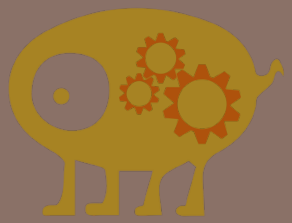


Data Mining
資料探勘

Clustering / Unsupervised learning

Hung-Yu Kao, Fall 2019

Unsupervised Learning

- Learning without a teacher
- Self-organization

Clustering

- K-mean
- Hierarchical clustering
- DBSCAN
- …

Anomaly Detection

- Outlier detection

Neural Network

- Autoencoder
- Generative Adversarial Network (GAN)
- SOM

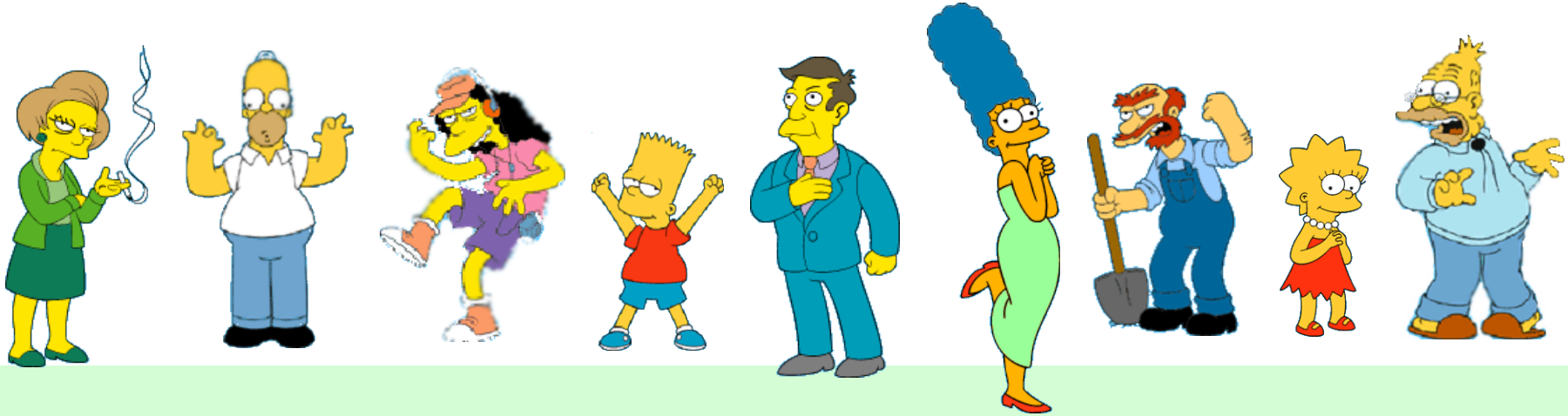
Learning approach

- Expectation Maximization (EM)
- PCA, MF, SVD

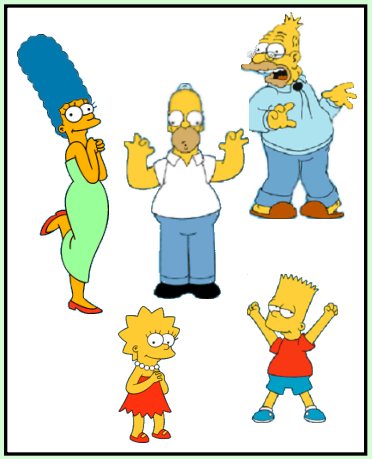
Clustering

- Clustering: process of grouping a set of physical or abstract objects into classes of similar objects
- Cluster: a collection of data objects that
  - are similar to one another within the same cluster
  - are dissimilar to the objects in other clusters
- Clustering: unsupervised classification
  - supervised classification: known #cluster & cluster labels
  - unsupervised classification: unknown #cluster & cluster labels

What is a natural grouping among these objects?

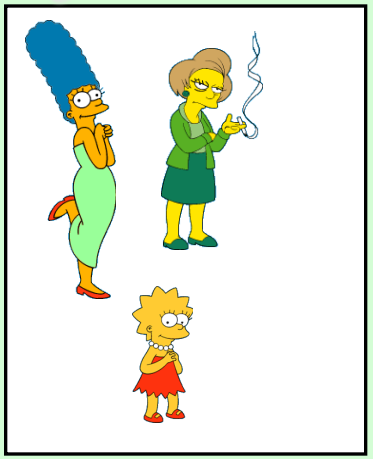


Clustering is subjective



Simpson's Family

School Employees



Females

Males

Good Clustering

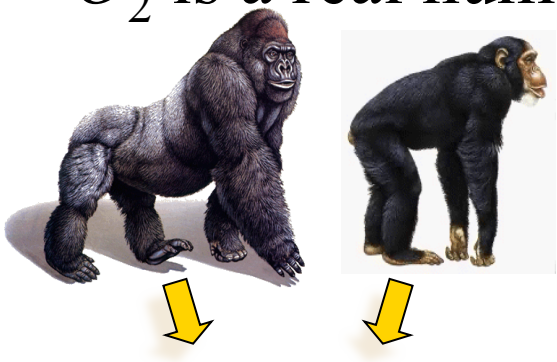
- Good clustering (produce high quality clusters)
  - intra-cluster similarity is high
  - inter-cluster class similarity is low
- Quality factors
  - similarity measure and its implementation
  - definition and *representation* of cluster chosen
  - clustering algorithm

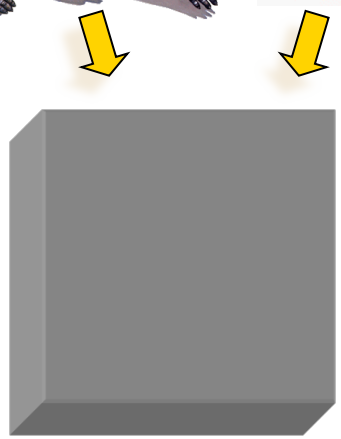
What is Similarity?

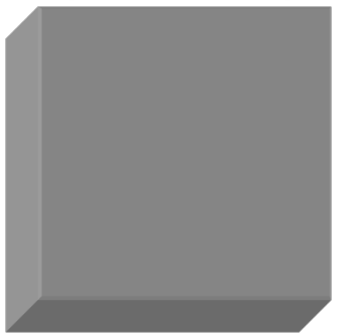
Similarity is hard to define, but… "We know it when we see it"

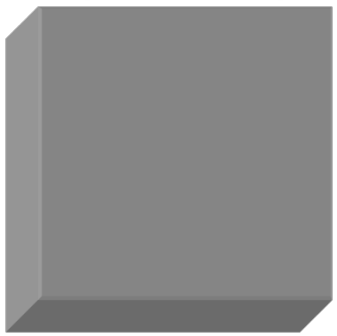
Defining Distance Measures

Definition: Let O_1 and O_2 be two objects from the universe of possible objects. The distance between O_1 and O_2 is a real number denoted by $D(O_1,O_2)$









0.23

3

342.7

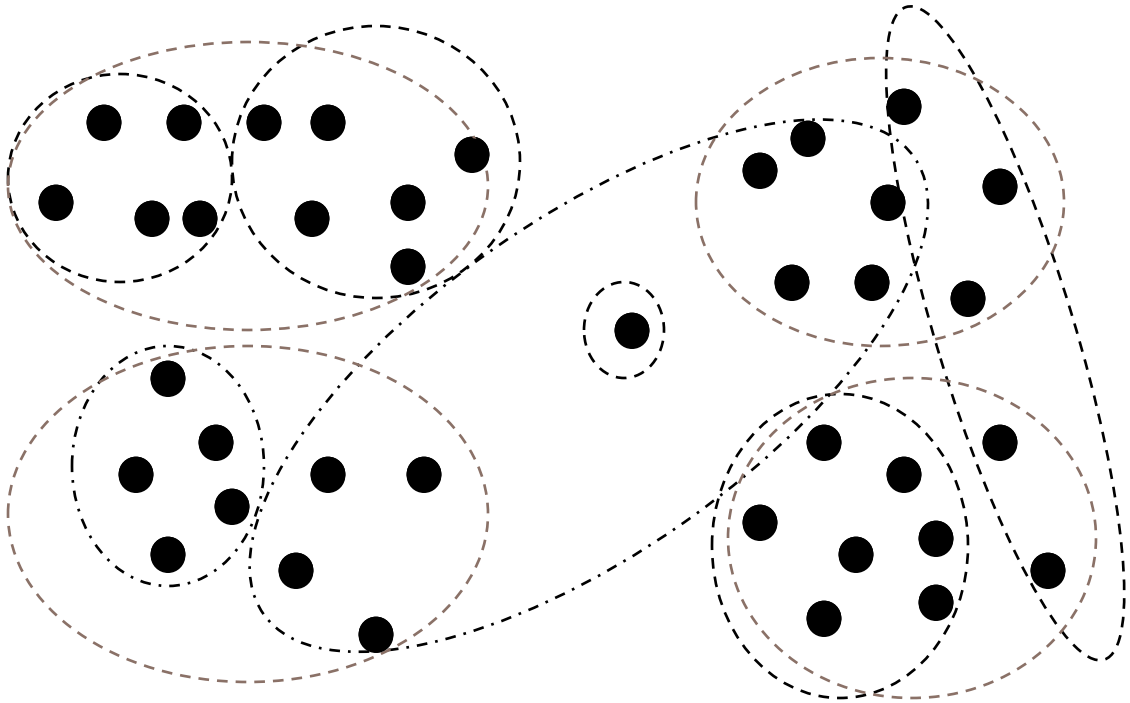
Typical Applications of Clustering Analysis

- Pattern Recognition
- Business: market segmentation
  - discover distinct group of customers
  - characterize customer groups
- Biology:
  - derive plant & animal taxonomies
  - categorizes genes with similar functionality
  - gain insight into structures inherent in populations
- Geography:
  - identification of area of similar land use
- Insurance:
  - identification of groups of insurance holders with high claim cost
- City-planning: identification of house group
- Document management: classify documents of WWW

Requirements of Clustering

- Scalability
- Dealing with different types of attributes (not only numerical data)
- Discovery of clusters with arbitrary shape (not only sphere)
- Minimal requirements for domain knowledge to input design parameters
- Ability to deal with noisy data
- Insensitivity to order of input records
- High dimensionality
- Constraint-based clustering
- Interpretability and usability

Clustering Houses



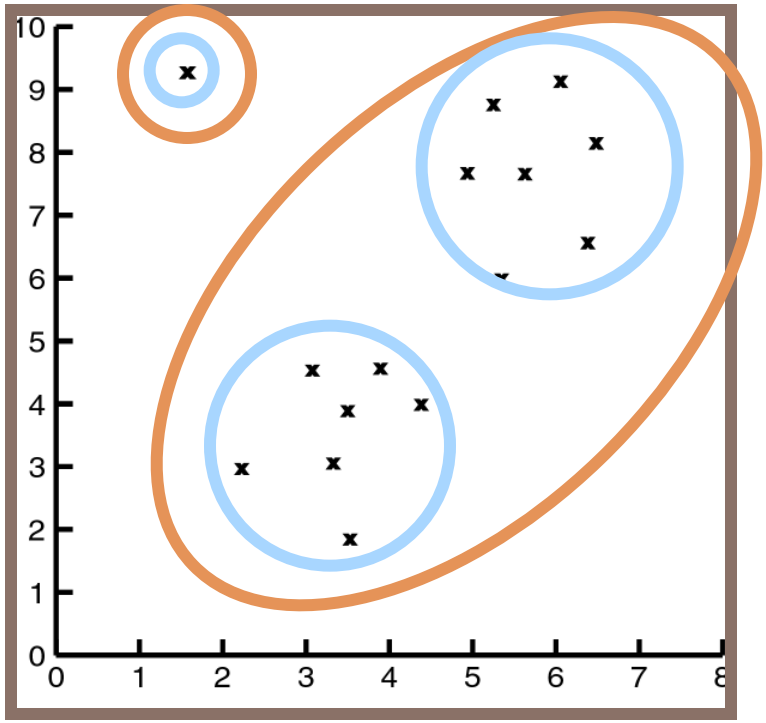
Geographic Distance Based

Size Based

Clustering Issues

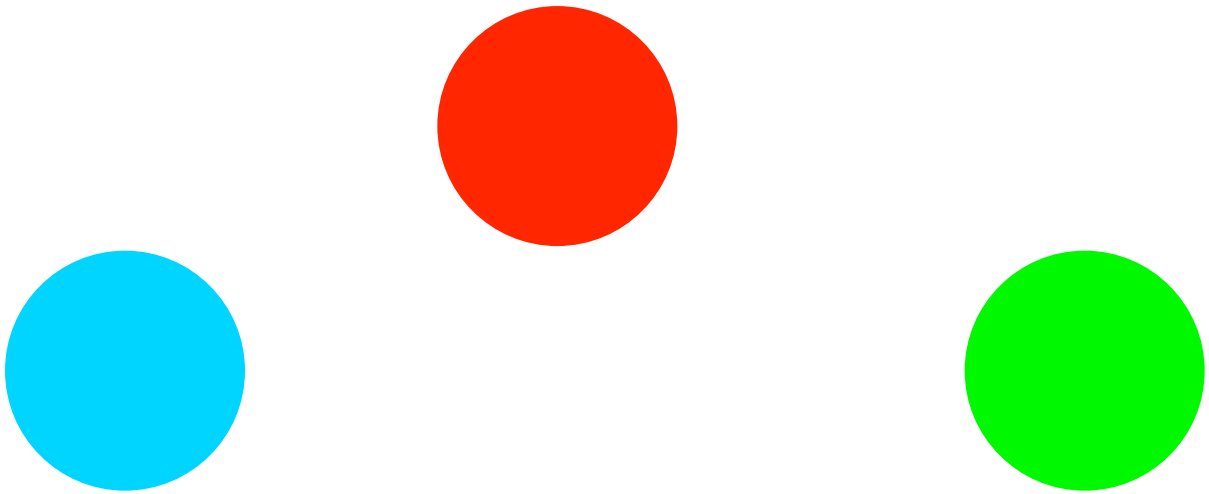
- Outlier handling
- Dynamic data
- Interpreting results (centroid meaning)
- Number of clusters (magic *k*)
- Data to be used
- Scalability

Impact of Outliers on Clustering



Types of Clusters: Well-Separated

- Well-Separated Clusters:
  - A cluster is a set of points such that any point in a cluster is closer (or more similar) to every other point in the cluster than to any point not in the cluster.



3 well-separated clusters

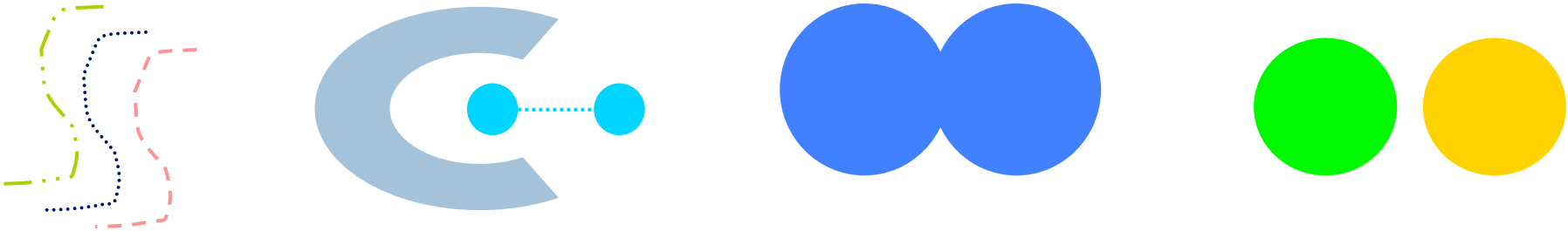
Types of Clusters: Center-Based

- Center-based
  - A cluster is a set of objects such that an object in a cluster is closer (more similar) to the "center" of a cluster, than to the center of any other cluster
  - The center of a cluster is often a centroid, the average of all the points in the cluster, or a medoid, the most "representative" point of a cluster

4 center-based clusters

Types of Clusters: Contiguity-Based

- Contiguous Cluster (Nearest neighbor or Transitive)
  - A cluster is a set of points such that a point in a cluster is closer (or more similar) to one or more other points in the cluster than to any point not in the cluster.

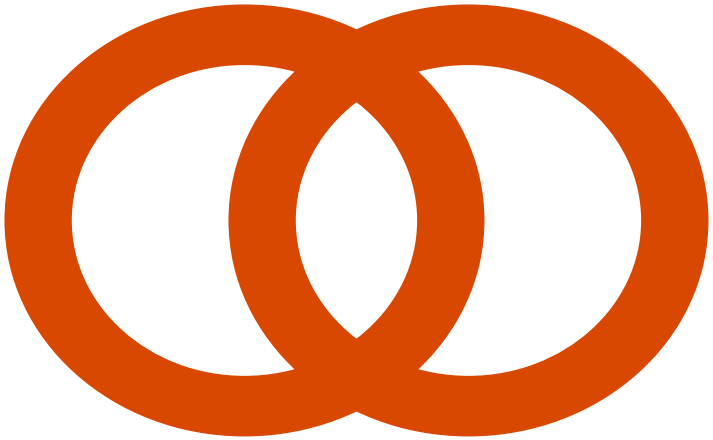


8 contiguous clusters

Clusters

Types of Clusters: Conceptual Clusters

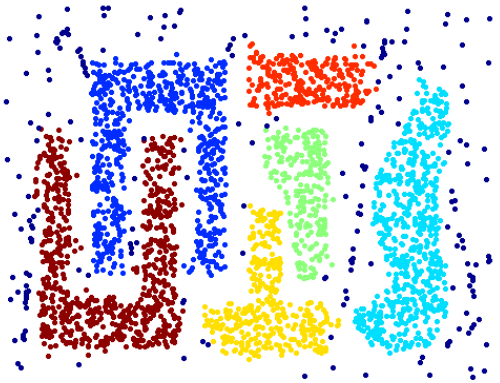
- Shared Property or Conceptual Clusters
  - Finds clusters that share some common property or represent a particular concept.

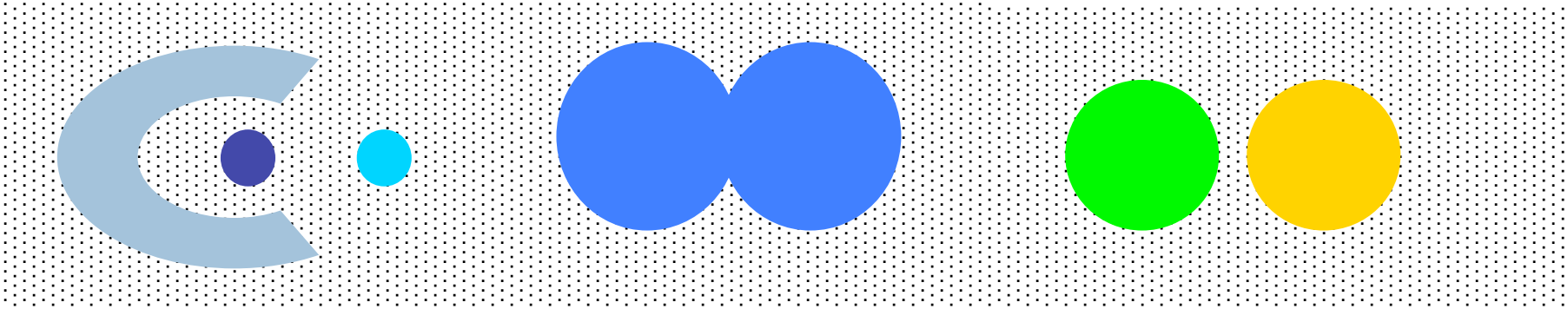


2 Overlapping Circles

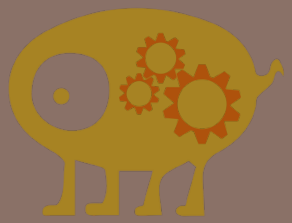
Types of Clusters: Density-Based

- Density-based
  - A cluster is a dense region of points, which is separated by low-density regions, from other regions of high density.
  - Used when the clusters are irregular or intertwined, and when noise and outliers are present.





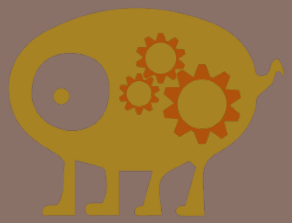
6 density-based clusters



APPROACHES OF CLUSTERING ALGORITHMS

Five Categories of Clustering Methods

- Partitioning algorithms
  - Construct various partitions and then evaluate them by some criterion.
- Hierarchy algorithms
  - Create a hierarchical decomposition of the set of data (or objects) using some criterion.
- Density-based
  - based on connectivity and density functions
- Grid-based
  - based on a multiple-level granularity structure
- Model-based
  - A model is hypothesized for each of the clusters and the idea is to find the best fit of that model to each other.



PARTITION-BASED CLUSTERING

Partitioning Algorithms: Basic Concept

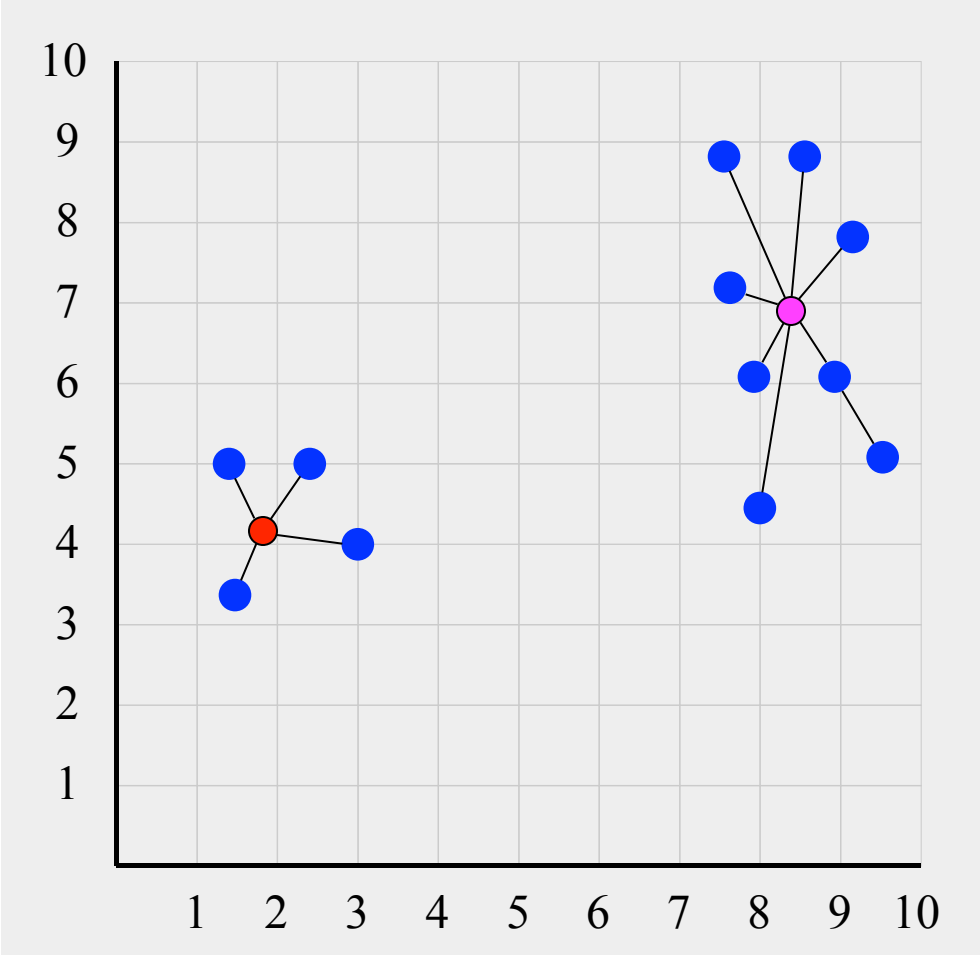
- Partitioning method: Construct a partition of a database ***D*** of ***n*** objects into a set of ***k*** clusters
- Given a *k*, find a partition of *k* clusters that optimizes the chosen partitioning criterion.
  - Global optimal: exhaustively enumerate all partitions. (NP-Hard!)
  - Heuristic methods.
    - k-means: each cluster is represented by the center of the cluster
    - k-medoids or PAM (Partition Around Medoids) : each cluster is represented by one of the objects in the cluster.

Squared Error

$$se_{K_i} = \sum_{j=1}^{m} \|t_{ij} - C_k\|^2$$

$$se_K = \sum_{j=1}^{k} se_{K_j}$$

Objective Function



The *K*-Means Clustering Method

- Given *k*, the *k-means* algorithm:
  1. Partition objects into *k* nonempty subsets
  2. Compute mean as the centroids of the clusters of the current partition
  3. Relocate each object to the nearest cluster
  4. Go back to Step 2, stop when no more new relocation

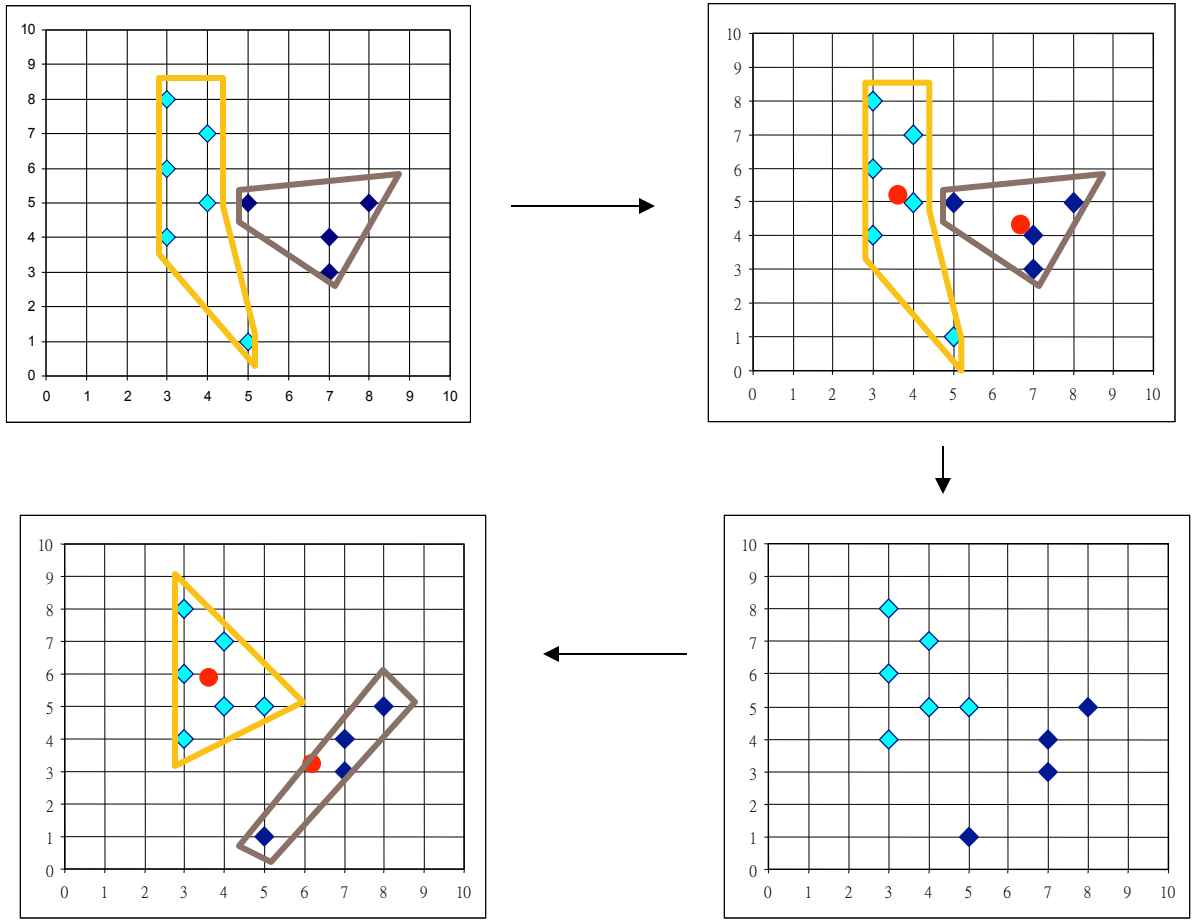
K-means Clustering

- Partitional clustering approach
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, k, must be specified
- The basic algorithm is very simple

1: Select K points as the initial centroids.
2: **repeat**
3: Form K clusters by assigning all points to the closest centroid.
4: Recompute the centroid of each cluster.
5: **until** The centroids don't change

K-Means Clustering Method

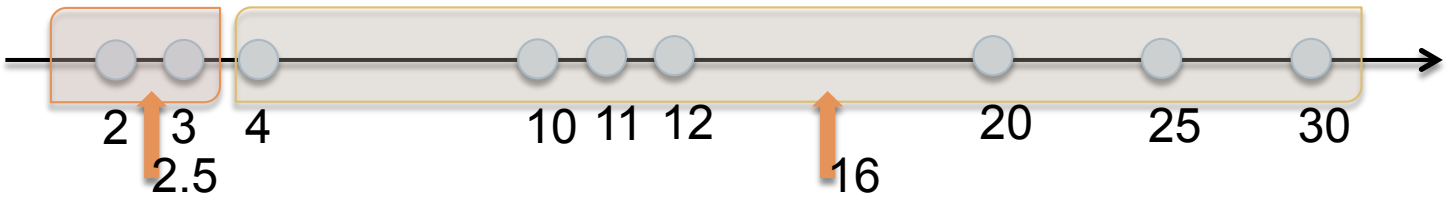
- Example



K-Means Example

- Given: {2,4,10,12,3,20,30,11,25}, k=2

C1	C2	M1	M2
{2,3}	{4,10,12,20,30,11,25}	2.5	16
{2,3,4}	{10,12,20,30,11,25}	3	18
{2,3,4,10}	{12,20,30,11,25}	4.75	19.6
{2,3,4,10,11,12}	{20,30,25}	7	25
{2,3,4,10,11,12}	{20,30,25}	7	25



K-means Clustering – Details

- Initial centroids are often chosen randomly.
  - Clusters produced vary from one run to another.
- The centroid is (typically) the mean of the points in the cluster.
- 'Closeness' is measured by Euclidean distance, cosine similarity, correlation, etc.
- K-means will converge for common similarity measures mentioned above.
- Most of the convergence happens in the first few iterations.
  - Often the stopping condition is changed to 'Until relatively few points change clusters'

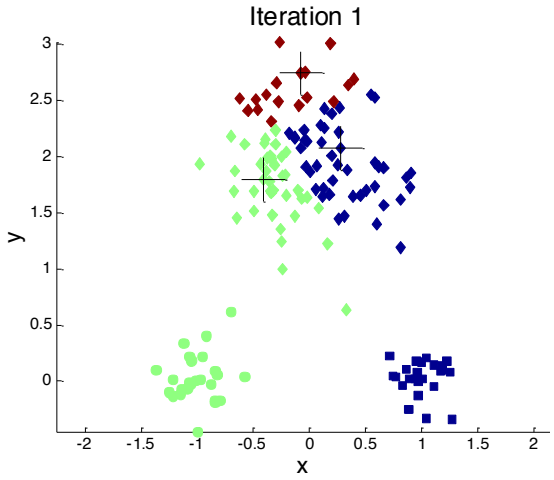
Comments on the *K-Means* Method

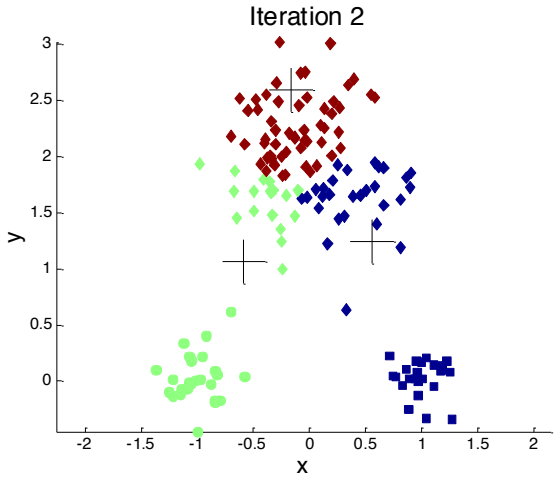
- Strength
  - Efficient, Complexity is O(n * K * I * d)
    - n = number of points, K = number of clusters, I = number of iterations, d = number of attributes
  - Often terminates at a *local optimum*
- Weakness
  - Applicable only when *mean* is defined (categorical data?)
  - Need to specify *k*, the *number* of clusters, in advance
  - Unable to handle noisy data and *outliers*

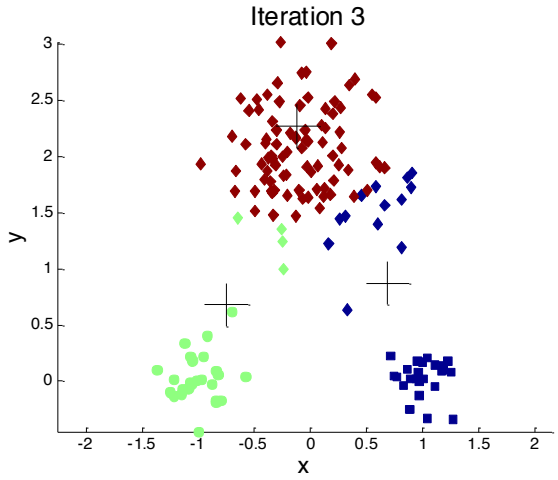
Variations of the *K-Means* Method

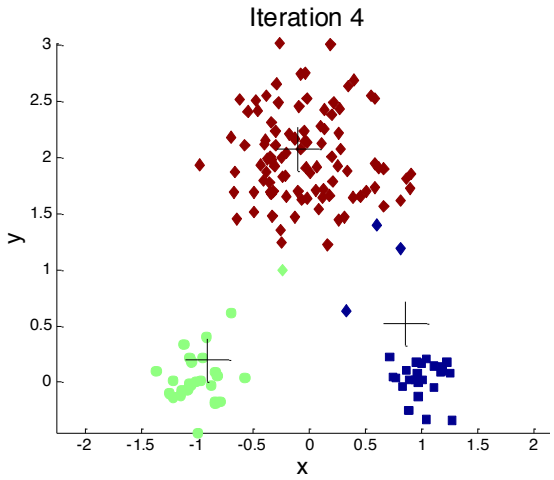
- Variants of the *k*-means
  - Selection of the initial *k* means
  - Dissimilarity calculations
  - Strategies to calculate cluster means
- Handling categorical data: *k-modes*
  - Replacing means of clusters with modes (distance=0 or 1)
  - *k-prototype*: a mixture of categorical and numerical data

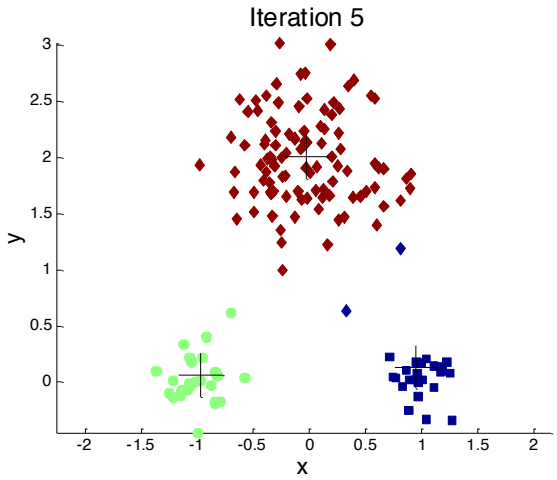
Importance of Choosing Initial Centroids

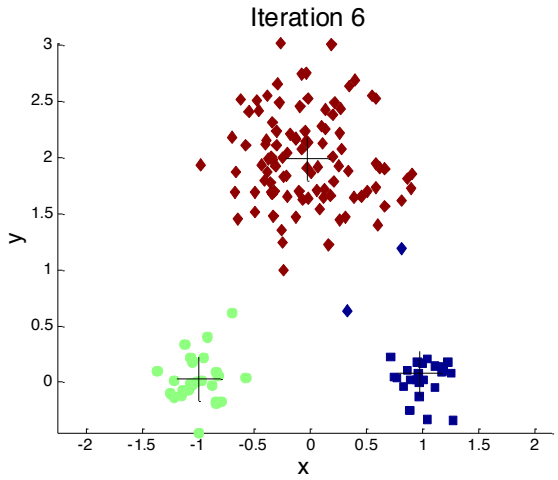




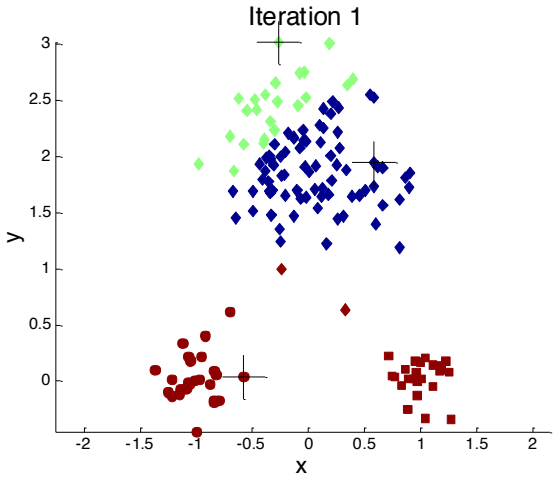


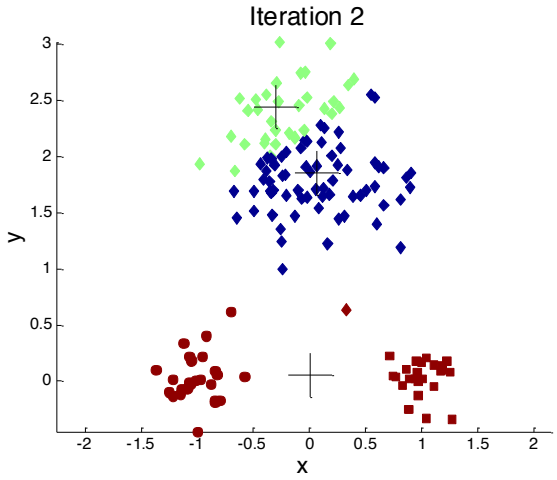


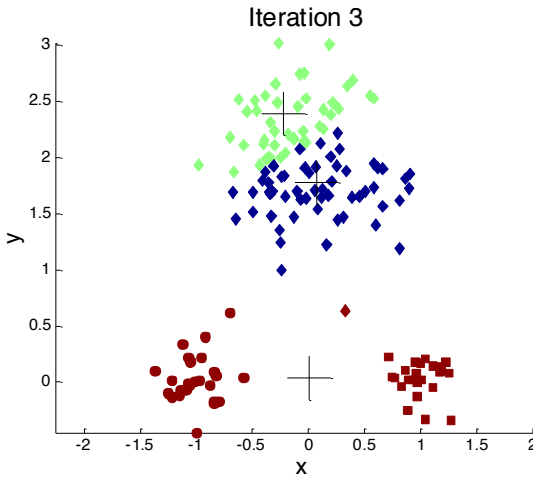


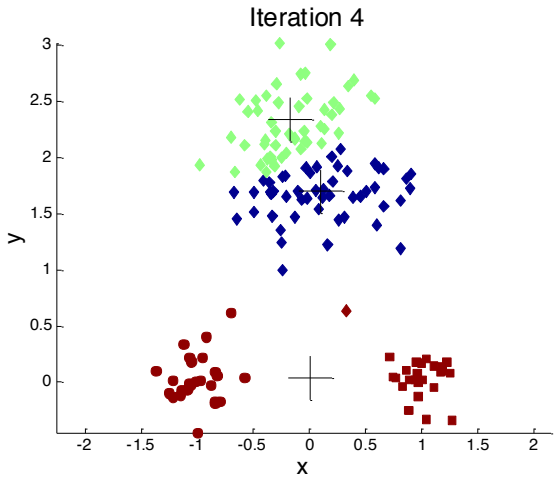


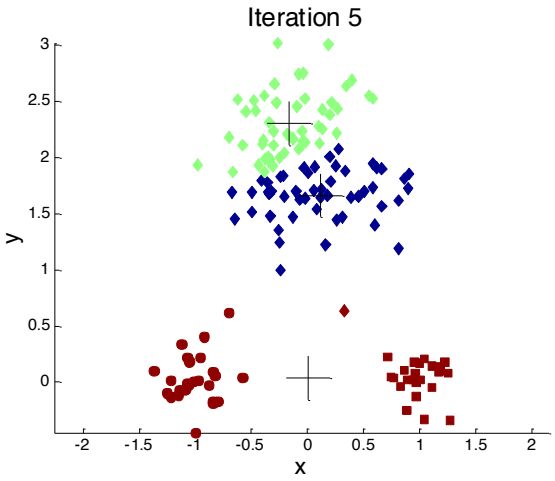
Importance of Choosing Initial Centroids …











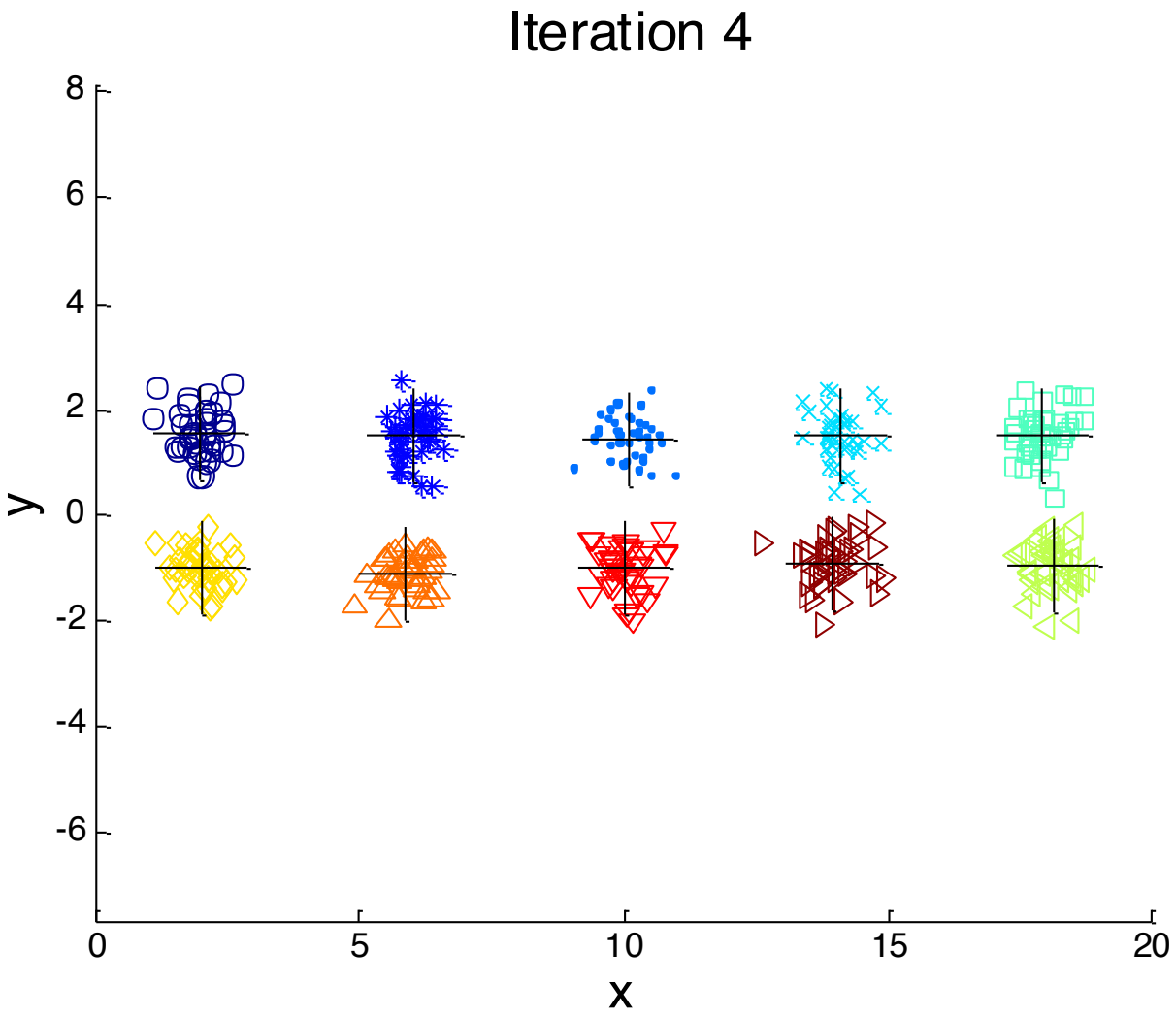
Problems with Selecting Initial Points

- If there are K 'real' clusters then the chance of selecting one centroid from each cluster is small.
  - Chance is relatively small when K is large
  - If clusters are the same size, n, then

$$P = \frac{\text{number of ways to select one centroid from each cluster}}{\text{number of ways to select } K \text{ centroids}} = \frac{K!n^K}{(Kn)^K} = \frac{K!}{K^K}$$

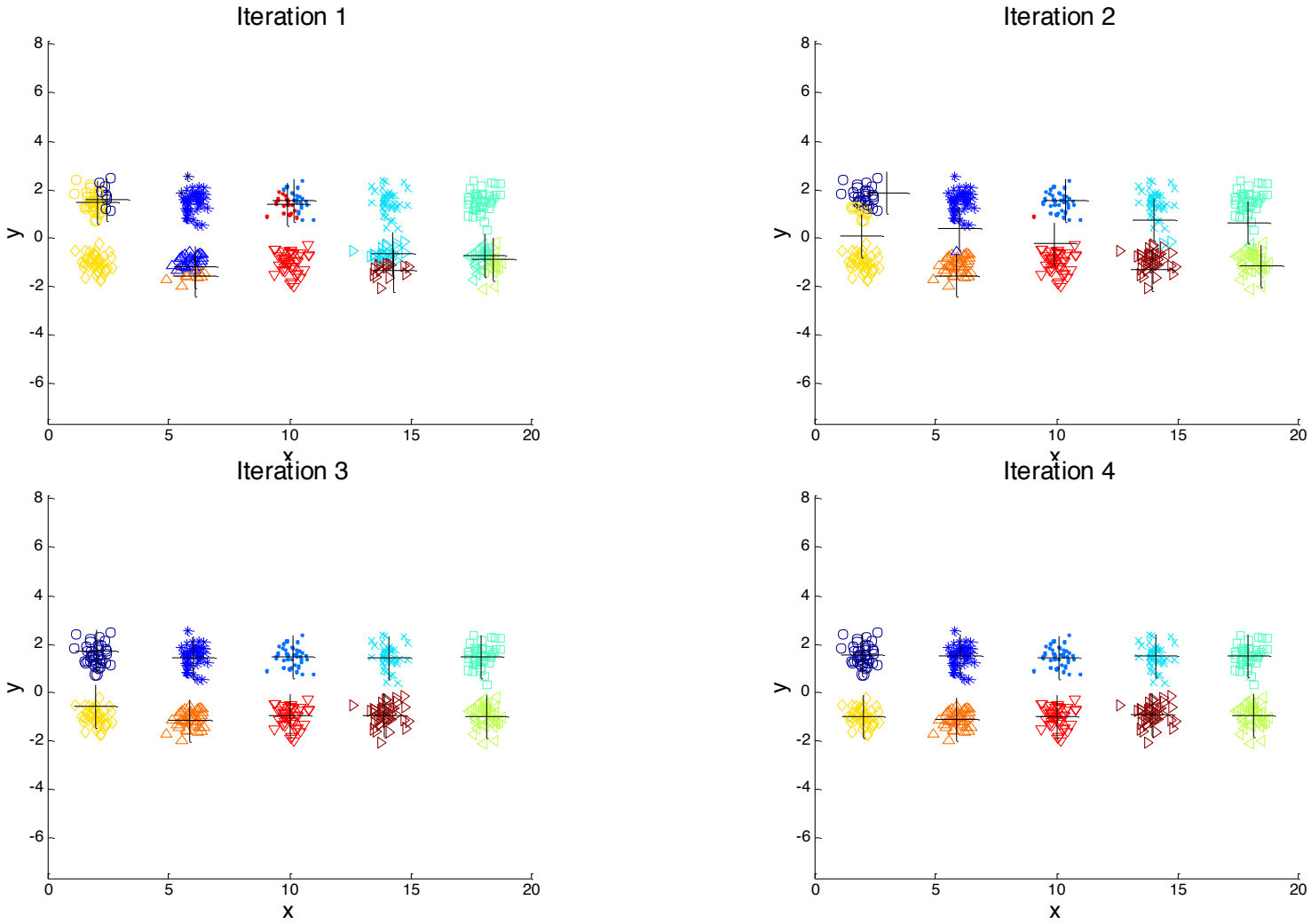
  - For example, if K = 10, then probability = $10!/10^{10} = 0.00036$
  - Sometimes the initial centroids will readjust themselves in 'right' way, and sometimes they don't
  - Consider an example of five pairs of clusters

10 Clusters Example



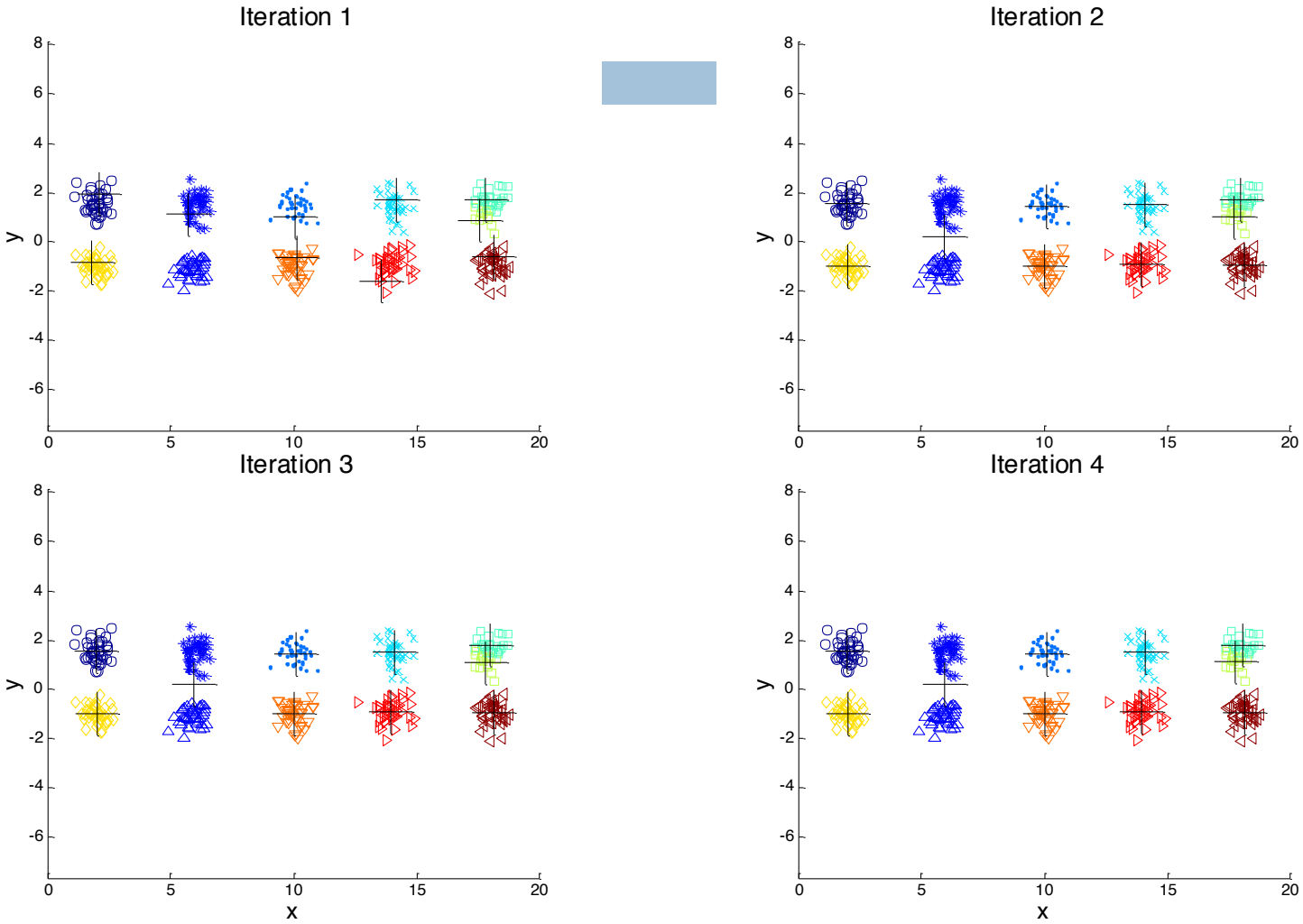
Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



Starting with two initial centroids in one cluster of each pair of clusters

10 Clusters Example



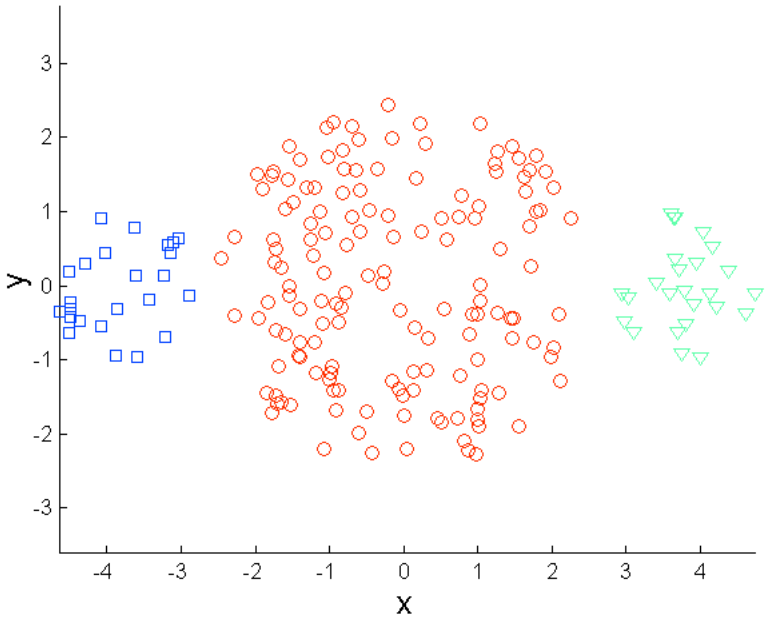
Starting with some pairs of clusters having three initial centroids, while other have only one.

Limitations of K-means

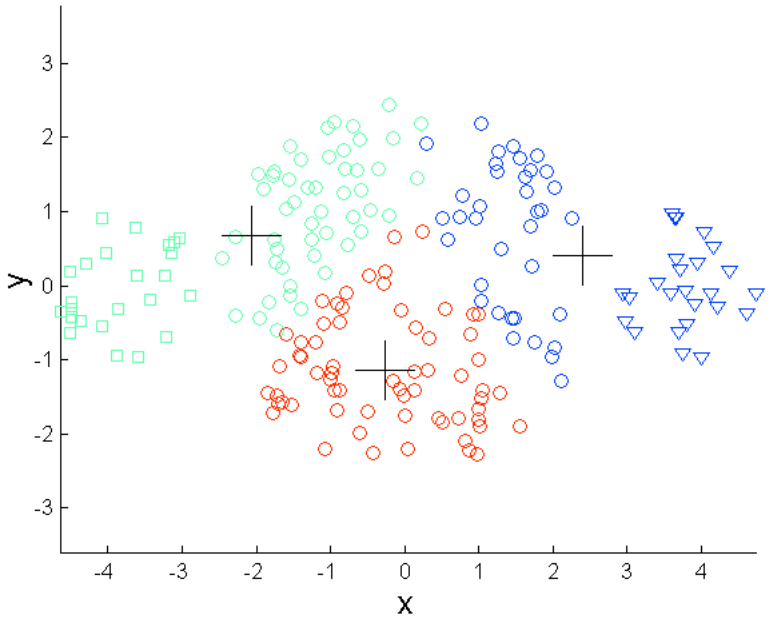
- K-means has problems when clusters are of differing
  - Sizes
  - Densities
  - Non-globular shapes

- K-means has problems when the data contains outliers.

Limitations of K-means: Differing Sizes

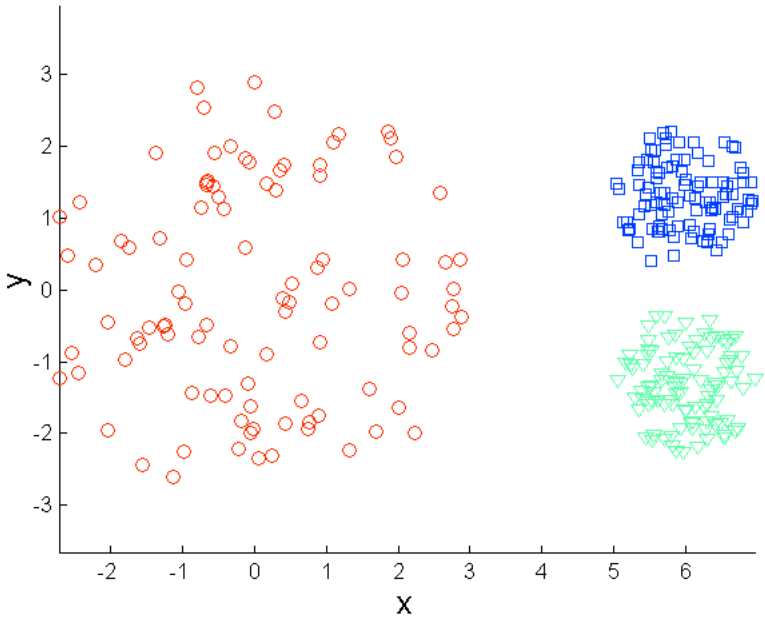


Original Points

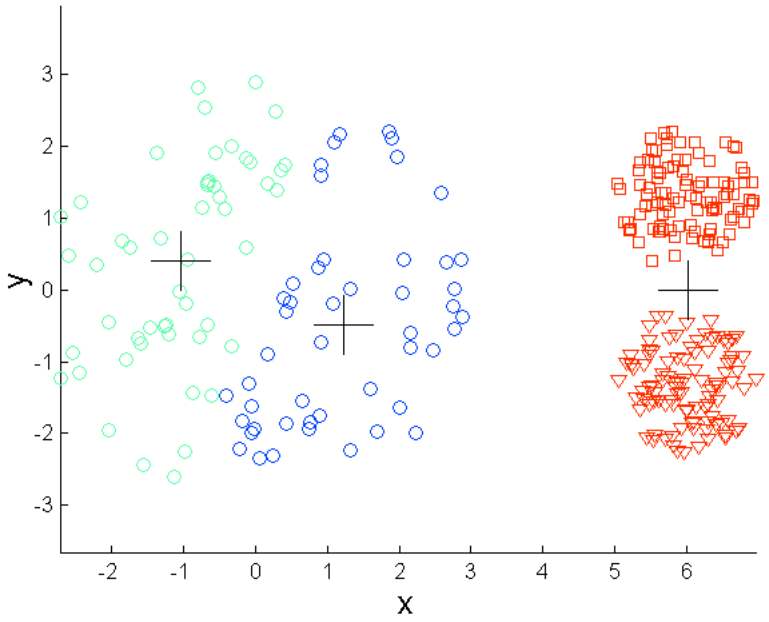


K-means (3 Clusters)

Limitations of K-means: Differing Density

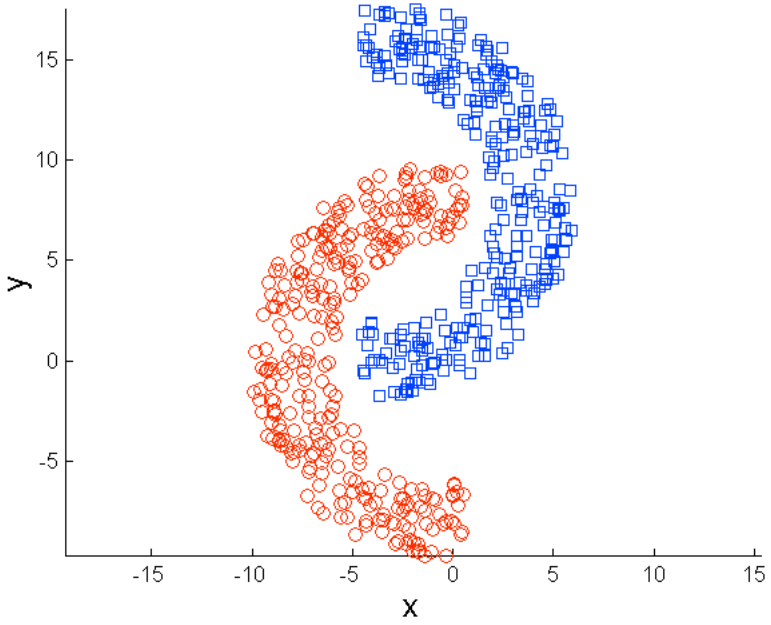


Original Points

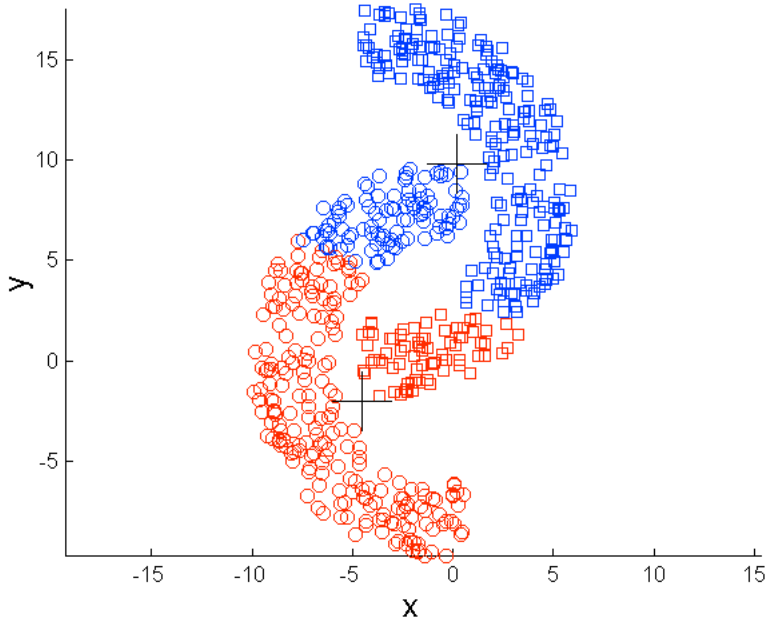


K-means (3 Clusters)

Limitations of K-means: Non-globular Shapes

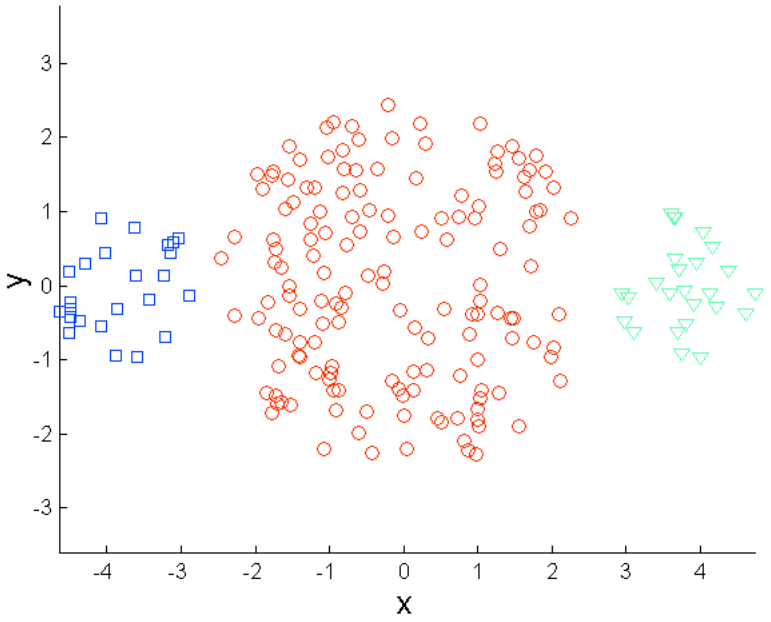


Original Points

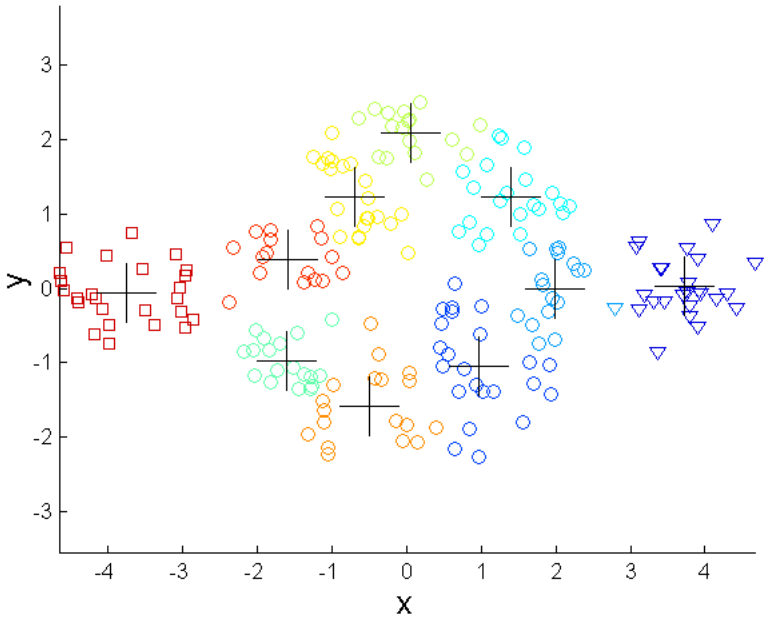


K-means (2 Clusters)

Overcoming K-means Limitations



Original Points

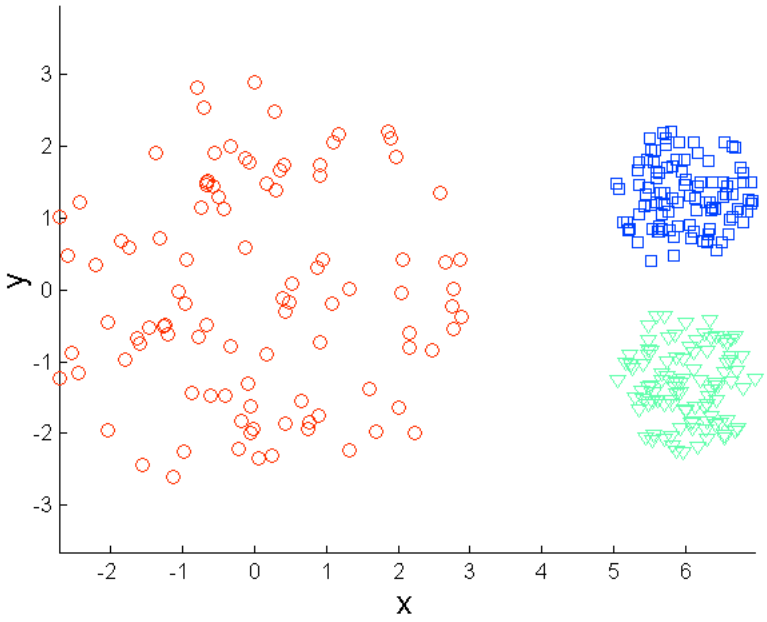


K-means Clusters

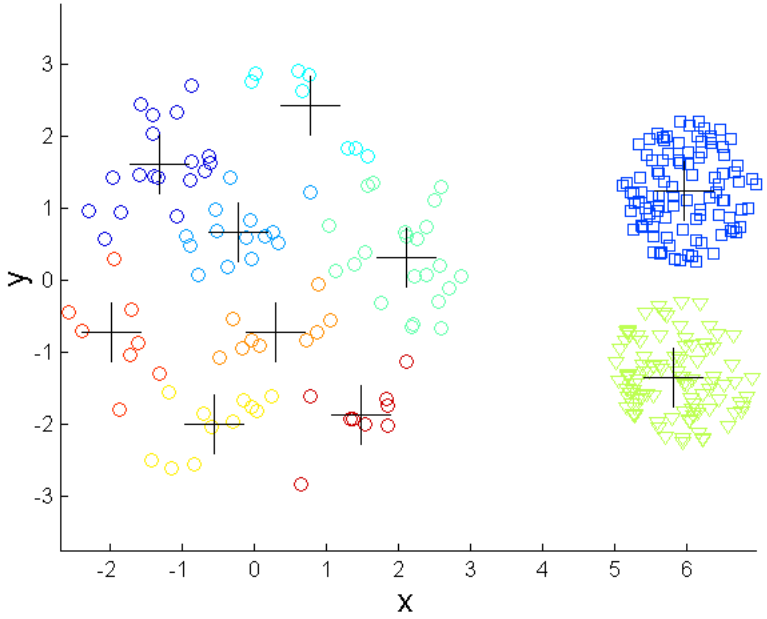
One solution is to use many clusters.

Find parts of clusters, but need to put together.

Overcoming K-means Limitations

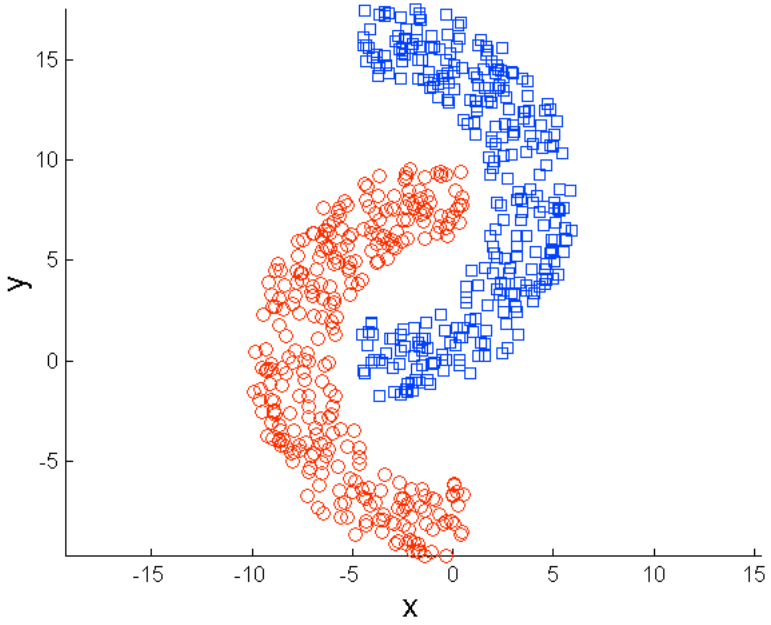


Original Points

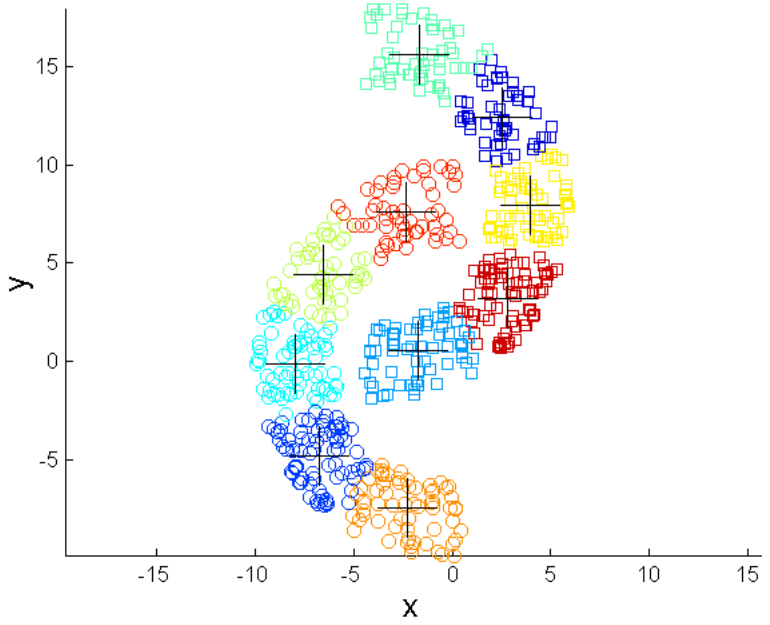


K-means Clusters

Overcoming K-means Limitations



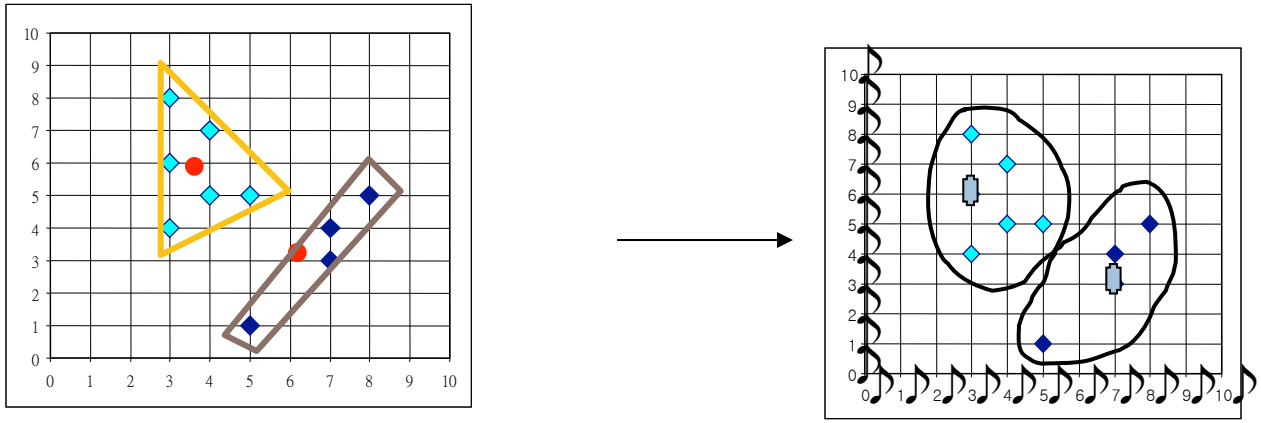
Original Points



K-means Clusters

What is the problem of K-Means Method?

- The k-means algorithm is sensitive to outliers !
  - Since an object with an extremely large value may substantially distort the distribution of the data.
- K-Medoids: Instead of taking the **mean** value of the object in a cluster as a reference point, **medoids** can be used, which is the **most centrally located** object in a cluster.

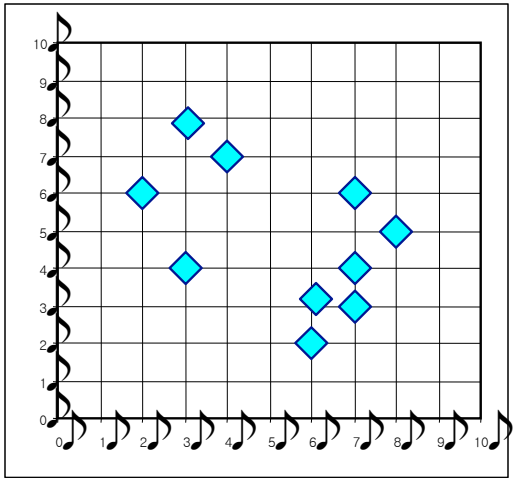


The *K-Medoids* Clustering Method

- Find *representative* objects, called medoids, in clusters
- *PAM* (Partitioning Around Medoids, 1987)
  - starts from an initial set of medoids and iteratively replaces one of the medoids by one of the non-medoids if it improves the total distance of the resulting clustering
  - *PAM* works effectively for small data sets, but does not scale well for large data sets
- *CLARA* (Kaufmann & Rousseeuw, 1990)
- *CLARANS* (Ng & Han, 1994): Randomized sampling
- Focusing + spatial data structure (Ester et al., 1995)

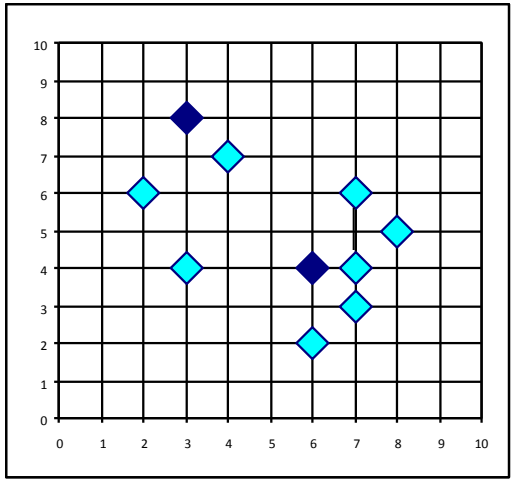
Typical K-medoids algorithm (PAM)

Total Cost = 20

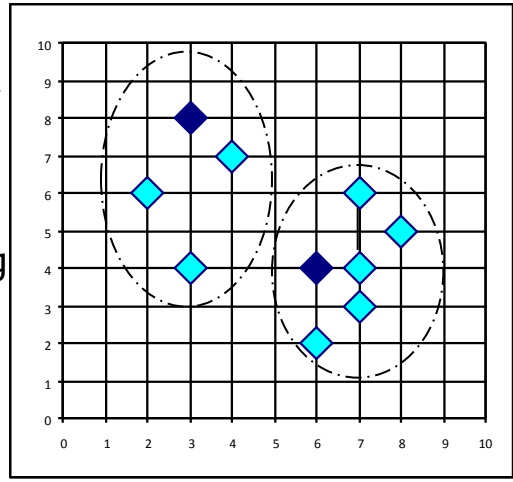


K=2

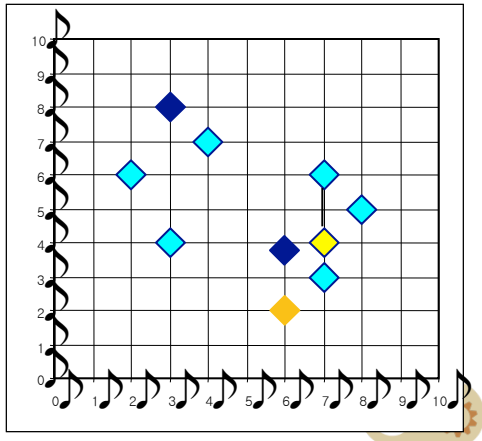
Arbitrary choose k object as initial medoids



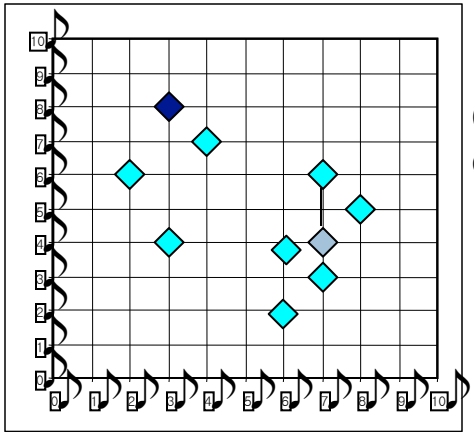
Assign each remaining object to nearest medoids



Randomly select a nonmedoid object,O_{ramdom}



Compute total cost of swapping



Swapping O and O_{ramdom}

If quality is improved.

Do loop

Until no change

What is the problem with PAM?

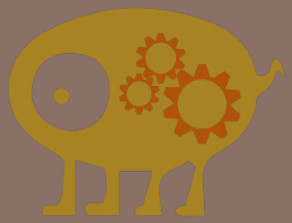
- PAM is more robust than k-means in the presence of noise and outliers
  - A medoid is less influenced by outliers or other extreme values than a mean
- PAM works efficiently for small data sets but does not **scale well** for large data sets

➔Sampling based method,

CLARA(Clustering LARge Applications)

CLARA (Clustering Large Applications)

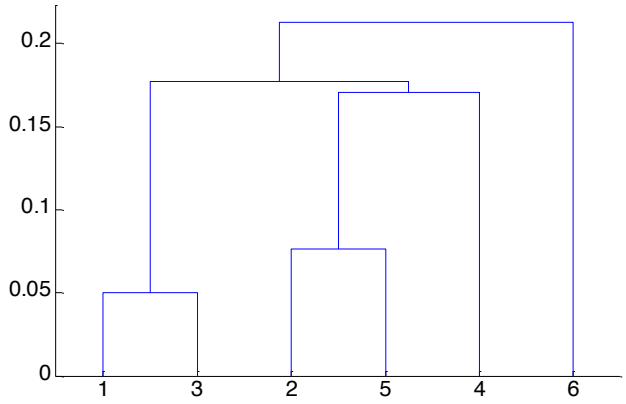
- Partition for large database
- Steps
  - draws *multiple samples* of the data set
  - applies *PAM* on each sample
  - gives the best clustering as the output
- Strength
  - deal with larger data sets than *PAM*
- Weakness
  - efficiency depends on the sample size
  - good clustering on samples => good clustering on whole data set ?

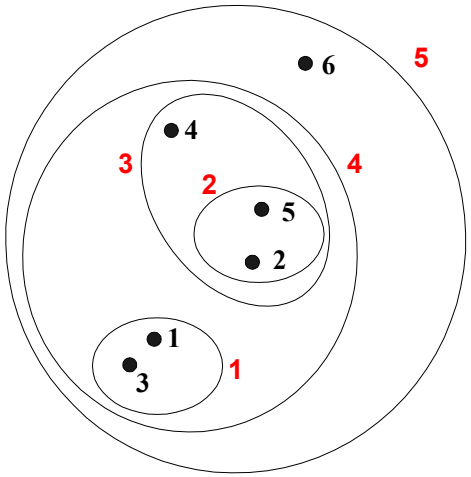


HIERARCHICAL CLUSTERING

Hierarchical Clustering

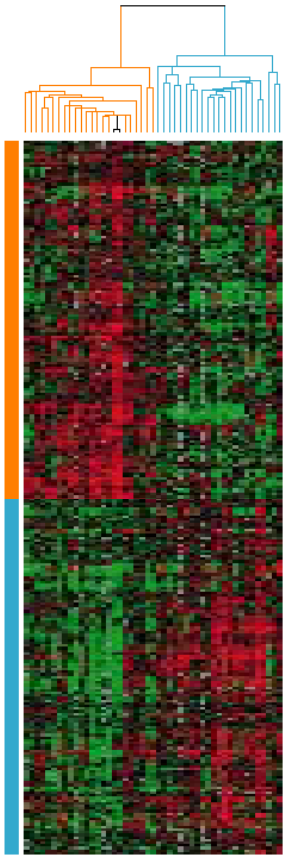
- Produces a set of nested clusters organized as a hierarchical tree
- Can be visualized as a dendrogram
  - A tree like diagram that records the sequences of merges or splits



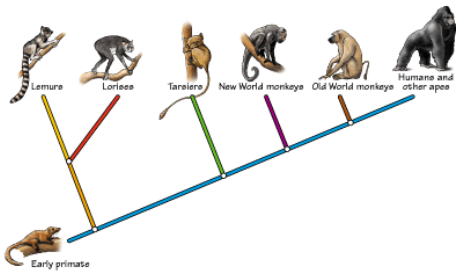


Strengths of Hierarchical Clustering

- Do not have to assume any particular number of clusters
  - Any desired number of clusters can be obtained by 'cutting' the dendogram at the proper level



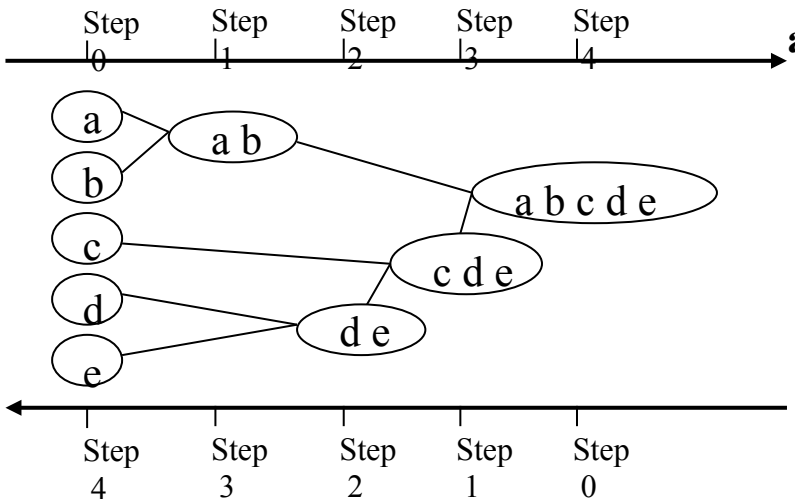
- They may correspond to meaningful taxonomies
  - Example in biological sciences (e.g., animal kingdom, phylogeny reconstruction, …)

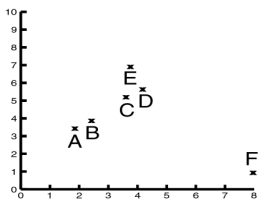


Hierarchical Clustering

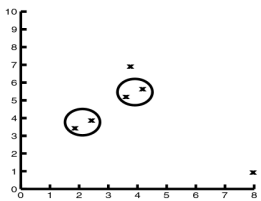
- Two main types of hierarchical clustering
  - Agglomerative:
    - Start with the points as individual clusters
    - At each step, merge the closest pair of clusters until only one cluster (or k clusters) left
  - Divisive:
    - Start with one, all-inclusive cluster
    - At each step, split a cluster until each cluster contains a point (or there are k clusters)
- Traditional hierarchical algorithms use a similarity or distance matrix
  - Merge or split one cluster at a time

Hierarchical Clustering

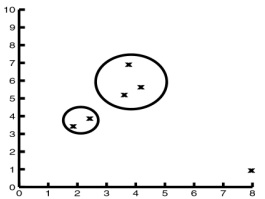




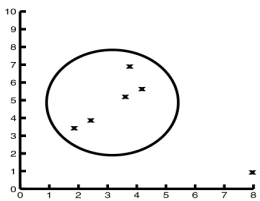
a) Six Clusters



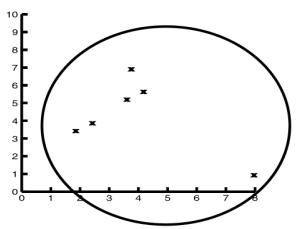
b) Four Clusters



c) Three Clusters



d) Two Clusters



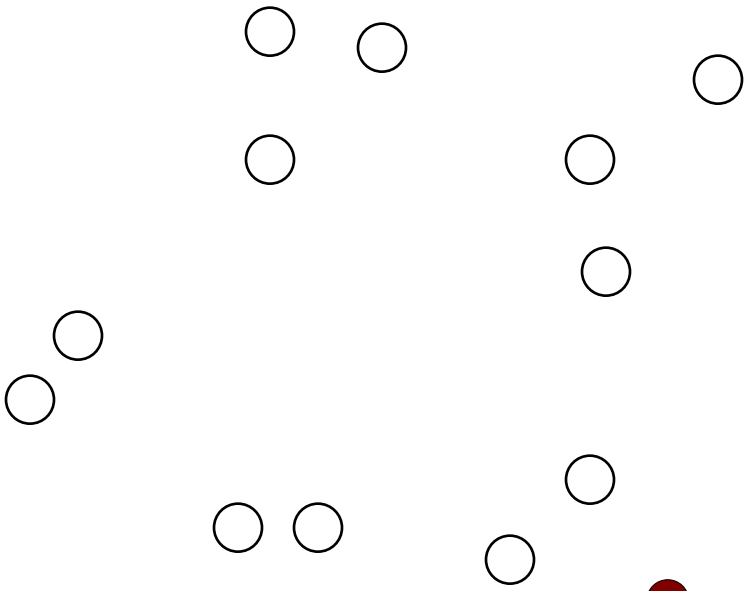
e) One Cluster

Agglomerative Clustering Algorithm

- More popular hierarchical clustering technique
- Basic algorithm is straightforward
  1. Compute the proximity matrix
  2. Let each data point be a cluster
  3. **Repeat**
  4. Merge the two closest clusters
  5. Update the proximity matrix
  6. **Until** only a single cluster remains
- Key operation is the computation of the proximity of two clusters
  - Different approaches to defining the distance between clusters distinguish the different algorithms

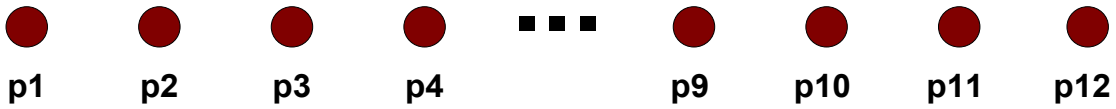
Starting Situation

- Start with clusters of individual points and a proximity matrix



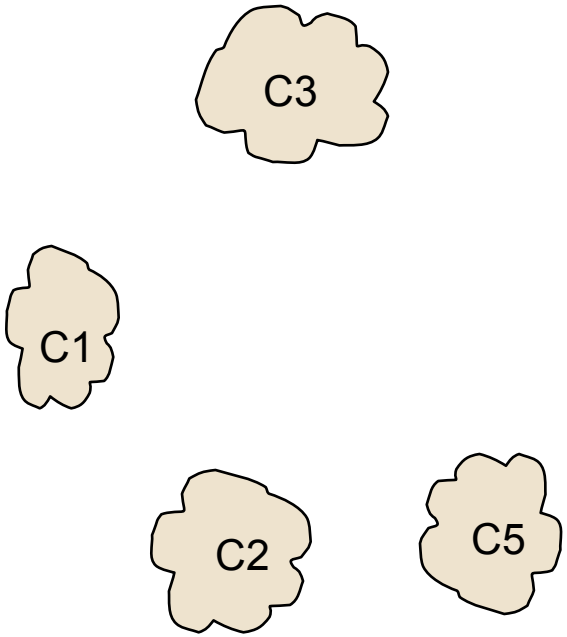
	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

Proximity Matrix



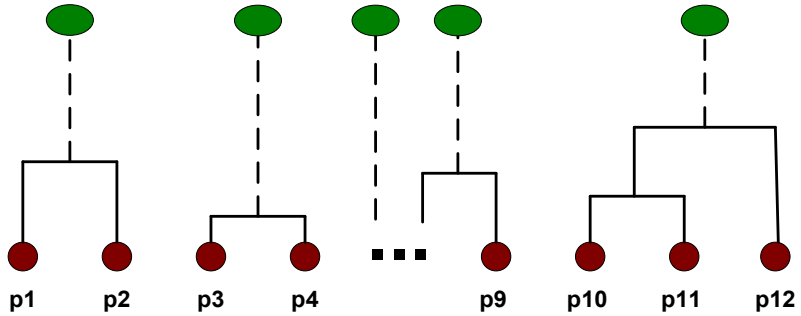
Intermediate Situation

- After some merging steps, we have some clusters



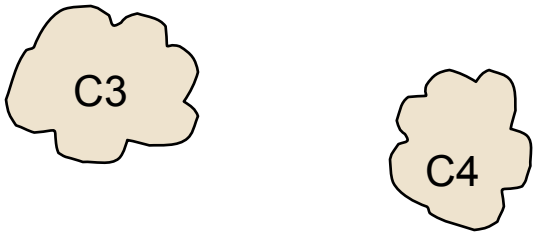
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

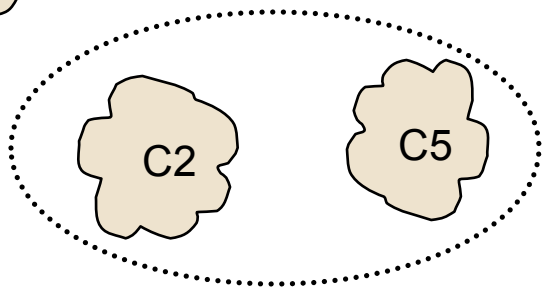
Proximity Matrix



Intermediate Situation

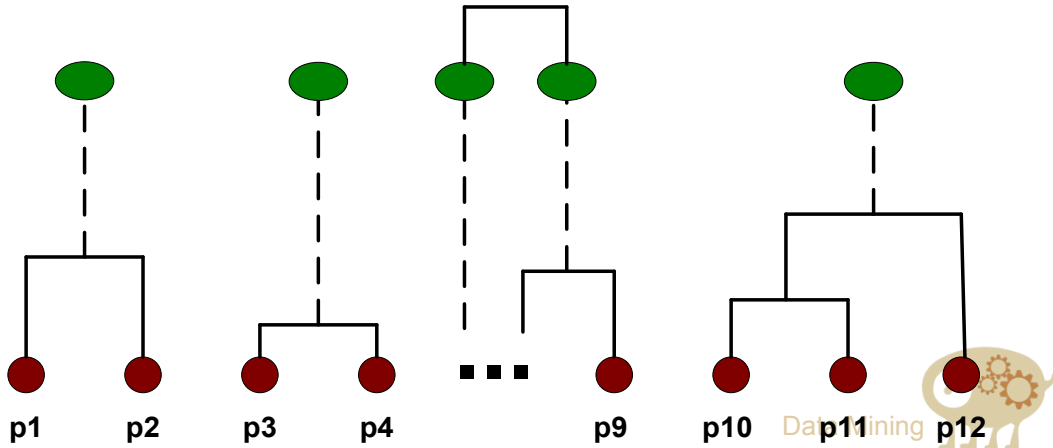
- We want to merge the two closest clusters (C2 and C5) and update the proximity matrix.





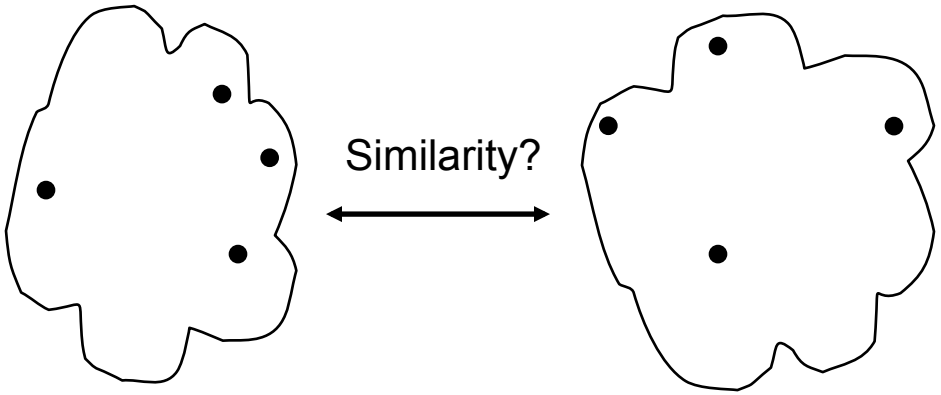
	C1	C2	C3	C4	C5
C1					
C2					
C3					
C4					
C5					

Proximity Matrix



The question is "How do we update the proximity matrix?"

How to Define Inter-Cluster Similarity

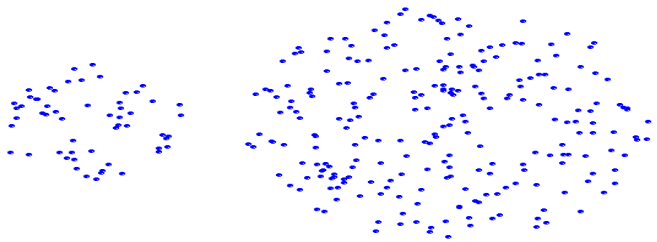


	p1	p2	p3	p4	p5	. . .
p1						
p2						
p3						
p4						
p5						
.						
.						
.						

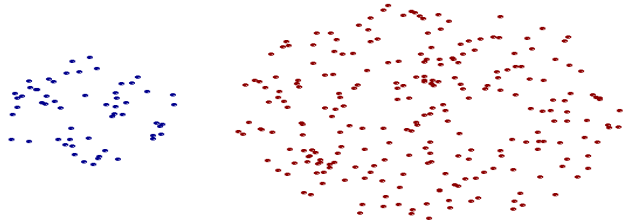
Proximity Matrix

- MIN
- MAX
- Group Average
- Distance Between Centroids
- Other methods driven by an objective function
  - Ward's Method uses squared error

Strength/Limitation of MIN

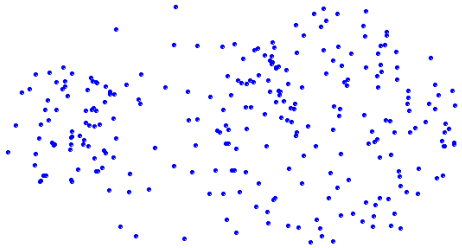


Original Points

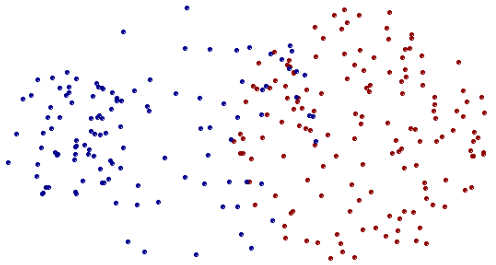


Two Clusters

- Can handle non-elliptical shapes



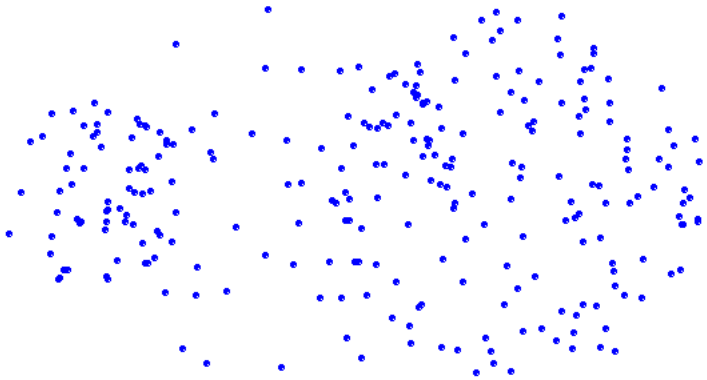
Original Points



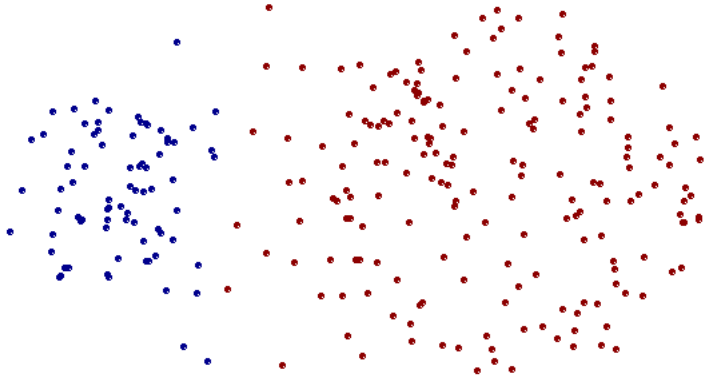
Two Clusters

- Sensitive to noise and outliers

Strength of MAX



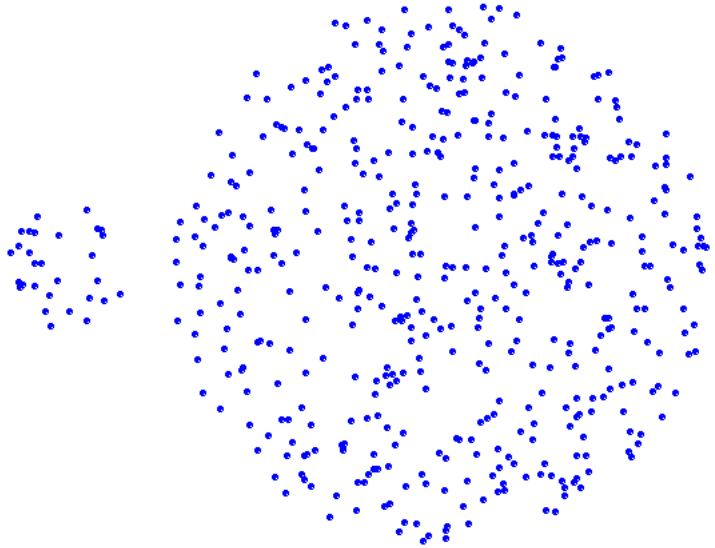
Original Points



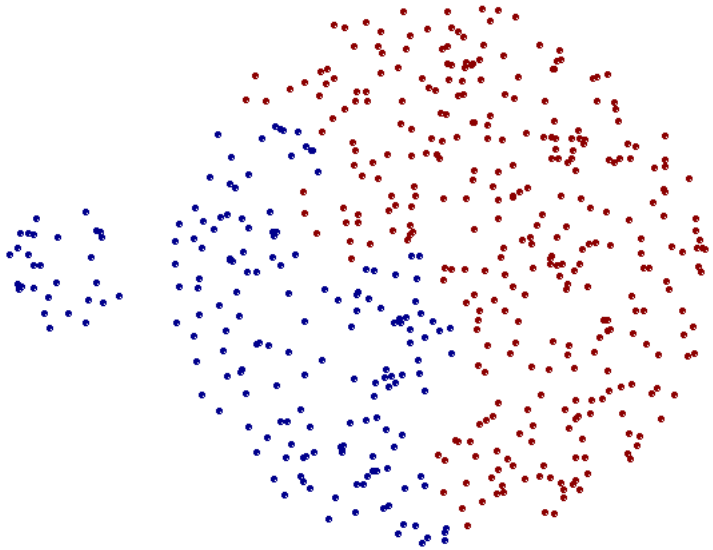
Two Clusters

- Less susceptible to noise and outliers

Limitations of MAX



Original Points



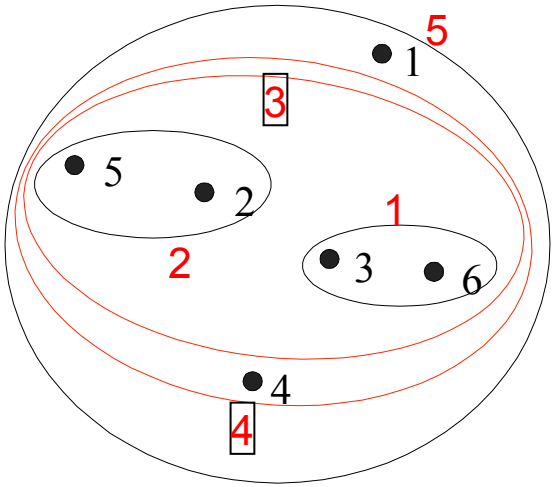
Two Clusters

- Tends to break large clusters
- Biased towards globular clusters

Cluster Similarity: Ward's Method

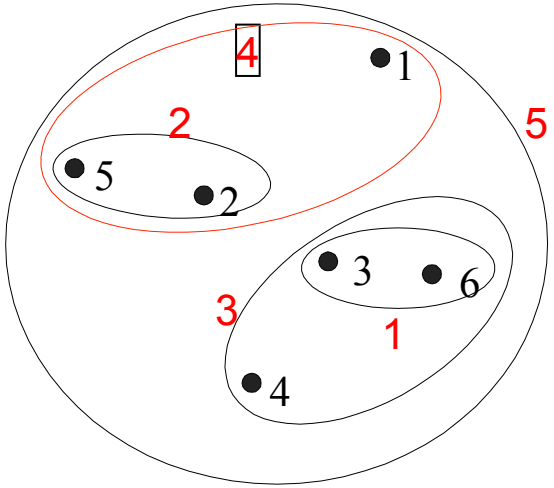
- Similarity of two clusters is based on the increase in squared error when two clusters are merged
  - Similar to group average if distance between points is distance squared
- Less susceptible to noise and outliers
- Biased towards globular clusters
- Hierarchical analogue of K-means
  - Can be used to initialize K-means

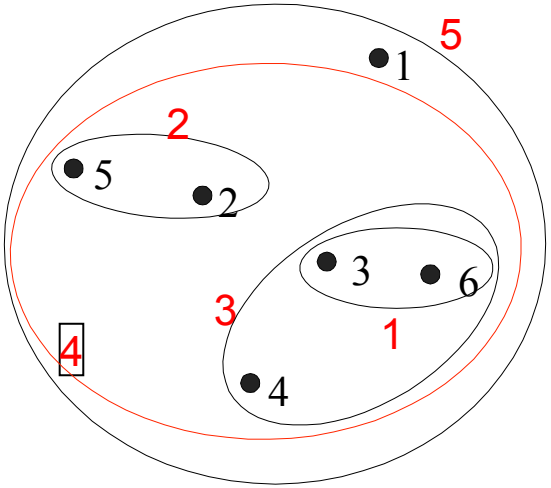
Hierarchical Clustering: Comparison



MIN

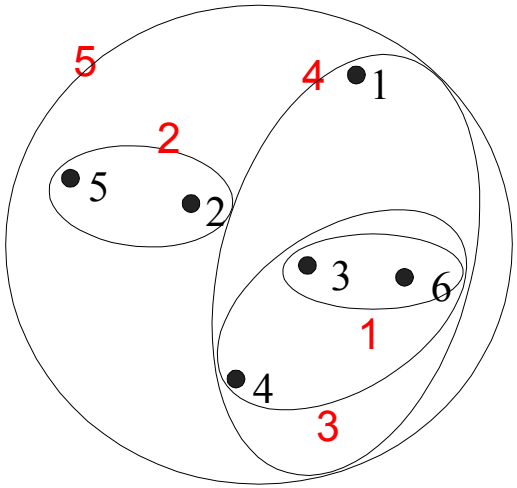
MAX





Group Average

Ward's Method



Hierarchical Clustering: Time and Space requirements

- $O(N^2)$ space since it uses the proximity matrix.
  - N is the number of points.

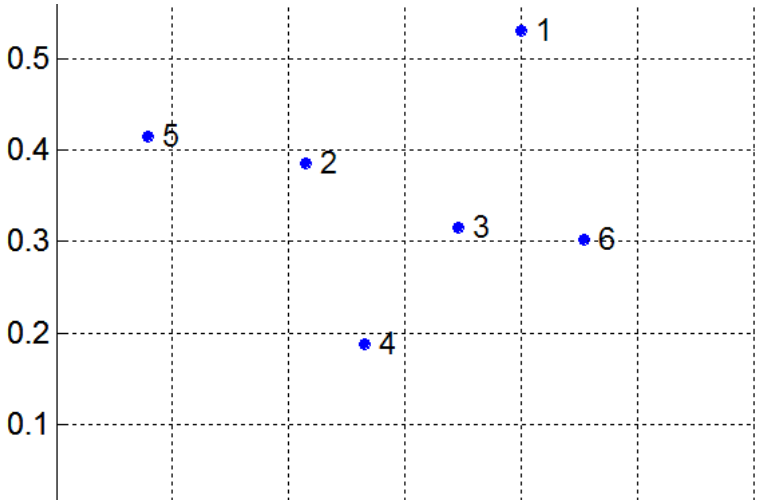
- $O(N^3)$ time in many cases
  - There are N steps and at each step the size, N^2, proximity matrix must be updated and searched
  - Complexity can be reduced to $O(N^2 \log(N))$ time for some approaches

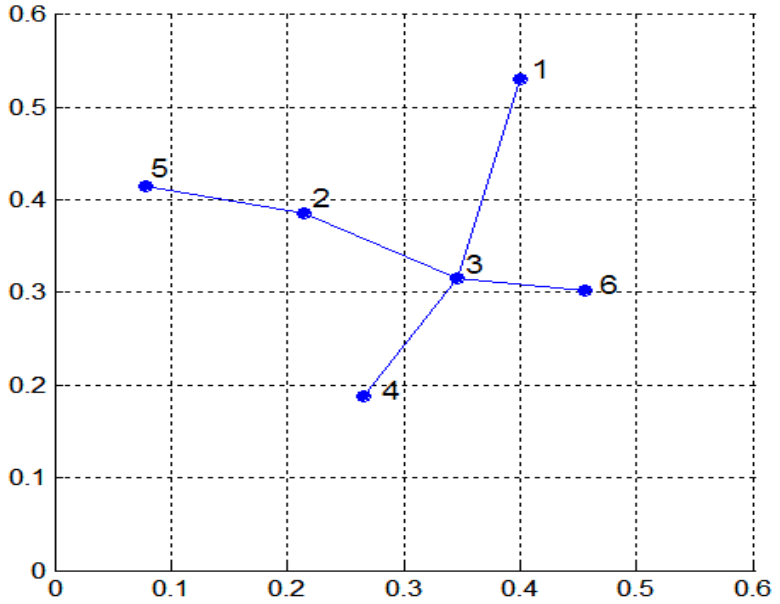
Hierarchical Clustering: Problems and Limitations

- Once a decision is made to combine two clusters, it cannot be undone (one direction)
- No objective function is directly minimized
- Different schemes have problems with one or more of the following:
  - Sensitivity to noise and outliers
  - Difficulty handling different sized clusters and convex shapes
  - Breaking large clusters

MST: Divisive Hierarchical Clustering

- Build MST (Minimum Spanning Tree)
  - Start with a tree that consists of any point
  - In successive steps, look for the closest pair of points (p, q) such that one point (p) is in the current tree but the other (q) is not
  - Add q to the tree and put an edge between p and q





MST: Divisive Hierarchical Clustering

- Use MST for constructing hierarchy of clusters

Algorithm 7.5 MST Divisive Hierarchical Clustering Algorithm

1: Compute a minimum spanning tree for the proximity graph.
2: **repeat**
3: Create a new cluster by breaking the link corresponding to the largest distance (smallest similarity).
4: **until** Only singleton clusters remain

Integration of hierarchical clustering & distance-based

- BIRCH (1996): uses CF-tree and incrementally adjusts the quality of sub-clusters

- CURE (1998): selects well-scattered points from the cluster and then shrinks them towards the center of the cluster by a specified fraction

Scalable Clustering Algorithms

- Clustering algorithms have been designed to handle very large datasets ⟸ which Issue ?
- E.g. the Birch algorithm
  - Main idea: use an in-memory R-tree to store points that are being clustered
  - Insert points one at a time into the R-tree, merging a new point with an existing cluster if is less than some δ distance away
  - If there are more leaf nodes than fit in memory, merge existing clusters that are close to each other
  - At the end of first pass we get a large number of clusters at the leaves of the R-tree
    - Merge clusters to reduce the number of clusters

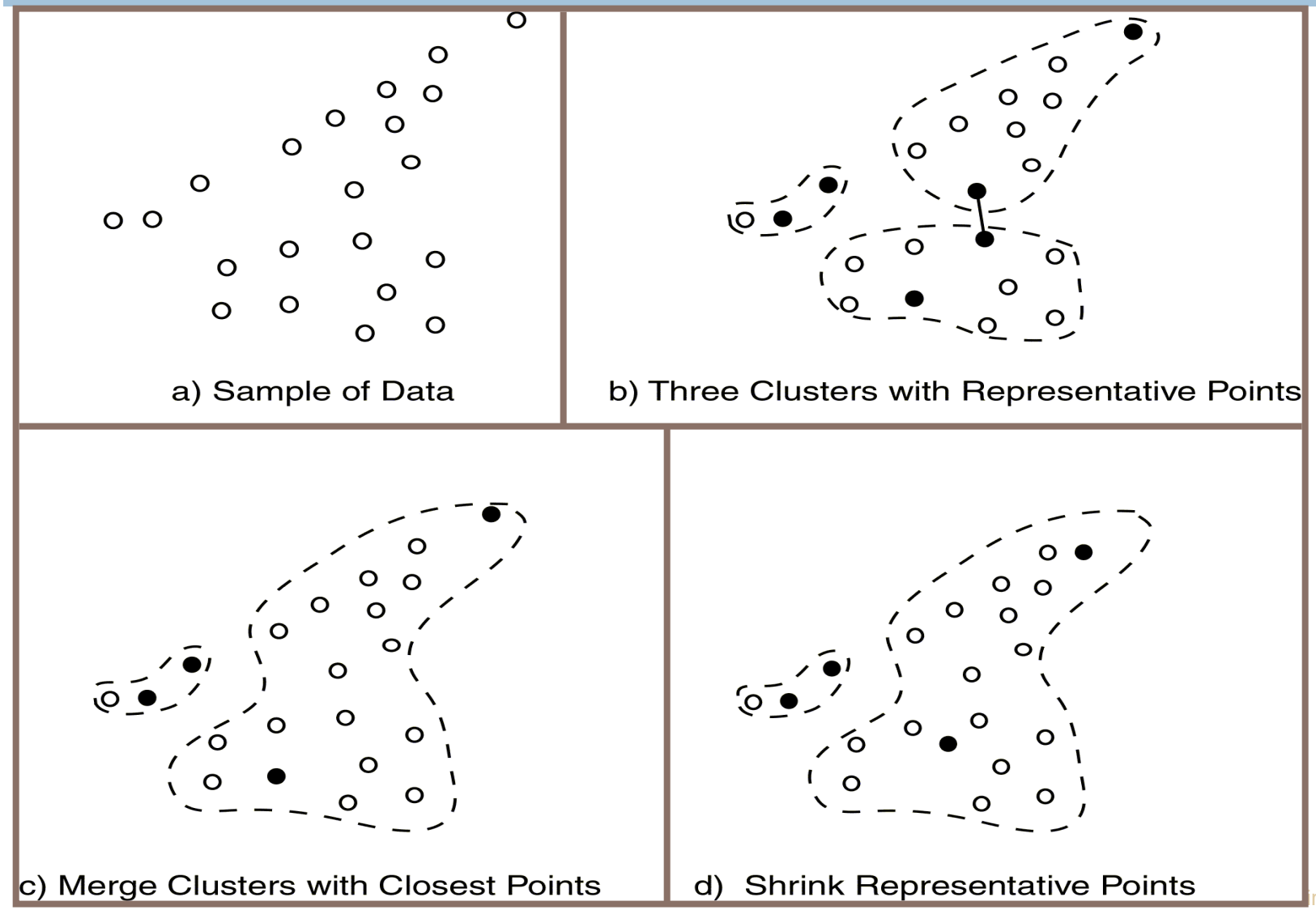
BIRCH

- Birch: Balanced Iterative Reducing and Clustering using Hierarchies
- Incrementally construct a CF (Clustering Feature) tree, a hierarchical data structure for multiphase clustering
- Steps
  - scan DB to build an initial in-memory CF tree
  - use an arbitrary clustering algorithm to cluster the leaf nodes of the CF-tree
- *Scales linearly*: finds a good clustering with a single scan

Cure

- Clustering Using Representatives (CURE)
  - Stops the creation of a cluster hierarchy if a level consists of *k* clusters
- Use many points to represent a cluster instead of only one
  - Uses multiple representative points to evaluate the distance between clusters, adjusts well to arbitrary shaped clusters and avoids single-link effect
  - Points will be well scattered

Cure Example



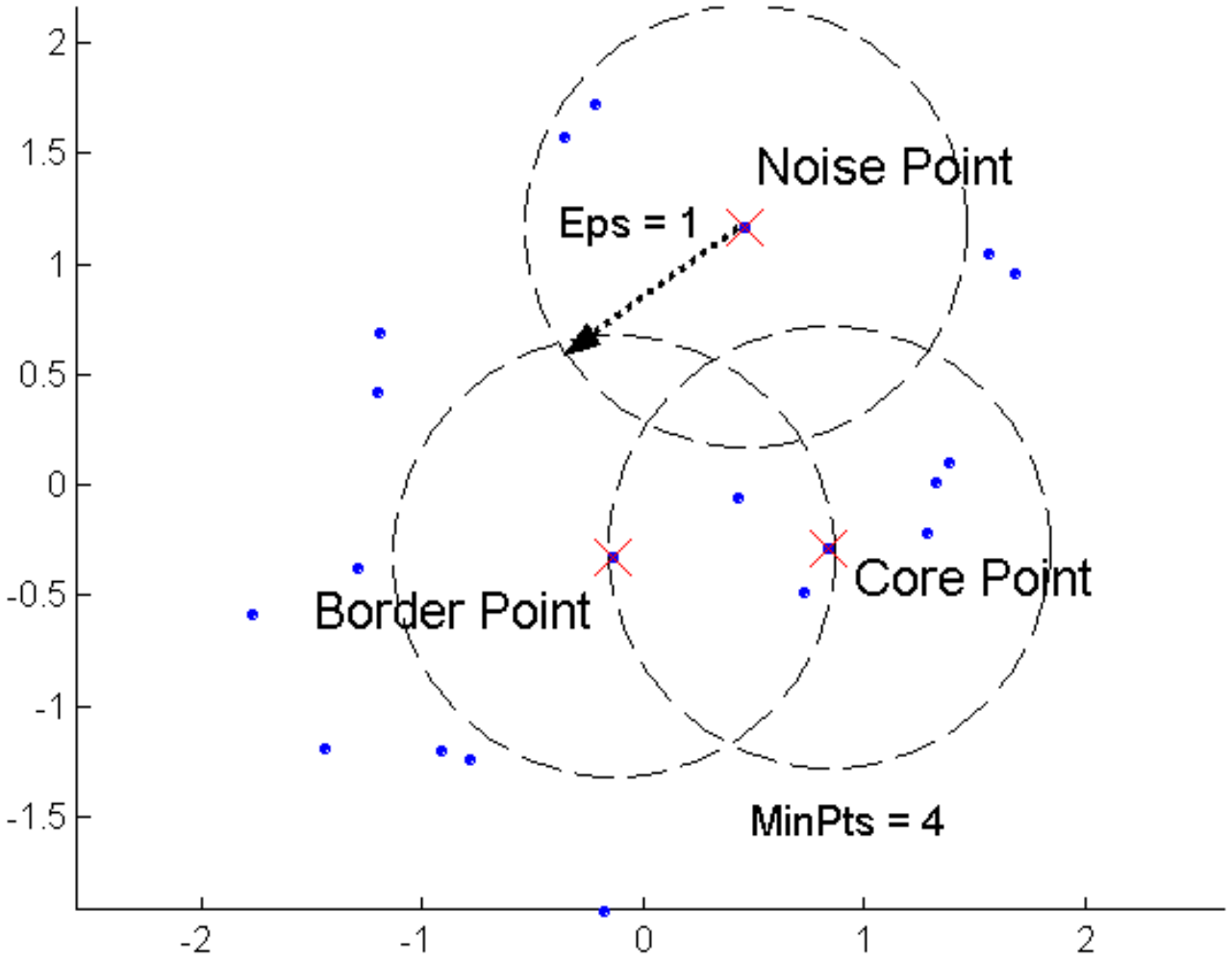
Algorithm of Cure

- Steps
  - Draw random sample *s*.
  - Partition sample to *p* partitions with size *s*/*p*
  - Partially cluster partitions into *s*/*pq* clusters
  - Eliminate outliers
  - Cluster partial clusters
  - Label data in disk

DBSCAN

- DBSCAN is a density-based algorithm.
  - Density = number of points within a specified radius (Eps)
  - A point is a core point if it has more than a specified number of points (MinPts) within Eps
    - These are points that are at the interior of a cluster
  - A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point
  - A noise point is any point that is not a core point or a border point.

DBSCAN: Core, Border, and Noise Points



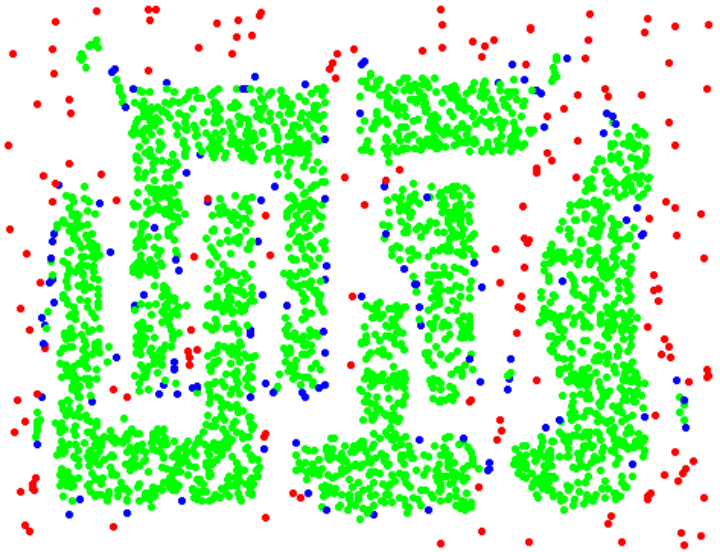
DBSCAN Algorithm

- Eliminate noise points

- $current\_cluster\_label \leftarrow 1$
  for all core points **do**
    if the core point has no cluster label **then**
      $current\_cluster\_label \leftarrow current\_cluster\_label + 1$
      Label the current core point with cluster label $current\_cluster\_label$
    end if
    for all points in the *Eps*-neighborhood, except i^{th} the point itself **do**
      if the point does not have a cluster label **then**
        Label the point with cluster label $current\_cluster\_label$
      end if
    end for
  end for

DBSCAN: Core, Border and Noise Points

Original Points

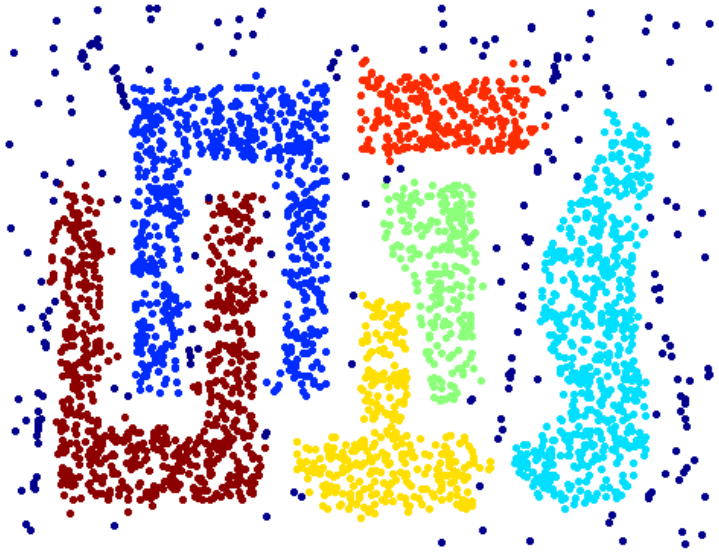


Point types: core, border and noise

Eps = 10, MinPts = 4

When DBSCAN Works Well

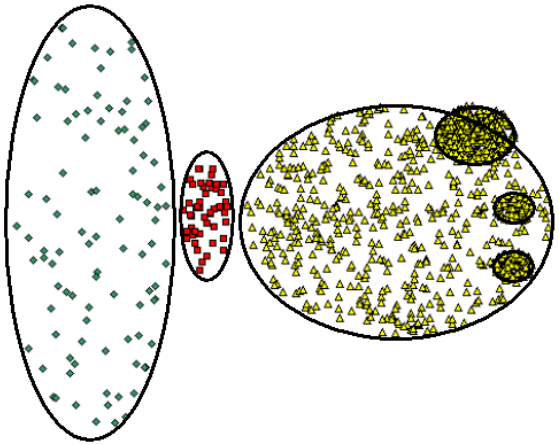
Original Points



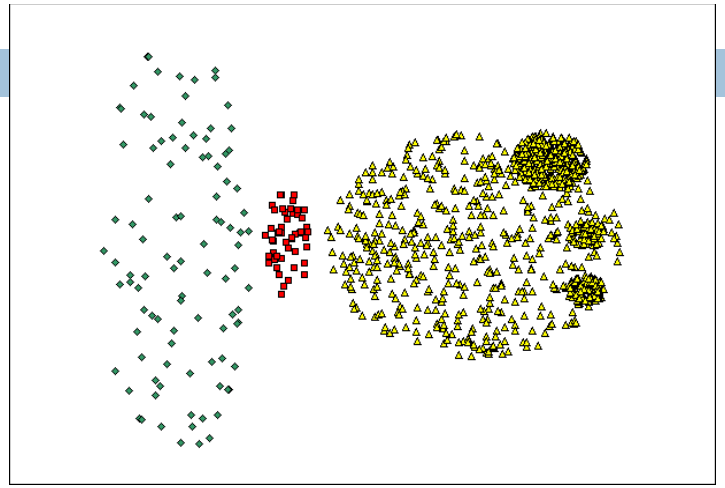
Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

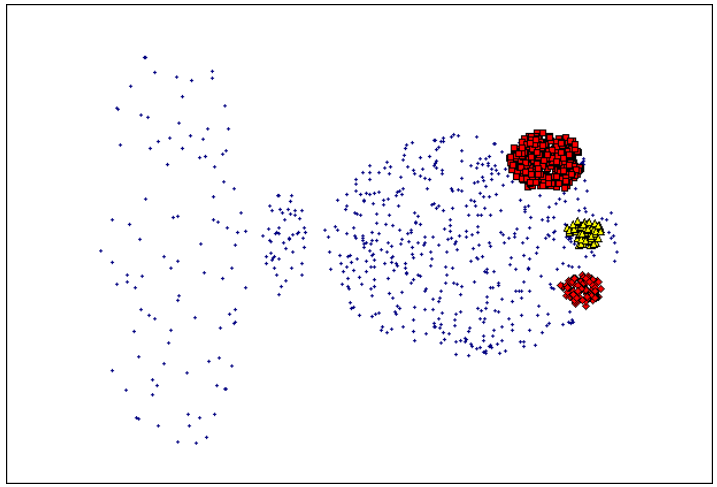
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).

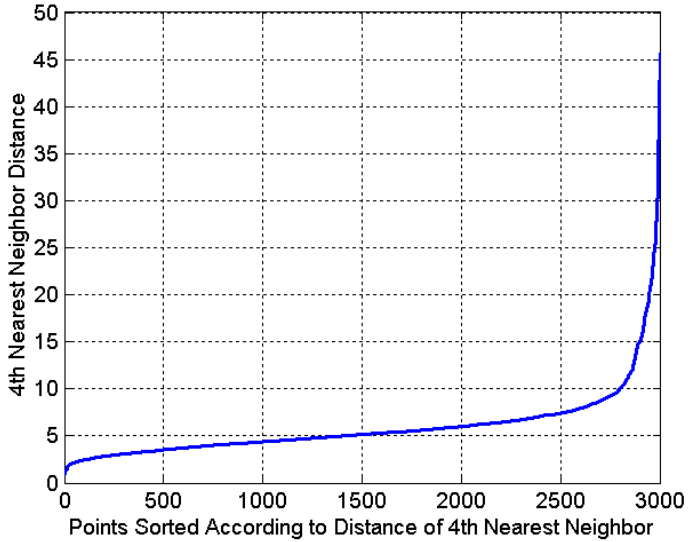


(MinPts=4, Eps=9.92)

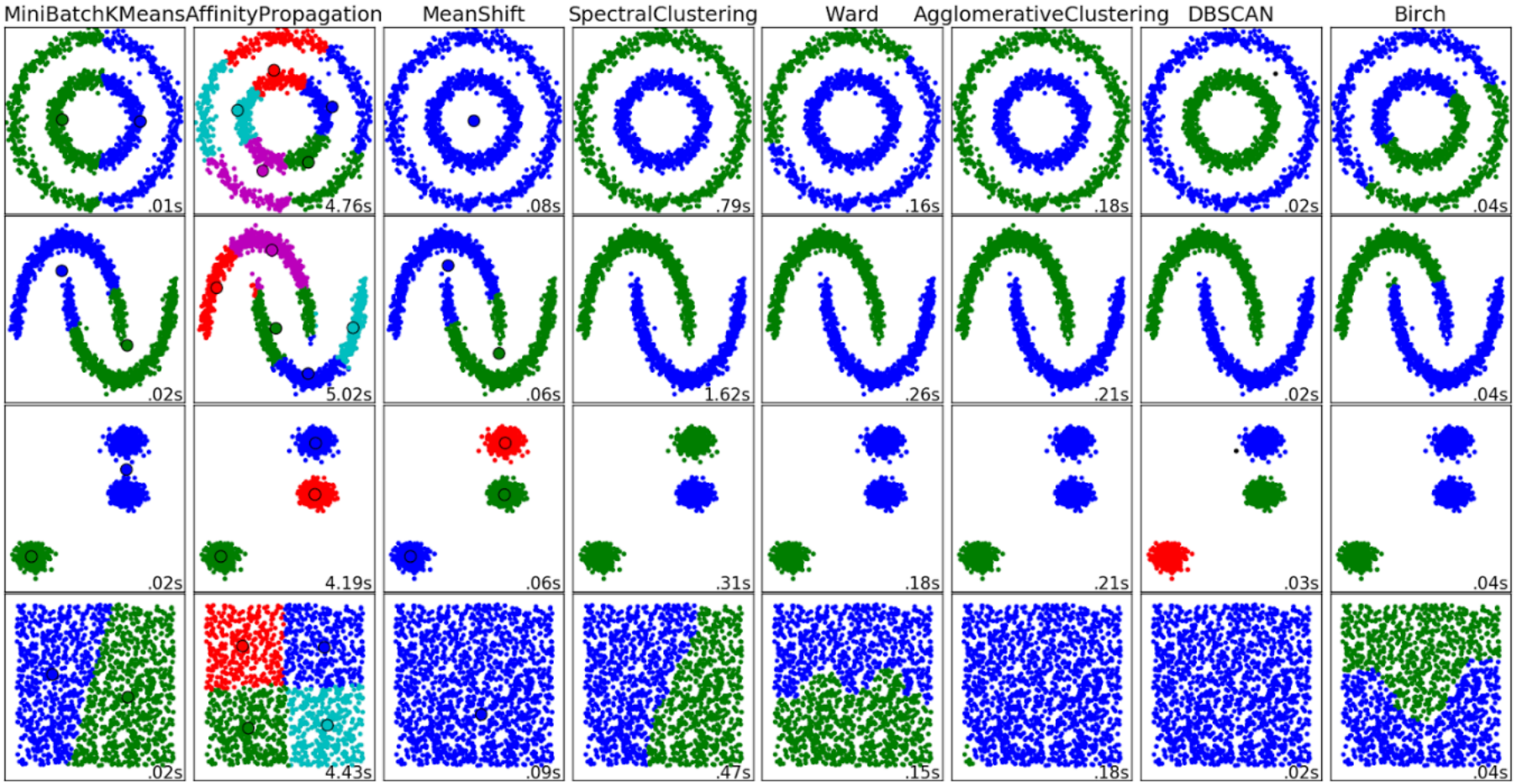
- Varying densities
- High-dimensional data

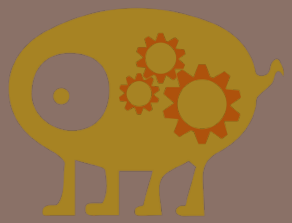
DBSCAN: Determining EPS and MinPts

- Idea is that for points in a cluster, their k^{th} nearest neighbors are at roughly the same distance
- Noise points have the k^{th} nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^{th} nearest neighbor



Clustering is subjective





UNSUPERVISED LEARNING

Supervised Learning vs Unsupervised Learning

01

What happens when our labels are noisy?

- Missing values.
- Labeled incorrectly.

02

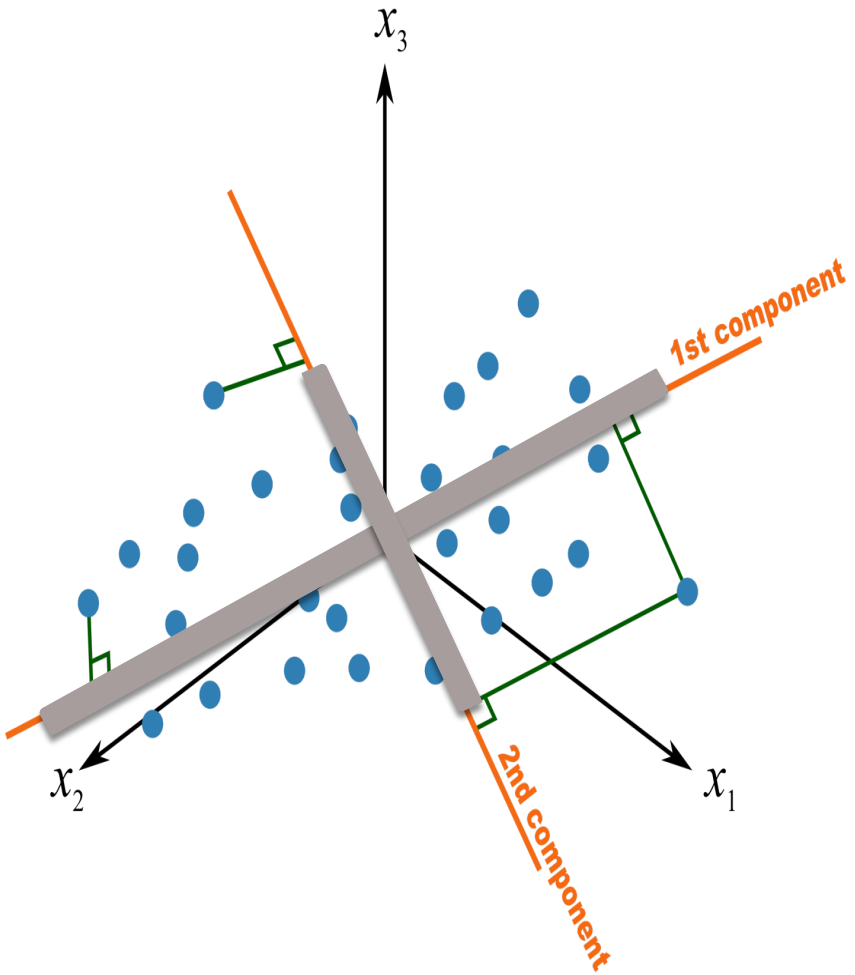
What happens where we don't have labels for training **at all**?

(or implicit labels)

PCA – Principal Component analysis

- Statistical approach for data compression and visualization
- Invented by Karl Pearson in 1901

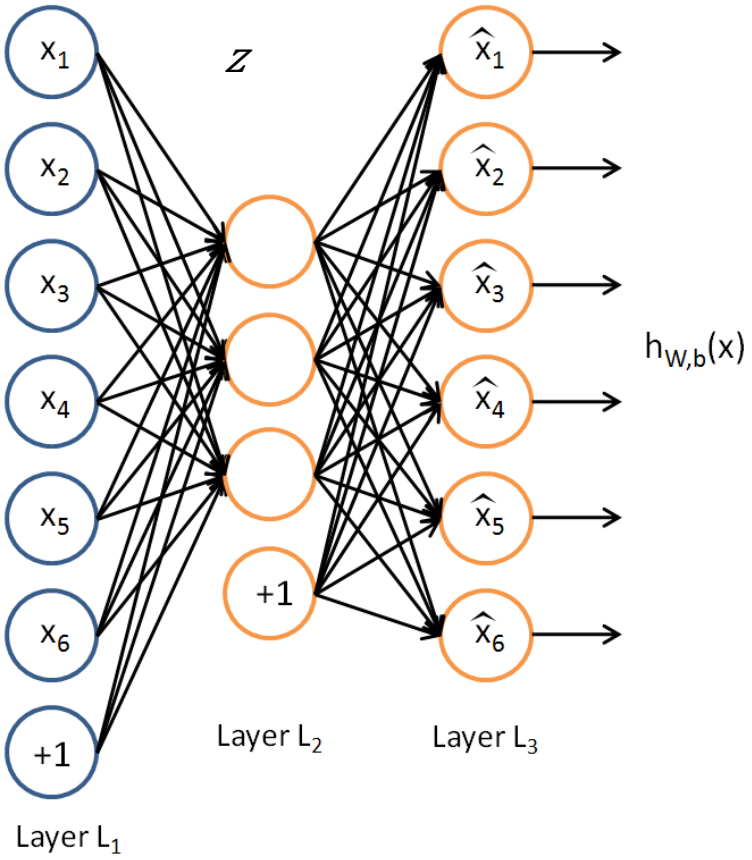
- Weakness: linear components only.



Traditional Autoencoder

- Unlike the **PCA** now we can use activation functions to achieve non-linearity.

- It has been shown that an AE without activation functions achieves the **PCA** capacity.



Uses

- The autoencoder idea was a part of NN history for decades (LeCun et al, 1987).

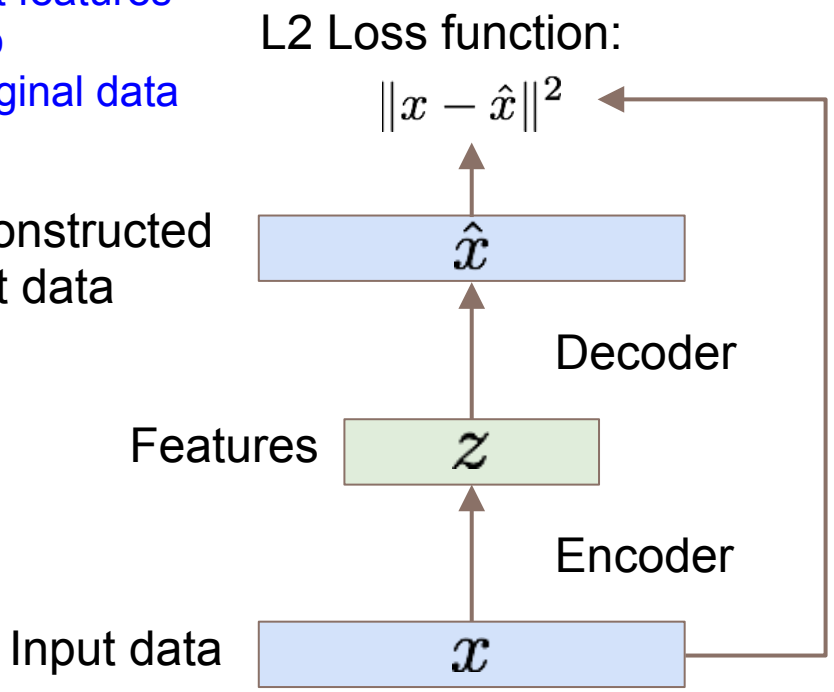
- Traditionally an autoencoder is used for **dimensionality reduction** and **feature/representation learning**.

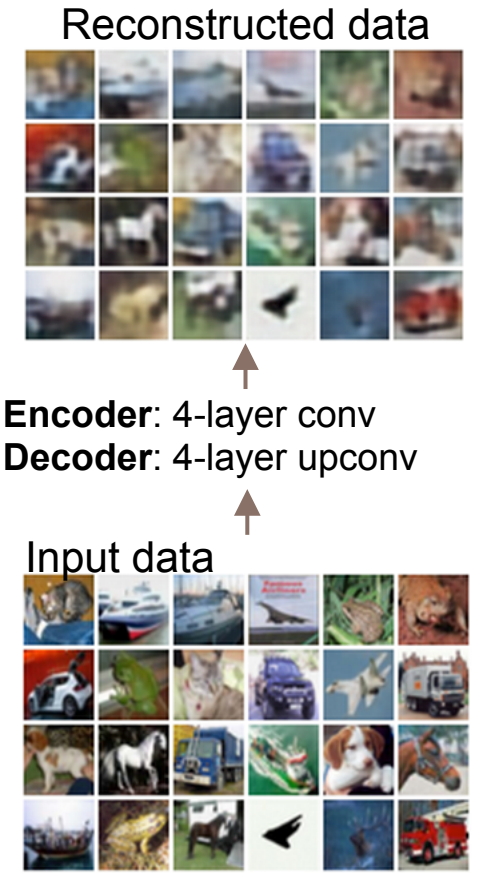
- Recently, the connection between autoencoders and latent space modeling has brought autoencoders to the front of generative modeling.

https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html - By Andrej Karpathy

Autoencoders

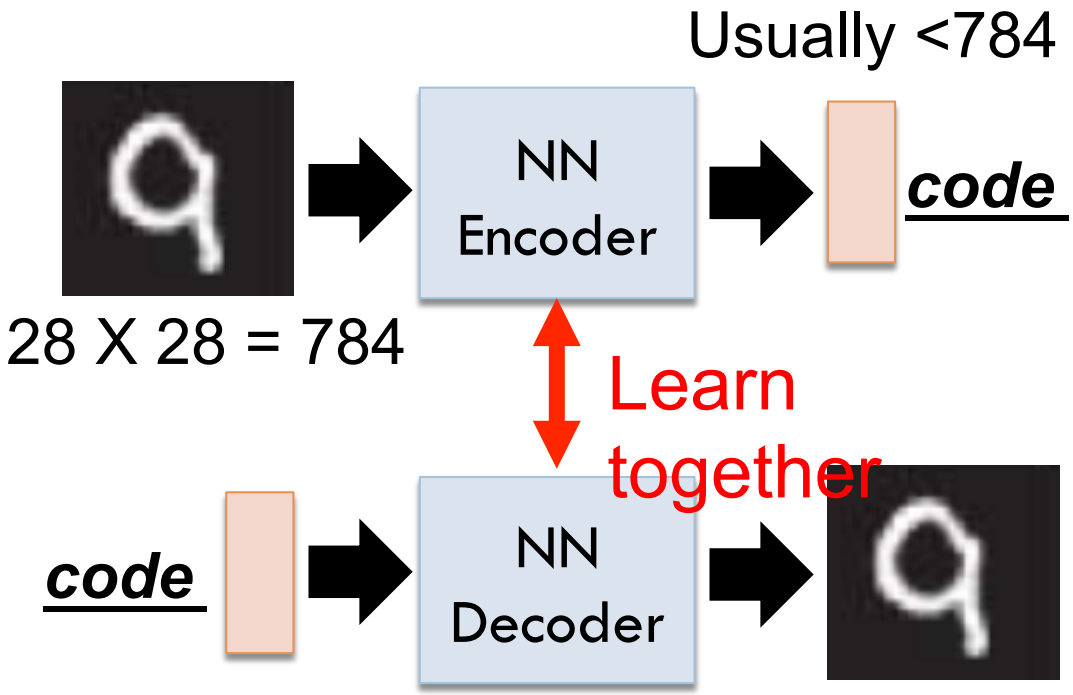
Train such that features can be used to reconstruct original data





Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

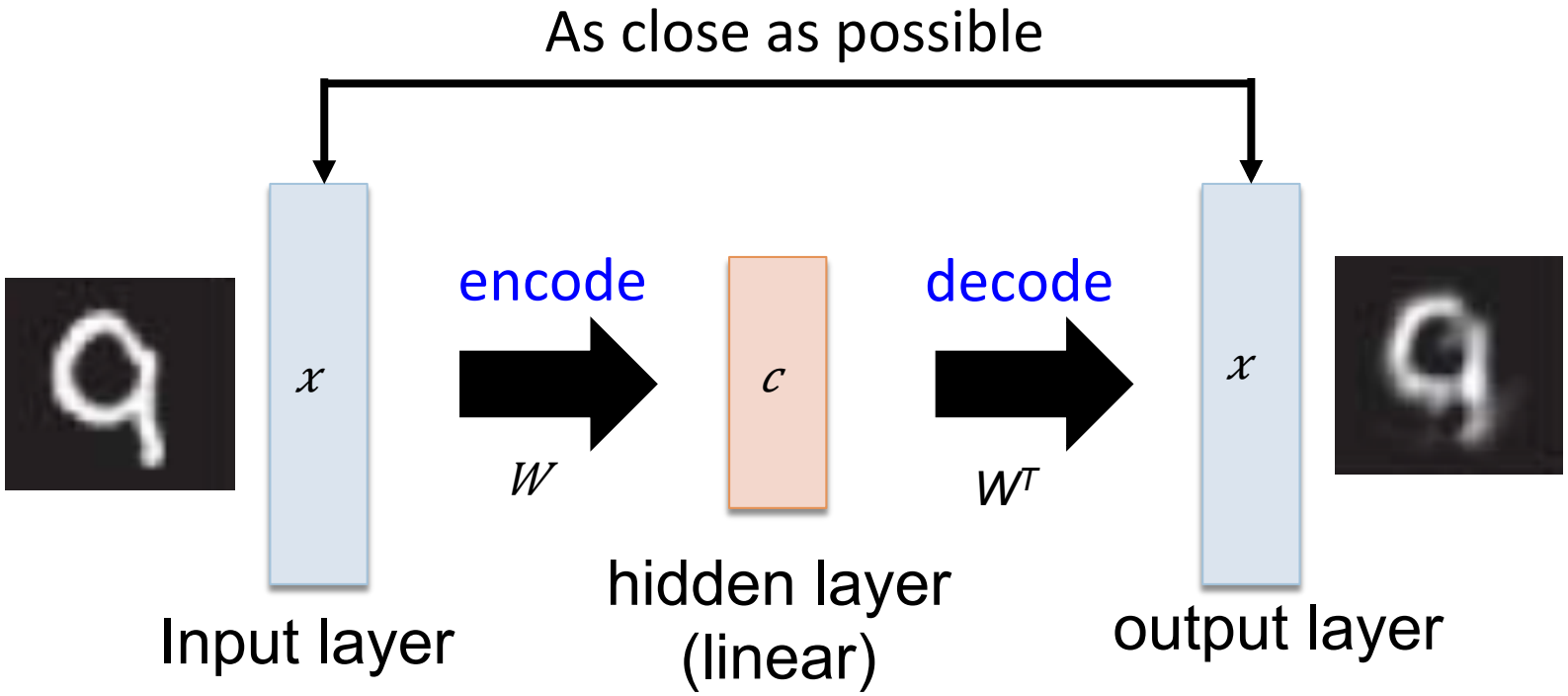
Autoencoder



Compact representation of the input object

Can reconstruct the original object

Recap: PCA

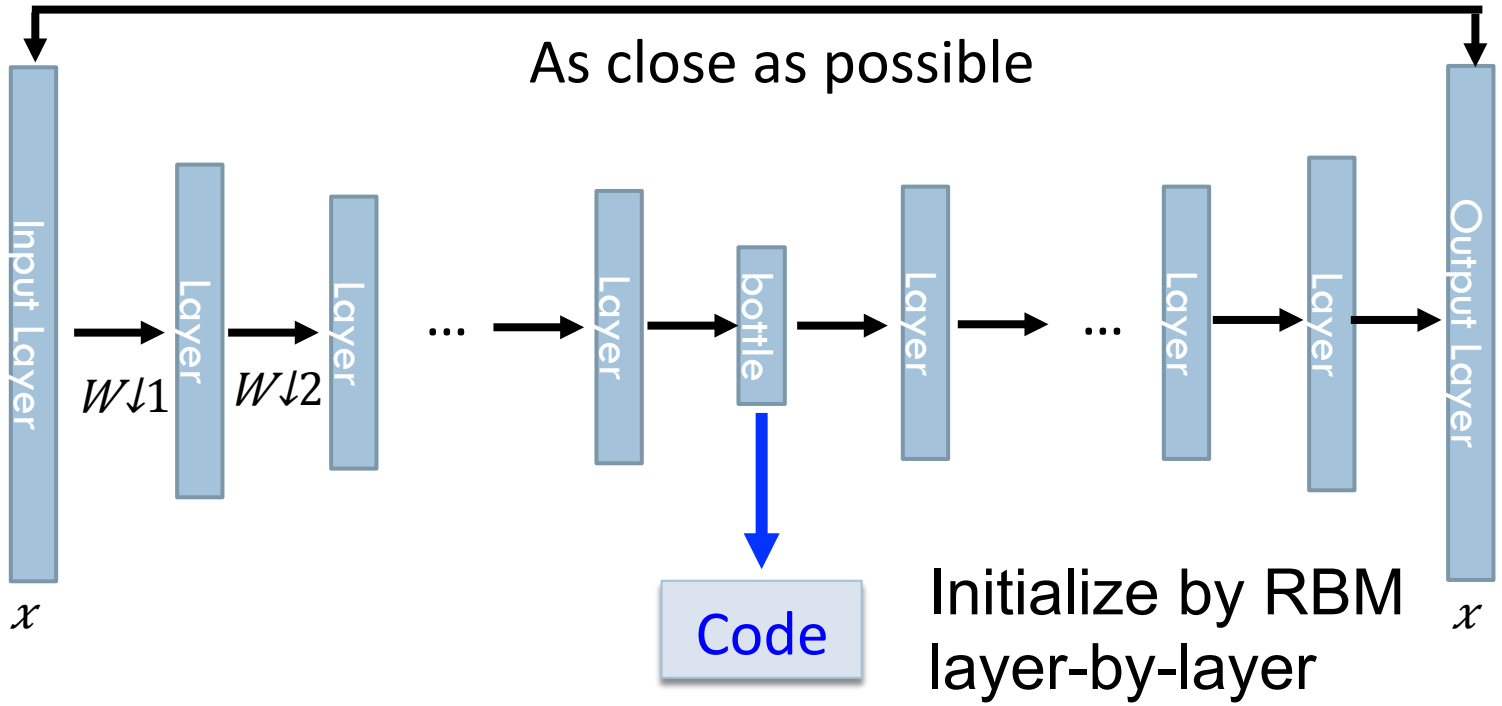


Output of the hidden layer is the code

Deep Autoencoder

Symmetric is not necessary.

- Of course, the auto-encoder can be deep

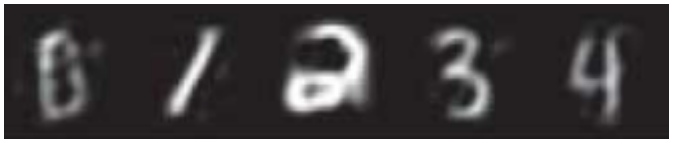


Reference: Hinton, Geoffrey E., and Ruslan R. Salakhutdinov. "Reducing the dimensionality of data with neural networks." *Science* 313.5786 (2006): 504-507

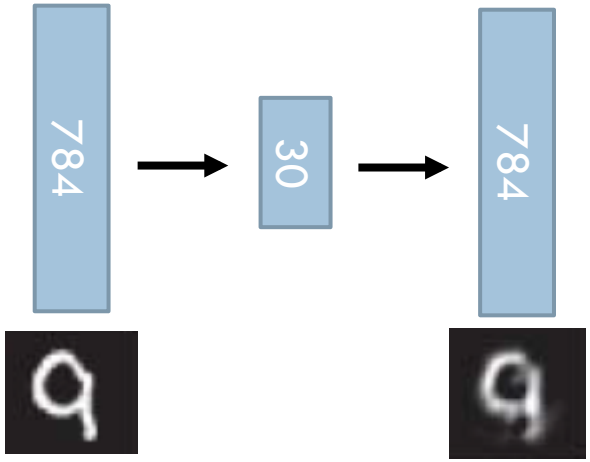
Deep Autoencoder

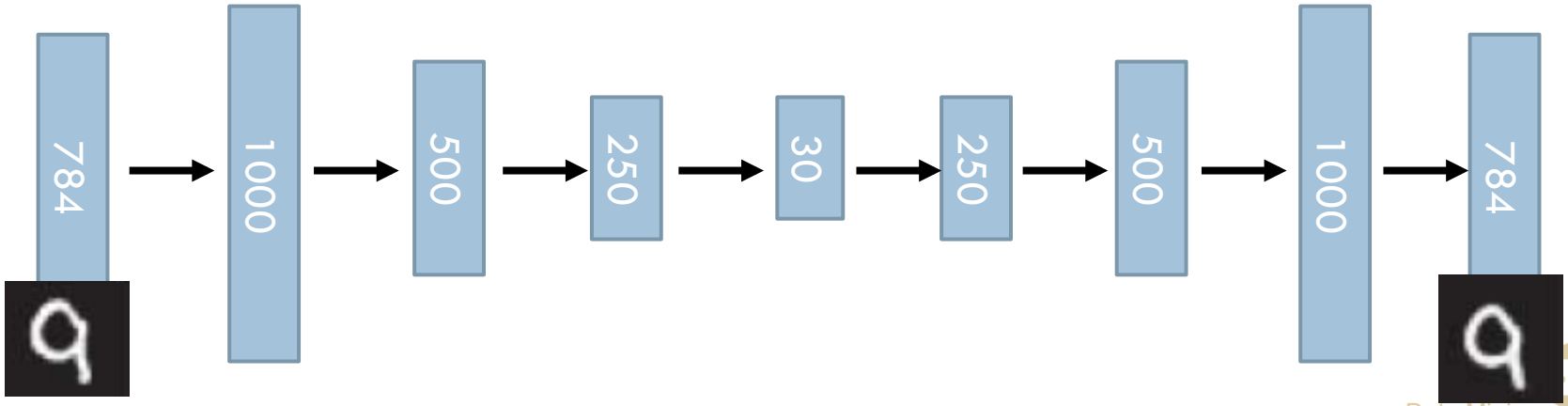
Original Image

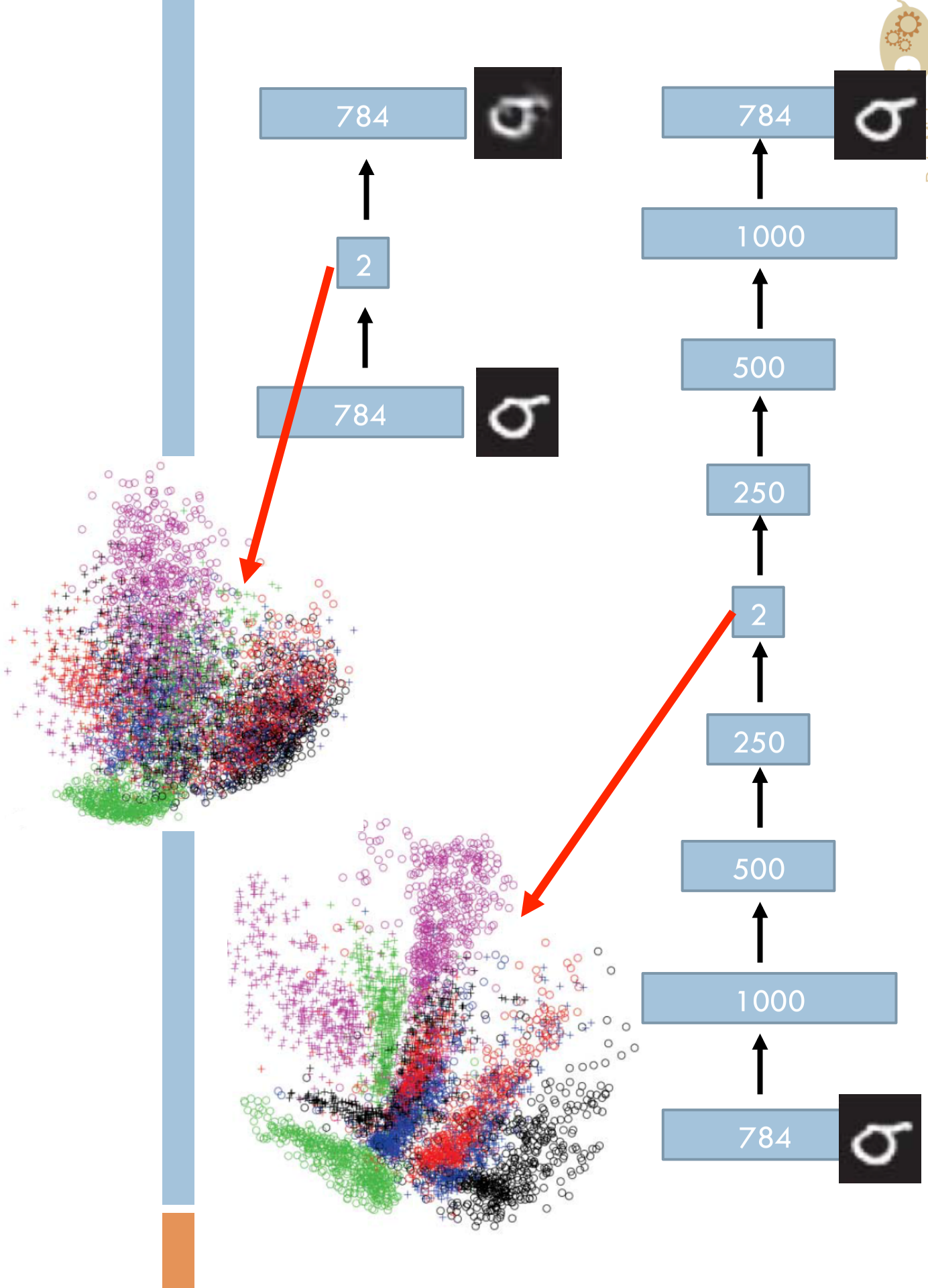
PCA



Deep Auto-encoder





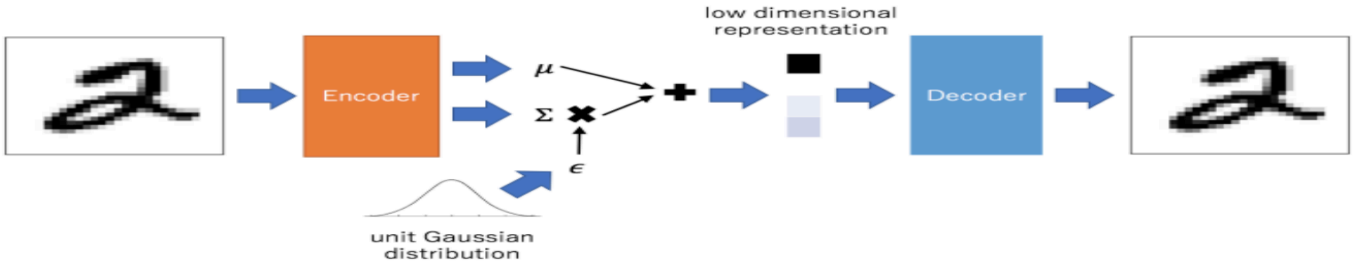

784
2
784
784
1000
500
250
2
250
500
1000
784

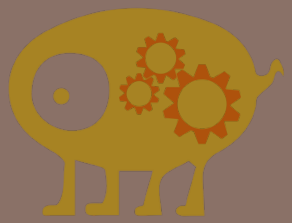
Variational Auto Encoders: Generating Data

Slide Credit: Fei-Fei Li, Justin Johnson, Serena Yeung, CS 231n

32x32 CIFAR-10

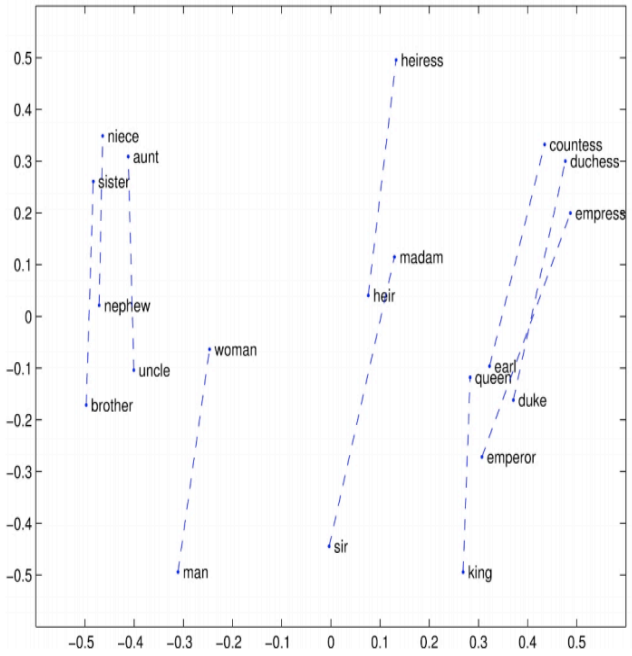
Labeled Faces in the Wild





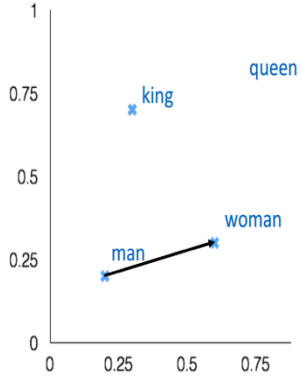
REPRESENTATION LEARNING

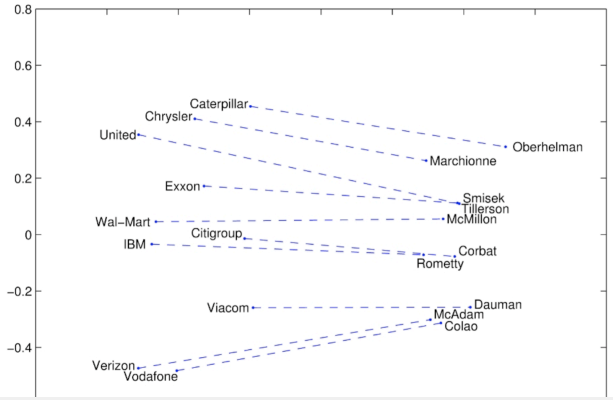
Learning Vector Representation of Words

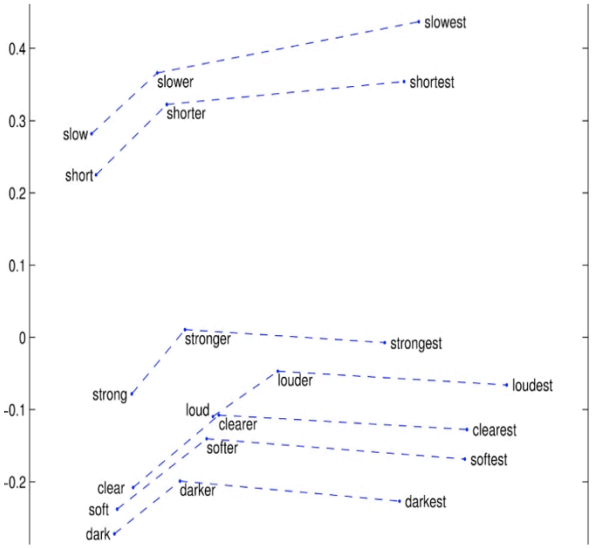


man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]







0. *frog*
1. frogs
2. toad
3. litoria
4. leptodactylidae
5. rana
6. lizard
7. eleutherodactylus

3. litoria

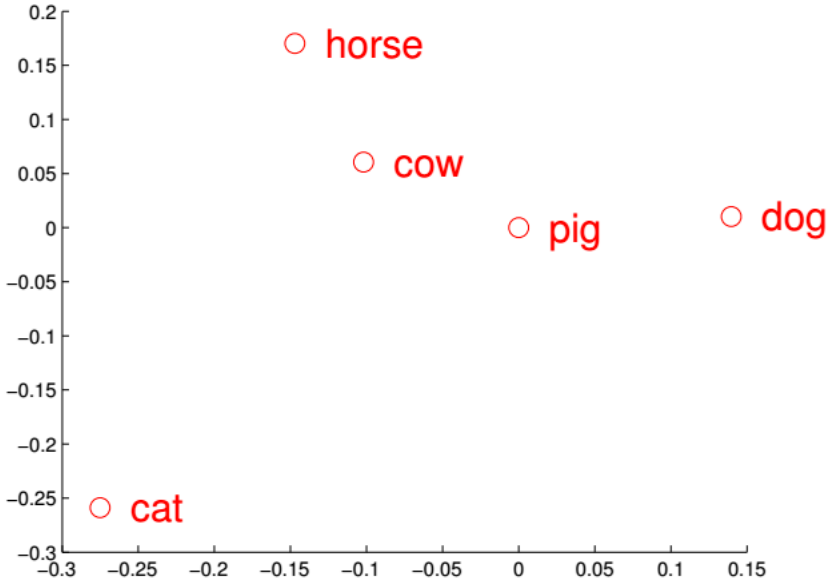
4. leptodactylidae

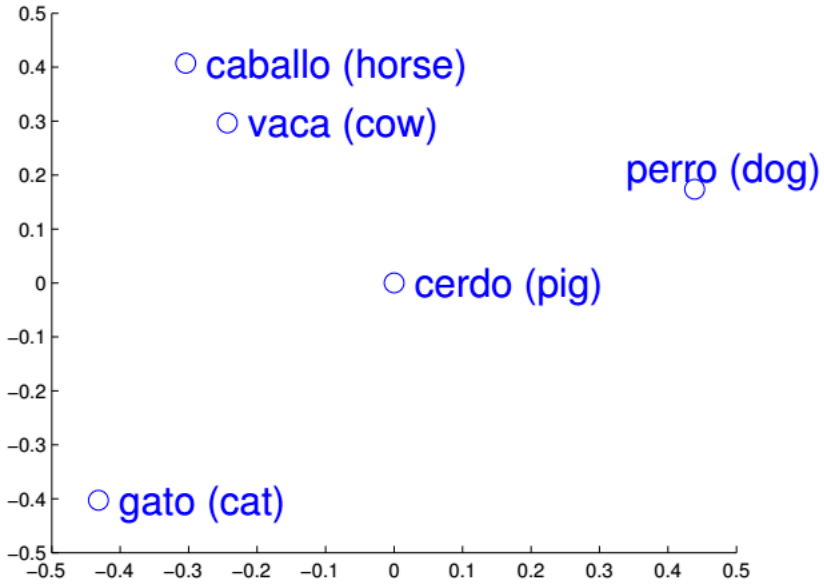
5. rana

7. eleutherodactylus

Representation Vector Usage

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au





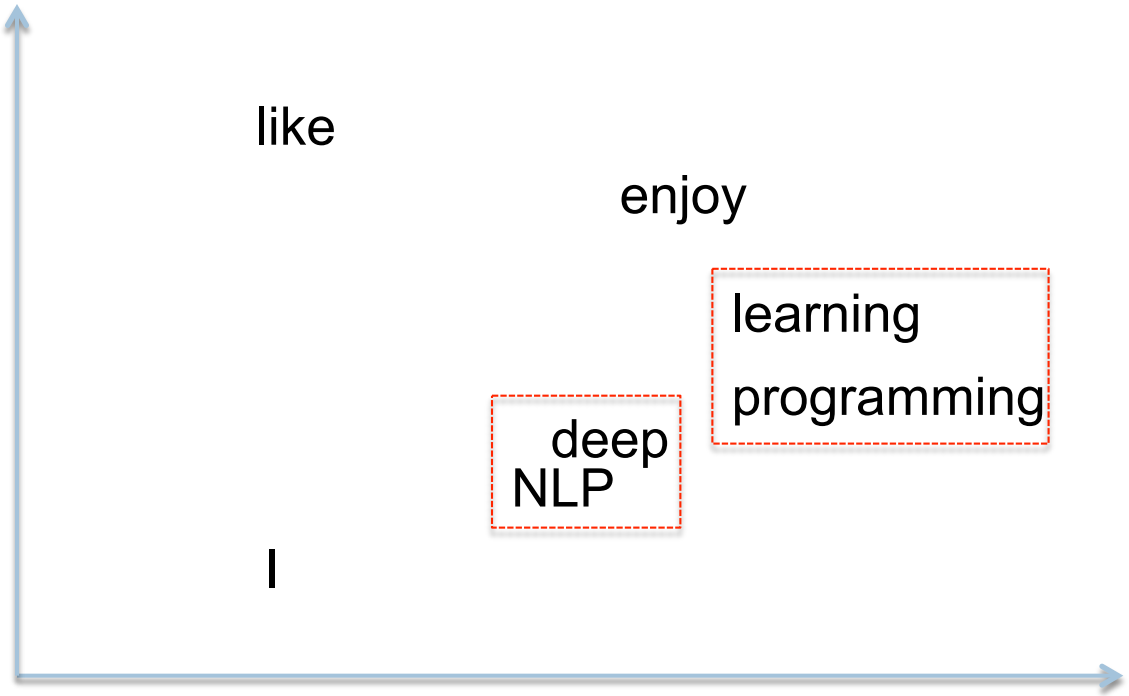
Simple example for word co-occurrence usage

- Training corpus
  - “I like deep learning.”, “I like NLP.”, “I enjoy programming."

cooccurence	I	like	enjoy	deep	learning	NLP	programming
I	0	2	1	0	0	0	0
like	2	0	0	1	0	1	0
enjoy	1	0	0	0	0	0	1
deep	0	1	0	0	1	0	0
learning	0	0	0	1	0	0	0
NLP	0	1	0	0	0	0	0
programming	0	0	1	0	0	0	0

Simple example for word co-occurrence usage

- SVD on this co-occurrence matrix
  - Reduce dimension
- Use the 2 biggest singular value to represent words



Latent Semantic Indexing(LSI)

- A statistical technique
- Uses linear algebra technique called *singular value decomposition (SVD)*
  - attempts to estimate the hidden structure that generates terms given concepts
  - discovers the most important associative patterns between words and concepts
- Data driven
  - A large collection of sentences or documents is employed

LSI and Text Documents

- Let **X** denote a term-document matrix

$$X = [x_1 \ldots x_n]^T$$

  - each row is the vector-space representation of a document
  - each column contains occurrences of a term in each document in the dataset

- Latent semantic indexing
  - compute the SVD of *X*: $X = U\Sigma V^T$
    - Σ - singular value matrix *(diagonal matrix)* $\hat{\Sigma}$
  - set to zero all but largest *K* singular values -
  - obtain the reconstruction of *X* by: $\hat{X} = U\hat{\Sigma}V^T$

LSI Example

- A collection of documents:

Linux OS

d1: Indian government goes for open-source software

d2: Debian 3.0 Woody released

d3: Wine 2.0 released with fixes for Gentoo 1.4 and Debian 3.0

d4: gnuPOD released: iPOD on Linux… with GPLed software

d5: Gentoo servers running at open-source mySQL database

d6: Dolly the sheep not totally identical clone

d7: DNA news: introduced low-cost human genome DNA chip

d8: Malaria-parasite genome database on the Web

d9: UK sets up genome bank to protect rare sheep breeds

d10: Dolly's DNA damaged

Genome news

LSI Example

- The term-document matrix X^T

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
open-source	1	0	0	0	1	0	0	0	0	0
software	1	0	0	1	0	0	0	0	0	0
Linux	0	0	0	1	0	0	0	0	0	0
released	0	1	1	1	0	0	0	0	0	0
Debian	0	1	1	0	0	0	0	0	0	0
Gentoo	0	0	1	0	1	0	0	0	0	0
database	0	0	0	0	1	0	0	1	0	0
Dolly	0	0	0	0	0	1	0	0	0	1
sheep	0	0	0	0	0	1	0	0	0	0
genome	0	0	0	0	0	0	1	1	1	0
DNA	0	0	0	0	0	0	2	0	0	1

LSI Example

- The reconstructed term-document matrix $\hat{X}^T$ after projecting on a subspace of dimension K=2
- Σ = diag(2.57, 2.49, 1.99, 1.9, 1.68, 1.53, 0.94, 0.66, 0.36, 0.10)
- $\hat{\Sigma}$ = diag(2.57, 2.49, 0, 0, 0, 0, 0, 0, 0, 0)

	d1	d2	d3	d4	d5	d6	d7	d8	d9	d10
open-source	0.34	0.28	0.38	0.42	0.24	0.00	0.04	0.07	0.02	0.01
software	0.44	0.37	0.50	0.55	0.31	-0.01	-0.03	0.06	0.00	-0.02
Linux	0.44	0.37	0.50	0.55	0.31	-0.01	-0.03	0.06	0.00	-0.02
released	0.63	0.53	0.72	0.79	0.45	-0.01	-0.05	0.09	-0.00	-0.04
Debian	0.39	0.33	0.44	0.48	0.28	-0.01	-0.03	0.06	0.00	-0.02
Gentoo	0.36	0.30	0.41	0.45	0.26	0.00	0.03	0.07	0.02	0.01
database	0.17	0.14	0.19	0.21	**0.14**	0.04	0.25	**0.11**	0.09	0.12
Dolly	-0.01	-0.01	-0.01	-0.02	0.03	0.08	0.45	0.13	0.14	0.21
sheep	-0.00	-0.00	-0.00	-0.01	0.03	0.06	0.34	0.10	0.11	0.16
genome	0.02	0.01	0.02	0.01	0.10	0.19	1.11	0.34	0.36	0.53
DNA	-0.03	-0.04	-0.04	-0.06	0.11	0.30	**1.70**	0.51	0.55	0.81

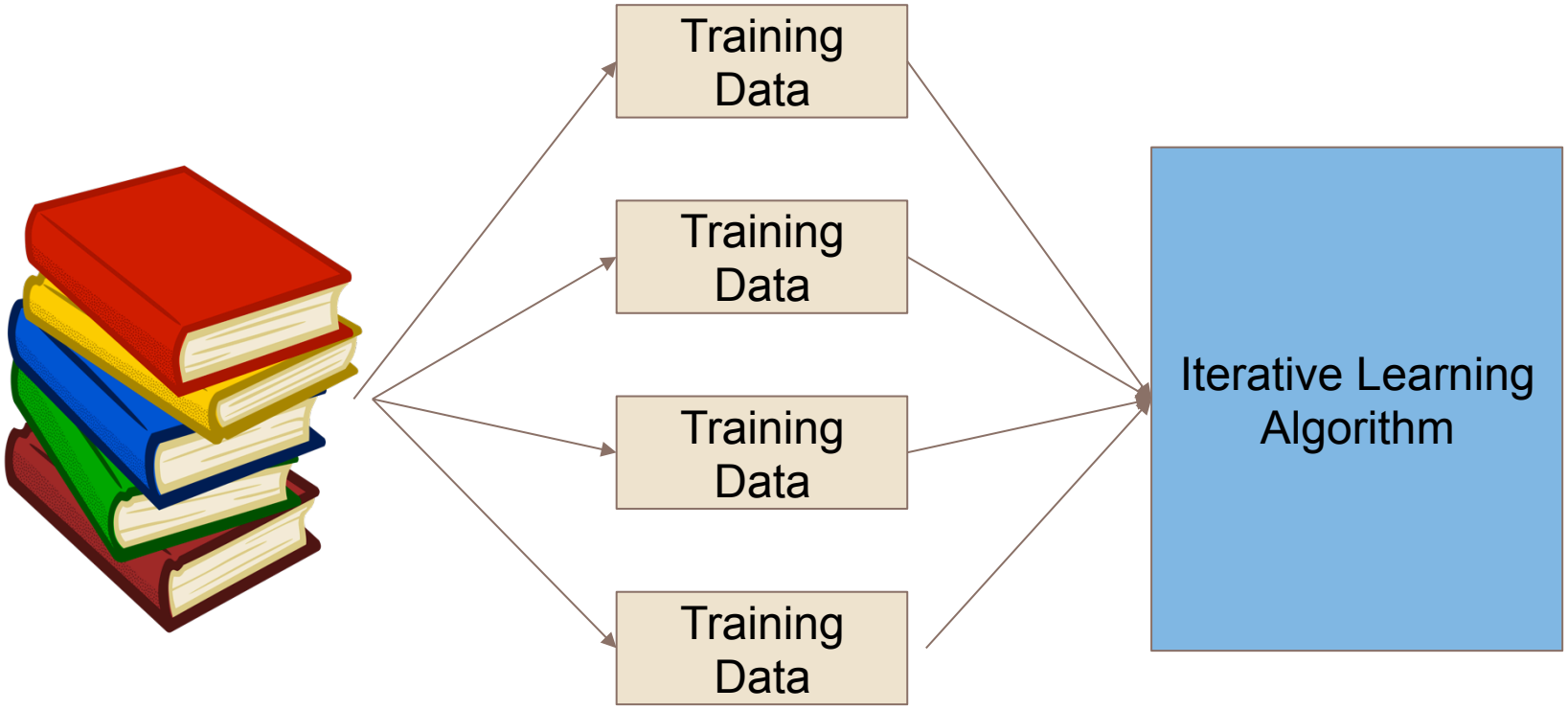
database	0	0	0	0	1	0	0	1	0	0

Problems

- Original matrix still has same dimension as our vocabulary size - potentially 100,000 x 100,000
- Matrix is extremely sparse
- We can perform Singular Value Decomposition (SVD) for dimensionality reduction, however at quadratic computational cost
- Adding a new word changes the entire matrix

Word2vec

Word2vec is an *iterative* training method: working with a huge matrix is cumbersome - instead, learning proceeds by considering small chunks of data at a time



Probabilistic Model: Some Choices

Unary language model

$$P(w_1, \cdots, w_n) = \prod_i P(w_i)$$

Ridiculous not to consider word order

Binary language model

$$P(w_1, \cdots, w_n) = \prod_i P(w_i | w_{i-1})$$

Better but still limited by short context distance

word2vec models (using window m to get more context)

Continuous Bag of Words (CBOW)

$$P(w_c | w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m})$$

The cat [center word] its fur

Skip-Gram (which we will focus on)

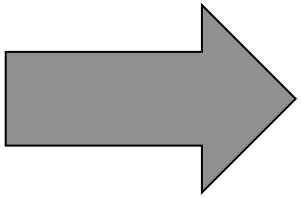
$$P(w_{c-m}, \ldots, w_{c-1}, w_{c+1}, \ldots, w_{c+m} | w_c)$$

[left context] licked [right context]

Skip-Gram: Training Data

Corpus

The cat licked its fur. The truck moved.



Context window size = 1

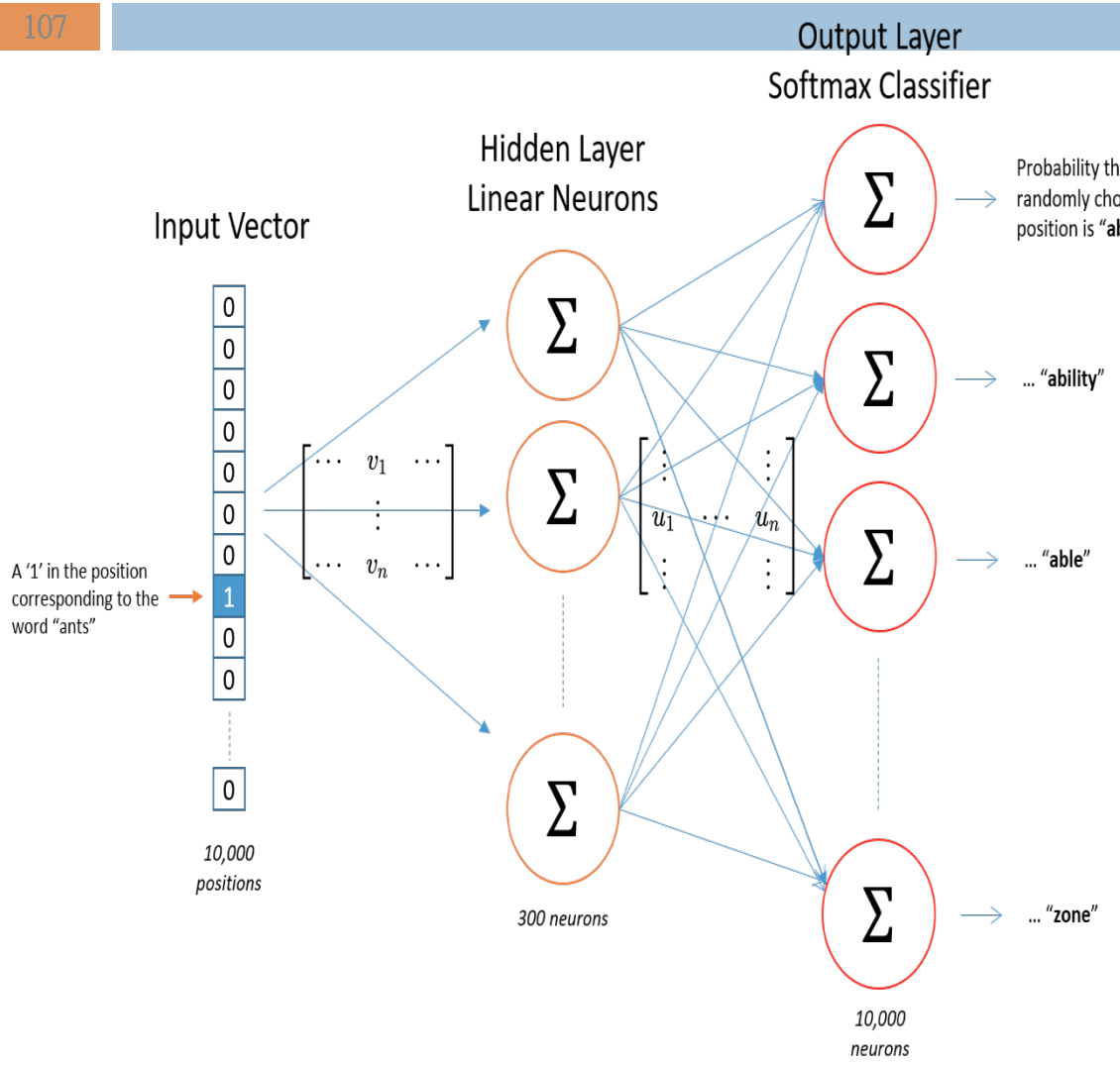
What if context size > 1?

- More word pairs
- Only pass two words to the model at a time

Training Word Pairs

Center	Context
the	cat
the	truck
cat	the
cat	licked
licked	cat
licked	its
…	…

Generate Probability Estimates



(1) Input vector selects input embedding from hidden layer matrix

(2) Softmax over multiplication with output matrix creates probability distribution over the vocabulary

This should be high for context words, low for others

Network Input

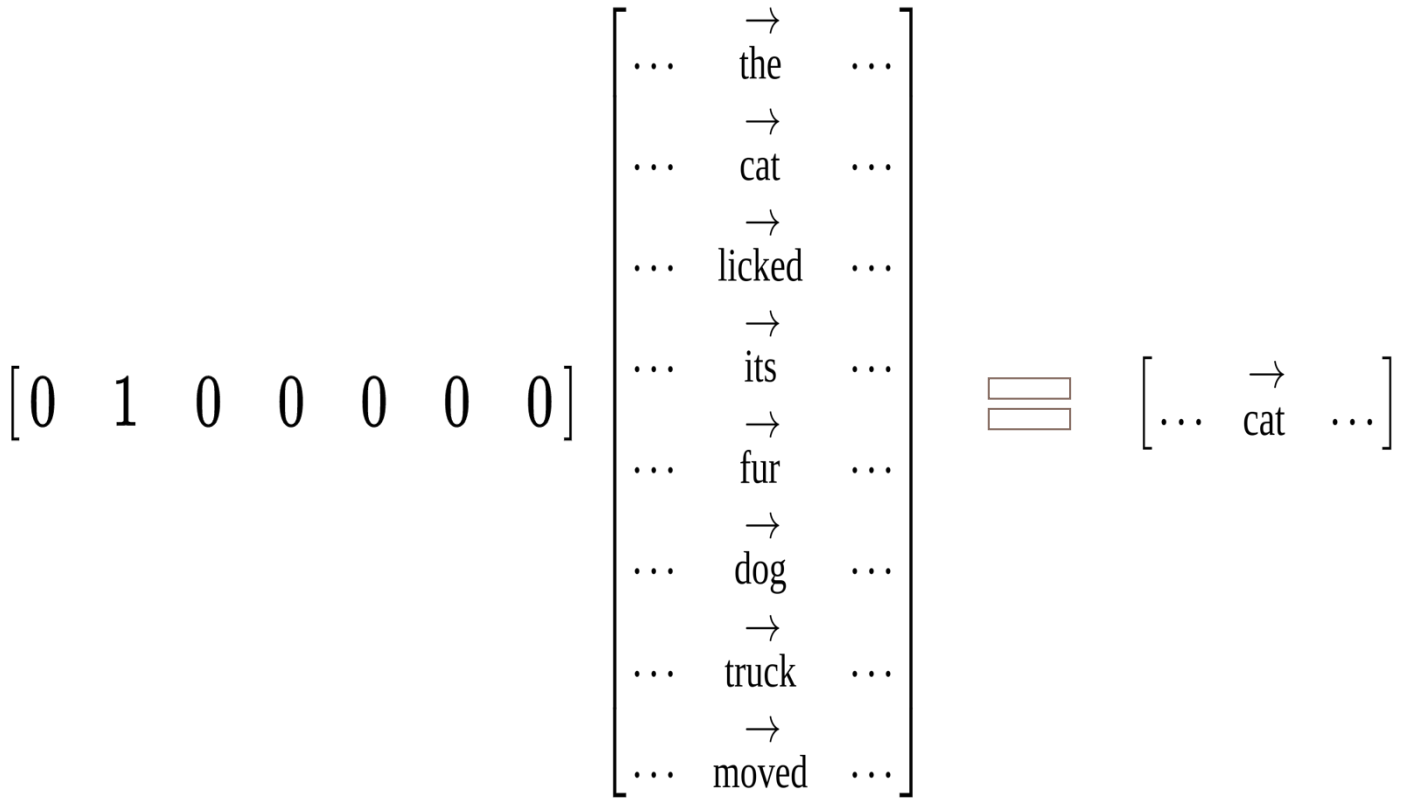
We imagine the words are encoded as "one-hot" vectors (all zeros except a one at the the word index in the embedding matrix)

```
vocab = {
    'the': 1,
    'cat': 2,
    'licked': 3,
    'its': 4,
    'fur': 5,
    'truck': 6,
    'dog': 7,
    'moved': 8}
```

$$\text{the} = \begin{bmatrix}1\\0\\0\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{cat} = \begin{bmatrix}0\\1\\0\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{licked} = \begin{bmatrix}0\\0\\1\\0\\0\\0\\0\\0\end{bmatrix} \quad \text{etc...}$$

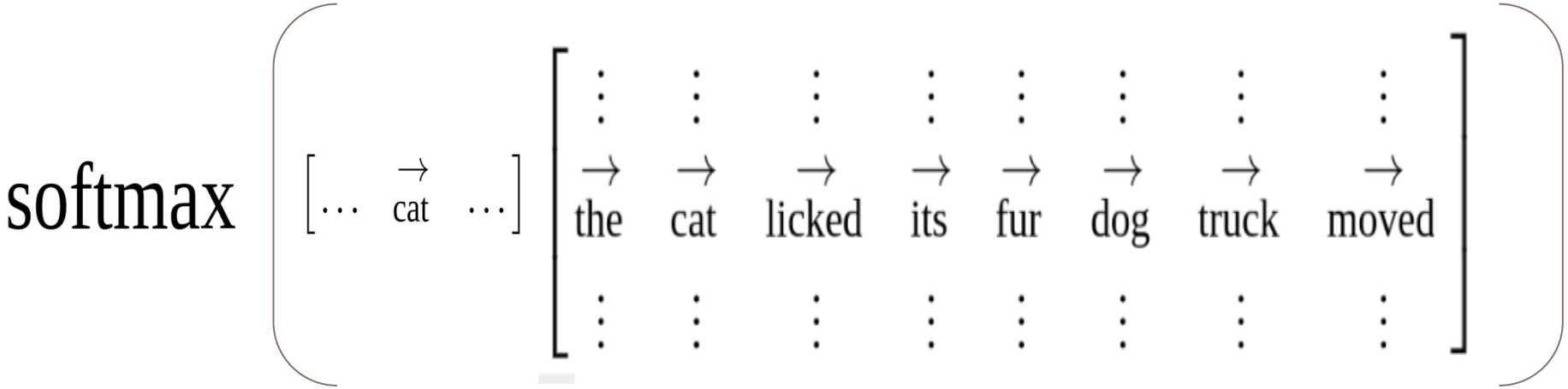
Embedding Lookup

In practice we don't do matrix multiplication, but just use the word index to pick out the vector.



Probability Over Vocab

Our word vector does a dot product with every word's output embedding, then applying softmax gives a probability distribution over the vocabulary



$$\text{probability word } w_i \text{ in context} = \hat{y}_i = P(w_i|\mathbf{v}_c, \mathbf{U}) = \frac{\exp(\mathbf{u}_{w_i}^T \mathbf{v}_c)}{\sum_{j=1}^{V} \exp(\mathbf{u}_{w_j}^T \mathbf{v}_c)}$$

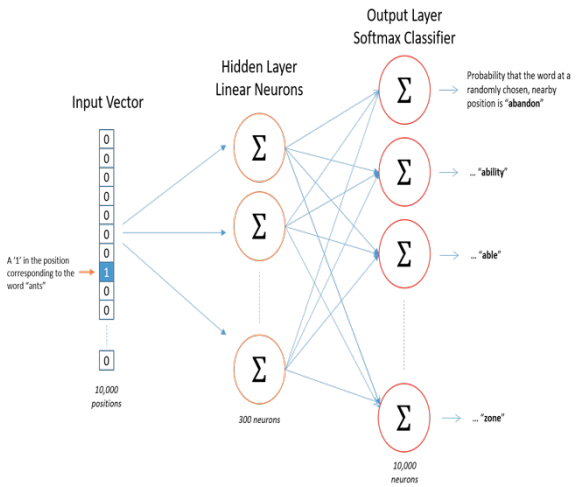
Calculate Error in Estimates

Training Pair:

$$\text{center word} = \text{cat} = [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0] = \text{input}$$

$$\text{context word} = \text{licked} = [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0] = \text{target}$$

$$[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]$$



$$\hat{\mathbf{y}} = [0.1\ \ 0.0\ \ 0.1\ \ 0.3\ \ 0.2\ \ 0.1\ \ 0.1\ \ 0.1]$$

Loss Function

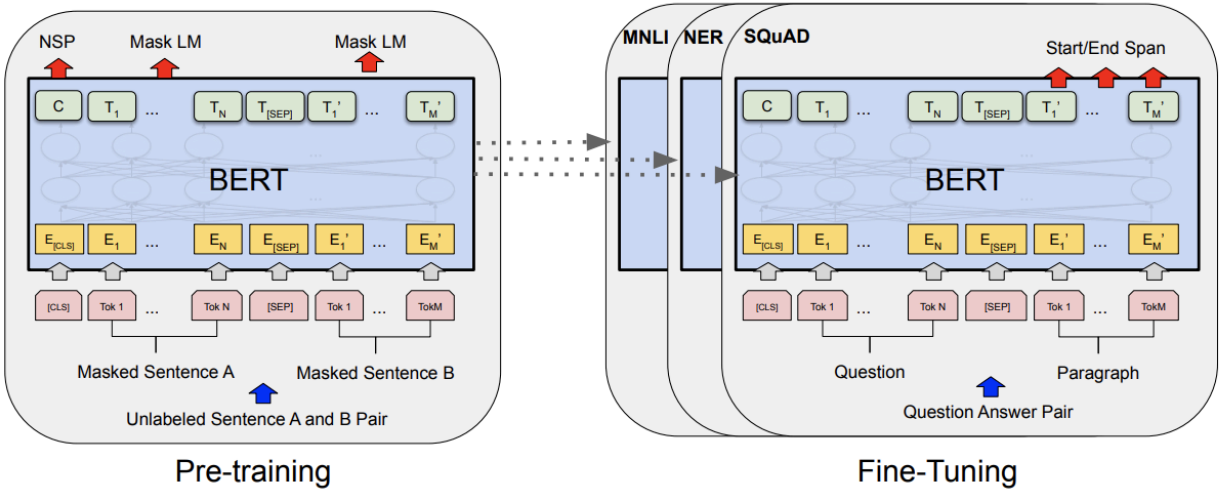
Cross entropy measures the distance between probability distributions

$$\mathrm{CE}(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=1}^{V} y_i \log(\hat{y}_i)$$

$$\sum \left(\begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} * \log \left(\begin{bmatrix} 0.1 & 0.0 & 0.1 & 0.3 & 0.2 & 0.1 & 0.1 & 0.1 \end{bmatrix} \right) \right)$$

BERT Series - BERT

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
- 2019-06, by **Google**
- Proceedings of the 2019 Conference of the North American of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)
- Citation counts (to 2019-10-21): 1831
- Authors: Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Touan



BERT Series - BERT

- Same as GPT, but:
  - Different **pretrain objective function**
    - MLM + NSP
  - Different **input format**
    - `[CLS] segment1 [SEP] segment2 [SEP]`
  - Different **fine-tune strategy**
    - Feed `[CLS]` into linear layer

BERT Series - BERT

- **Masked Language Model (MLM)**
  - Definition:
    - Given a text and replace part of the words in text with `[MASK]` token, what are the original words being replaced?
  - Masking strategy:
    - Mask only **15%** of input
    - For the masked tokens, **80%** will be replaced with token `[MASK]`
    - For the masked tokens, **10%** will be replaced with **random** tokens
    - For the masked tokens, **10%** will be replaced with **original** tokens
  - Data will be preprocessed (or randomly masked) only **once**
    - Each epoch train on the same masked texts

BERT Series - BERT

- Masked Language Model (MLM)
  - For example:
    - Original sentence: "National Cheng Kung University is the best university in Taiwan."
    - Masked sentence: "National Cheng **[MASK]** University is the best **[MASK]** in Taiwan."
    - What is the probability of first **[MASK]** token being "Kung" ?
      - BERT will maximize the probability of the first **[MASK]** token being "Kung"
      - Or maximize the log-likelyhood of the first **[MASK]** token being "Kung"
    - What is the probability of second **[MASK]** token being "university" ?
      - BERT will maximize the probability of the second **[MASK]** token being "university"
      - Or maximize the log-likelyhood of the second **[MASK]** token being "university"

BERT Series - BERT

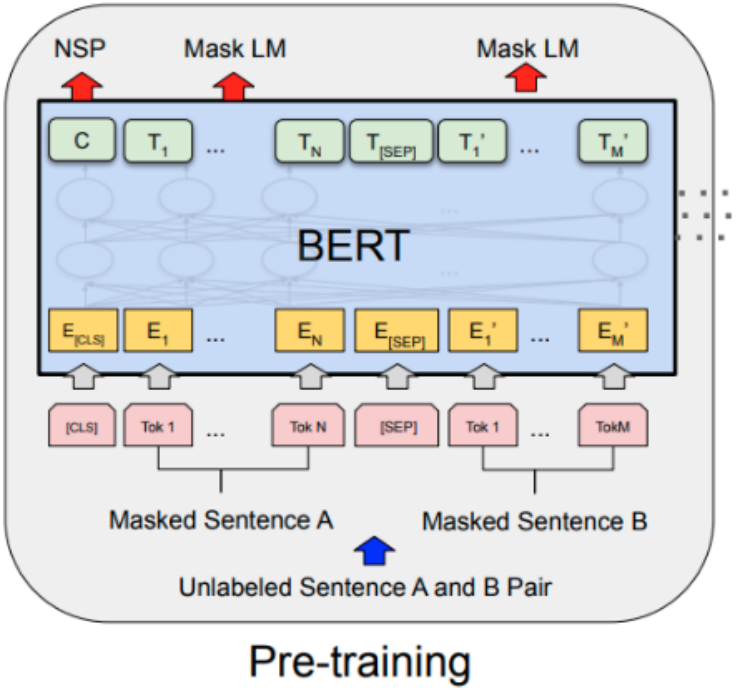
- Next Sentence Prediction (NSP)
  - Definition:
    - Given two sentences **s1**, **s2**, are the two sentences coming from same article?
    - If **s1** and **s2** are coming from **same** article, output **true**
    - If **s1** and **s2** are **not** coming from same **article**, output **false**
    - Binary classification problem
  - Data will be preprocessed (or sampled from large text corpus) only **once**
    - **50%** will sample **s1**, **s2** from **same article**, or label = **`IsNext`**
    - **50%** will sample **s1**, **s2** from **two random articles**, or label = **`NotNext`**
    - Each epoch train on the same pairs **s1**, **s2**
  - Using **`[CLS]`** token to predict
    - Feed **`[CLS]`** token into linear layer
  - Concate two sentences **s1**, **s2** with **`[SEP]`** tokens
    - Format: **`[CLS]`** **`segment1 [SEP] segment2 [SEP]`**

BERT Series - BERT

- Next Sentence Prediction (NSP)
  - Example 1:
    - Input: "**[CLS]** the man went to the store **[SEP]** he bought a gallon of milk **[SEP]**"
    - Label: **IsNext**
  - Example 2:
    - Input: "**[CLS]** the man went to the store **[SEP]** NCKU is the best **[SEP]**"
    - Label: **NotNext**

BERT Series - BERT

- Unsupervised **Pretraining**



Pre-training