

Clustering algorithms

Self Organizing Maps

Plan

1. Clustering
2. K-means
3. SOM
4. HAC
5. Example of applications

Clustering

Clustering is the classification of objects into different groups, or more precisely, the partitioning of a data set into subsets (clusters), so that the data in each subset (ideally) share some common trait - often according to some defined distance measure.

Types of clustering:

1. **Hierarchical algorithms**: these find successive clusters

using previously established clusters.

1. Agglomerative ("bottom-up"): Agglomerative algorithms begin with each element as a separate cluster and merge them into successively larger clusters.

2. Divisive ("top-down"): Divisive algorithms begin with the whole set and proceed to divide it into successively smaller clusters.

Types of clustering:

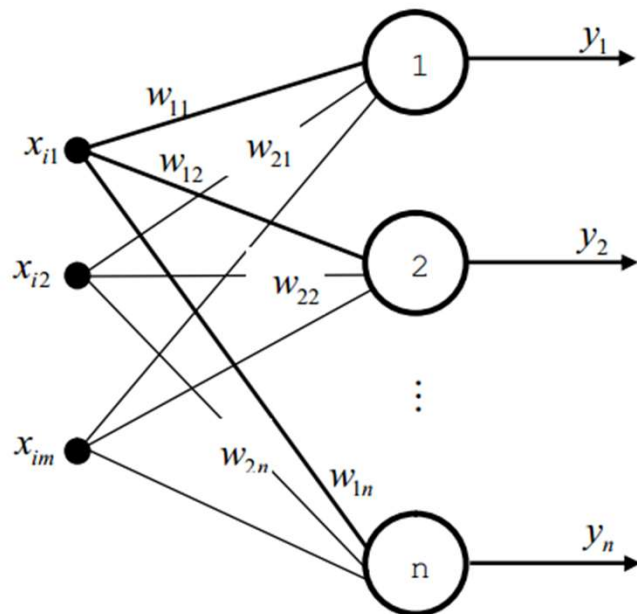
2. Partitional clustering: Partitional algorithms determine all clusters at once. They include:

***K*-means and SOM and others**

Fuzzy c-means clustering

QT clustering algorithm

SOM Unit



Unsupervised learning

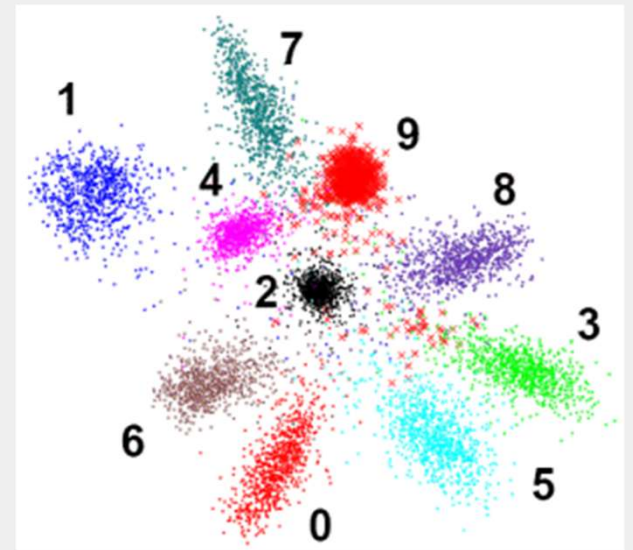
We have some dataset

Objects are independent

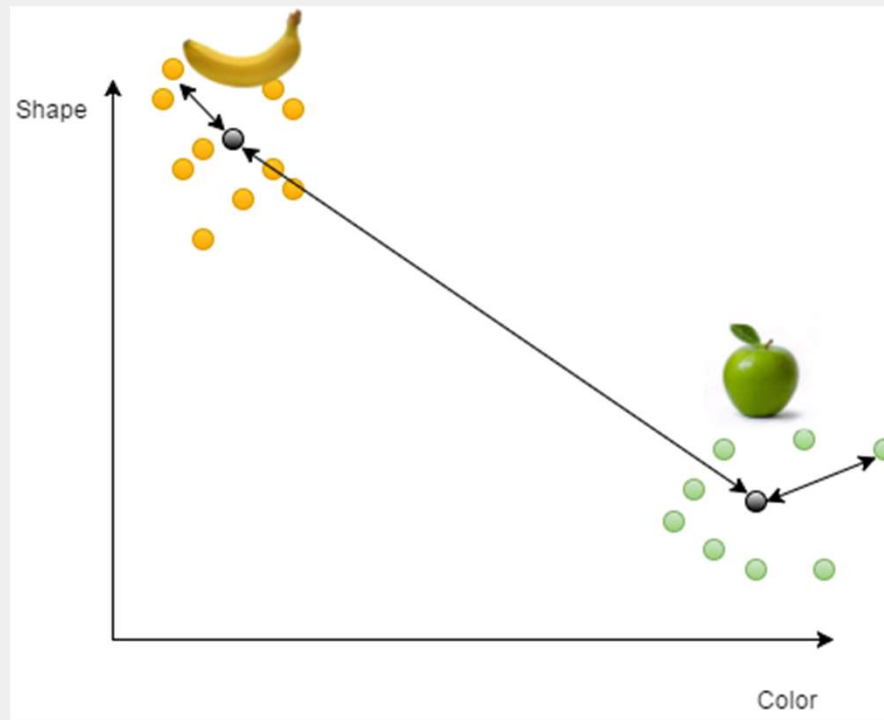
Obey some stat law

$$X_m = \{\mathbf{x}_1, \dots, \mathbf{x}_m\}$$

$$\mathbf{x}_i \sim P(\mathbf{x})$$

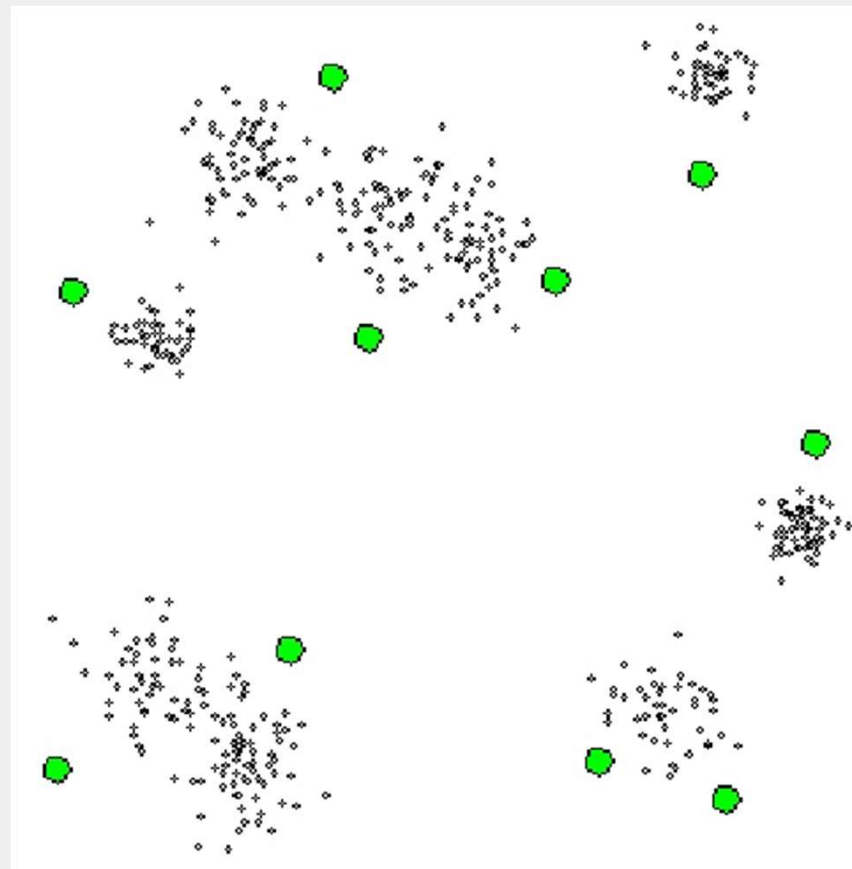


Unsupervised learning



Competitive Learning

Winner take all (WTA)



K-means Clustering

What is clustering?

Why would we want to cluster?

How would you determine clusters?

How can you do this efficiently?

K-means Clustering

Strengths

Simple iterative method

User provides “K”

Weaknesses

Often too simple → bad results

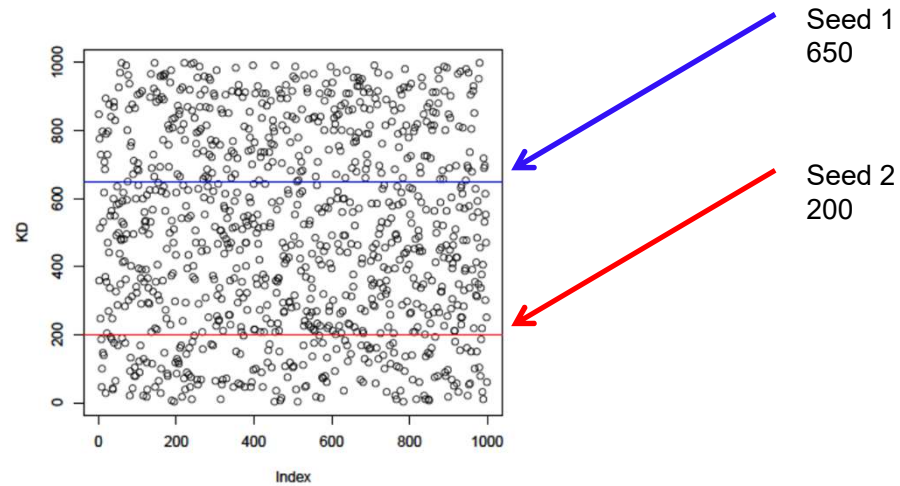
Difficult to guess the correct “K”

K-means Clustering

Basic Algorithm:

Step 0: select K

Step 1: randomly select initial cluster seeds



K-means Clustering

An initial cluster seed represents the “mean value” of its cluster.

In the preceding figure:

Cluster seed 1 = 650

Cluster seed 2 = 200

K-means Clustering

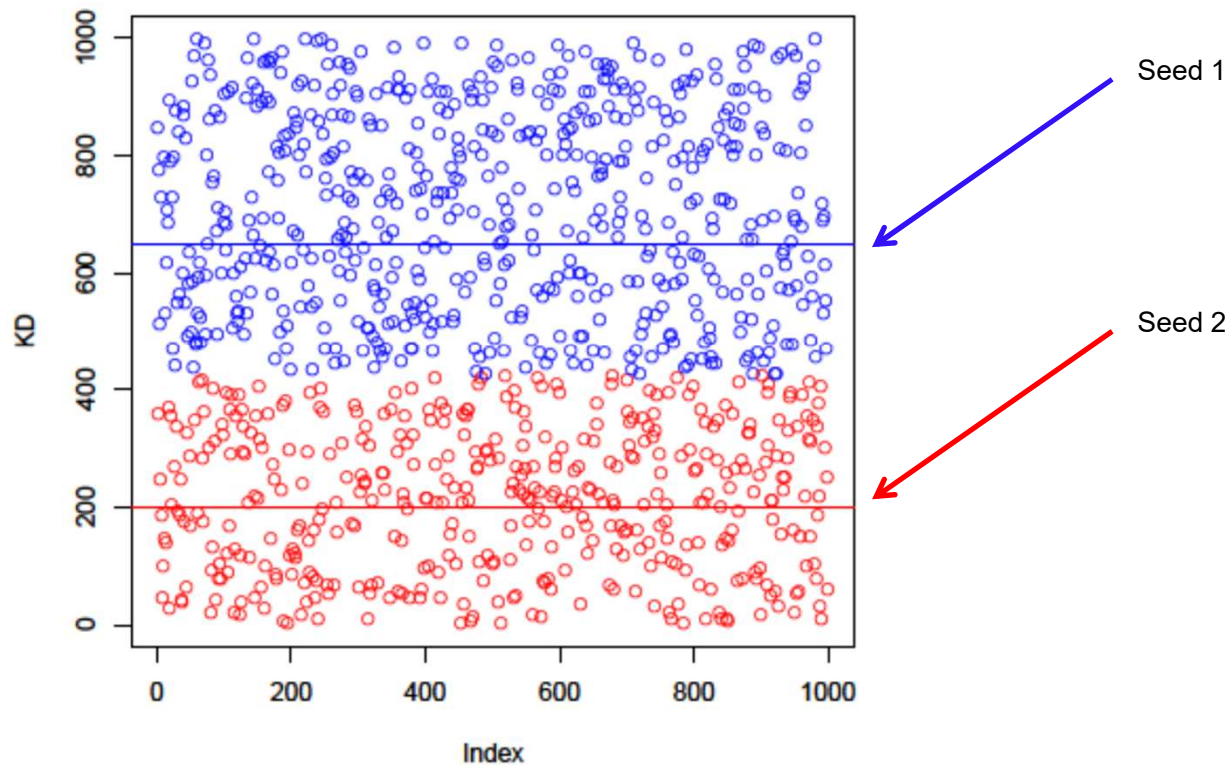
Step 2: calculate distance from each object to each cluster seed.

What type of distance should we use?

Squared Euclidean distance

Step 3: Assign each object to the closest cluster

K-means Clustering



K-means Clustering

Iterate:

Calculate distance from objects to cluster centroids.

Assign objects to closest cluster

Recalculate new centroids

Stop based on convergence criteria

No change in clusters

Max iterations

K-means Issues

Distance measure is squared Euclidean

Scale should be similar in all dimensions

Rescale data?

Not good for nominal data. Why?

Approach tries to minimize the within-cluster sum of squares error (WCSS)

Implicit assumption that SSE is similar for each group

WCSS

The over all WCSS is given by: $\sum_{i=1}^k \sum_{x \in C_i} \|x - \mu_i\|^2$

The goal is to find the smallest WCSS

Does this depend on the initial seed values?

Possibly.

Bottom Line

K-means

Easy to use

Need to know K

May need to scale data

Good initial method

Local optima

No guarantee of optimal solution

Repeat with different starting values

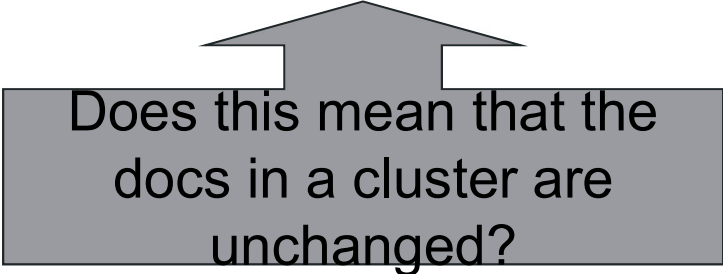
Termination conditions

Several possibilities, e.g.,

A fixed number of iterations.

Doc partition unchanged.

Centroid positions don't change.



Does this mean that the
docs in a cluster are
unchanged?

Convergence

Why should the K -means algorithm ever reach a *fixed point*?

A state in which clusters don't change.

K -means is a special case of a general procedure known as the *Expectation Maximization (EM) algorithm*.

EM is known to converge.

Theoretically, number of iterations could be large.

Typically converges quickly

Time Complexity

Computing distance between doc and cluster is $O(m)$ where m is the dimensionality of the vectors.

Reassigning clusters: $O(Kn)$ distance computations, or $O(Knm)$.

Computing centroids: Each doc gets added once to some centroid: $O(nm)$.

Assume these two steps are each done once for I iterations: $O(IKnm)$.

Seed Choice

Results can vary based on random seed selection.

Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.

Select good seeds using a heuristic (e.g., doc least similar to any existing mean)

Try out multiple starting points

Initialize with the results of another method.

How Many Clusters?

Number of clusters K is given

Partition n docs into predetermined number of clusters

Finding the “right” number of clusters is part of the problem

Given data, partition into an “appropriate” number of subsets.

E.g., for query results - ideal value of K not known up front - though UI may impose limits.

Can usually take an algorithm for one flavor and convert to the other.

K not specified in advance

Say, the results of a query.

Solve an optimization problem: penalize having lots of clusters

application dependent, e.g., compressed summary of search results list.

Tradeoff between having more clusters (better focus within each cluster) and having too many clusters

K not specified in advance

Given a clustering, define the Benefit for a doc to be some inverse distance to its centroid

Define the Total Benefit to be the sum of the individual doc Benefits.

Penalize lots of clusters

For each cluster, we have a Cost C .

Thus for a clustering with K clusters, the Total Cost is KC .

Define the Value of a clustering to be =

Total Benefit - Total Cost.

Find the clustering of highest value, over all choices of K .

Total benefit increases with increasing K . But can stop when it doesn't increase by "much". The Cost term enforces this.

What's SOM?

Neural network for unstructured data



Kohonen, Teuvo (1982). "Self-Organized Formation of Topologically Correct Feature Maps". *Biological Cybernetics* 43 (1): 59–69. doi:10.1007/bf00337288.

WEBSOM

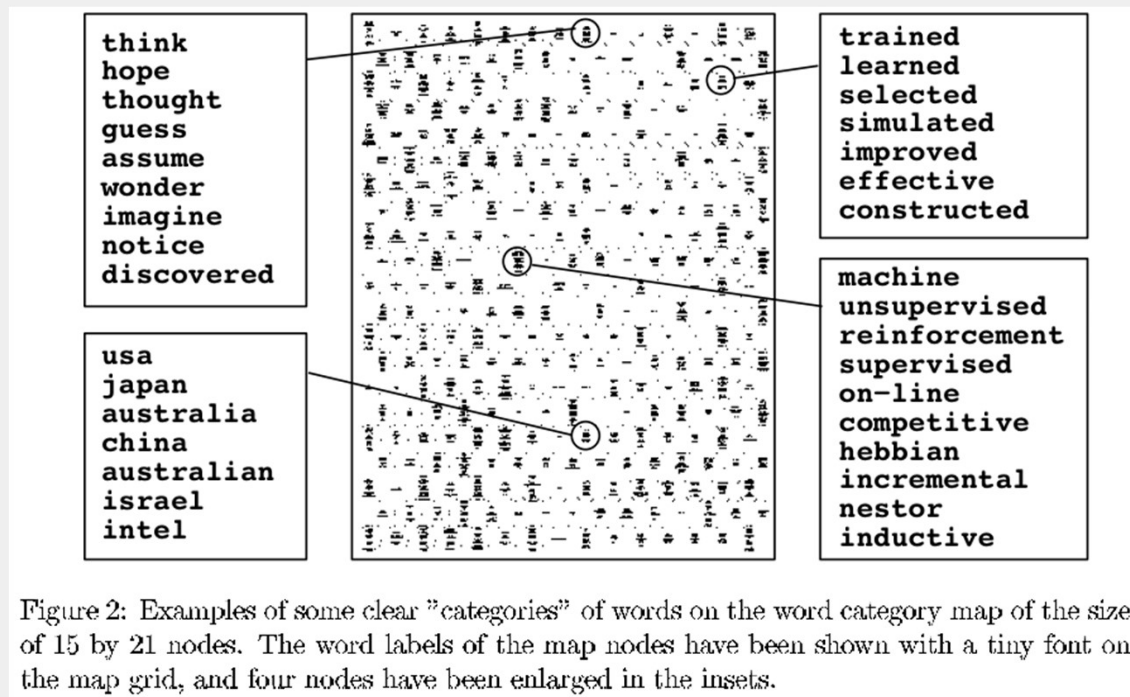
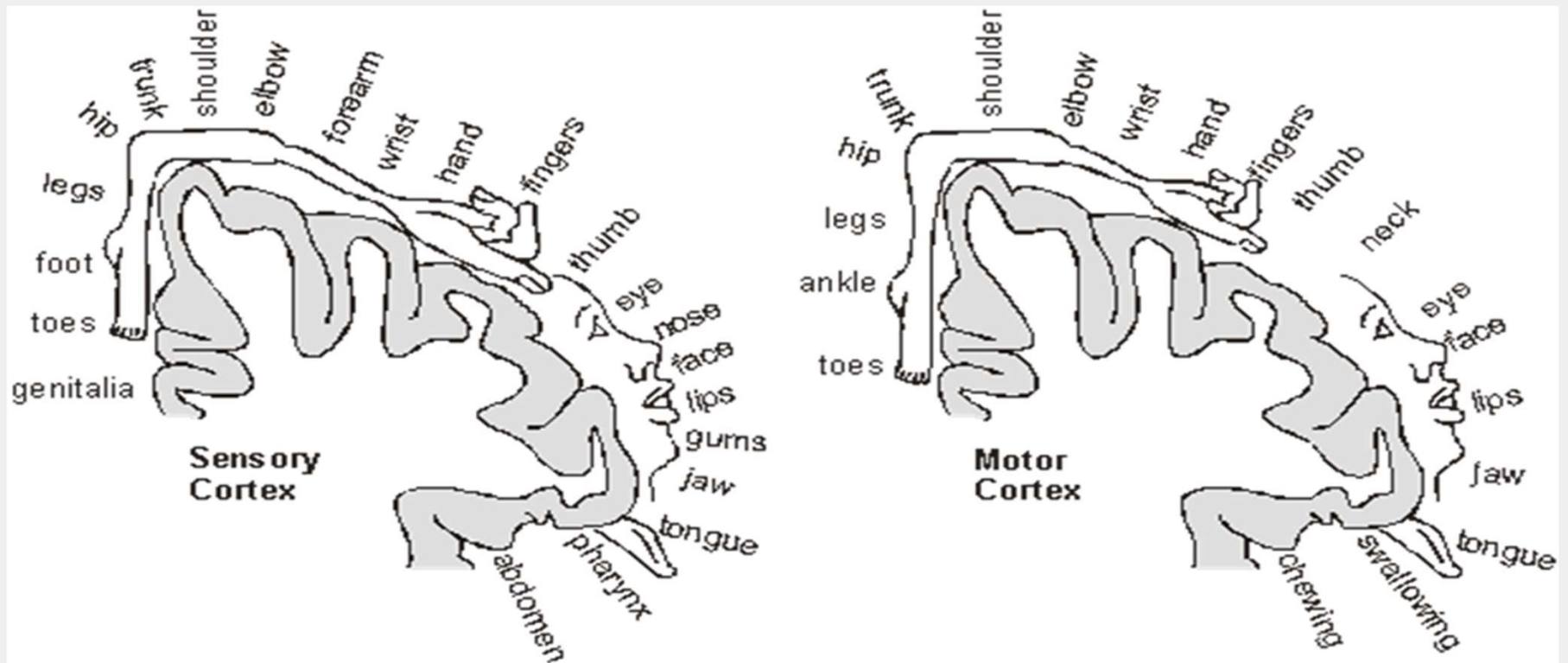
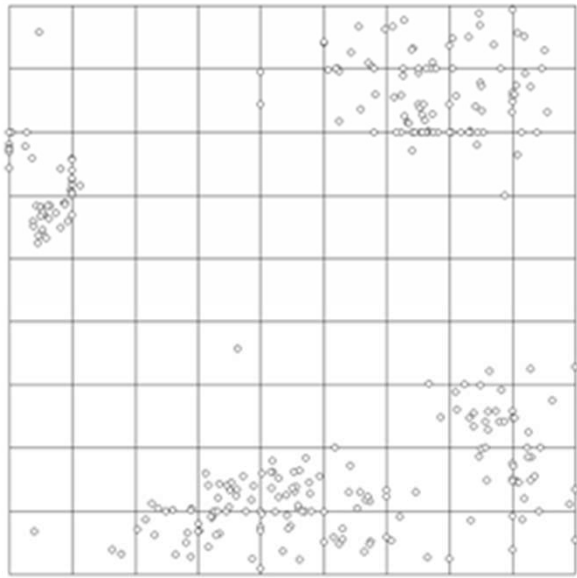


Figure 2: Examples of some clear "categories" of words on the word category map of the size of 15 by 21 nodes. The word labels of the map nodes have been shown with a tiny font on the map grid, and four nodes have been enlarged in the insets.

Humuncule



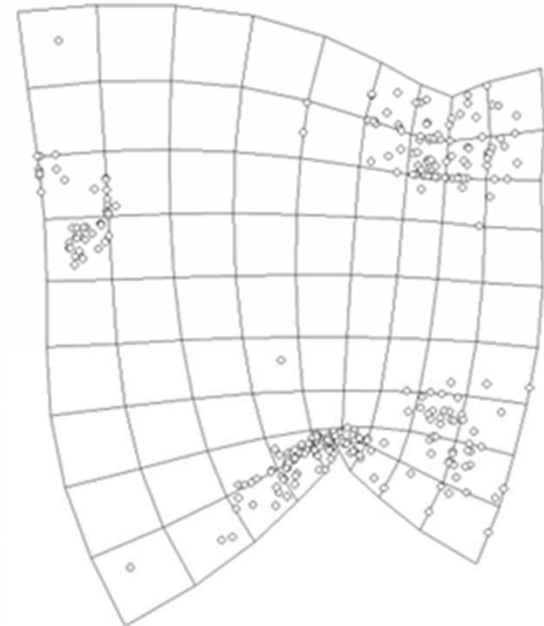
SOM Grid



a)



b)



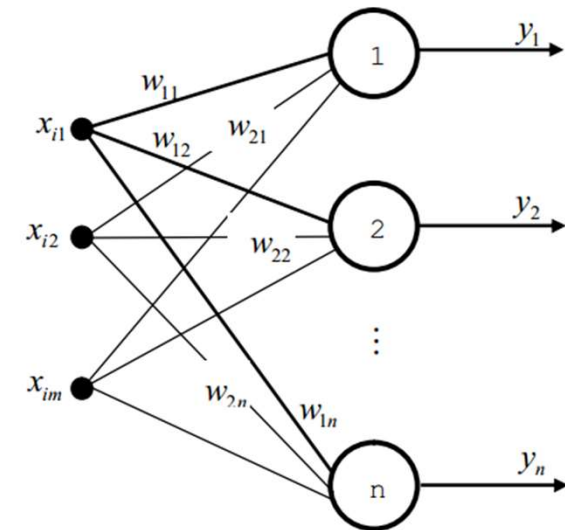
B)

Algo

1. Create a set of vectors organized in a $1/2 \times$ dimensional array to initiate them with random values.
2. Represent the input of the vector map from the training sample. Determine the winner.
3. Stimulate the winner and maybe some of his friends!
4. Repeat steps 1 through 3 many times over.
5. The result is a weighted kahonen card

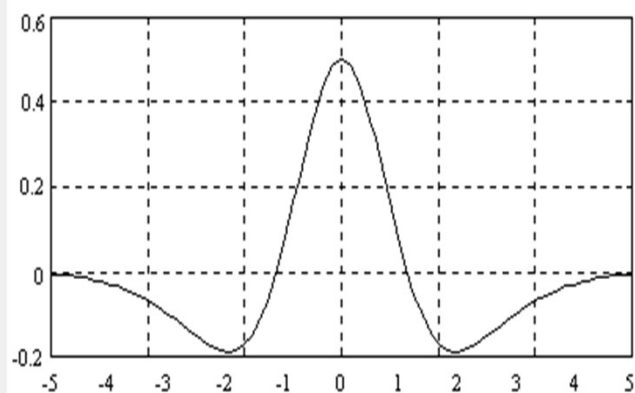
Algo

$$\|x(n) - w_b(n)\| = \min_{i \in V_M} \{\|x - w_i(n)\|\} , \quad (1)$$

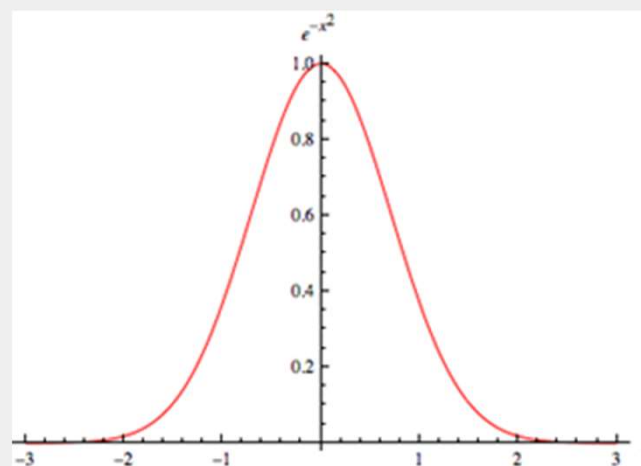


$$w_i(n+1) = w_i(n) + \gamma(n)h_{ib}(n)(x(n) - w_i(n)) , \quad (2)$$

Algo



$$\psi(t) = \frac{2}{\sqrt{3\sigma\pi^{\frac{1}{4}}}} \left(1 - \frac{t^2}{\sigma^2}\right) e^{\frac{-t^2}{2\sigma^2}}$$



$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

Рисунок 2. (Слева) Вейвлет мексиканская шляпа. (Справа) Гауссиана. Используются в качестве функции соседства между нейронами SOM.

Algo

- Algorithm neural network - competitive training without a teacher
- It is used for visualization of many-dimensional data and finding hidden statistically significant regularities.
- The neural network forms a flat lattice with various variations or a chain.
- Net of the hidden layer. The input and output layer is the same.
- Lateral connections between neurons are preserved

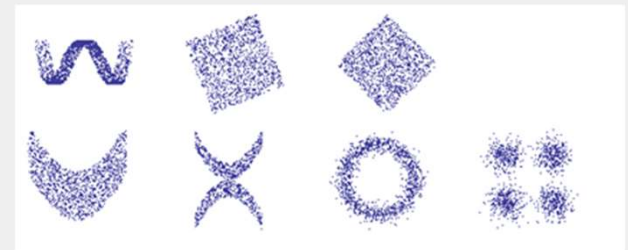
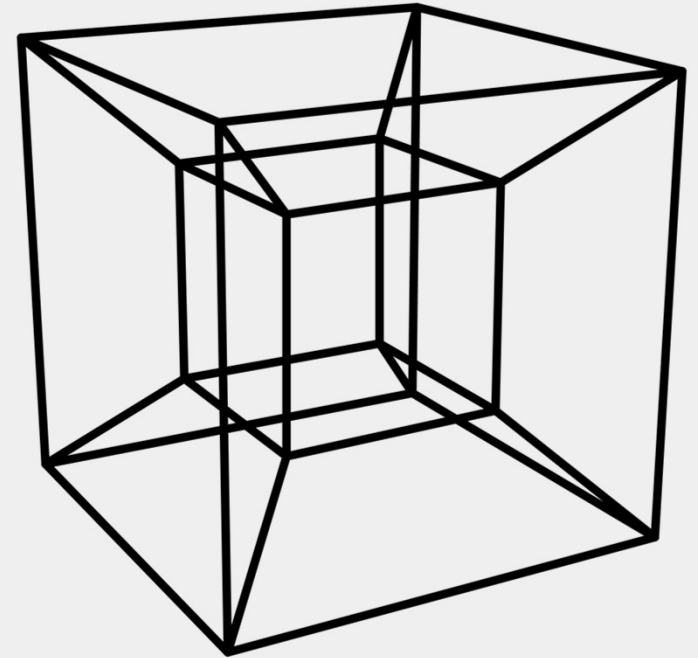
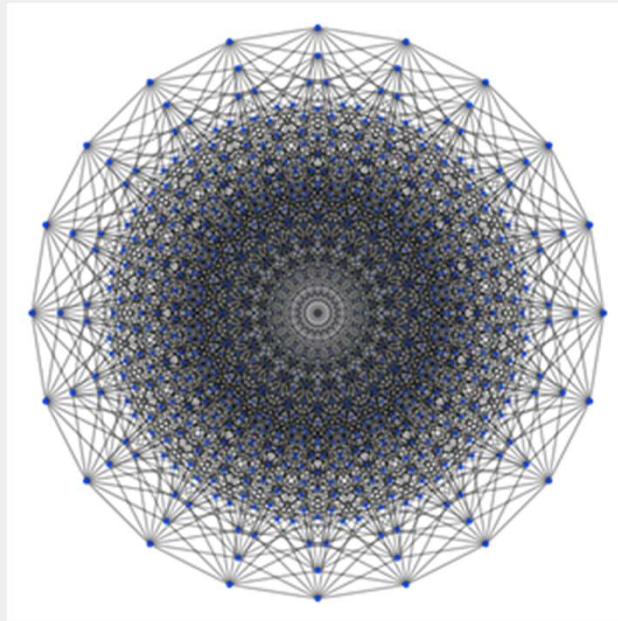
Shortcomings

1. Dead neurons - could not capture data
2. Boundary effects - pulling the grid to the center of the cloud

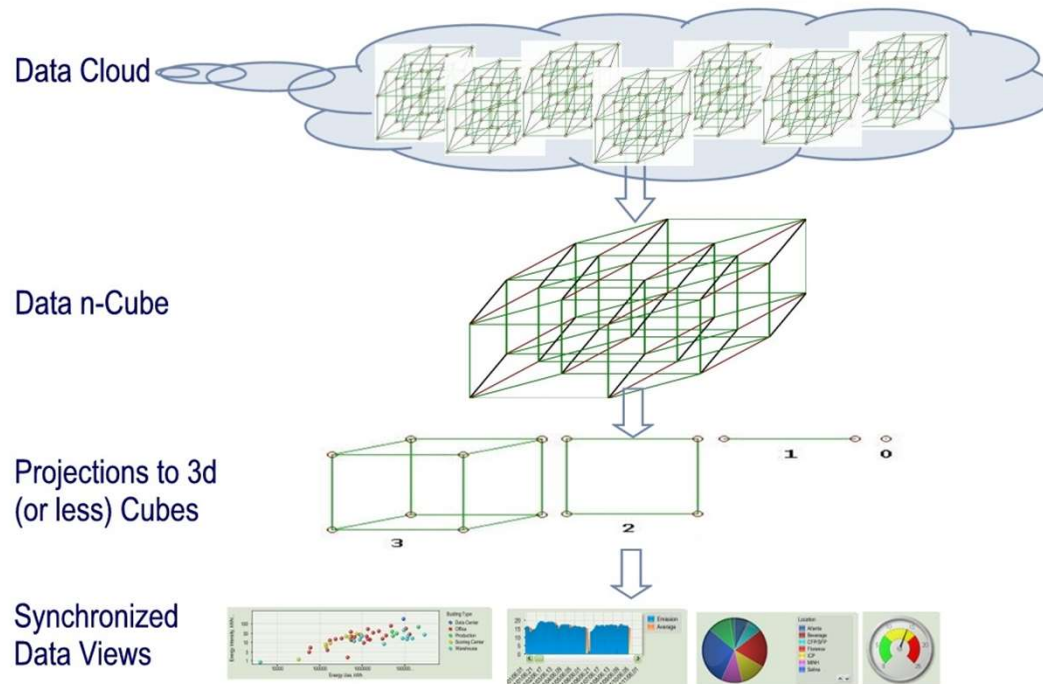
Data

3. Slim points in the data cloud Can be displayed on opposite edges of the map.

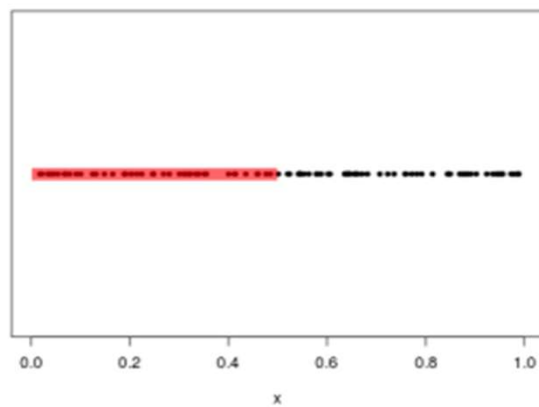
Reduction of dimension



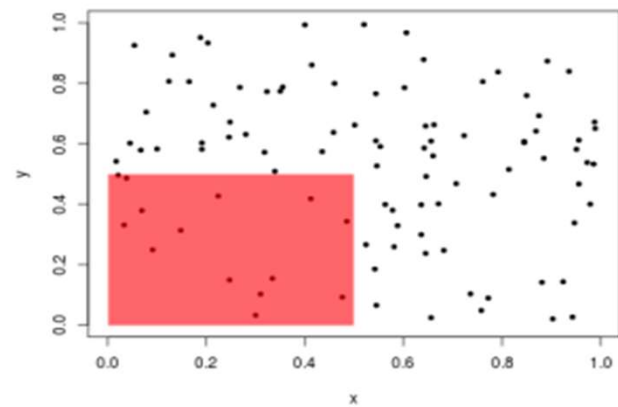
Dimension reduction



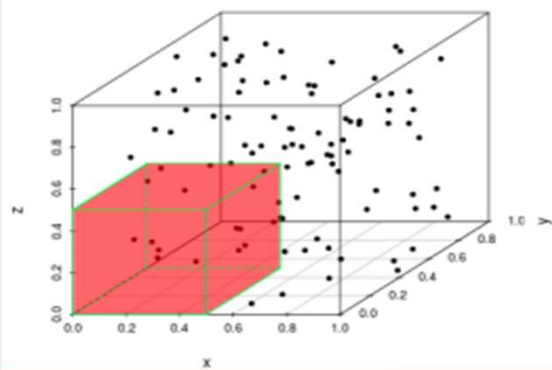
1-D: 42% of data captured.



2-D: 14% of data captured.

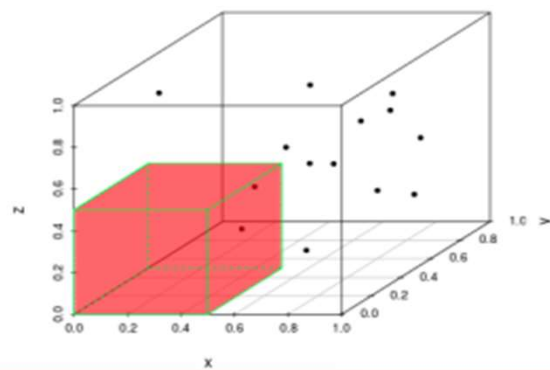


3-D: 7% of data captured.

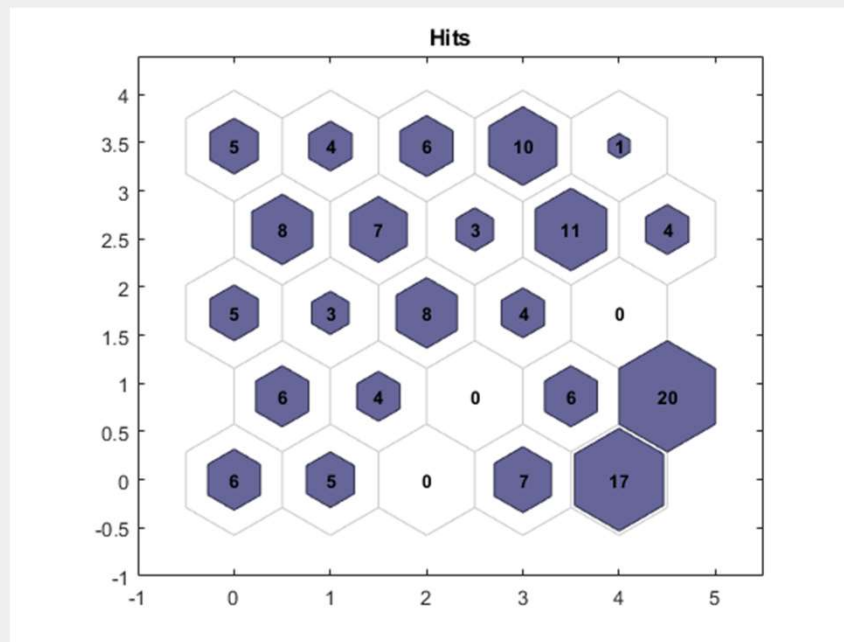


4-D: 3% of data captured.

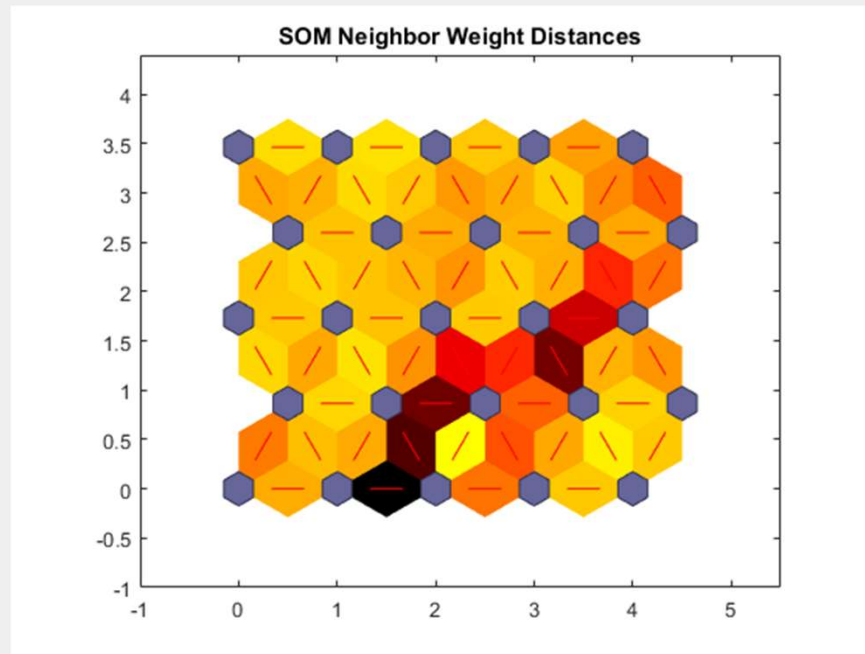
$t = 0$



Visualization



Visualization



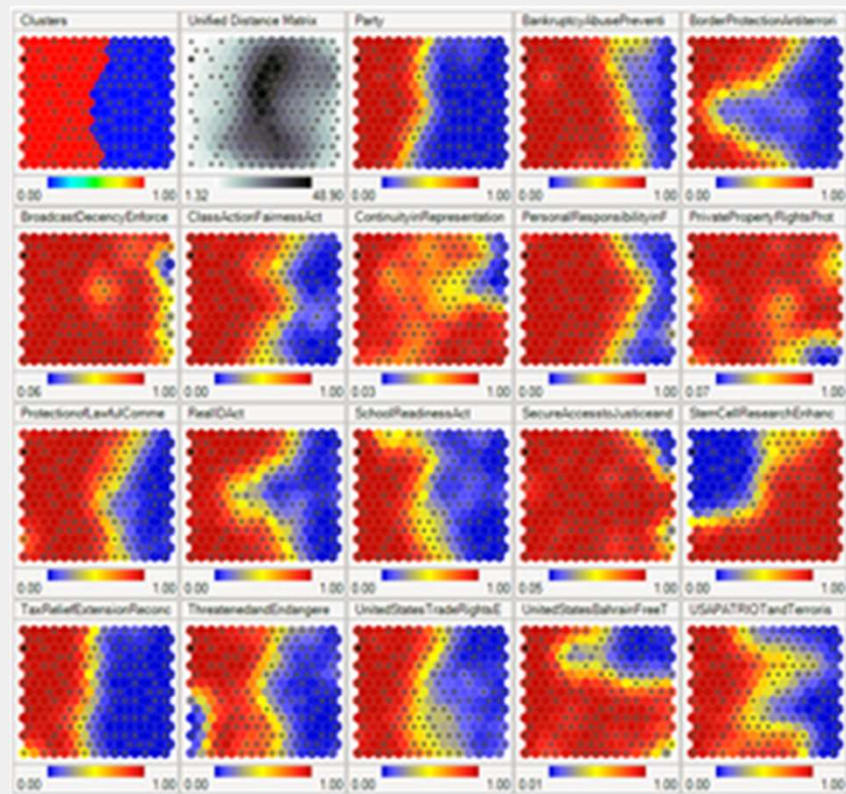
Visualization



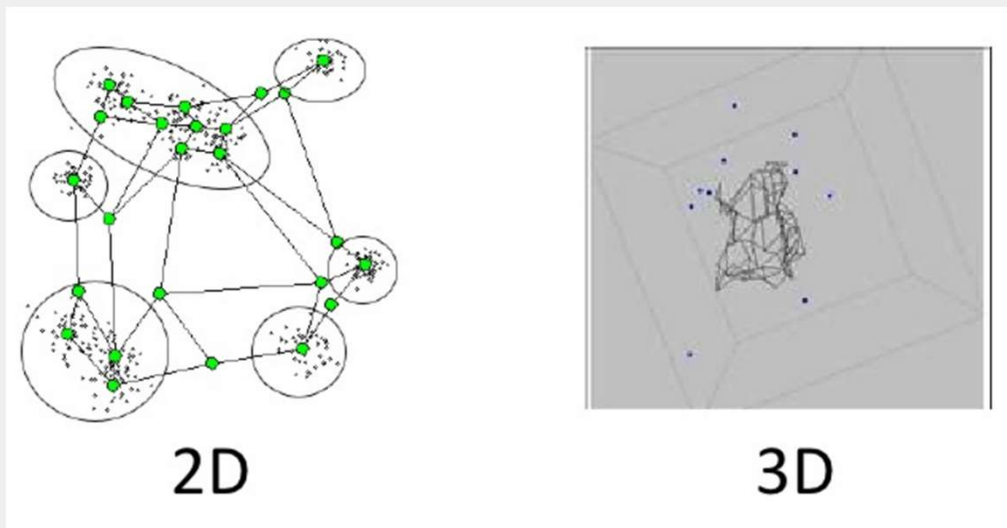
**Синий более низкое
значение**

Красное более высокое значение

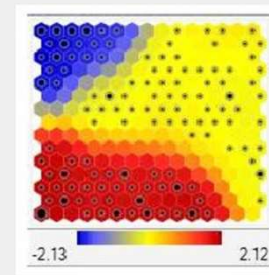
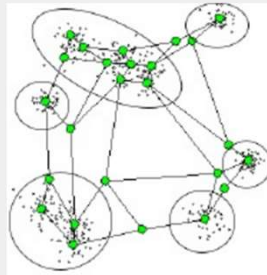
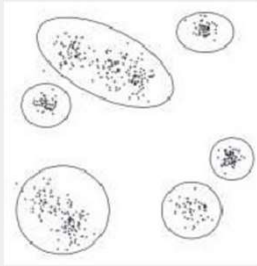
Visualization



Visualization



Visualization



$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ \vdots \\ \vdots \\ x_n \end{bmatrix} \quad \longrightarrow \quad \mathbf{BMU} = \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ \vdots \\ \vdots \\ u_n \end{bmatrix} \quad \longrightarrow \quad \mathbf{U} = [u_x, u_y]$$

Example of application

Поиск:

[Найти](#)

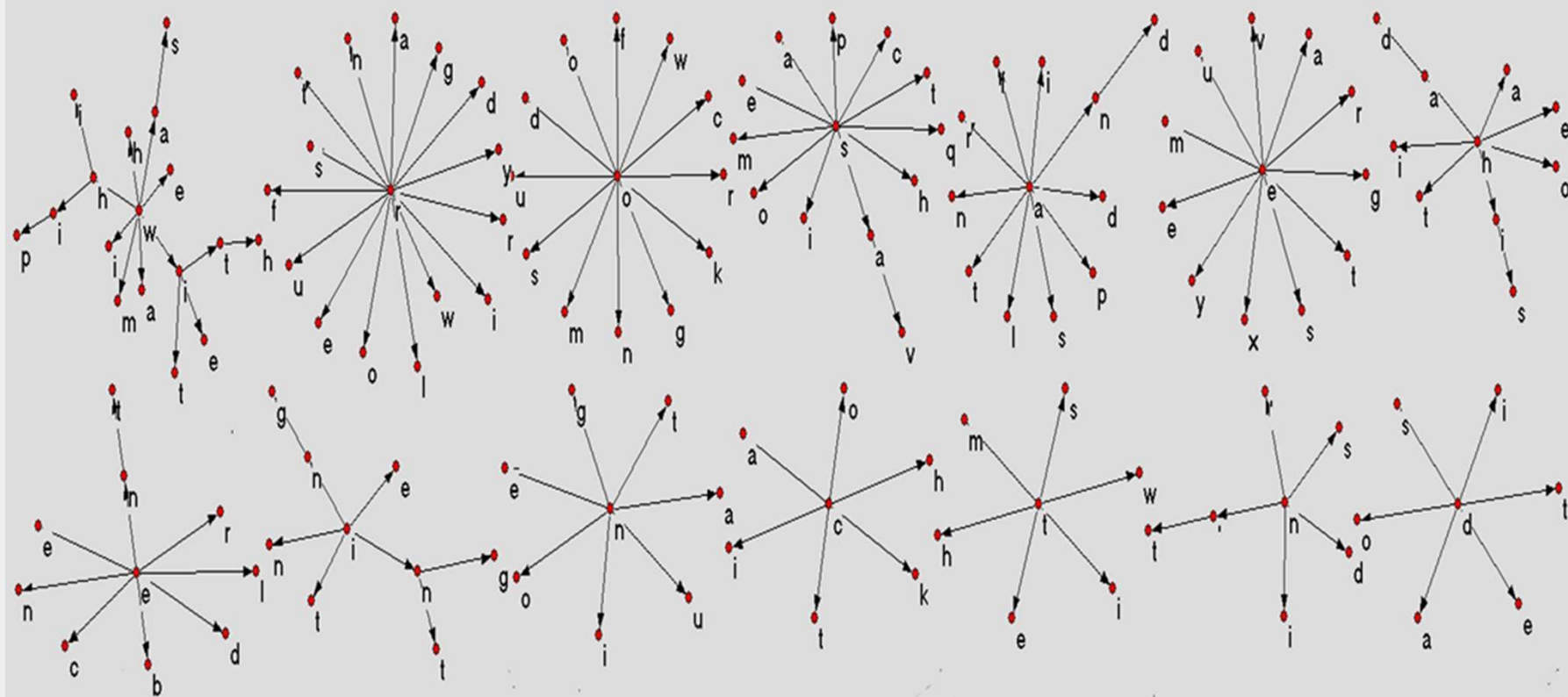
Выдача

ID	Текст статьи
http://365info...	Любителям Тенгрианства любите и верьте во что вы хотите но не н...
http://365info...	убивают еще и второго "зайца" – дискредитируют исламскую рели...
http://365info...	убивают еще и второго "зайца" – дискредитируют исламскую религию, потому что масс...
http://365info...	Мусульманин это человек придерживающийся религии ИСЛАМ , то
http://365info...	Не нужно мешать ссда историю и ИСЛАМ , прикрывая свою похоть
http://365info...	с удовольствием бы провел время в уютном кафе , угостив друга вк
http://365info...	Запад хочет удобного ИСЛАМА
http://365info...	Восстановить Аральское море в полной мере уже не представляется
http://365info...	Я слабо в ИСЛАМЕ разбираюсь , но даже моих слабых знаний доста...
http://365info...	Созидающая сила новой религии была такова , что всего лишь за дв...

Характерные слова

Классификация

Example of application



Example of application

Снижение размерности изображения в задаче распознавания лиц.

SOM показал себя лучше для заданного набора изображений, чем другие методы (PCA, ICA)

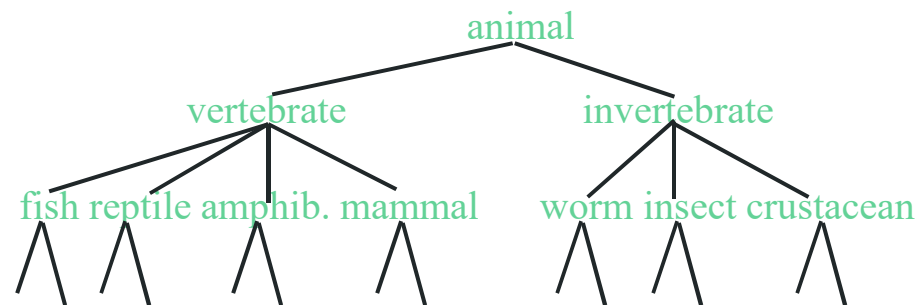
Быстродействие системы снижается в случае увеличения и



Journal of Multimedia, May 2008 by Dinesh Kumar, C.S. Rai, Shakti Kumar

Hierarchical Clustering

Build a tree-based hierarchical taxonomy (*dendrogram*) from a set of documents.



How could you do this with k-means?

Hierarchical Clustering algorithms

Agglomerative (bottom-up):

Start with each document being a single cluster.

Eventually all documents belong to the same cluster.

Divisive (top-down):

Start with all documents belong to the same cluster.

Eventually each node forms a cluster on its own.

Could be a recursive application of k-means like algorithms

Hierarchical Agglomerative Clustering (HAC)

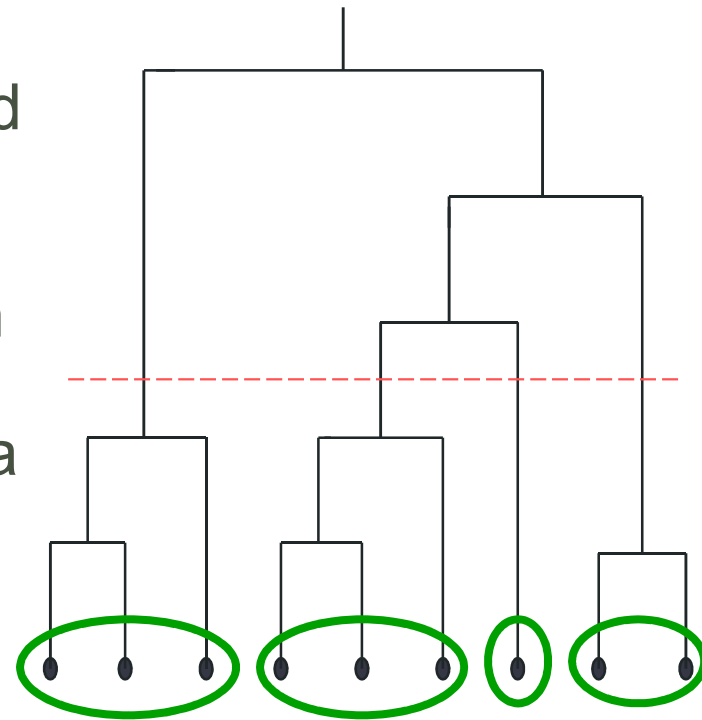
Assumes a similarity function for determining the similarity of two instances.

Starts with all instances in a separate cluster and then repeatedly joins the two clusters that are most similar until there is only one cluster.

The history of merging forms a binary tree or hierarchy.

Dendrogram: Hierarchical Clustering

- Clustering obtained by cutting the dendrogram at a desired level: each **connected** component forms a cluster.



Hierarchical Agglomerative Clustering (HAC)

Starts with each doc in a separate cluster

then repeatedly joins the closest pair of clusters, until there is only one cluster.

The history of merging forms a binary tree or hierarchy.

How to measure distance of clusters??

Closest pair of clusters

Many variants to defining closest pair of clusters

Single-link

Distance of the “*closest*” *points* (single-link)

Complete-link

Distance of the “furthest” points

Centroid

Distance of the centroids (centers of gravity)

(Average-link)

Single Link Agglomerative Clustering

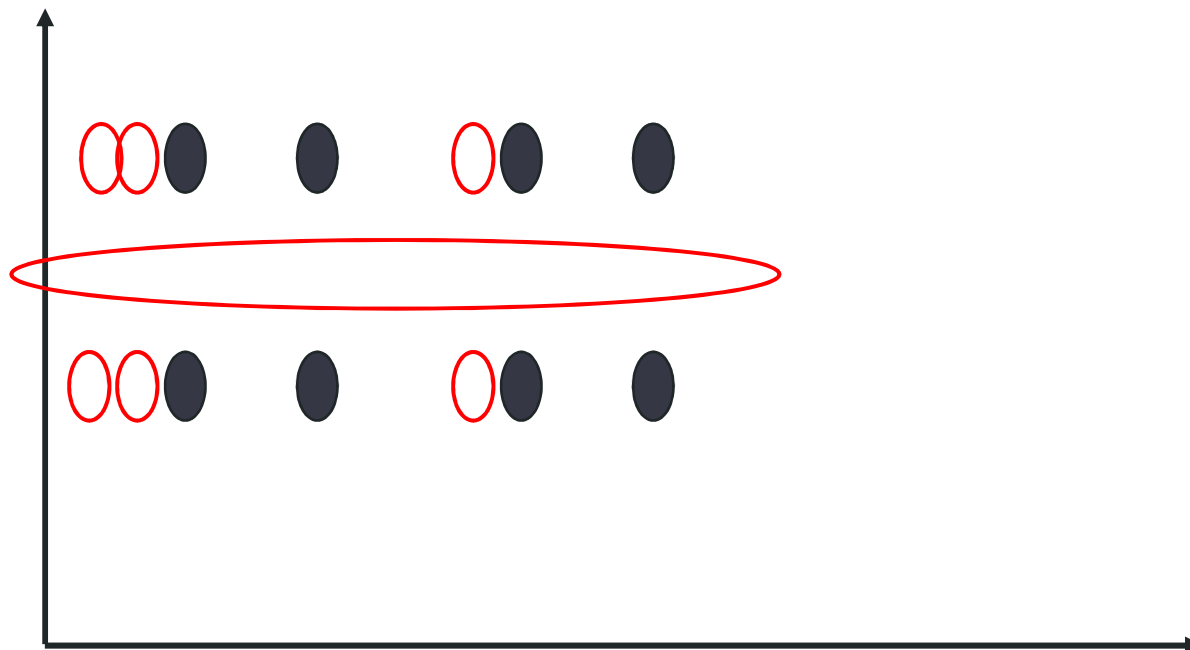
Use maximum similarity of pairs:

$$\textit{sim}(c_i, c_j) = \max_{x \in c_i, y \in c_j} \textit{sim}(x, y)$$

Can result in “straggly” (long and thin) clusters due to chaining effect. $\textit{sim}((c_i \cup c_j), c_k) = \max(\textit{sim}(c_i, c_k), \textit{sim}(c_j, c_k))$

After merging c_i and c_j , the similarity of the resulting cluster to another cluster, c_k , is:

Single Link Example



Complete Link Agglomerative Clustering

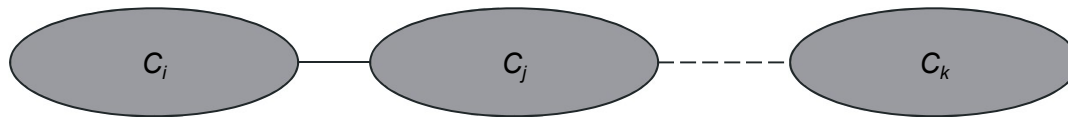
Use minimum similarity of pairs:

$$\text{sim}(c_i, c_j) = \min_{x \in c_i, y \in c_j} \text{sim}(x, y)$$

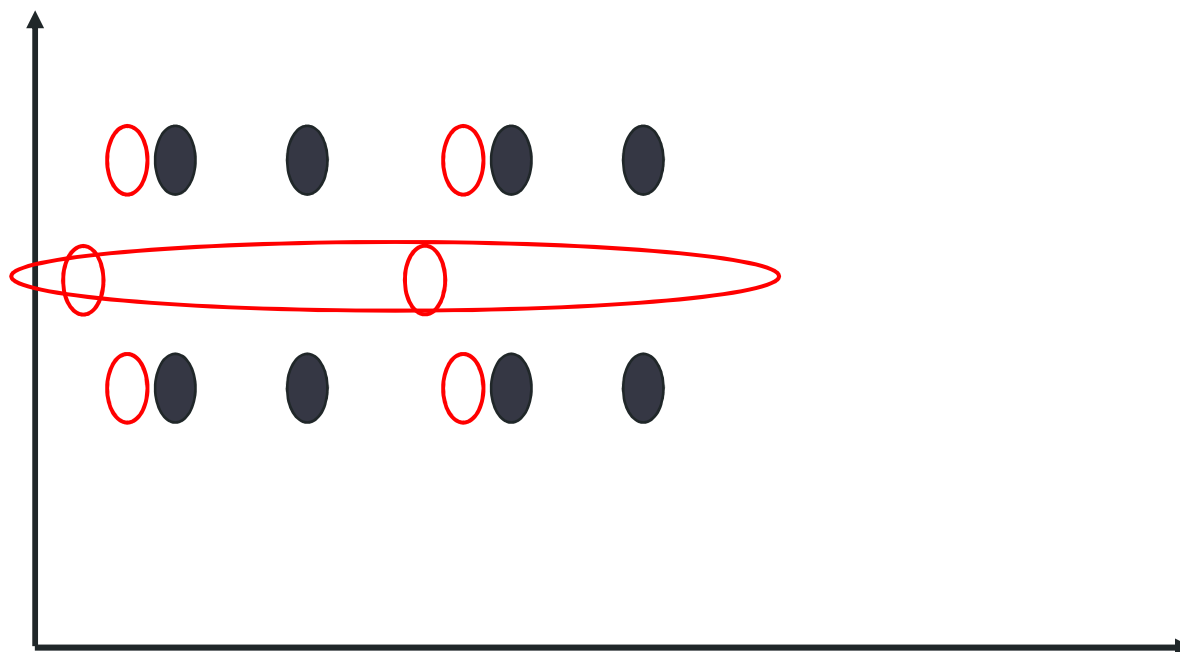
Makes “tighter,” spherical clusters that are typically preferable.

After merging c_i and c_j , the similarity of the resulting cluster to another cluster, c_k , is:

$$\text{sim}((c_i \cup c_j), c_k) = \min(\text{sim}(c_i, c_k), \text{sim}(c_j, c_k))$$



Complete Link Example



Key notion: *cluster representative*

We want a notion of a representative point in a cluster

Representative should be some sort of “typical” or central point in the cluster, e.g.,

point inducing smallest radii to docs in cluster

smallest squared distances, etc.

point that is the “average” of all docs in the cluster

Centroid or center of gravity

Centroid-based Similarity

Always maintain average of vectors in each cluster:

$$\vec{s}(c_j) = \frac{\sum_{\vec{x} \in c_j} \vec{x}}{|c_j|}$$

Compute similarity of clusters by:

$$\textit{sim}(c_i, c_j) = \textit{sim}(s(c_i), s(c_j))$$

Computational Complexity

In the first iteration, all HAC methods need to compute similarity of all pairs of n individual instances which is $O(mn^2)$.

In each of the subsequent $n-2$ merging iterations, compute the distance between the most recently created cluster and all other existing clusters.

Maintaining of heap of distances allows this to be $O(mn^2 \log n)$

Major issue - labeling

After clustering algorithm finds clusters - how can they be useful to the end user?

Need pithy label for each cluster

In search results, say “Animal” or “Car” in the *jaguar* example.

In topic trees, need navigational cues.

Often done by hand, a posteriori.

How would you do this?

How to Label Clusters

Show titles of typical documents

Titles are easy to scan

Authors create them for quick scanning!

But you can only show a few titles which may not fully represent cluster

Show words/phrases prominent in cluster

More likely to fully represent cluster

Use distinguishing words/phrases

Differential labeling

But harder to scan

Labeling

Common heuristics - list 5-10 most frequent terms in the centroid vector.

Drop stop-words; stem.

Differential labeling by frequent terms

Within a collection “Computers”, clusters all have the word **computer** as frequent term.

Discriminant analysis of centroids.

Perhaps better: distinctive noun phrase

Scaling up to large datasets

Fahlman, Scott & Lebiere, Christian (1989). The cascade-correlation learning architecture. In Touretzky, D., editor, Advances in Neural Information Processing Systems (volume 2), (pp. 524-532), San Mateo, CA. Morgan Kaufmann.

Fahlman, S.E. and Lebiere, C., “The Cascade Correlation Learning Architecture,” NIPS, Vol. 2, pp. 524-532, Morgan Kaufmann, 1990.

Fahlman, S. E. (1991) The recurrent cascade-correlation learning architecture. In Lippman, R.P. Moody, J.E., and Touretzky, D.S., editors, NIPS 3, 190-205.

Efficient large-scale clustering

17M biomedical papers in Medline

Each paper contains ~20 citations

Clustering 340M citations

$\sim 10^{17}$ distance calculations for naïve HAC

Expensive Distance Metric for Text

String edit distance

Compute with dynamic programming

Costs for character:

insertion

deletion

substitution

...

	S	e	c	a	t	
S c o t t	0.0	0.7	1.4	2.1	2.8	3.5
	0.7	0.0	0.7	1.1	1.4	1.8
	1.4	0.7	1.0	0.7	1.4	1.8
	2.1	1.1	1.7	1.4	1.7	2.4
	2.8	1.4	2.1	1.8	2.4	1.7
	3.5	1.8	2.4	2.1	2.8	2.4

String edit (Levenshtein) distance

Distance is **shortest sequence of edit commands** that transform s to t .

Simplest set of operations:

Copy character from s over to t

Delete a character in s (cost 1)

Insert a character in t (cost 1)

Substitute one character for another (cost 1)

Levenshtein distance - example

distance("William Cohen", "Willliam Cohon")

[illegible]

Levenshtein distance - example

distance("William Cohen", "Willliam Cohon")

[illegible]

Computing Levenstein distance

$D(i,j)$ = score of **best** alignment from $s1..si$ to $t1..tj$

$$= \min \left\{ \begin{array}{ll} D(i-1,j-1), \text{ if } s_i=t_j & //copy \\ D(i-1,j-1)+1, \text{ if } s_i \neq t_j & //substitute \\ D(i-1,j)+1 & //insert \\ D(i,j-1)+1 & //delete \end{array} \right.$$

Computing Levenstein distance

	C	O	H	E	N
M	1	2	3	4	5
C	1	2	3	4	5
C	2	2	3	4	5
O	3	2	3	4	5
H	4	3	2	3	4
N	5	4	3	3	3

= $D(s, t)$

Computing Levenshtein distance

A *trace* indicates where the min value came from, and can be used to find edit operations and/or a best *alignment* (may be more than 1)

	C	O	H	E	N
M	1	2	3	4	5
C	1↑	2	3	4	5
C	2↑	3	3	4	5
O	3↖	2	3	4	5
H	4	3↖	2	3	4
N	5	4	3←	3↖	3

Large Clustering Problems

Many examples

Many clusters

Many dimensions

Example Domains

- Text
- Images
- Protein structure

The Canopies Approach

Two distance metrics: cheap & expensive

First Pass

very inexpensive distance metric

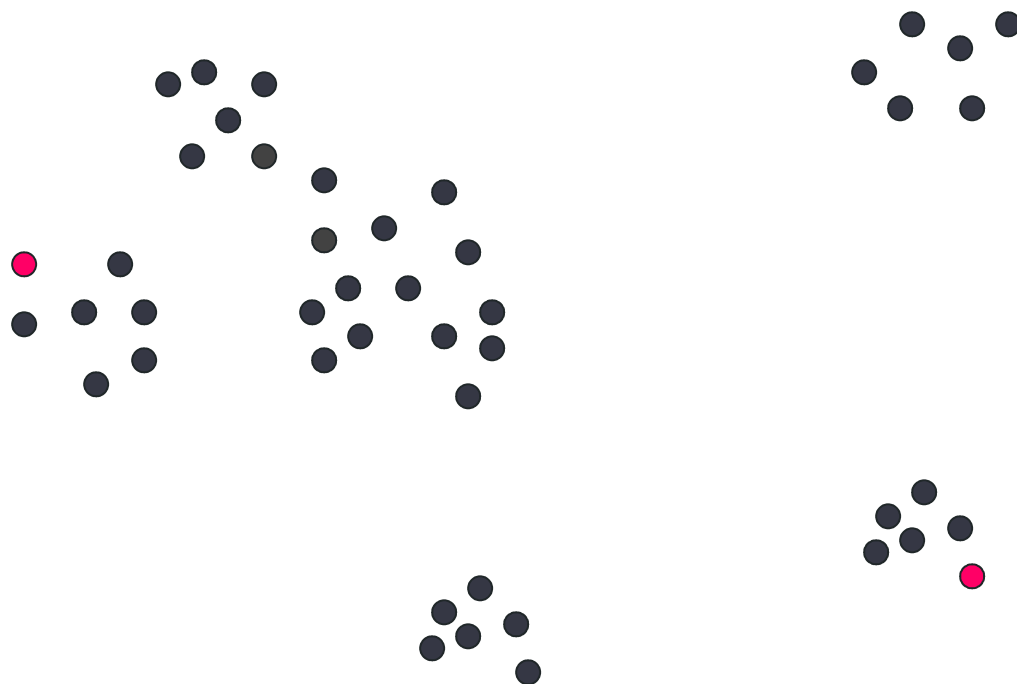
create overlapping canopies

Second Pass

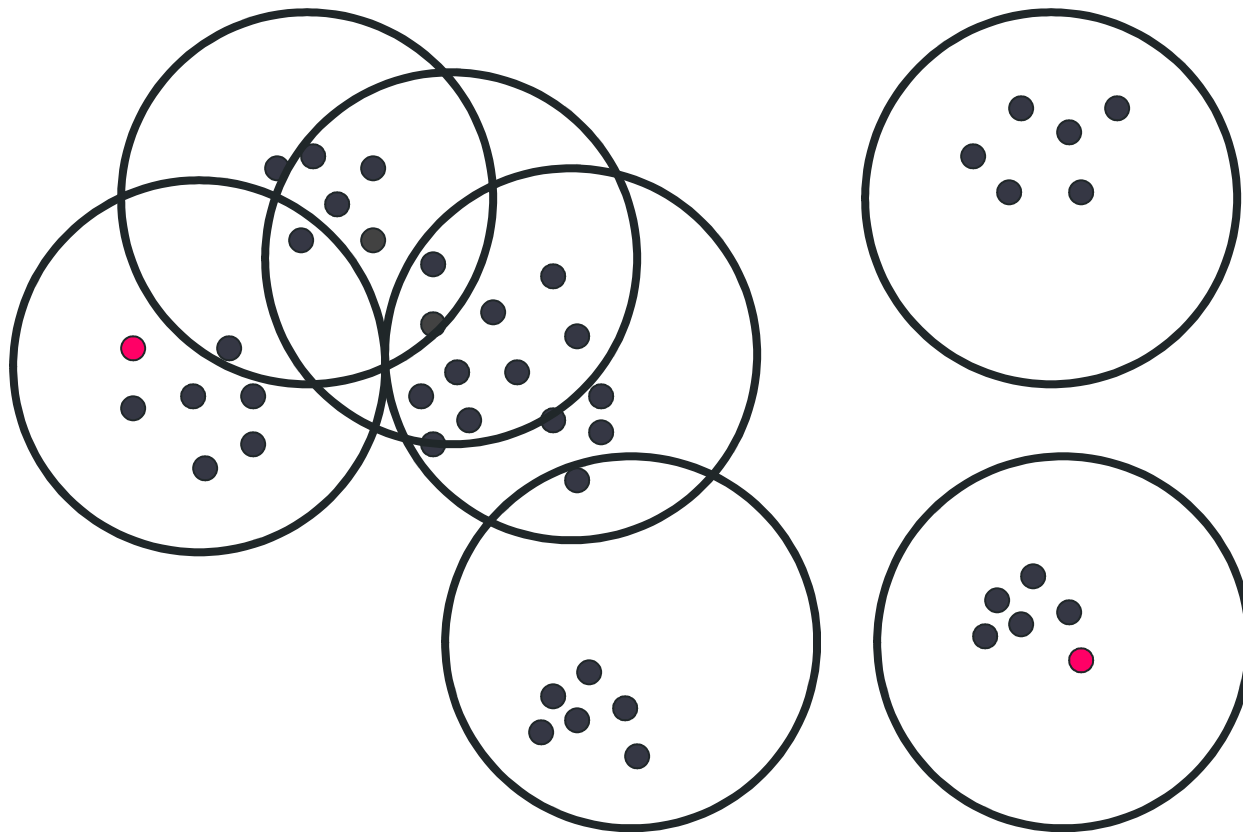
expensive, accurate distance metric

canopies determine which distances calculated

Illustrating Canopies



Overlapping Canopies



Creating canopies with two thresholds

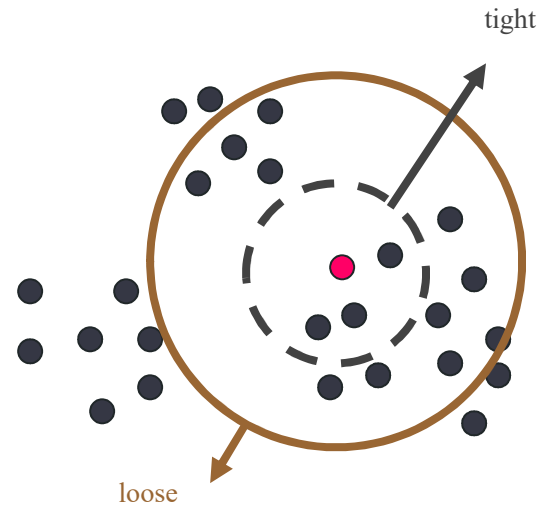
Put all points in D

Loop:

Pick a point X from D

Put points within
 K_{loose} of X in canopy

Remove points within K_{tight} of X from D

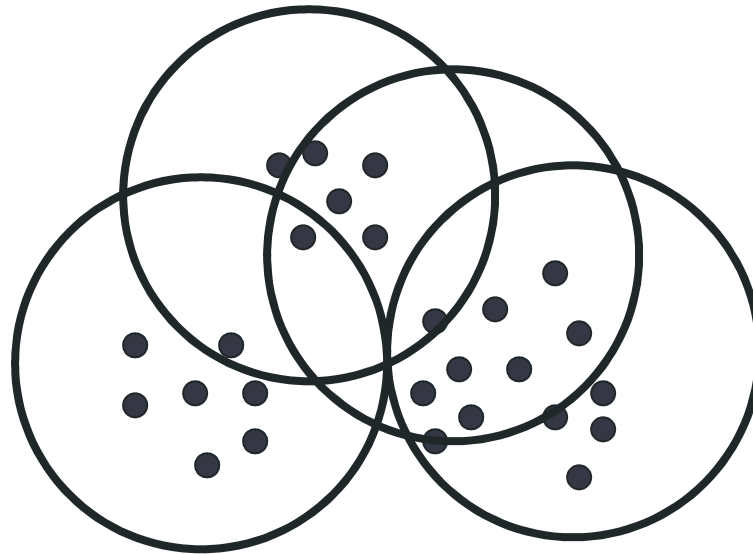


Using canopies with HAC

Calculate expensive distances
between points in the same canopy

All other distances default to infinity

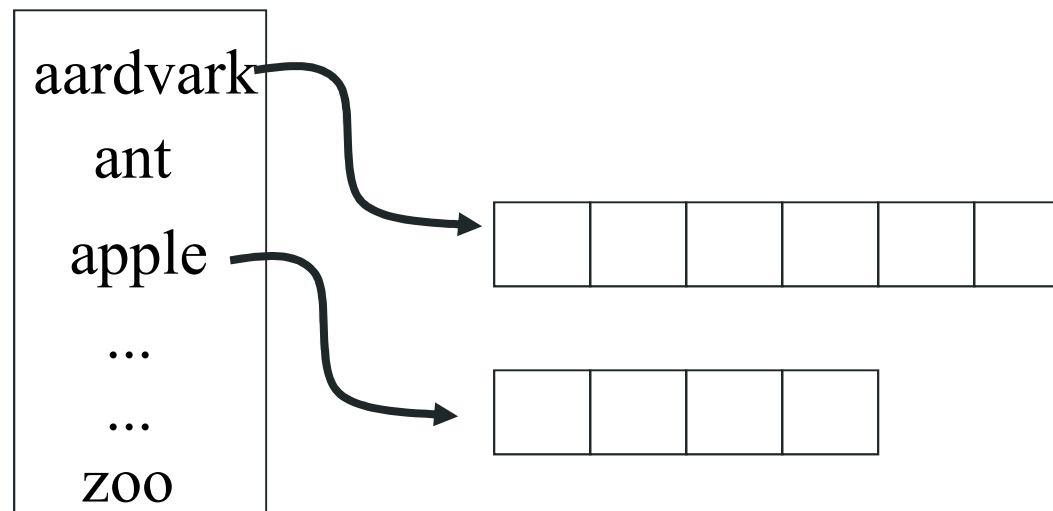
Use finite distances and iteratively
merge closest



Inexpensive Distance Metric for Text

Word-level matching (TFIDF)

Inexpensive using an inverted index



Expensive Distance Metric for Text

String edit distance

Compute with dynamic programming

Costs for character:

insertion

deletion

substitution

...

		S	e	c	a	t
S c o t t	0.0	0.7	1.4	2.1	2.8	3.5
	0.7	0.0	0.7	1.1	1.4	1.8
	1.4	0.7	1.0	0.7	1.4	1.8
	2.1	1.1	1.7	1.4	1.7	2.4
	2.8	1.4	2.1	1.8	2.4	1.7
	3.5	1.8	2.4	2.1	2.8	2.4

Computational Savings

inexpensive metric \ll expensive metric

Canopy creation nearly for free

Number of canopies: c (large)

Number of canopies per point: f (small but > 1)

fn/c points per canopy (if evenly spread)

$O(c(fn/c)^2)$ distance calculations initially

Canopies

Two distance metrics

cheap and approximate

expensive and accurate

Two-pass clustering

create overlapping canopies

full clustering with limited distances

Preserving Good Clustering

Small, disjoint canopies



big time savings

Large, overlapping canopies



original accurate clustering

Goal: fast and accurate

For every cluster, there exists a canopy such that all points in the cluster are in the canopy