UNSW SYDNEY Australia's Global University

COMP9321

Data Services Engineering

Term 1, 2019

Week 5 Lecture 2

Unsupervised Learning

COMP9321 2019T1

Supervised learning

label

$label_1$

$label_3$

$label_4$

$label_5$

model/
predictor

Supervised learning: given labeled examples

Unsupervised learning

Unupervised learning: given data, i.e. examples, but no labels

Unsupervised Learning

Definition of Unsupervised Learning:

Learning useful structure *without* labeled classes, optimization criterion, feedback signal, or any other information beyond the raw data

Unsupervised learning applications

learn clusters/groups without any label

customer segmentation (i.e. grouping)

image compression

bioinformatics: learn motifs

find important features

…

Today's Agenda

- **K-Means**
- **Apriori**

Clustering

Unsupervised Learning

Clustering

Clustering:

– Unsupervised learning

– Requires data, but no labels

– Detect patterns

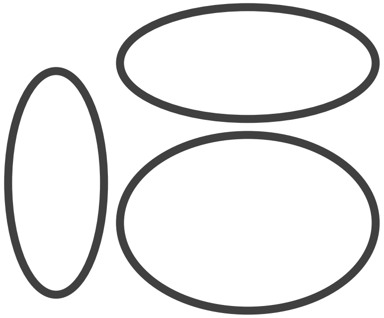
Motivations of Clustering

- exploratory data analysis
– understanding general characteristics of data
– visualizing data
- generalization – infer something about an instance (e.g. a gene) based on how it relates to other instances

Paradigms

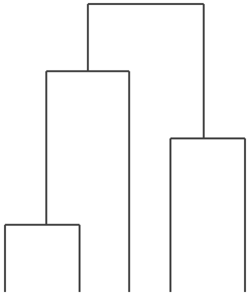
Flat algorithms

- Usually start with a random (partial) partitioning
- Refine it iteratively
  - K means clustering
  - Model based clustering
- Spectral clustering



Hierarchical algorithms

- Bottom-up, agglomerative
- Top-down, divisive



UNSW SYDNEY

Paradigms

Hard clustering: Each example belongs to exactly one cluster

Soft clustering: An example can belong to more than one cluster (probabilistic)

- Makes more sense for applications like creating browsable hierarchies
- You may want to put a pair of sneakers in two clusters: (i) sports apparel and (ii) shoes

Disjunctive (an instance can belong to multiple clusters) vs. non-disjunctive

Deterministic (same clusters produced every time for a given data set) vs. stochastic

Clustering: Image Segmentation

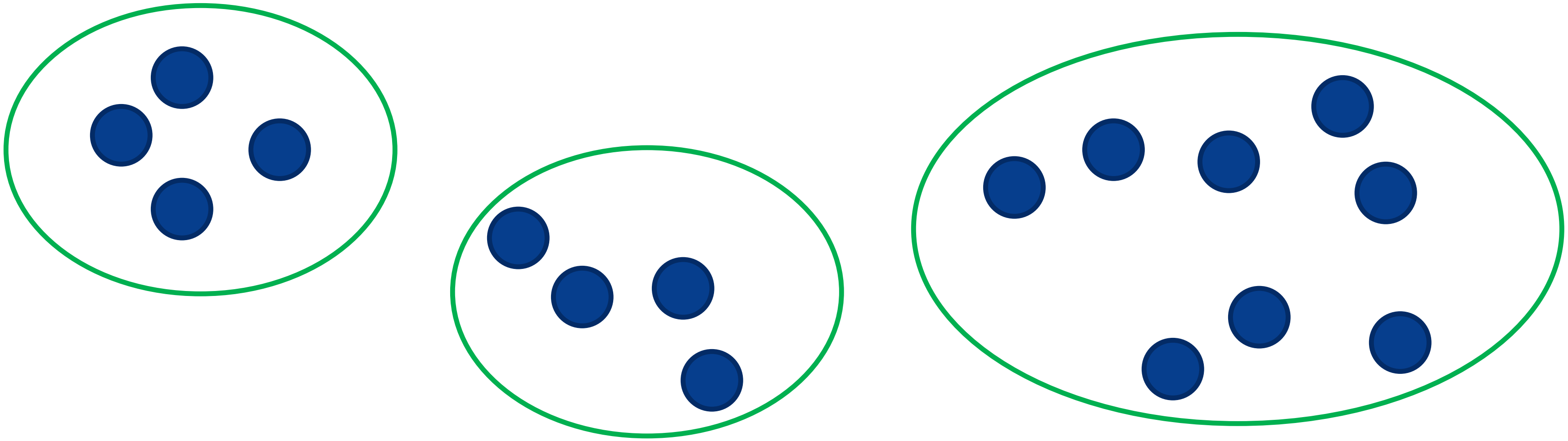
Break up the image into meaningful or perceptually similar regions

Figure from: James Hayes

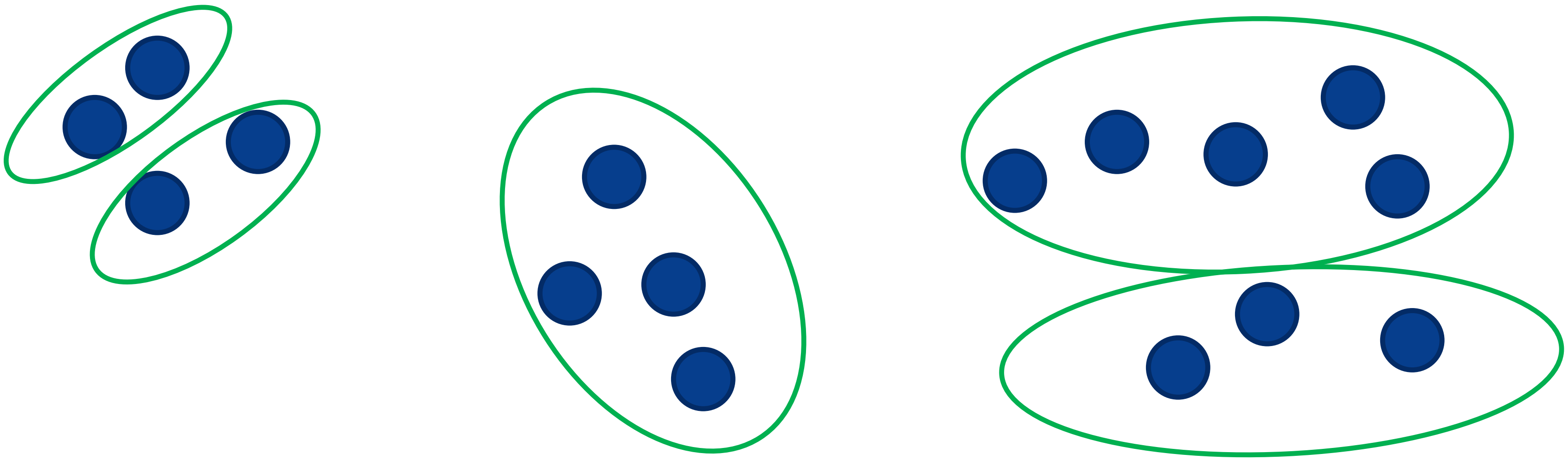
Clustering: Edge Detection

Figure from: James Hayes

Basic Idea of Clustering



Basic Idea of Clustering



Basic Idea of Clustering

Group together similar data points (instances)

- How to measure the similarity?
- ✓ What could similar mean?
- How many clusters do we need?

K-means

Most well-known and popular clustering algorithm:

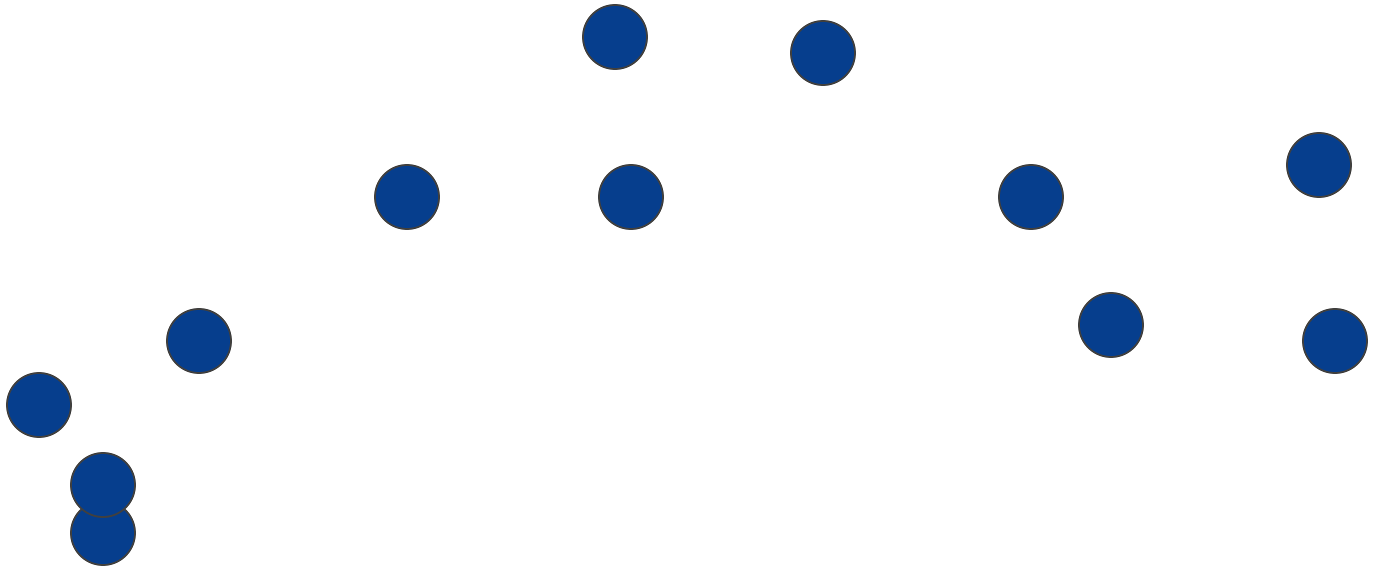
Step 1. Start with some initial cluster centers (k random points)

Step 2. Iterate:

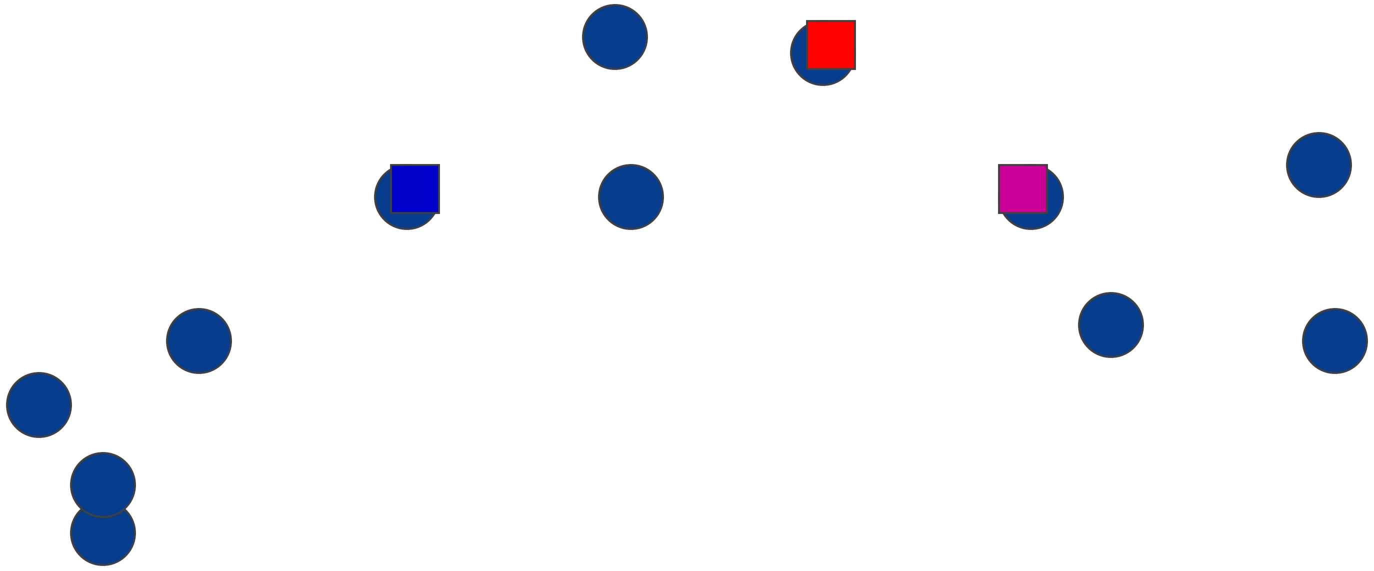
- Assign/cluster each example to closest center
- Recalculate and change centers as the mean of the points in the cluster.

Step 3. Stop when no points' assignments change

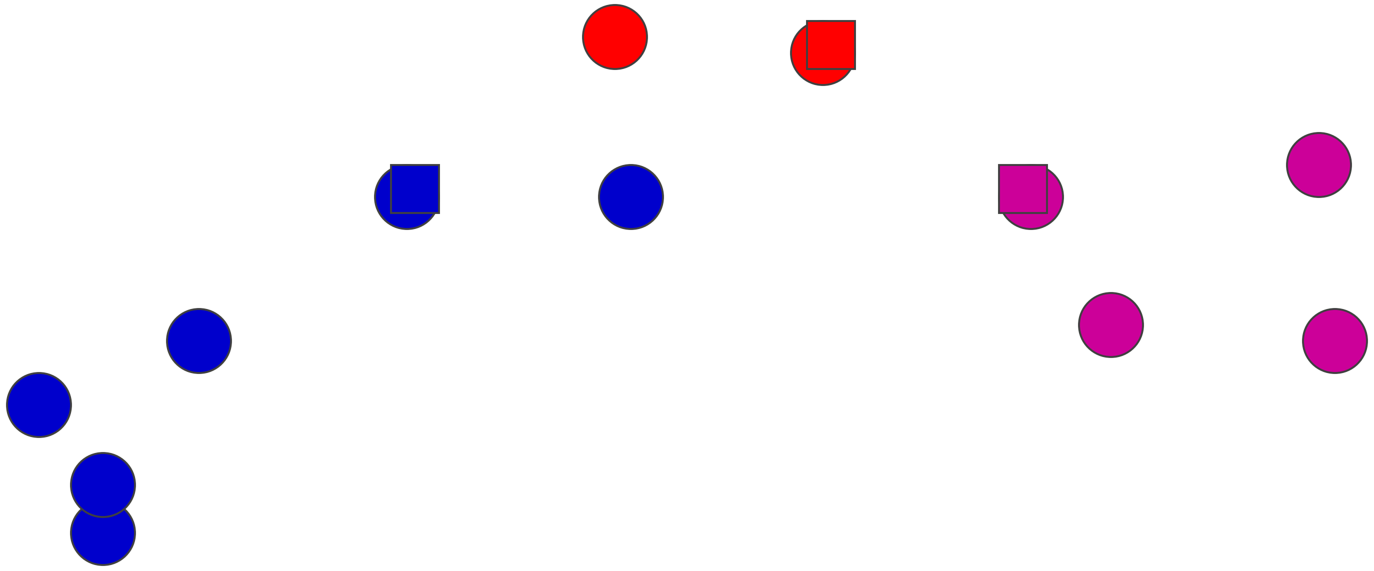
K-means: an example



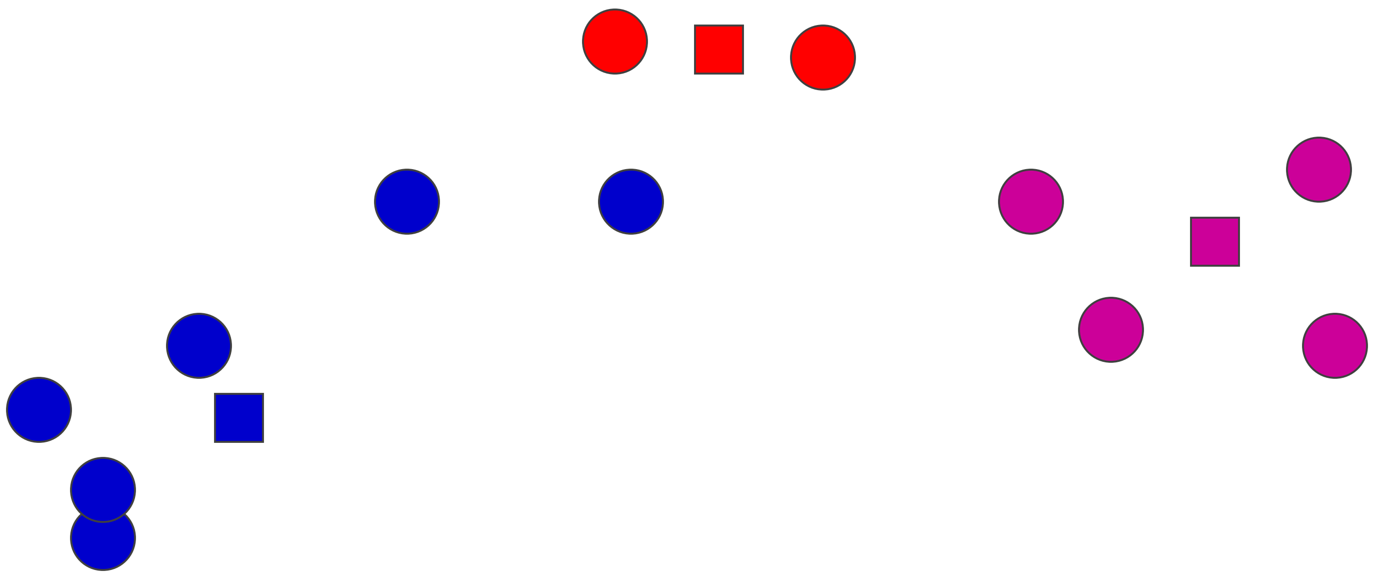
K-means: Initialize centers randomly



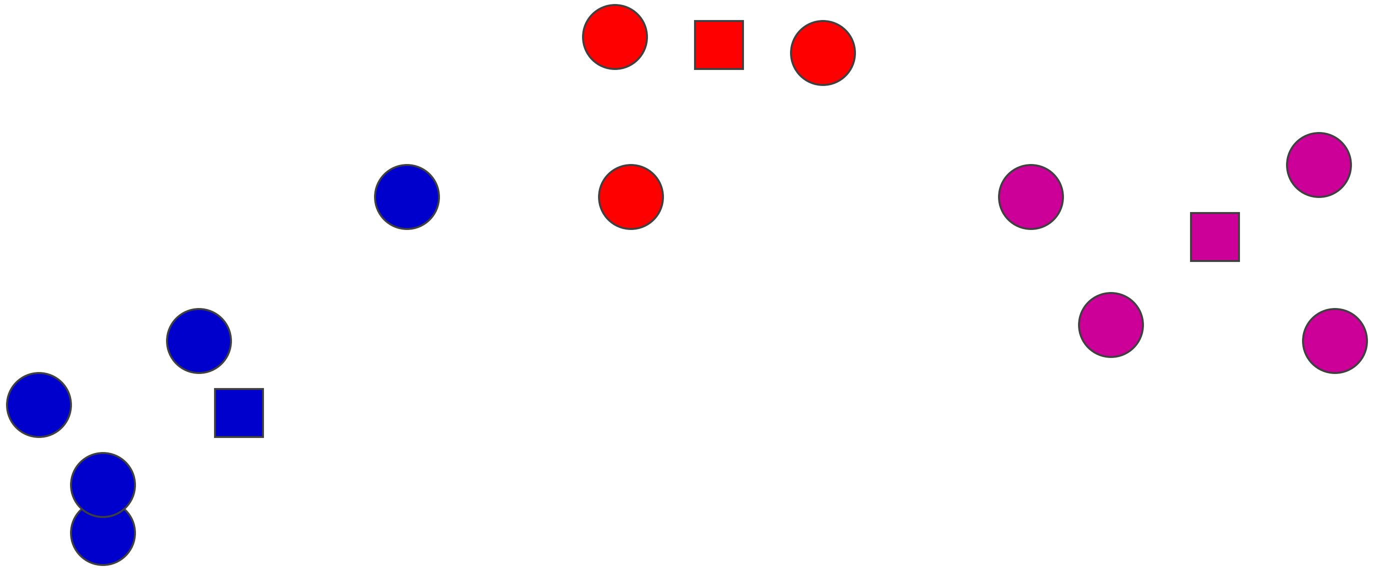
K-means: assign points to nearest center



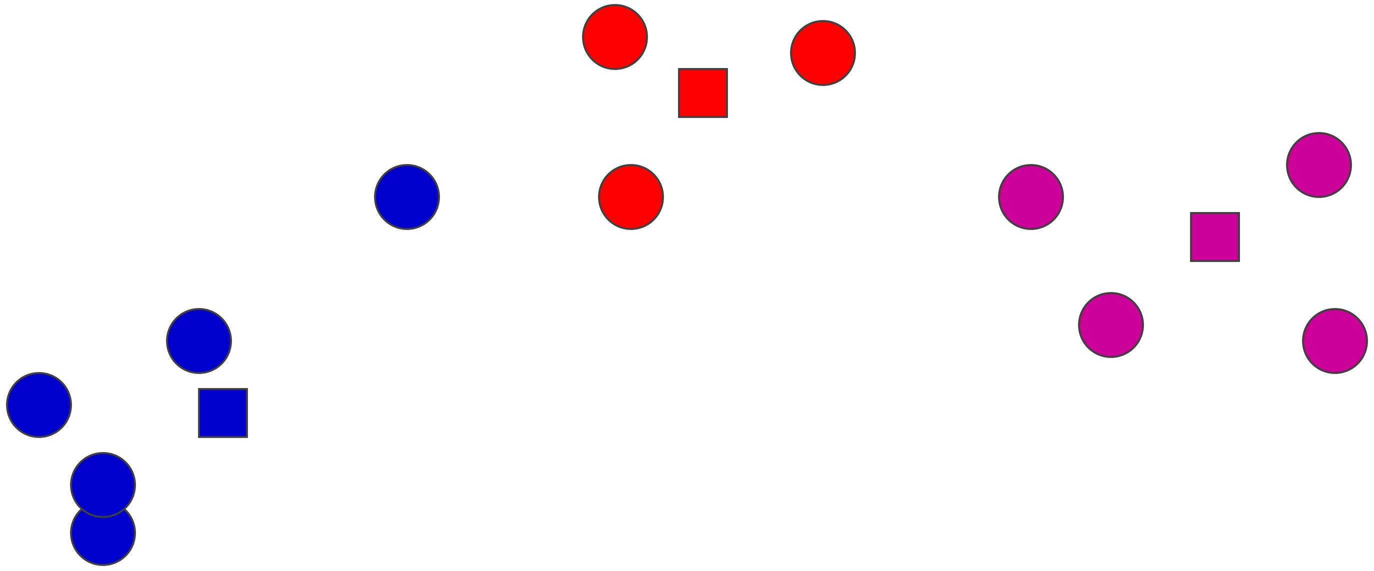
K-means: readjust centers



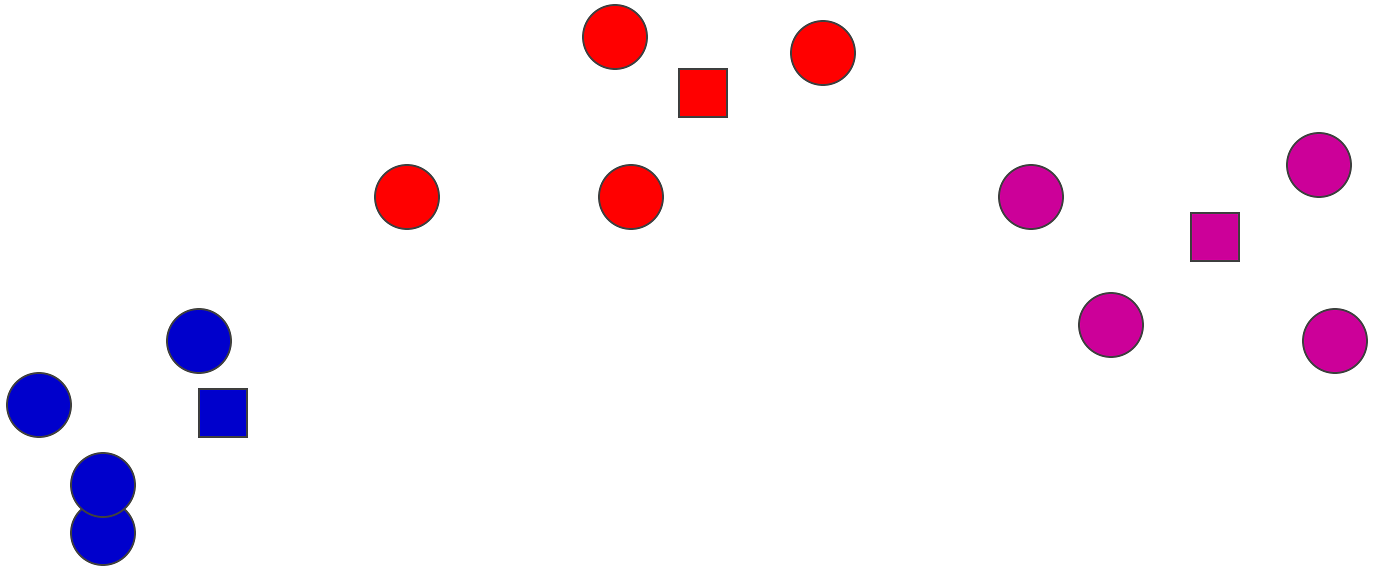
K-means: assign points to nearest center



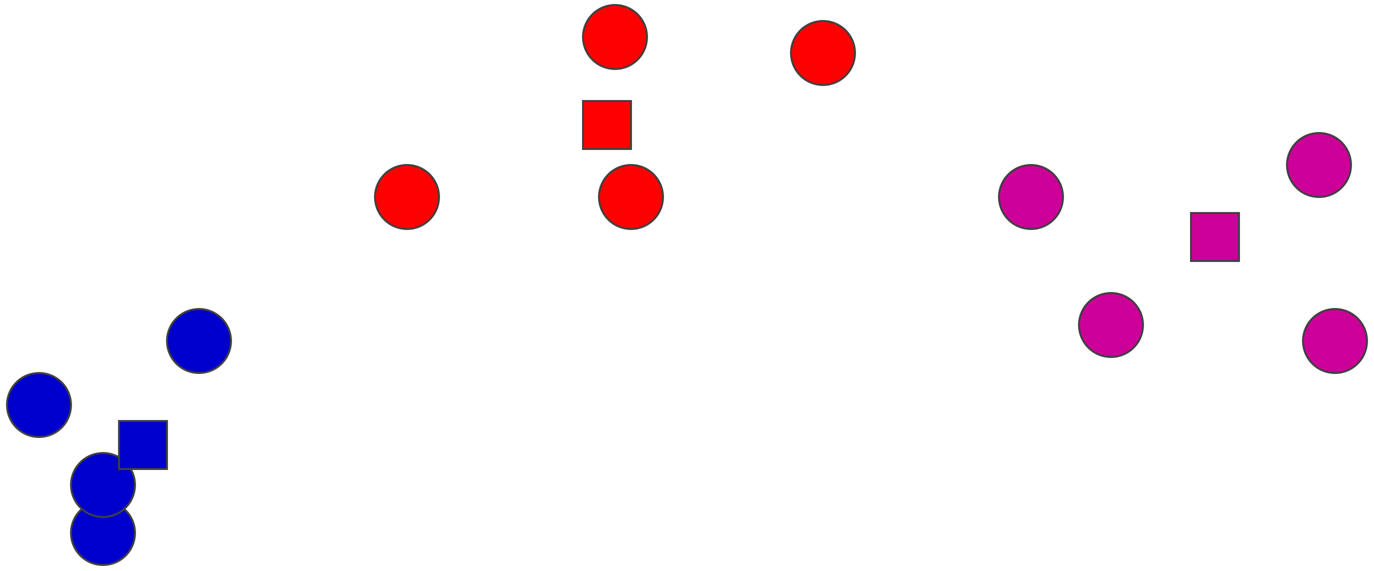
K-means: readjust centers



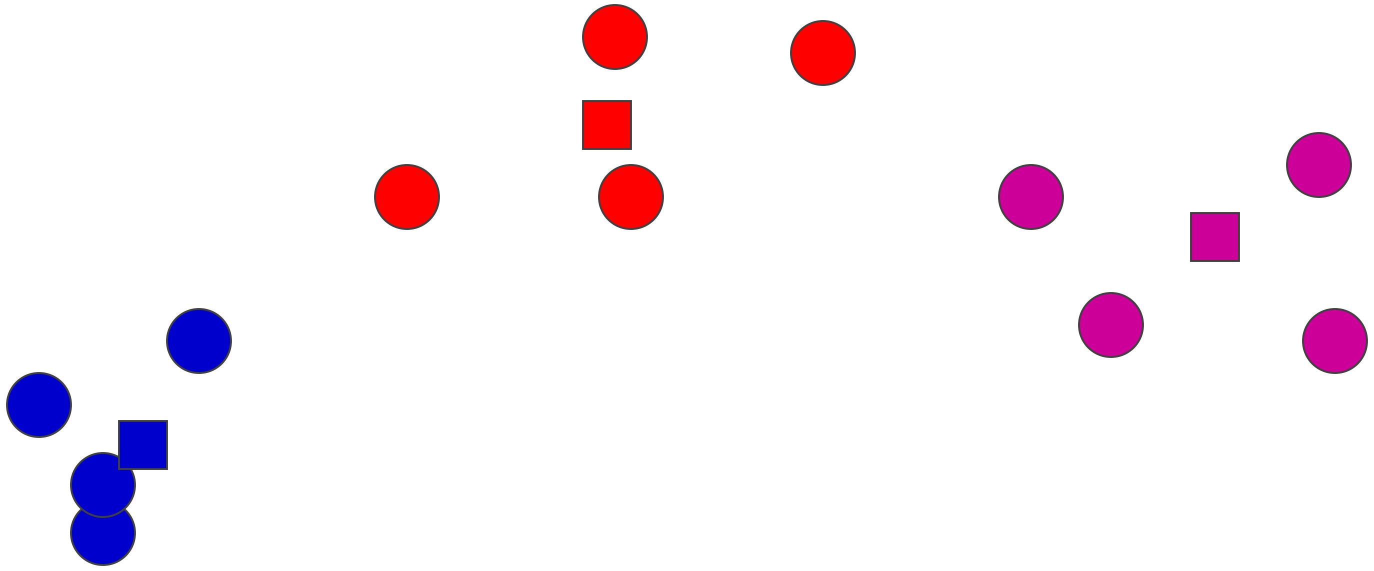
K-means: assign points to nearest center



K-means: readjust centers



K-means: assign points to nearest center

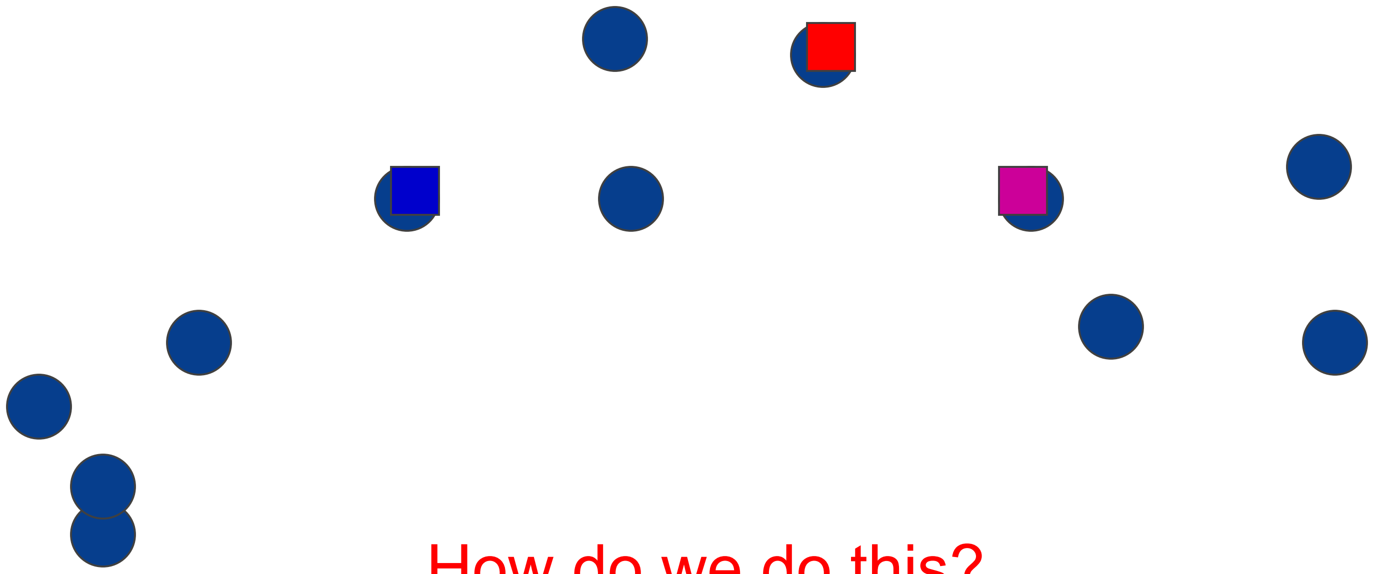


No changes: Done

K-means

Iterate:

- **Assign/cluster each example to closest center**
- Recalculate centers as the mean of the points in a cluster



How do we do this?

UNSW SYDNEY

K-means

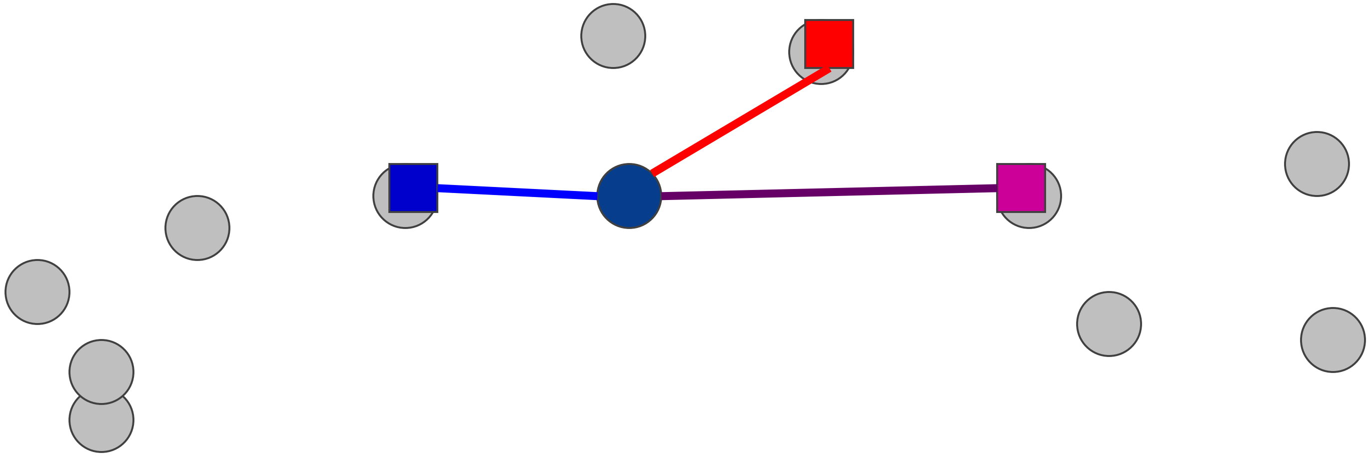
Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:

  - get distance to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



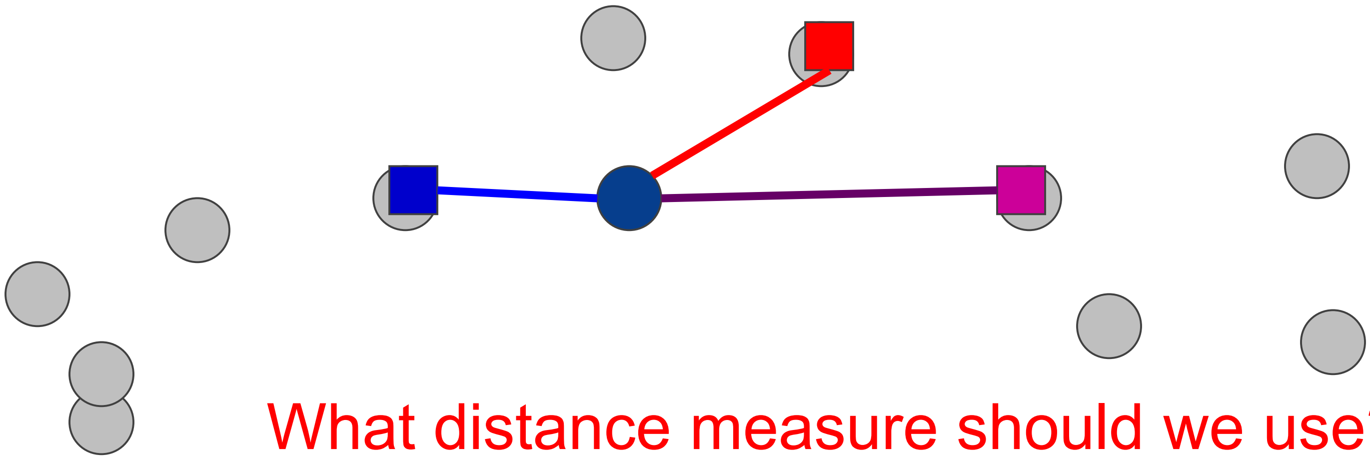
K-means

Iterate:

- **Assign/cluster each example to closest center**

  iterate over each point:
  - get **distance** to each cluster center
  - assign to closest center (hard cluster)

- Recalculate centers as the mean of the points in a cluster



What distance measure should we use?

Distance measures

Euclidean:

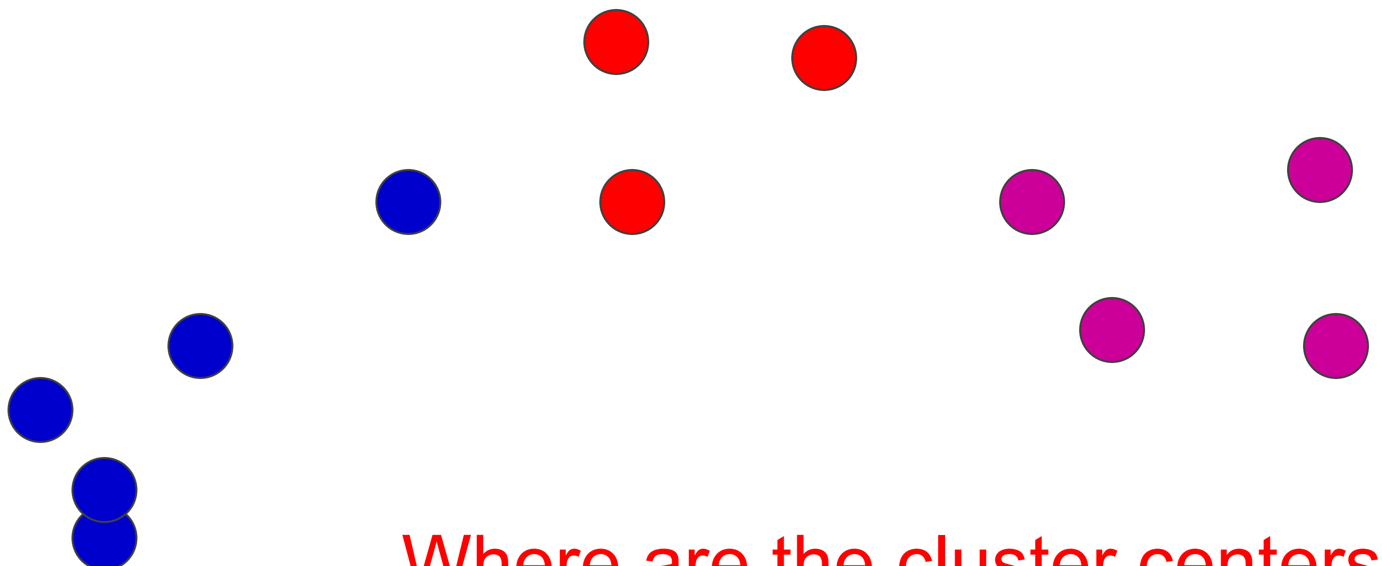
$$d(x,y)=\sqrt{\sum_{i=1}^{n}(x_i - y_i)^2}$$

good for spatial data

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

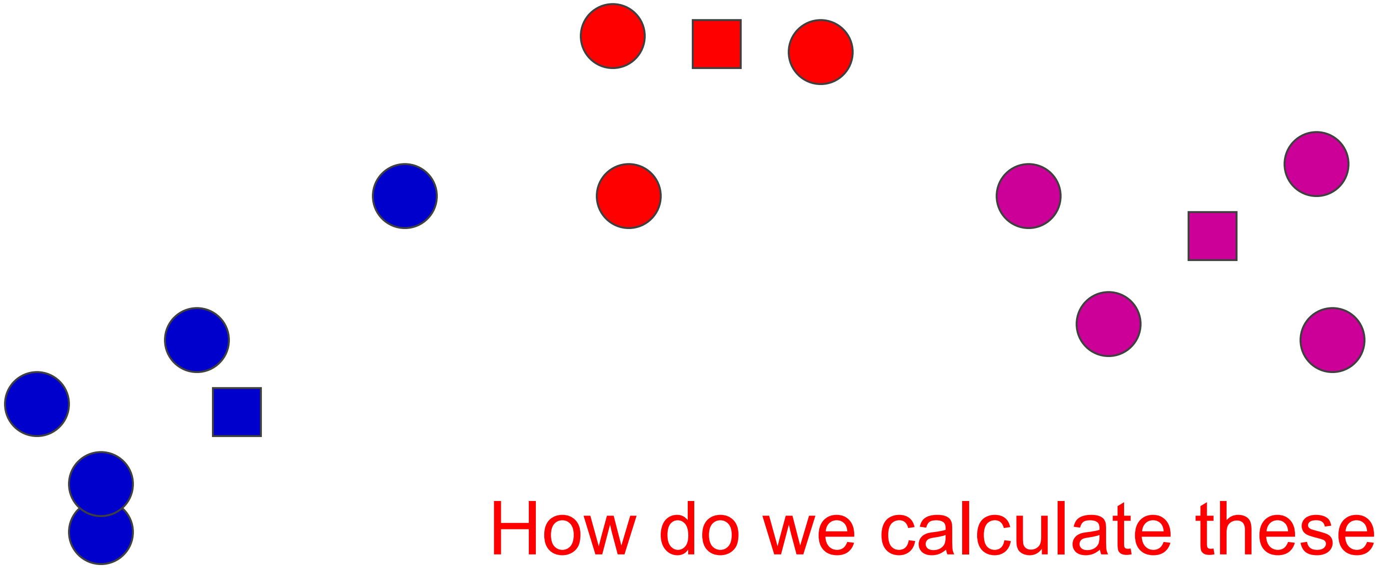


Where are the cluster centers?

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster



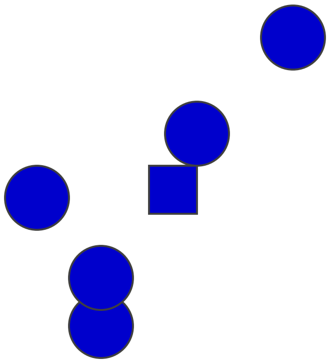
How do we calculate these?

K-means

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

e.g., for a set of instances that have been assigned to a cluster c_j, we re-compute the mean of the cluster as follow



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

K-means

$$\text{given : a set } X = \{\vec{x}_1 \ldots \vec{x}_n\} \text{ of instances}$$

$$\text{select } k \text{ initial cluster centers } \vec{f}_1 \ldots \vec{f}_k$$

$$\text{while stopping criterion not true do}$$

$$\text{for all clusters } c_j \text{ do}$$

// determine which instances are assigned to this cluster

$$c_j = \left\{ \vec{x}_i \mid \forall f_l \ \operatorname{dist}\left(\vec{x}_i, \vec{f}_j\right) < \operatorname{dist}\left(\vec{x}_i, \vec{f}_l\right) \right\}$$

$$\text{for all means } \vec{f}_j \text{ do}$$

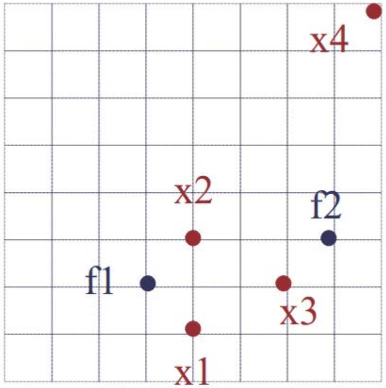
// update the cluster center

$$\vec{f}_j = \mu(c_j)$$

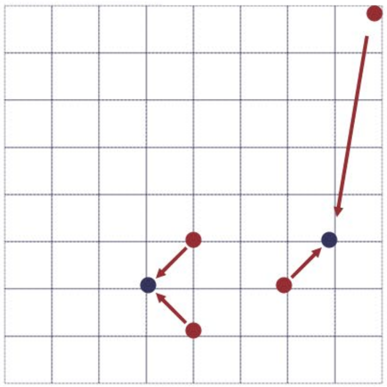
Run an example together ~~

Initialization: 4 points, 2 clusters and distance function

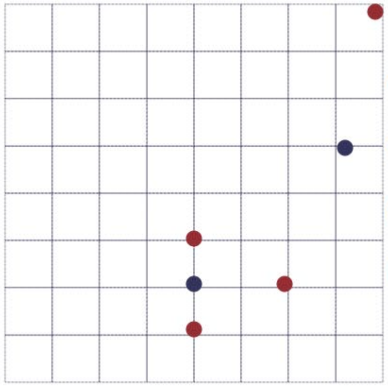
$$\text{dist}(x_i, x_j) = \sum_e \left| x_{i,e} - x_{j,e} \right|$$



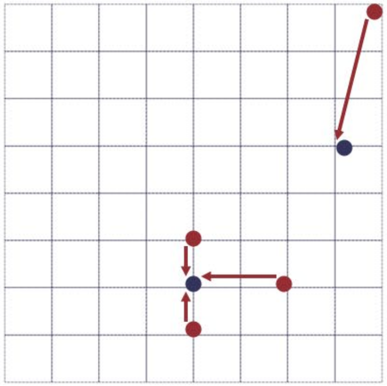
$$dist(x_1, f_1) = 2, \quad dist(x_1, f_2) = 5$$
$$dist(x_2, f_1) = 2, \quad dist(x_2, f_2) = 3$$
$$dist(x_3, f_1) = 3, \quad dist(x_3, f_2) = 2$$
$$dist(x_4, f_1) = 11, \quad dist(x_4, f_2) = 6$$



$$f_1 = \left\langle \frac{4+4}{2}, \frac{1+3}{2} \right\rangle = \langle 4,2 \rangle$$
$$f_2 = \left\langle \frac{6+8}{2}, \frac{2+8}{2} \right\rangle = \langle 7,5 \rangle$$



$$dist(x_1, f_1) = 1, \quad dist(x_1, f_2) = 7$$
$$dist(x_2, f_1) = 1, \quad dist(x_2, f_2) = 5$$
$$dist(x_3, f_1) = 2, \quad dist(x_3, f_2) = 4$$
$$dist(x_4, f_1) = 10, \quad dist(x_4, f_2) = 4$$



$$f_1 = \left\langle \frac{4+4+6}{3}, \frac{1+3+2}{3} \right\rangle = \langle 4.67,2 \rangle$$
$$f_2 = \left\langle \frac{8}{1}, \frac{8}{1} \right\rangle = \langle 8,8 \rangle$$

K-means loss function

K-means tries to minimize what is called the “k-means” loss function:

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

that is, the sum of the squared distances from each point to the associated cluster center

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Does each step of k-means move towards reducing this loss function (or at least not increasing)?

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

This isn't quite a complete proof/argument, but:

1. Any other assignment would end up in a larger loss
2. The mean of a set of values minimizes the squared error

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

Does this mean that k-means will always find the minimum loss/clustering?

Minimizing k-means loss

Iterate:

1. Assign/cluster each example to closest center
2. Recalculate centers as the mean of the points in a cluster

$$loss = \sum_{i=1}^{n} d(x_i, \mu_k)^2 \quad \text{where } \mu_k \text{ is cluster center for } x_i$$

NO! It will find *a minimum*.

Unfortunately, the k-means loss function is generally not convex and for most problems has many, many minima

We're only guaranteed to find one of them

Properties of K-means

Guaranteed to converge in a finite number of iterations

Running time per iteration

1. Assign data points to closest cluster center O(KN) time
2. Change the cluster center to the average of its assigned points O(N)

K-means variations/parameters

Start with some initial cluster centers

Iterate:

- Assign/cluster each example to closest center
- Recalculate centers as the mean of the points in a cluster

What are some other variations/parameters we haven't specified?

K-means variations/parameters

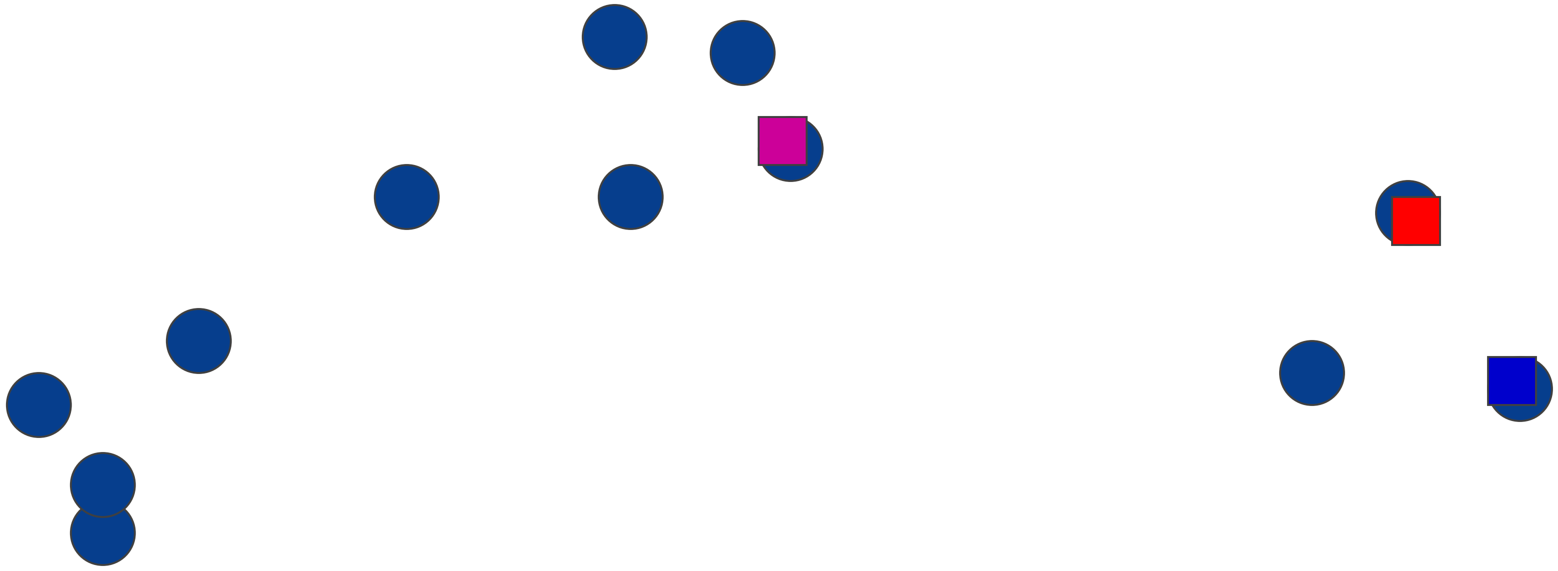
Initial (seed) cluster centers

Convergence

- A fixed number of iterations
- partitions unchanged
- Cluster centers don’t change

K!

K-means: Initialize centers randomly



What would happen here?

Seed selection ideas?

Seed choice

Results can vary drastically based on random seed selection

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

Common heuristics

- Random centers in the space
- Randomly pick examples
- Points least similar to any existing center (furthest centers heuristic)
- **Try out multiple starting points**
- Initialize with the results of another clustering method

Furthest centers heuristic

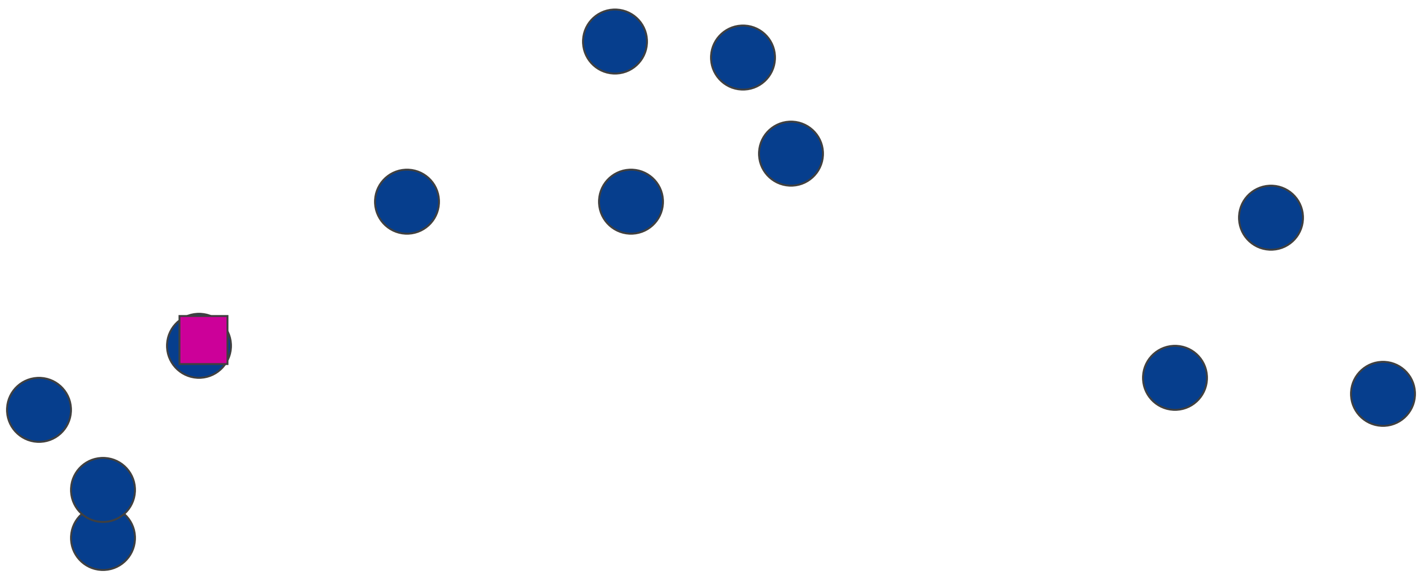
μ_1 = pick random point

for i = 2 to K:

μ_i = point that is furthest from **any** previous centers

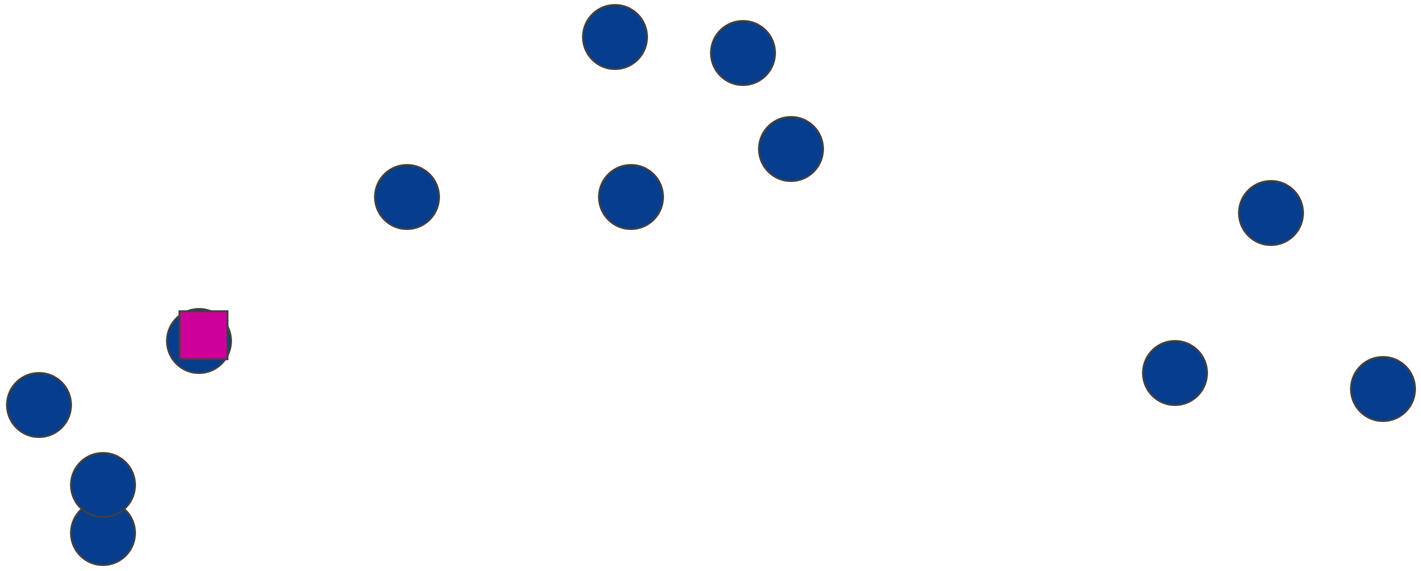
$$\mu_i = \underbrace{\underset{x}{\arg\max}}_{\text{point with the largest distance to any previous center}} \underbrace{\min_{\mu_j : 1 < j < i} d(x, \mu_j)}_{\text{smallest distance from x to any previous center}}$$

K-means: Initialize furthest from centers



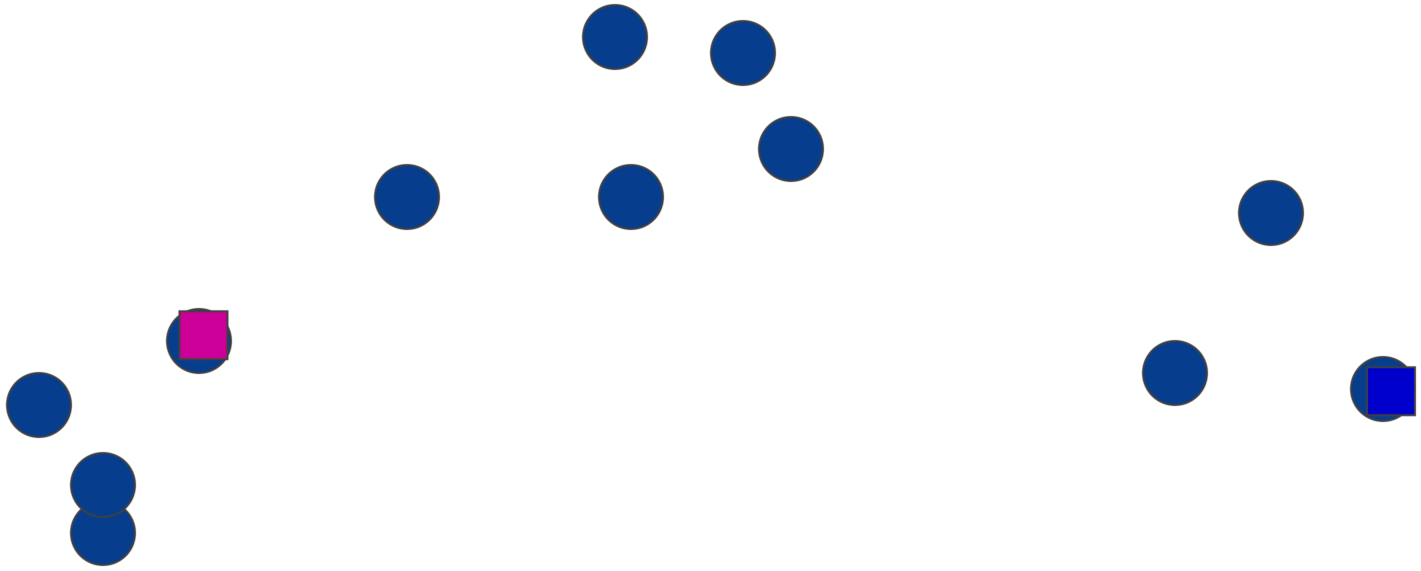
Pick a random point for the first center

K-means: Initialize furthest from centers



What point will be chosen next?

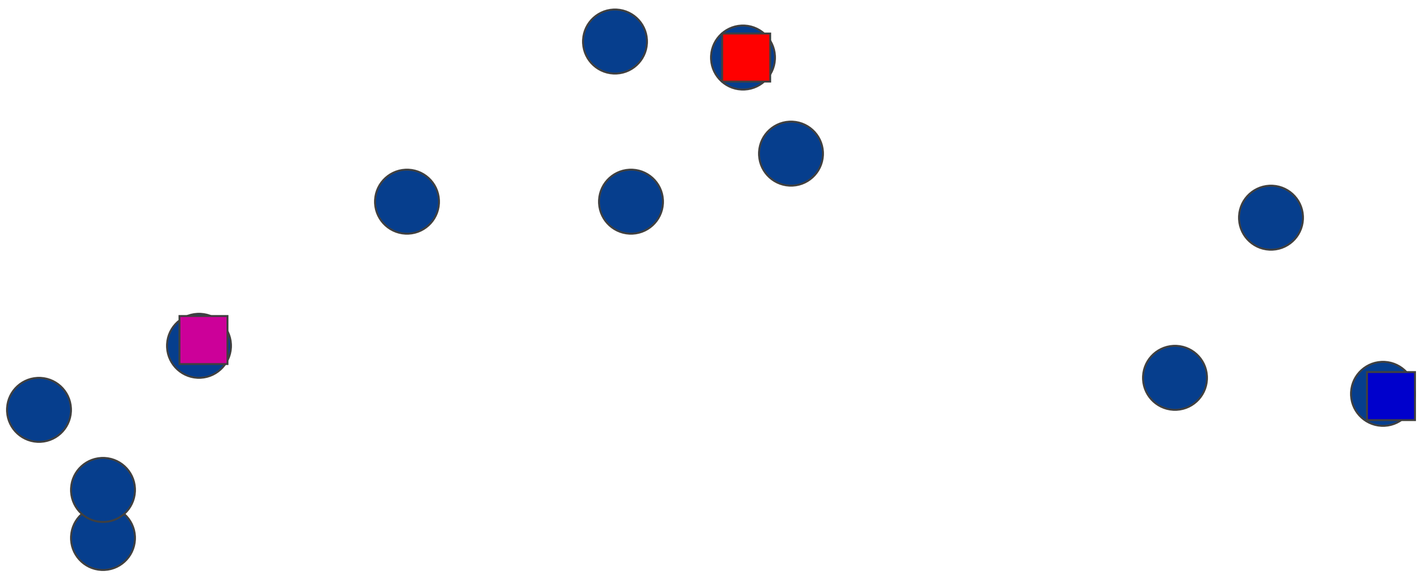
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

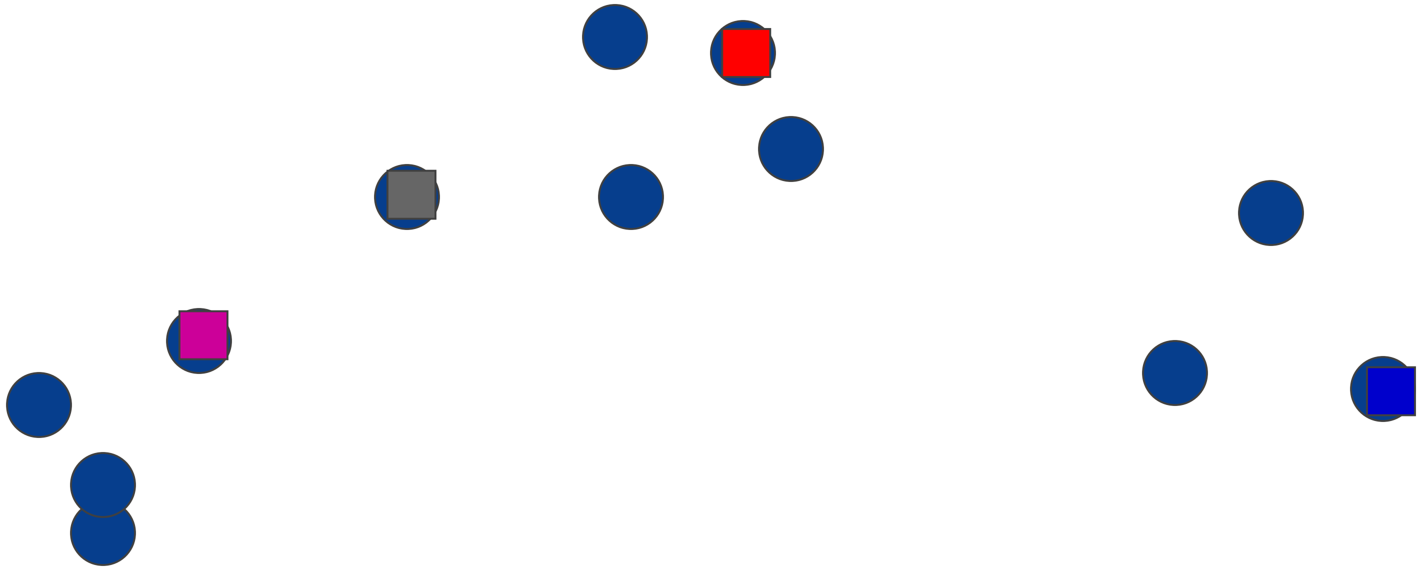
K-means: Initialize furthest from centers



Furthest point from center

What point will be chosen next?

K-means: Initialize furthest from centers

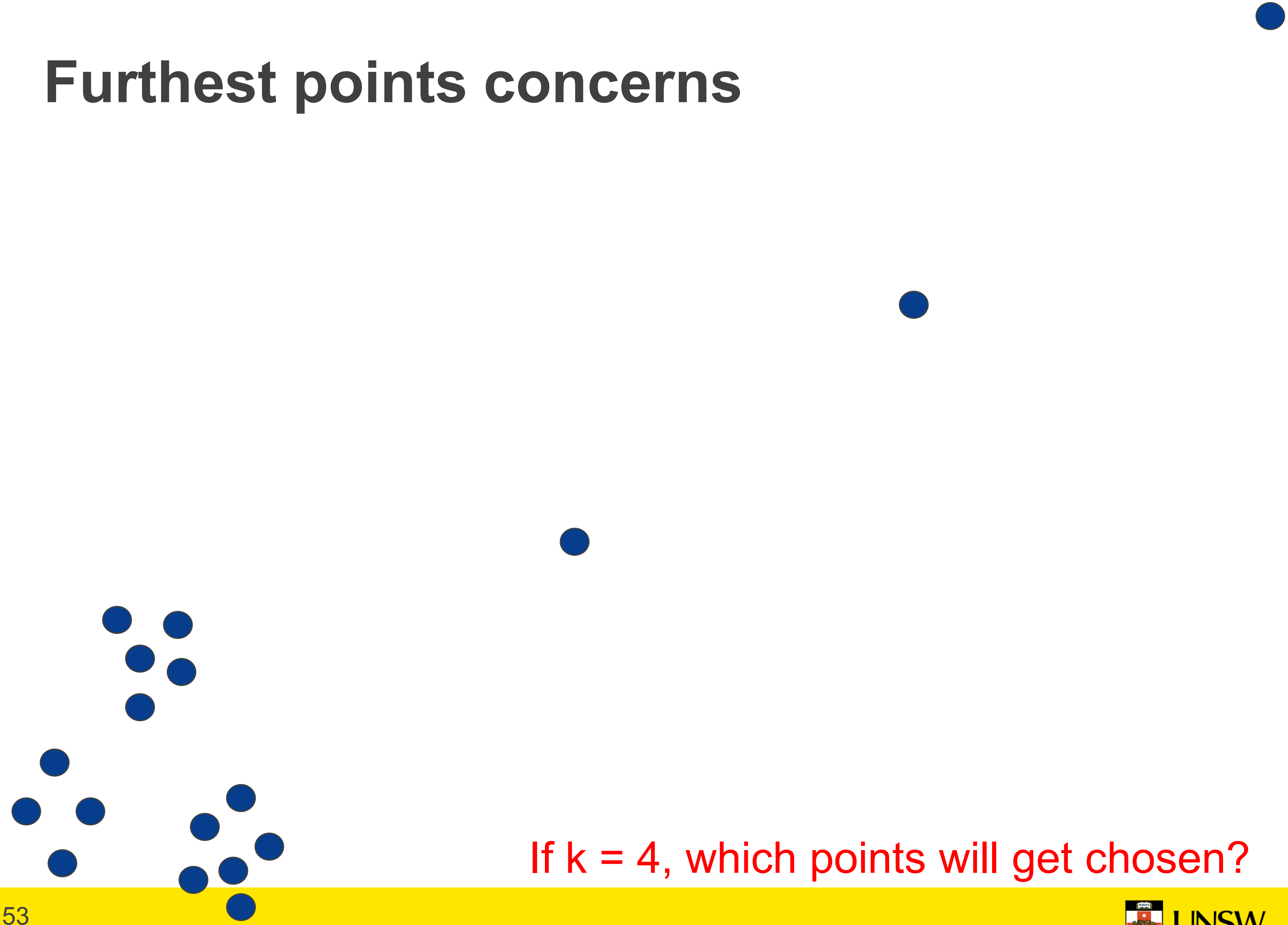


Furthest point from center

Any issues/concerns with this approach?

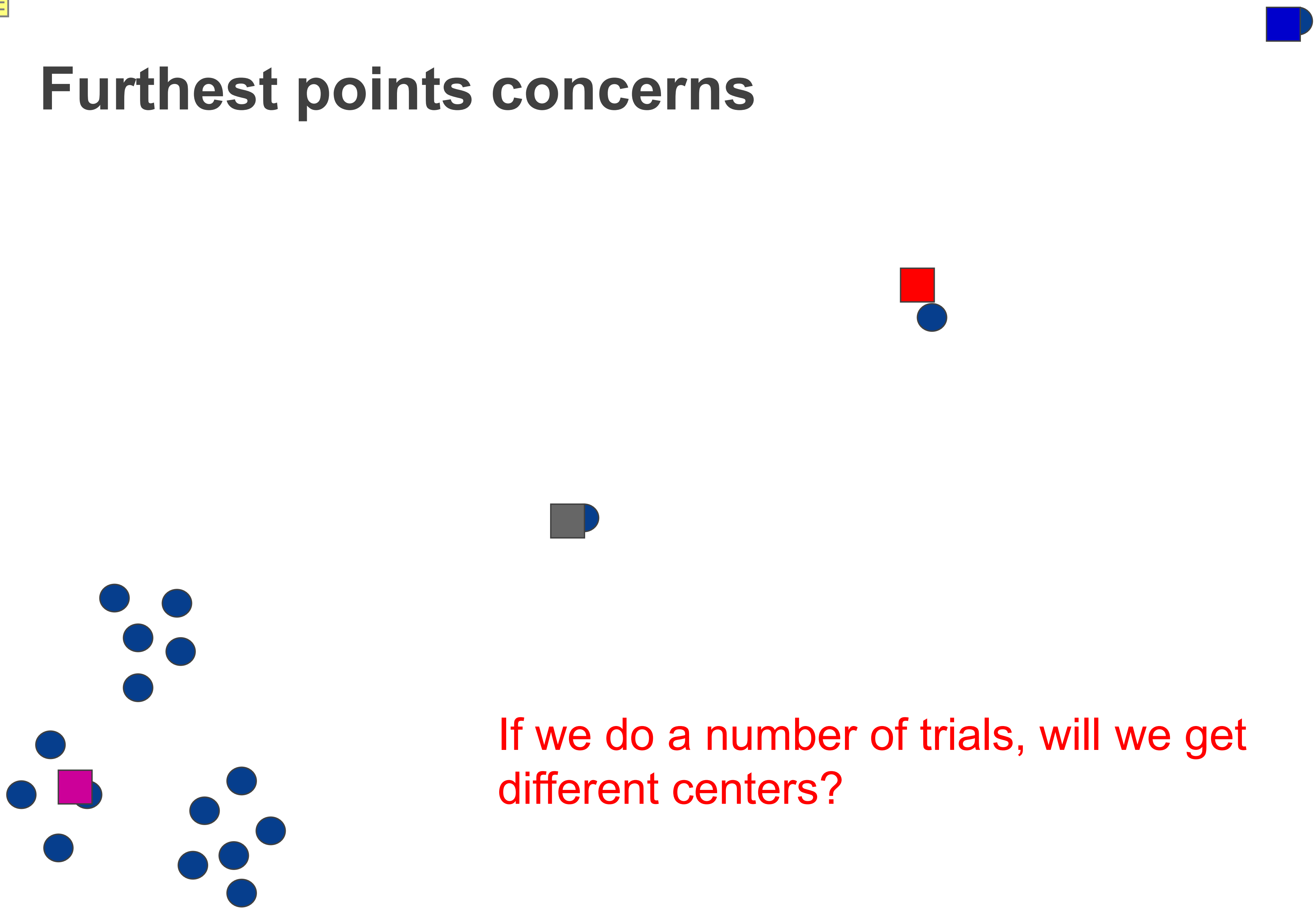
UNSW SYDNEY

Furthest points concerns



If k = 4, which points will get chosen?

Furthest points concerns



If we do a number of trials, will we get different centers?

Furthest points concerns

Doesn’t deal well with outliers

K-means++

μ_1 = pick random point

for k = 2 to **K**:

for i = 1 to **N**:

$s_i = \min d(x_i, \mu_{1\ldots k-1})$ // smallest distance to any center

μ_k = randomly pick point ***proportionate*** to ***s***

How does this help?

K-means++

μ_1 = pick random point

for k = 2 to **K**:

for i = 1 to **N**:

$s_i = \min d(x_i, \mu_{1...k-1})$ // smallest distance to any center

μ_k = randomly pick point ***proportionate*** to ***s***

- Makes it possible to select other points
  - if #points >> #outliers, we will pick good points
- Makes it non-deterministic, which will help with random runs
- Nice theoretical guarantees!

What Is A Good Clustering?

Internal criterion: A good clustering will produce high quality clusters in which:

- the intra-class (that is, intra-cluster) similarity is high
- the inter-class similarity is low
- The measured quality of a clustering depends on both the document representation and the similarity measure used

Clustering Evaluation

- Intra-cluster cohesion (compactness):

– Cohesion measures how near the data points in a cluster are to the cluster centroid.

– Sum of squared error (SSE) is a commonly used measure.

- Inter-cluster separation (isolation):

– Separation means that different cluster centroids should be far away from one another.

- In most applications, expert judgments are still the key

Association rule

Unsupervised Learning

Mining Association Rules

- What is Association rule mining
- Apriori Algorithm
- Additional Measures of rule interestingness

What Is Association Rule Mining?

- Finding frequent patterns, associations, correlations, or causal structures among sets of items in transaction databases

- Understand customer buying habits by finding associations and correlations between the different items that customers place in their “shopping basket”

- Applications
  - Basket data analysis, cross-marketing, catalog design, loss-leader analysis, web log analysis, fraud detection (supervisor->examiner)

What Is Association Rule Mining?

Rule form

Antecedent → Consequent [support, confidence]

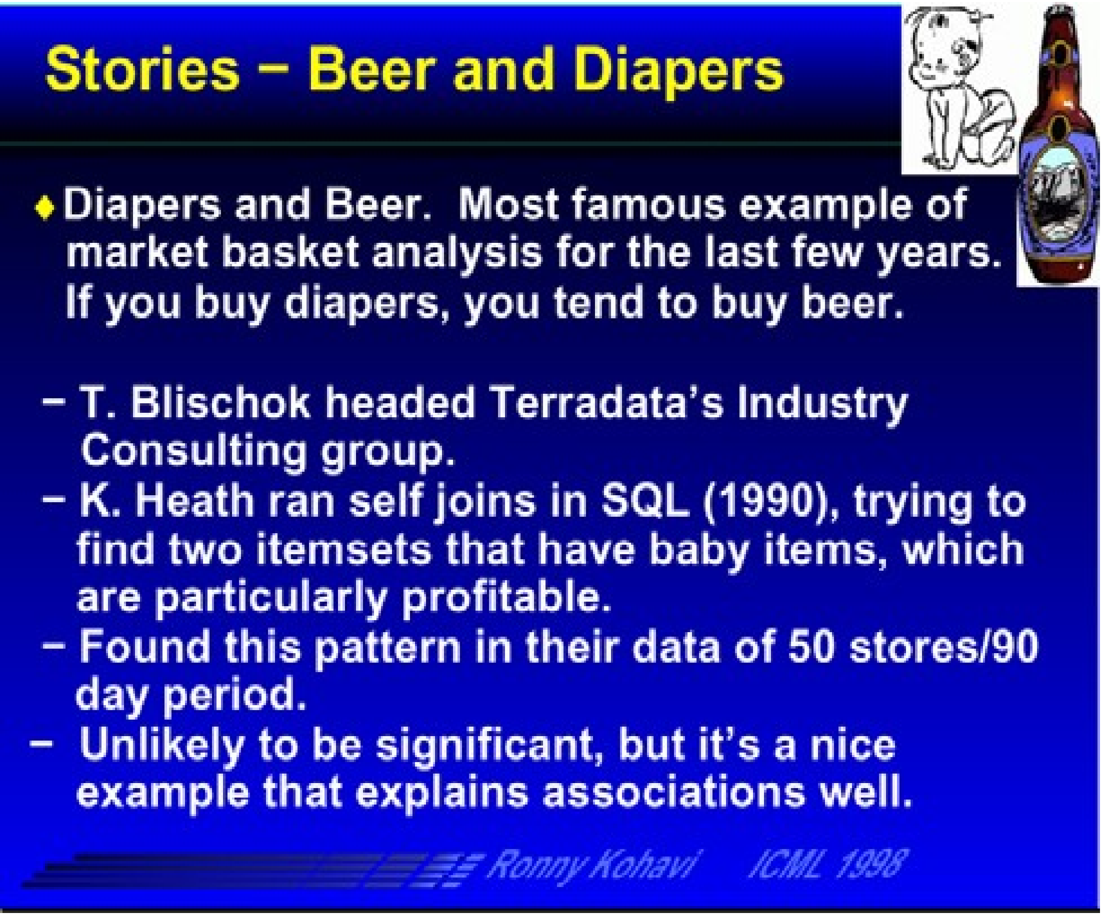
(support and confidence are user defined measures of interestingness)

Examples

buys(x, “computer”) → buys(x, “financial management software”) [0.5%, 60%]

age(x, “30..39”) ^ income(x, “42..48K”) → buys(x, “car”) [1%,75%]

How can Association Rules be used?



- Probably mom was calling dad at work to buy diapers on way home and he decided to buy a six-pack as well.
- The retailer could move diapers and beers to separate places and position high profit items of interest to young fathers along the path.

How can Association Rules be used?

Let the rule discovered be

$\{\text{Bagels,...}\} \rightarrow \{\text{Potato Chips}\}$

Potato chips as consequent → Can be used to determine what should be done to boost its sales

Bagels in the antecedent → Can be used to see which products would be affected if the store discontinues selling bagels

Bagels in antecedent and Potato chips in the consequent → Can be used to see what products should be sold with Bagels to promote sale of Potato Chips

Association Rule: Basic Concepts

Given:

- (1) database of *transactions*,
- (2) each transaction is a *list of items* purchased by a customer in a visit

Find:

- all rules that correlate the presence of one set of items (itemset) with that of another set of items
- E.g., 98% of people who purchase tires and auto accessories also get automotive services done

Rule basic Measures: Support and Confidence

$A \rightarrow B$ [s, c]

Support: denotes the frequency of the rule within transactions. A high value means that the rule involve a great part of database.

$$\text{support}(A \rightarrow B\ [\,s, c\,]) = p(A \rightarrow B)$$

Confidence: denotes the percentage of transactions containing A which contain also B. It is an estimation of conditioned probability .

$$\text{confidence}(A \rightarrow B\ [\,s, c\,]) = p(B|A) = \text{sup}(A,B)/\text{sup}(A).$$

Example

Trans. Id	Purchased Items
1	A,D
2	A,C
3	A,B,C
4	B,E,F

Itemset:

A,B or B,E,F

Support of an itemset:

Sup(A,B)=1 Sup(A,C)=2

Frequent pattern:

Given min. sup=2,

{A,C} is a frequent pattern

9

For minimum support = 50% and minimum confidence = 50%, we have the following rules

$A \rightarrow C$ with 50% support and 66% confidence

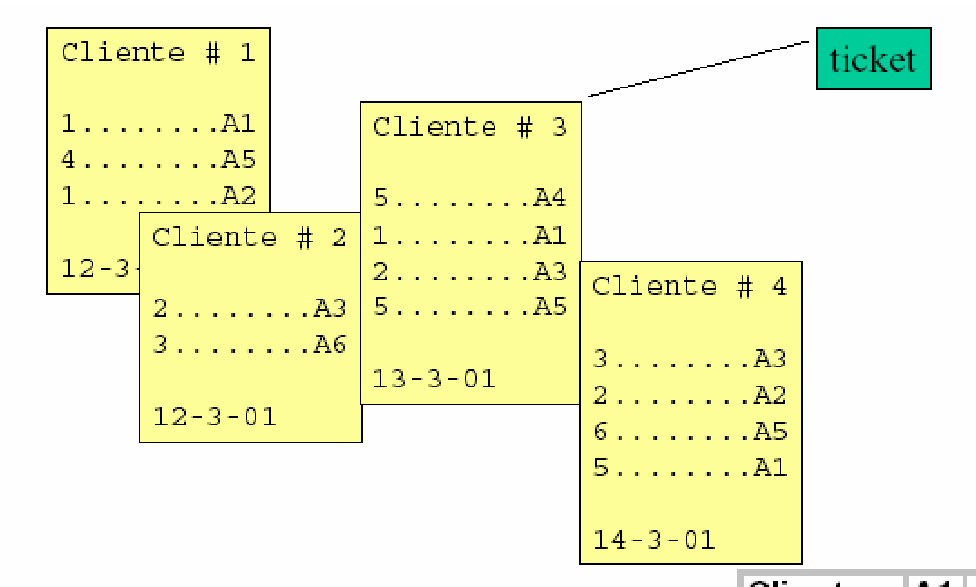
$C \rightarrow A$ with 50% support and 100% confidence

Mining Association Rules

- What is Association rule mining

- Apriori Algorithm

Boolean association rules



Each transaction is represented by a Boolean vector

Cliente	A1	A2	A3	A4	A5	A6	A7	A8	A9	A10	A11	A12	A13
1	1	1	0	0	1	0	0	0	0	0	0	0	0
2	0	0	1	0	0	1	0	0	0	0	0	0	0
3	1	0	1	1	1	0	0	0	0	0	0	0	0
4	1	1	1	0	1	0	0	0	0	0	0	0	0
5	0	0	1	0	0	1	0	1	1	1	0	0	0
6	0	1	0	0	0	0	0	1	0	1	0	0	0
7	1	0	0	0	0	0	1	1	0	1	0	1	1
8	0	1	0	0	0	0	0	1	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	1	0	1	0

Mining Association Rules - An Example

Transaction ID	Items Bought
2000	A,B,C
1000	A,C
4000	A,D
5000	B,E,F

Min. support 50%
Min. confidence 50%

Frequent Itemset	Support
{A}	75%
{B}	50%
{C}	50%
{A,C}	50%

For rule $A \rightarrow C$:

support = support({A , C }) = 50%

confidence = support({A , C }) / support({A }) = 66.6%

The Apriori principle

Any subset of a frequent itemset must be frequent

A transaction containing {beer, diaper, nuts} also contains {beer, diaper}

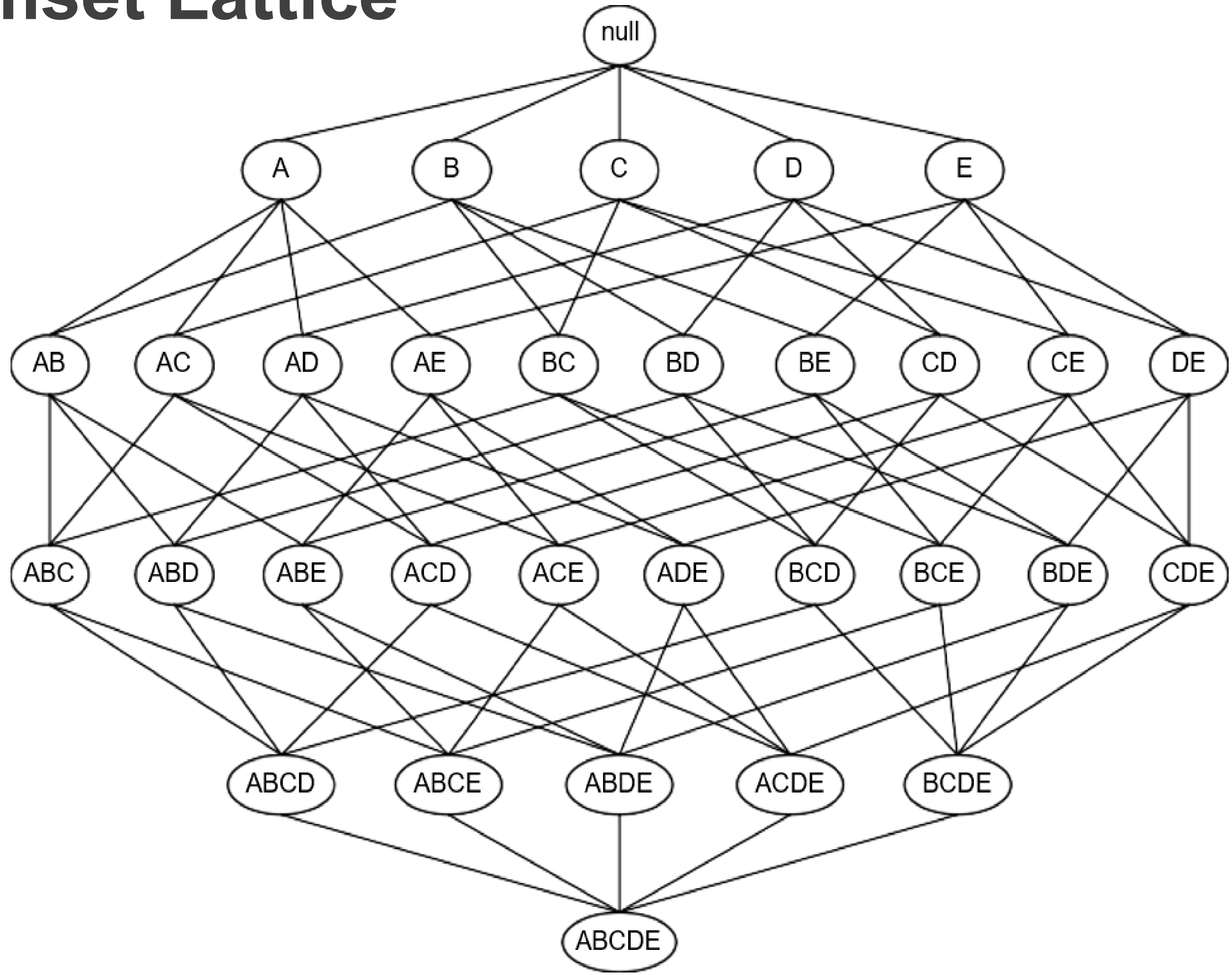
{beer, diaper, nuts} is frequent
➔ {beer, diaper} must also be frequent

The Apriori principle

No superset of any infrequent itemset should be generated or tested

Many item combinations can be pruned

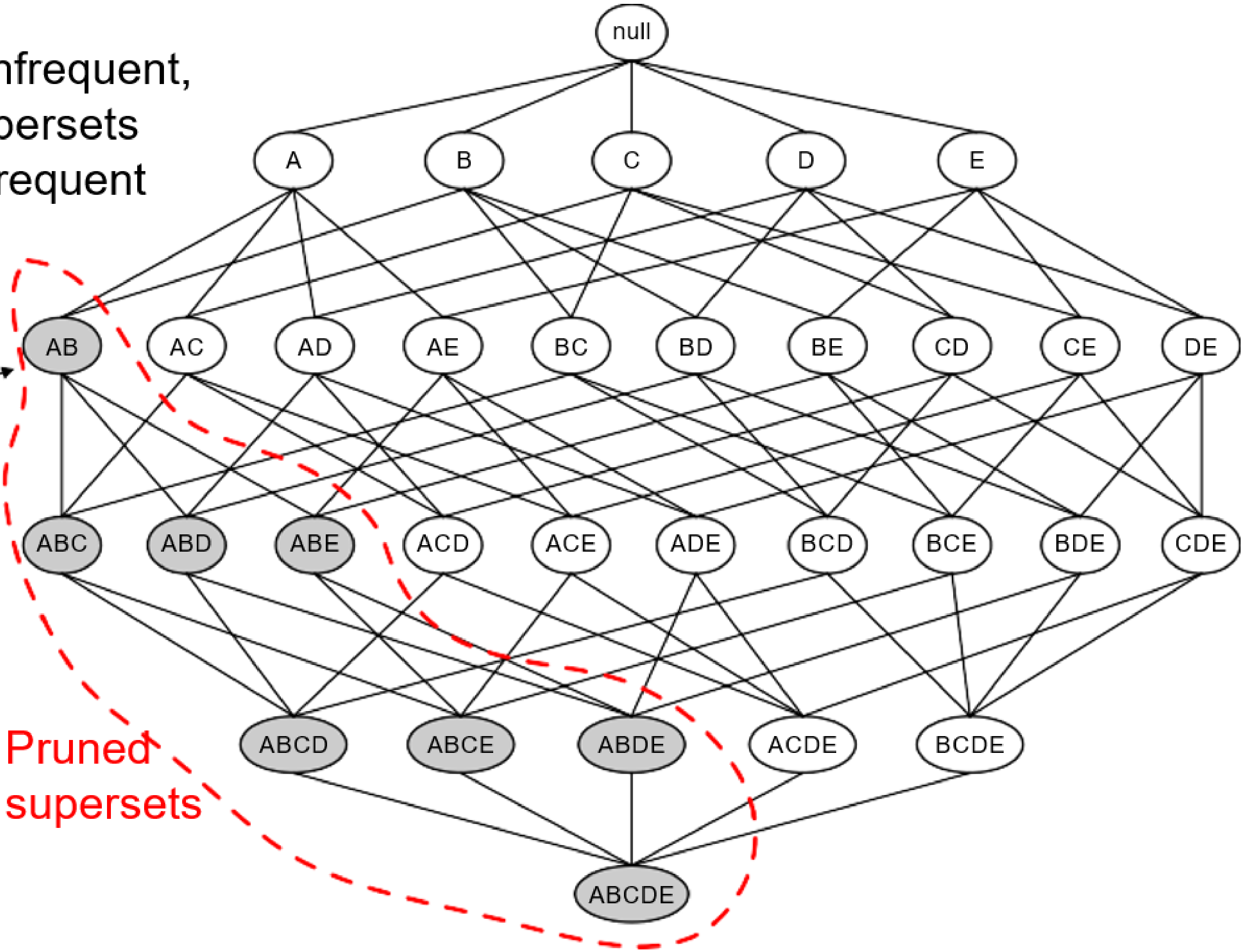
Itemset Lattice



Apriori principle for pruning candidates

If an itemset is infrequent, then all of its supersets must also be infrequent

Found to be Infrequent



Pruned supersets

Mining Frequent Itemsets (the Key Step)

- Find the *frequent itemsets:* the sets of items that have minimum support
  - A subset of a frequent itemset must also be a frequent itemset
    - Generate length (k+1) candidate itemsets from length k frequent itemsets, and
    - Test the candidates against DB to determine which are in fact frequent
- Use the frequent itemsets to generate association rules.
  - Generation is straightforward

The Apriori Algorithm — Example

Min. support: 2 transactions

Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

Scan D

C_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{4}	1
{5}	3

L_1

itemset	sup.
{1}	2
{2}	3
{3}	3
{5}	3

C_2

itemset
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

Scan D

C_2

itemset	sup
{1 2}	1
{1 3}	2
{1 5}	1
{2 3}	2
{2 5}	3
{3 5}	2

L_2

itemset	sup
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

itemset
{2 3 5}

Scan D

L_3

itemset	sup
{2 3 5}	2

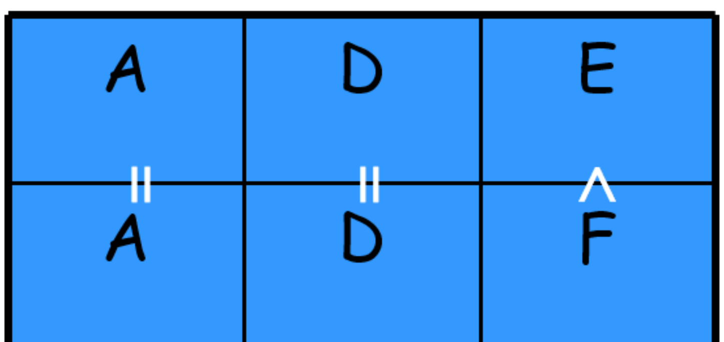
How to Generate Candidates?

The items in L_{k-1} are <u>listed in an order</u>

<u>Step 1</u>: self-joining L_{k-1}

- insert into $\boldsymbol{C_k}$
- select $p.item_1$, $p.item_2$, …, $p.item_{k-1}$, $q.item_{k-1}$
- from L_{k-1} p, L_{k-1} q
- where $p.item_1=q.item_1$, …, $p.item_{k-2}=q.item_{k-2}$,

  $p.item_{k-1} < q.item_{k-1}$

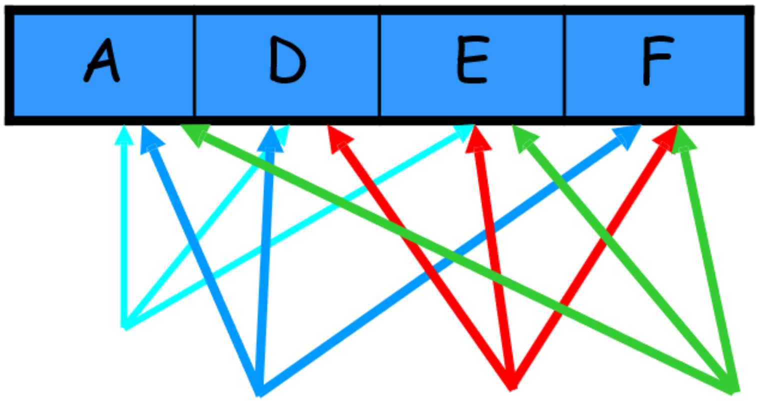


Step 2: pruning

for all *itemsets c in* C_k do

for all *(k-1)-subsets s of c* do

if *(s is not in* L_{k-1}*)* then delete *c* from C_k



Example of Generating Candidates

L_3={*abc, abd, acd, ace, bcd*}

Self-joining: L_3*L_3

abcd from *abc* and *abd*

acde from *acd* and *ace*

Pruning *(before counting its support)*:

acde is removed because *ade* is not in L_3

C_4={*abcd*}

The Apriori Algorithm

C_k: Candidate itemset of size k L_k : frequent itemset of size k

Join Step: C_k is generated by joining L_{k-1} with itself

Prune Step: Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset

Algorithm:

L_1 = {frequent items};
for (k = 1; L_k != $\emptyset$; k++) **do begin**
C_{k+1} = candidates generated from L_k;
for each transaction t in database do
increment the count of all candidates in C_{k+1} that are contained in t
L_{k+1} = candidates in C_{k+1} with min_support
end
return L = $\cup_k L_k$;

Generating AR from frequent intemsets

Confidence $(A \rightarrow B) = P(B|A) = \dfrac{\text{support\_count}(\{A,B\})}{\text{support\_count}(\{A\})}$

For every frequent itemset x, generate all non-empty subsets of x

For every non-empty subset s of x, output the rule "s $\rightarrow$ (x-s) " if

$$\frac{\text{support\_count}(\{x\})}{\text{support\_count}(\{s\})} \geq \text{min\_conf}$$

Generating AR from frequent intemsets

Given a frequent itemset L, find all non-empty subsets $f \subset L$ such that $f \rightarrow L - f$ satisfies the minimum confidence requirement

If {A,B,C,D} is a frequent itemset, candidate rules:

> $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$, $BCD \rightarrow A$, $A \rightarrow BCD$, $B \rightarrow ACD$, $C \rightarrow ABD$, $D \rightarrow ABC$, $AB \rightarrow CD$, $AC \rightarrow BD$, $AD \rightarrow BC$, $BC \rightarrow AD$, $BD \rightarrow AC$, $CD \rightarrow AB$

If $|L| = k$, then there are $2^k - 2$ candidate association rules (ignoring $L \rightarrow \varnothing$ and $\varnothing \rightarrow L$)

From Frequent Itemsets to Association Rules

Q: Given frequent set {A,B,E}, what are possible association rules?

- $A \to B, E$
- $A, B \to E$
- $A, E \to B$
- $B \to A, E$
- $B, E \to A$
- $E \to A, B$
- $\_\_\_ \to A,B,E$ (empty rule), or true $\to A,B,E$

Generating Rules: example

Trans-ID	Items
1	ACD
2	BCE
3	ABCE
4	BE
5	ABCE

Min_support: 60%
Min_confidence: 75%

Frequent Itemset	Support
{**BCE**},{AC}	60%
{BC},{CE},{A}	60%
{BE},{B},{C},{E}	80%

Rule	Conf.
{BC} =>{E}	100%
{BE} =>{C}	75%
{CE} =>{B}	100%
{B} =>{CE}	75%
{C} =>{BE}	75%
{E} =>{BC}	75%

Database TDB

Tid	Items
10	A, C, D
20	B, C, E
30	A, B, C, E
40	B, E

1^{st} scan

C_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{D}	1
{E}	3

L_1

Itemset	sup
{A}	2
{B}	3
{C}	3
{E}	3

C_2

Itemset
{A, B}
{A, C}
{A, E}
{B, C}
{B, E}
{C, E}

2^{nd} scan

C_2

Itemset	sup
{A, B}	1
{A, C}	2
{A, E}	1
{B, C}	2
{B, E}	3
{C, E}	2

L_2

Itemset	sup
{A, C}	2
{B, C}	2
{B, E}	3
{C, E}	2

C_3

Itemset
{B, C, E}

3^{rd} scan

L_3

Itemset	sup
{B, C, E}	2

Challenges of Frequent Pattern Mining

- Challenges
  - Multiple scans of transaction database
  - Huge number of candidates
  - Tedious workload of support counting for candidates
- Improving Apriori: general ideas
  - Reduce number of transaction database scans
  - Shrink number of candidates
  - Facilitate support counting of candidates