- A learning algorithm produces a classifier that can classify new points.
- Note that we're now plotting the (two-dimensional) feature vector rather than the raw input, since the learning algorithms only depend on the feature vectors.
- However, the main challenge with supervised learning is that it can be expensive to collect the labels for data.

Clustering (unsupervised learning)



Intuition: Want to assign nearby points to same cluster

Unlabeled data is very cheap to obtain

- In contrast, in clustering, you are only given **unlabeled** training examples.
- Our goal is to assign each point to a cluster. In this case, there are two clusters, 1 (blue) and 2 (orange).
- Intuitively, nearby points should be assigned to the same cluster.
- The advantage of unsupervised learning is that unlabeled data is often very cheap and almost free to obtain, especially text or images on the web.

Clustering task

Definition: clustering

Input: training points

$$\mathcal{D}_{\text{train}} = [x_1, \ldots, x_n]$$

Output: assignment of each point to a cluster

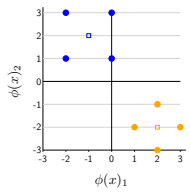
$$\mathbf{z} = [z_1, \ldots, z_n] \text{ where } z_i \in \{1, \ldots, K\}$$

- Formally, the task of clustering is to take a set of points as input and return a partitioning of the points into K clusters.
- We will represent the partitioning using an **assignment vector** $\mathbf{z} = [z_1, \ldots, z_n]$.
- For each i, $z_i \in \{1, \ldots, K\}$ specifies which of the K clusters point i is assigned to.

Centroids

Each cluster $k = 1, \ldots, K$ is represented by a **centroid** $\mu_k \in \mathbb{R}^d$

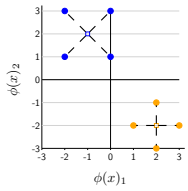
$$\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]$$



Intuition: want each point $\phi(x_i)$ to be close to its assigned centroid μ_{z_i}

- What makes a cluster? The key assumption is that each cluster k is represented by a **centroid** μ_k.
- Now the intuition is that we want each point $\phi(x_i)$ to be close to its assigned centroid μ_{z_i}.

K-means objective

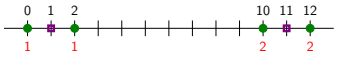


$$\text{Loss}_{\text{kmeans}}(\mathbf{z}, \boldsymbol{\mu}) = \sum_{i=1}^{n} \|\phi(x_i) - \mu_{z_i}\|^2$$

$$\min_{\mathbf{z}} \min_{\boldsymbol{\mu}} \text{Loss}_{\text{kmeans}}(\mathbf{z}, \mu)$$

- To formalize this, we define the K-means objective (distinct from the K-means algorithm).
- The variables are the assignments $\mathbf{z}$ and centroids $\boldsymbol{\mu}$.
- We examine the squared distance (dashed lines) from a point $\phi(x_i)$ to the centroid of its assigned cluster μ_{z_i}. Summing over all these squared distances gives the K-means objective.
- This loss can be interpreted as a reconstruction loss: imagine replacing each data point by its assigned centroid. Then the objective captures how lossy this compression was.
- Now our goal is to minimize the K-means loss.

Alternating minimization from optimum



If know centroids $\mu_1 = 1$, $\mu_2 = 11$:

$z_1 = \arg\min\{(0-1)^2, (0-11)^2\} = 1$
$z_2 = \arg\min\{(2-1)^2, (2-11)^2\} = 1$
$z_3 = \arg\min\{(10-1)^2, (10-11)^2\} = 2$
$z_4 = \arg\min\{(12-1)^2, (12-11)^2\} = 2$

If know assignments $z_1 = z_2 = 1$, $z_3 = z_4 = 2$:

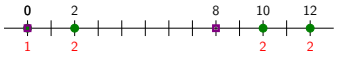
$\mu_1 = \arg\min_\mu (0-\mu)^2 + (2-\mu)^2 = 1$
$\mu_2 = \arg\min_\mu (10-\mu)^2 + (12-\mu)^2 = 11$

- Before we present the K-means algorithm, let us form some intuitions.
- Consider the following one-dimensional clustering problem with 4 points. Intuitively there are two clusters.
- Suppose we know the centroids. Then for each point the assignment that minimizes the K-means loss is the closer of the two centroids.
- Suppose we know the assignments. Then for each cluster, we average the points that are assigned to that cluster.

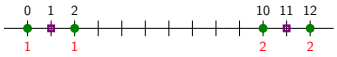
Alternating minimization from random initialization

Initialize $\boldsymbol{\mu}$:

Iteration 1:



Iteration 2:



Converged.

- But of course we don't know either the centroids or assignments.
- So we simply start with an arbitrary setting of the centroids.
- Then alternate between choosing the best assignments given the centroids, and choosing the best centroids given the assignments.
- This is the K-means algorithm.

K-means algorithm

Algorithm: K-means

Initialize $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]$ randomly.

For $t = 1, \ldots, T$:

Step 1: set assignments $\mathbf{z}$ given $\boldsymbol{\mu}$

For each point $i = 1, \ldots, n$:

$$z_i \leftarrow \arg\min_{k=1,\ldots,K} \|\phi(x_i) - \mu_k\|^2$$

Step 2: set centroids $\boldsymbol{\mu}$ given $\mathbf{z}$

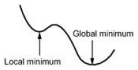
For each cluster $k = 1, \ldots, K$:

$$\mu_k \leftarrow \frac{1}{|\{i : z_i = k\}|} \sum_{i:z_i=k} \phi(x_i)$$

- Now we can state the K-means algorithm formally. We start by initializing all the centroids randomly. Then, we iteratively alternate back and forth between steps 1 and 2, optimizing $\mathbf{z}$ given $\boldsymbol{\mu}$ and vice-versa.
- **Step 1** of K-means fixes the centroids $\boldsymbol{\mu}$. Then we can optimize the K-means objective with respect to $\mathbf{z}$ alone quite easily. It is easy to show that the best label for z_i is the cluster k that minimizes the distance to the centroid μ_k (which is fixed).
- **Step 2** turns things around and fixes the assignments $\mathbf{z}$. We can again look at the K-means objective function and optimize it with respect to the centroids $\boldsymbol{\mu}$. The best μ_k is to place the centroid at the average of all the points assigned to cluster k.

Local minima

K-means is guaranteed to converge to a local minimum, but is not guaranteed to find the global minimum.



[demo: getting stuck in local optima, seed = 100]

Solutions:

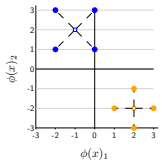
- Run multiple times from different random initializations
- Initialize with a heuristic (K-means++)

- K-means is guaranteed to decrease the loss function each iteration and will converge to a local minimum, but it is not guaranteed to find the global minimum, so one must exercise caution when applying K-means.
- Advanced: One solution is to simply run K-means several times from multiple random initializations and then choose the solution that has the lowest loss.
- Advanced: Or we could try to be smarter in how we initialize K-means. K-means++ is an initialization scheme which places centroids on training points so that these centroids tend to be distant from one another.

Summary

Clustering: discover structure in unlabeled data

K-means objective:



assignments $\mathbf{z}$

K-means algorithm:

centroids $\boldsymbol{\mu}$

Unsupervised learning use cases:

- Data exploration and discovery
- Providing representations to downstream supervised learning

- In summary, K-means is a simple and widely-used method for discovering cluster structure in data.
- Note that K-means can mean two things: the objective and the algorithm.
- Given points we define the K-means objective as the sum of the squared differences between a point and its assigned centroid.
- We also defined the K-means algorithm, which performs alternating optimization on the K-means objective.
- Finally, clustering is just one instance of unsupervised learning, which seeks to learn models from the wealth of unlabeled data alone. Unsupervised learning can be used in two ways: exploring a dataset which has not been labeled (let the data speak), and learning representations (discrete clusters or continuous embeddings) useful for downstream supervised applications.