Second Review

Read also the midterm review (First Review).

The final exam covers both.

This is only a subset of slides.

The exam covers all material posted; not only these slides.

Support Vector Machines

Linear classifier

$$h(\mathbf{x}) = sign(\mathbf{w} \cdot \mathbf{x} + b)$$

This time will write b explicitly.

Which outputs +1 or –1.

Say:

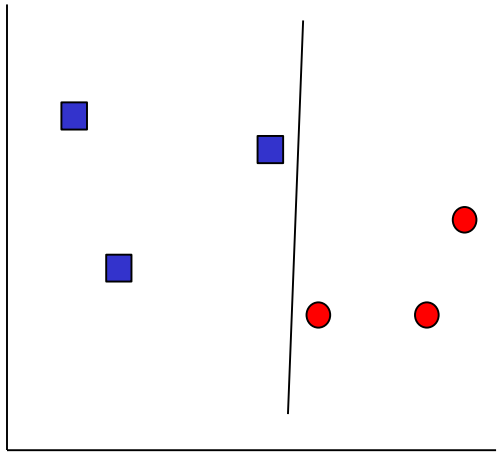
+1 corresponds to blue, and

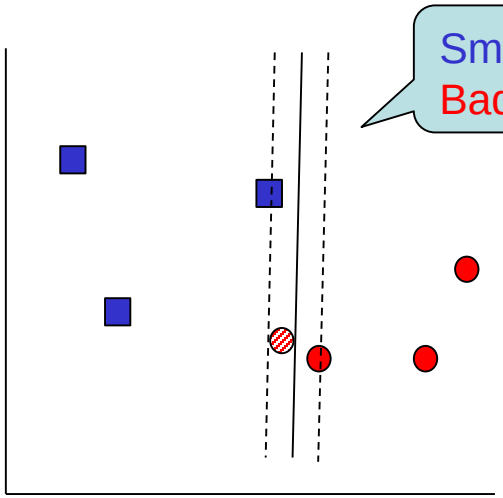
-1 to red, or vice versa.

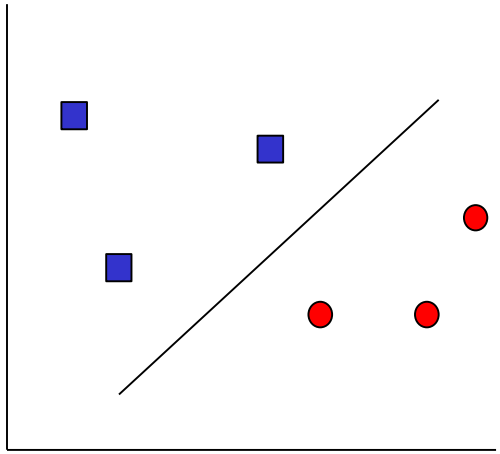
There are many lines however that do the job.

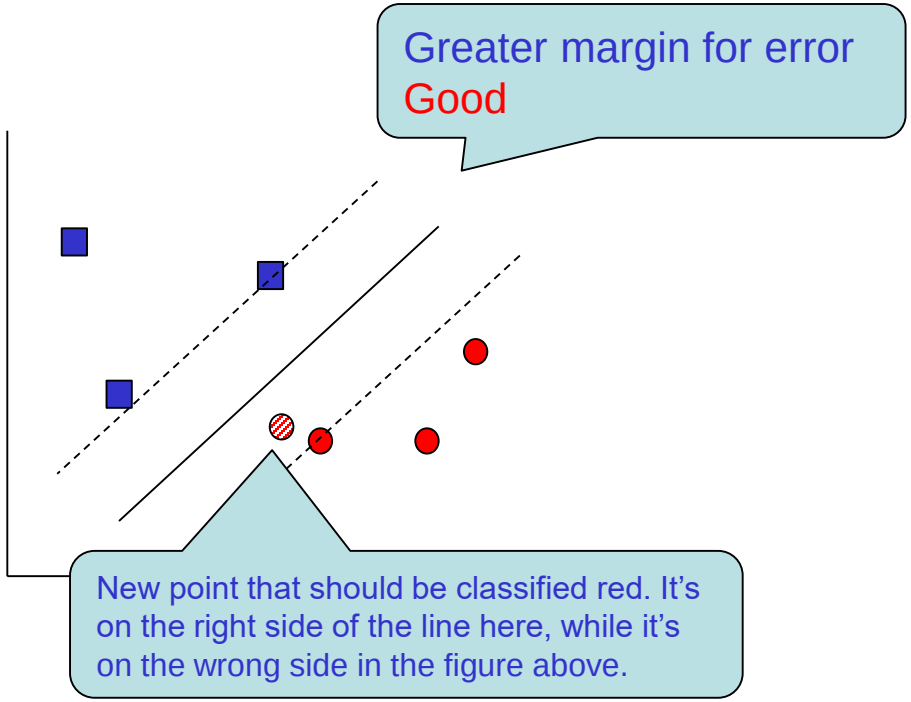
Which one to choose?

Margin for different separators









Scale Invariance

- We rescale **w** and *b* (without changing the line) such that:

$$\mathbf{w} \cdot \mathbf{x}^{k_1} + b = 1$$

for the closest point(s) to the line on the +1 side, and

$$\mathbf{w} \cdot \mathbf{x}^{k_2} + b = -1$$

for the closest point(s) to the line on the -1 side.

Closest points are called "support vectors".

Margin

- **Digression.** The distance of a point to a line: $\mathbf{w} \cdot \mathbf{x} + b = 0$

$$\frac{\left|\mathbf{w} \cdot \mathbf{x}^k + b\right|}{\|\mathbf{w}\|} = \frac{\left|\mathbf{w} \cdot \mathbf{x}^k + b\right|}{\sqrt{\mathbf{w} \cdot \mathbf{w}}}$$

Consider the **closest** points to the line we are building **on each side** respectively. Their distances to the line are:

$$\frac{\left|\mathbf{w} \cdot \mathbf{x}^{k_1} + b\right|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|} \qquad \frac{\left|\mathbf{w} \cdot \mathbf{x}^{k_2} + b\right|}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$$

$\frac{1}{\|\mathbf{w}\|}$ is called *margin*.

We want a line that maximizes margin.

Optimization problem

- Maximizing margin is equivalent to minimizing:

$$\frac{1}{2}\mathbf{w}\cdot\mathbf{w}$$

and this is subject to constraints:

$$y^k\left(\mathbf{w}\cdot\mathbf{x}^k+b\right)\geq 1 \qquad k\in[1,n]$$

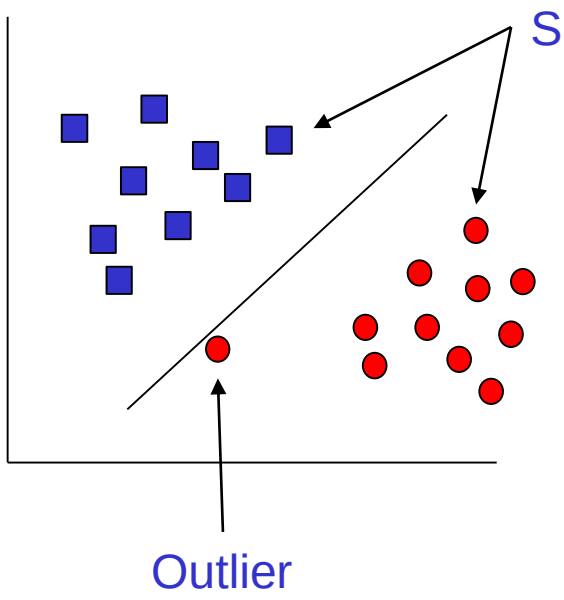
We denote by w* and *b the solutions to the constrained optimization problem.**

y^k is the class, +1 or -1 of training point $\mathbf{x}^k$.

$\mathbf{x}^k$ is a training point. We have *n* of them. They are **constant** vectors, not variable.

The constraints are satisfied iff the line properly separates the training points. We assume for now that the training points are linearly separable.

When there are a few outlier points



If we were to “fully” consider the outlier (make it’s constraint inequality true), the line would have to be more on the left, with much less margin.

To address this, we will **cut some slack** to constraint inequalities, so that they don’t push the line to much.

Modified Optimization – Soft Margin

$$\min_{\mathbf{w},b,\xi} \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_{k=1}^{n}\xi_k$$

Subject to

$$y^k\left(\mathbf{w}\cdot\mathbf{x}^k + b\right) \geq 1 - \xi_k \qquad k \in [1,n]$$

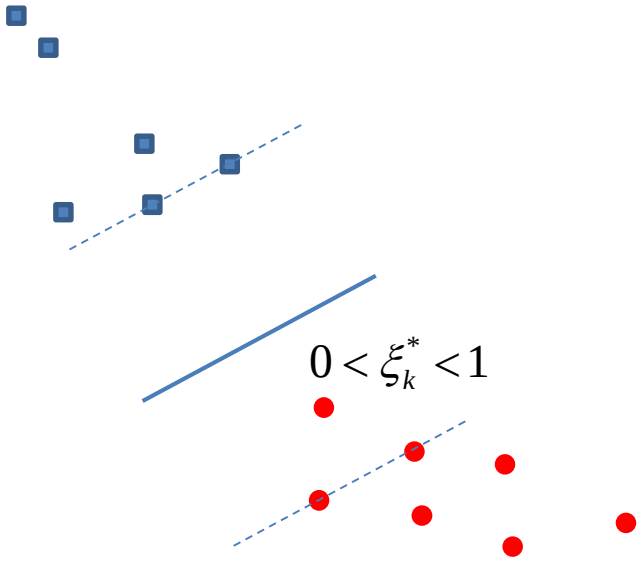
$$\xi_k \geq 0$$

If we don't add this term, ξ_k's can take any big value, making the right-hand side of the inequalities arbitrary small, and such, rendering the constraints useless because we again can produce **w=0** as a solution.

C is a hyper parameter that controls how much "slack" we want to give the inequalities to be true.

Margin Errors

$$\xi_k^* > 0$$



Because $\mathbf{x}^k$ still on the right side of the separator line, i.e.

$$0 < y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) < 1$$

Recall

$$y^k\left(\mathbf{w}^* \cdot \mathbf{x}^k + b^*\right) = 1 - \xi_k^*$$

ROC Curves

Confusion Matrix

	P	N
P	#True Positives (TP)	#False Negatives (FN)
N	#False Positives (FP)	#True Negatives (TN)

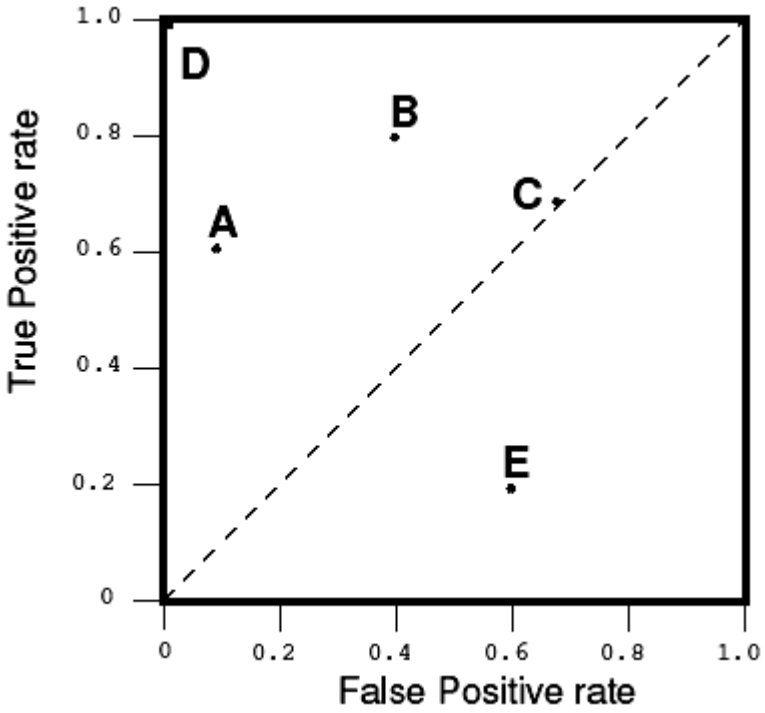
Classified as (predicted class)

Actual class

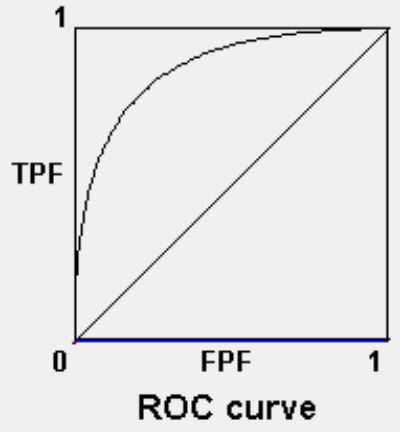
True positive rate is
TPR = P correctly classified / P = TP / P

False positive rate is
FPR = P incorrectly classified / N = FP / N

ROC Space



Curves in ROC space

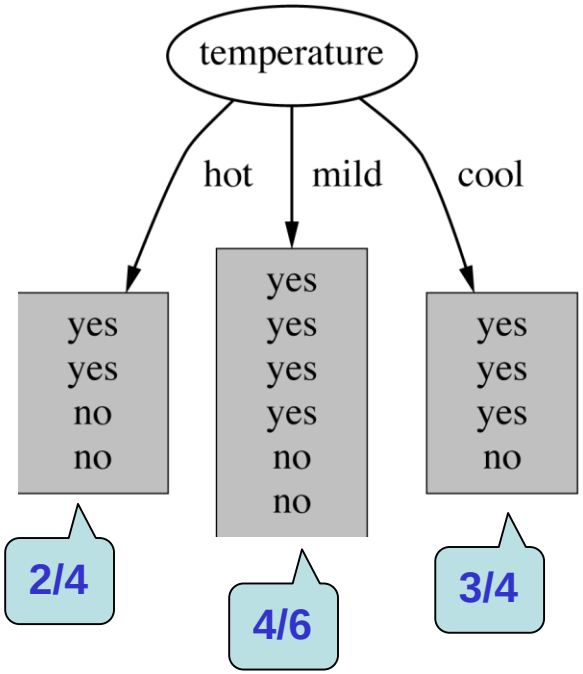


- Many classifiers, such as decision trees or rule sets, are designed to produce only a class decision, i.e., a Y or N on each instance.
  - When such a discrete classifier is applied to a test set, it yields a single **confusion matrix**, which in turn corresponds to **one** ROC point.

- Some classifiers, such as a Naive Bayes, or Logistic Regression, yield a probability or score.
  - A scoring classier can be used with a threshold to produce a discrete (binary) classier:
    - if the classifier output is above the threshold, the classier produces a Y,
    - else a N.
  - Each threshold value produces a different point in ROC space (corresponding to a different confusion matrix).

Creating Scoring Classifiers

- **E.g**, decision tree:
  - Class proportions may serve as a score.



- E.g. SVM:
  - build logistic regression model on the distances of the test points to the separating hyperplane.
    - Distances will serve as a single predictor attribute for generating Log. Reg. probabilities for the class values.

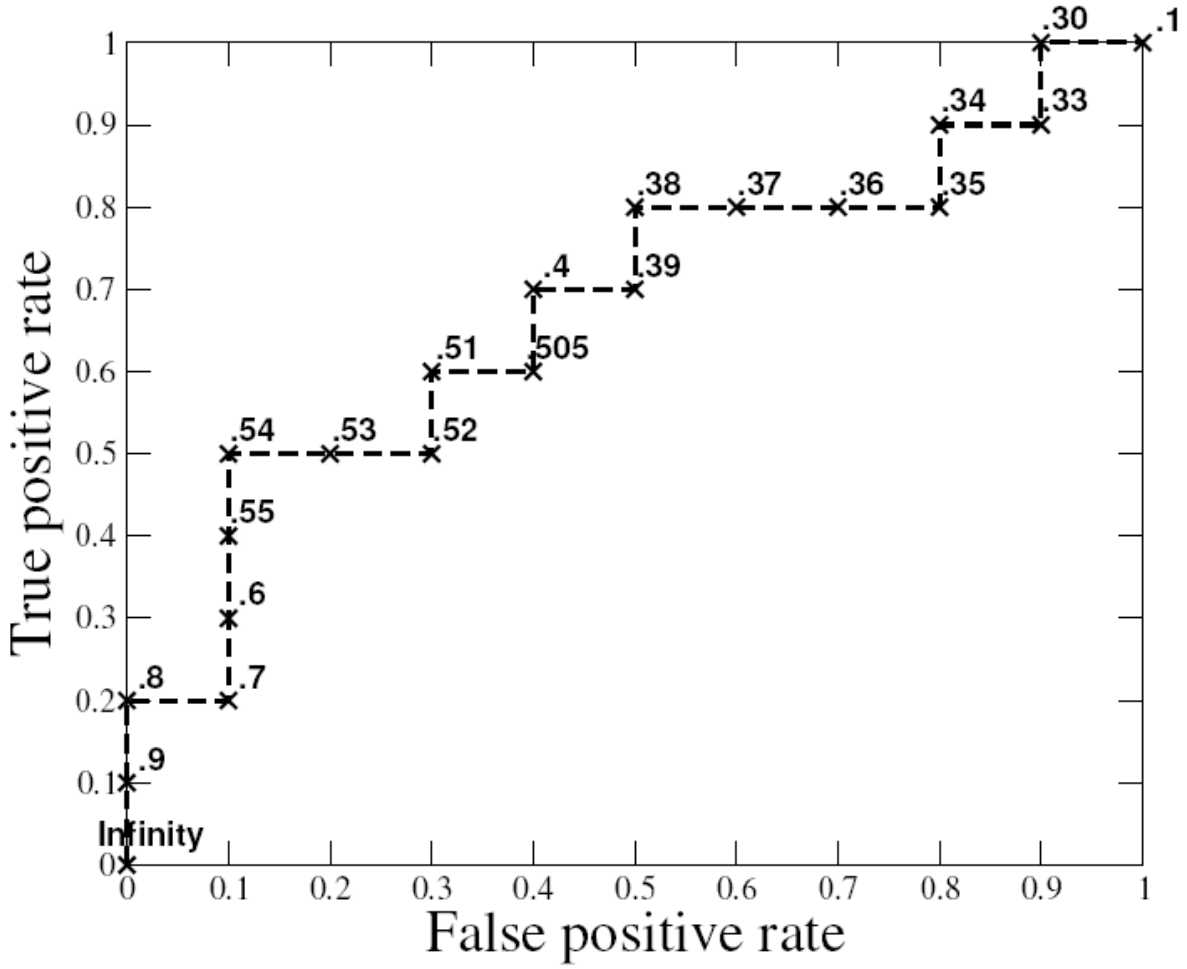
Algorithm

- sort test instances decreasing by their scores and
- move down the list (lowering the threshold), processing one instance at a time and
- update TPR and FPR as we go.

Example of a test set and the scores

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

Example



Question: Why do we have increments of 0.1 for both TPR and FPR? When would the increments be different?

A threshold of +inf produces the point (0; 0).

As we lower the threshold to 0.9 the first positive instance is classified positive, yielding (0;0.1).

As the threshold is further reduced, the curve climbs up and to the right, ending up at (1;1) with a threshold of 0.1.

Lowering this threshold corresponds to moving from the "conservative" to the "liberal" areas of the graph.

Precision and Recall

- **Precision** = TP / (TP+FP)
- **Recall** = TP / P (same as TPR)

- **F measure** is the harmonic mean of the two:
  - Fmeasure = 2 / [(1/Precision) + (1/Recall)] =
    2*Precision*Recall / (Precision + Recall)

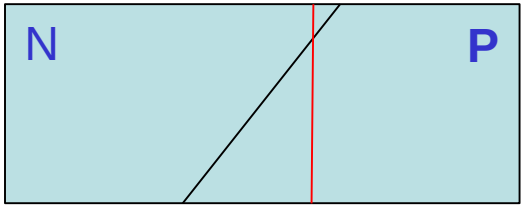
- Example: A dumb classifier that always say "Yes" has a 100% recall, but very bad precision.

- Example: Consider now a classifier that has a recall of 90% and a precision of 40%.

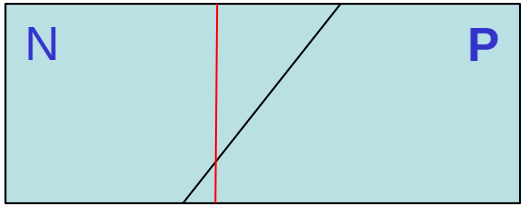
- The F measure for the first classifier is: 2*1*0.1/(1+0.1) ~0.18
- The F measure for the second classifier is: 2*0.9*0.4/(0.9+0.4) ~0.55
- So, the F measure clearly shows that the second classifier is much better.

Precision and Recall Examples

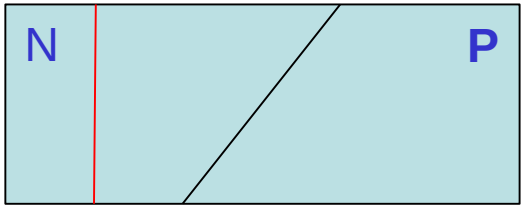
The red line is the classification line. Everything on the right is classified as "P". Everything on the left is classified as N.



Low recall, high precision.



Higher recall, lower precision.



Highest recall (100%), quite low precision.

Sensitivity and Specificity

Popular in the medical domain.

- **Sensitivity** = TP / P (same as TPR and Recall)
  - Example: Fraction of patients with cancer that the classifier rightly predicts.
- **Specificity** = TN / N
  - Example: Fraction of patients without cancer that the classier rightly rules out.

- Often doctors like to increase sensitivity at the expense of specificity.
  - Typically, achieved by lowering the threshold for predicting "Yes".

Random Classifier Baselines

We would like to see how well a classifier compares against a random classifier

Carefully review: **eval_03_baselines.pdf**

Mining Associations

Apriori Algorithm

Apriori Principle

- If an itemset A is frequent, then each itemset $B \subset A$ is frequent.

- Given an itemset A, if we can find an itemset $B \subset A$ that's not frequent, then A cannot be frequent.

Apriori Algorithm

- For each k, we construct two sets of k –itemsets:
  - C_k = candidate k - itemsets = those that might be frequent (support $\geq s$) based on information from the pass for k –1.
  - F_k = the set of truly frequent k - itemsets.

Apriori Algorithm

- Let $k=1$
- Generate frequent itemsets of length 1
- Repeat until no new frequent itemsets are found

  $k=k+1$

  1. **Generate** length k candidate itemsets from length k-1 frequent itemsets
  2. **Prune** candidate itemsets containing subsets of length k-1 that are infrequent
  3. **Count** the support of each candidate by scanning the DB and eliminate candidates that are infrequent, leaving only those that are frequent

Another Example

Min_sup_count = 2

TID	List of item ID's
T1	1, 2, 5
T2	2, 4
T3	2, 3
T4	1, 2, 4
T5	1, 3
T6	2, 3
T7	1, 3
T8	1, 2, 3, 5
T9	1, 2, 3

C1

Itemset
{1}
{2}
{3}
{4}
{5}

F1

Itemset	Sup. count
{1}	6
{2}	7
{3}	6
{4}	2
{5}	2

Generate C2 from F1×F1

Min_sup_count = 2 **F1** **C2**

TID	List of item D's
T1	1, 2, 5
T2	2, 4
T3	2, 3
T4	1, 2, 4
T5	1, 3
T6	2, 3
T7	1, 3
T8	1, 2, 3, 5
T9	1, 2, 3

Itemset	Sup. count
{1}	6
{2}	7
{3}	6
{4}	2
{5}	2

Itemset
{1,2}
{1,3}
{1,4}
{1,5}
{2,3}
{2,4}
{2,5}
{3,4}
{3,5}
{4,5}

Itemset	Sup. C
{1,2}	4
{1,3}	4
{1,4}	1
{1,5}	2
{2,3}	4
{2,4}	2
{2,5}	2
{3,4}	0
{3,5}	1
{4,5}	0

Generate C3 from F2×F2

Min_sup_count = 2

TID	List of item ID's
T1	1, 2, 5
T2	2, 4
T3	2, 3
T4	1, 2, 4
T5	1, 3
T6	2, 3
T7	1, 3
T8	1, 2, 3, 5
T9	1, 2, 3

F2

Itemset	Sup. C
{1,2}	4
{1,3}	4
{1,5}	2
{2,3}	4
{2,4}	2
{2,5}	2

C3

Itemset
{1,2,3}
{1,2,5}
{1,3,5}
{2,3,4}
{2,3,5}
{2,4,5}

Prune

Itemset
{1,2,3}
{1,2,5}
~~{1,3,5}~~
~~{2,3,4}~~
~~{2,3,5}~~
~~{2,4,5}~~

F3

Itemset	Sup. C
{1,2,3}	2
{1,2,5}	2

Generate C4 from F3×F3

Min_sup_count = 2

TID	List of item ID's
T1	1, 2, 5
T2	2, 4
T3	2, 3
T4	1, 2, 4
T5	1, 3
T6	2, 3
T7	1, 3
T8	1, 2, 3, 5
T9	1, 2, 3

C4

Itemset
{1,2,3,5}

{1,2,3,5} is pruned because {2,3,5} is infrequent

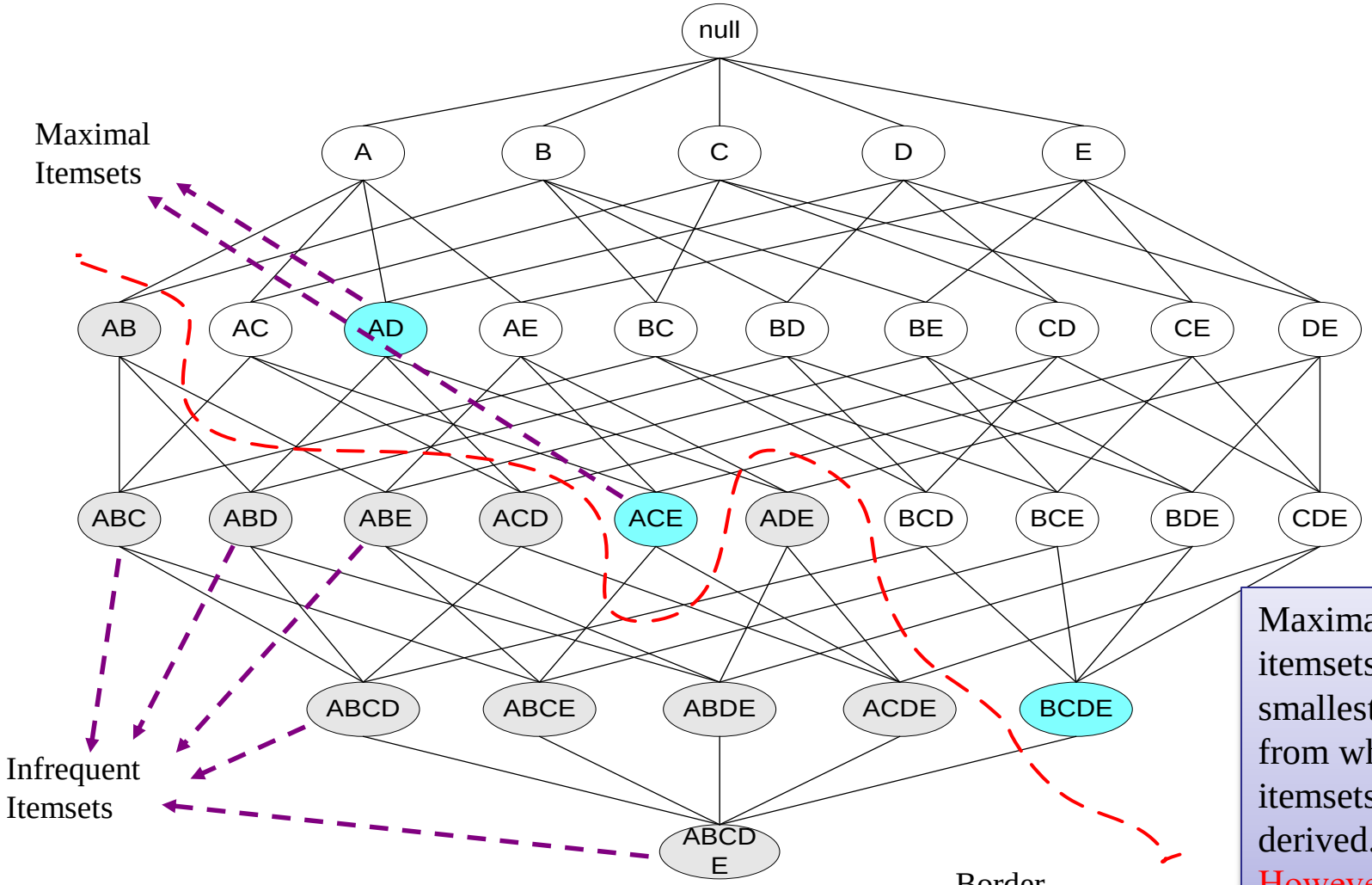
F3

Itemset	Sup. C
{1,2,3}	2
{1,2,5}	2

Compact Representation of Frequent Itemsets

Maximal Frequent Itemsets

A freq. itemset is maximal freq. if none of its immediate supersets is frequent



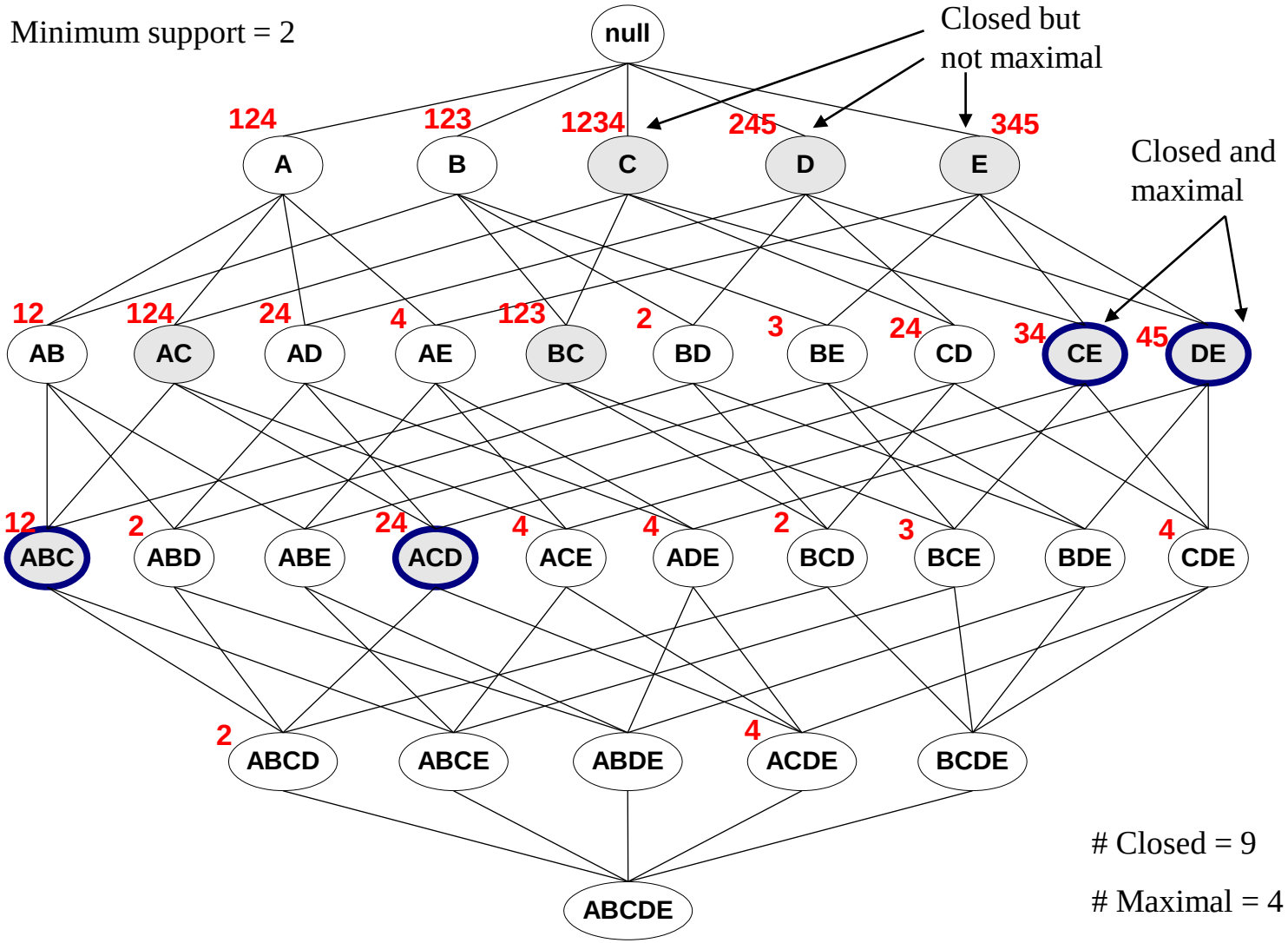
Maximal frequent itemsets form the smallest set of itemsets from which all frequent itemsets can be derived.

However, we can't derive support counts.

Closed Frequent Itemsets

A freq. itemset is closed frequent if none of its immediate supersets has same supp.

TID	Items
1	ABC
2	ABCD
3	BCE
4	ACDE
5	DE



Closed = 9

Maximal = 4

Deriving Frequent Itemsets From Closed Frequent Itemsets

- Consider the subsets of the closed itemsets.

- Let *itset* be such subset. The support of *itset* must be equal to the **largest support** among its supersets.

Example

Closed = {ABC:3, ACD:4, CE:6, DE:7}

F3 = {ABC:3, ACD:4}

F2 = {AB:3, AC:4, BC:3, AD:4, CD:4, CE:6, DE:7}

F1 = {A:4, B:3, C:6, D:7, E:7}

Computing Frequent Closed Itemsets

- Use the Apriori Algorithm.
- After computing, say F_k and F_{k+1},
  - Check for itemsets in F_k that have a support equal to the support of one of their supersets in F_{k+1}.
  - Purge all such itemsets from F_k.

Association Analysis (2)
FPTree/FPGrowth

FP-Tree/FP-Growth Algorithm

Building the FP-Tree

1. Scan data to determine the support count of each item.

   Infrequent items are discarded, while the frequent items are sorted in decreasing order of support counts.

2. Make a second pass over the data to construct the FPtree.

   As the transactions are read, their items are sorted according to the above order.

First scan – determine frequent 1-itemsets, then build header

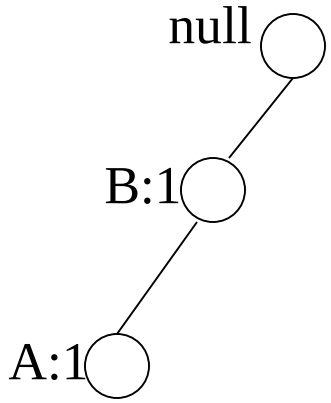
TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

B	8
A	7
C	7
D	5
E	3

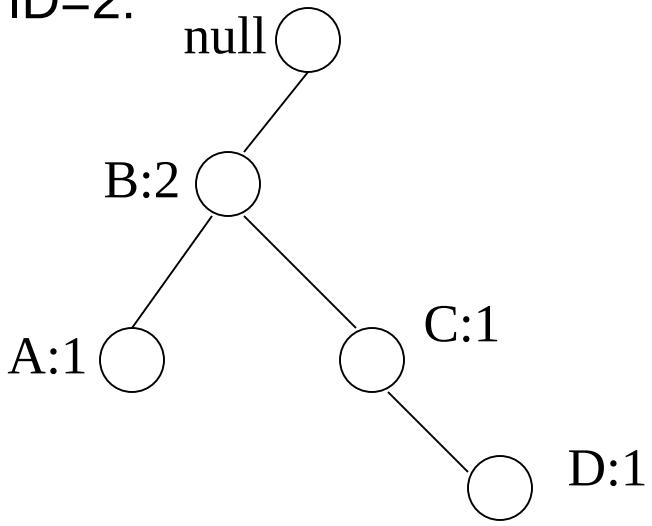
FP-tree construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

After reading TID=1:



After reading TID=2:



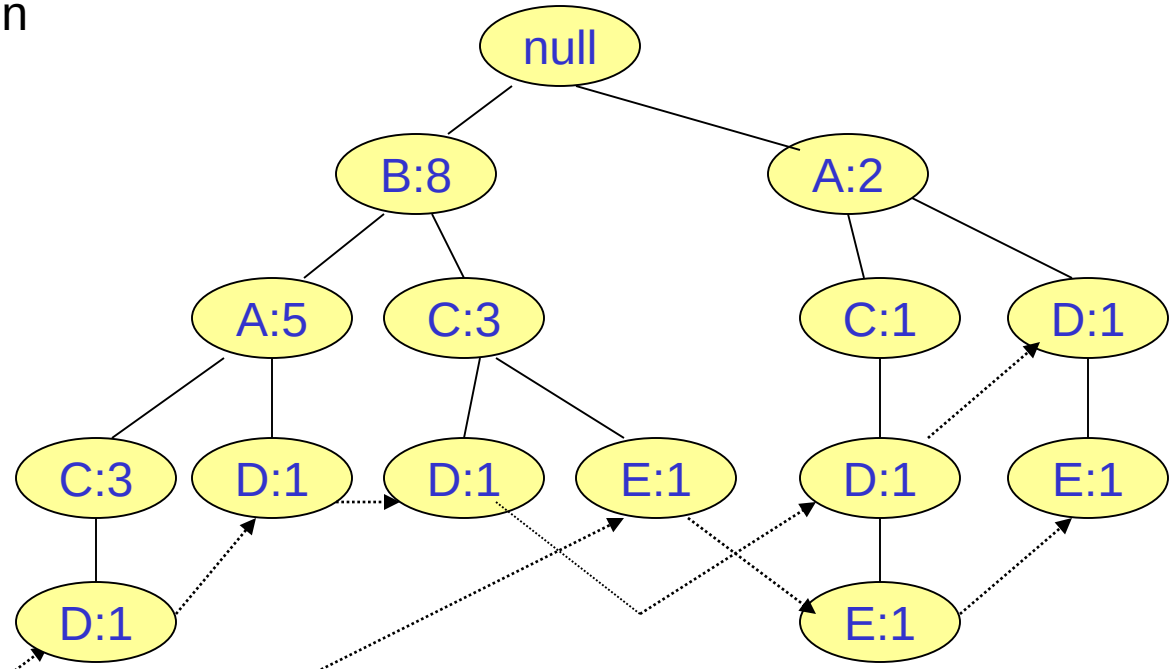
FP-Tree Construction

TID	Items
1	{A,B}
2	{B,C,D}
3	{A,C,D,E}
4	{A,D,E}
5	{A,B,C}
6	{A,B,C,D}
7	{B,C}
8	{A,B,C}
9	{A,B,D}
10	{B,C,E}

Transaction Database

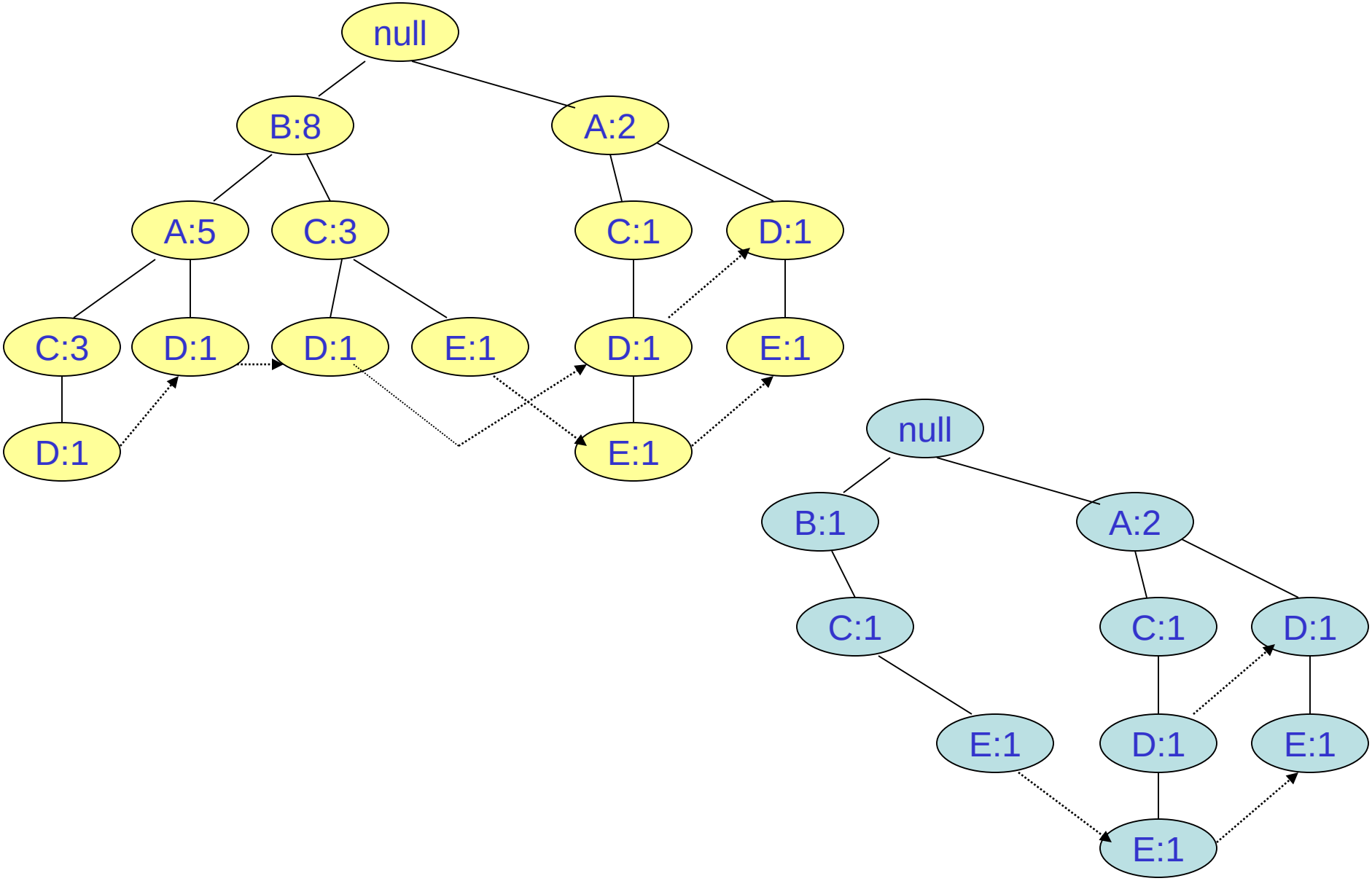
Header table

Item	Pointer
B	8
A	7
C	7
D	5
E	3



Chain pointers help in quickly finding all the paths of the tree containing some given item.

Paths (transactions) containing node E (last node in the header table)



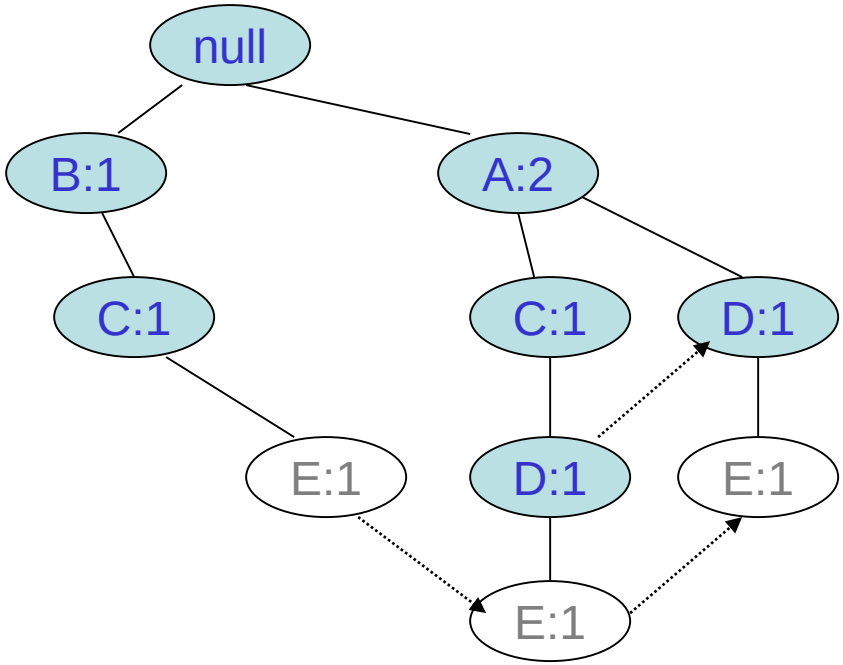
Conditional FP-Tree for E

- FP-Growth builds a conditional FP-Tree for E, which is the tree of itemsets ending in E.

- It is not the tree obtained in the previous slide as result of deleting nodes from the original tree. Why?

- Because the order of the items can change.
  - E.g. now, C has a higher count than B.

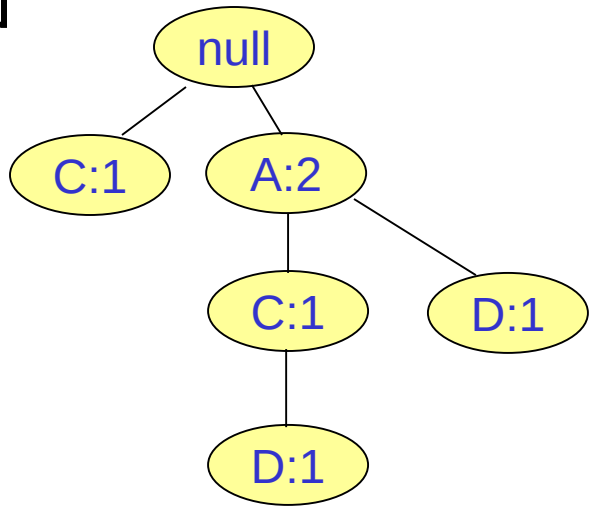
Suffix E



(New) Header table

A	2
C	2
D	2

Conditional FP-Tree for suffix E



The set of paths ending in E.

Insert each path (after truncating E) into a new tree.

B doesn't survive because it has support 1, which is lower than min support of 2.

We continue recursively.

Base of recursion: When the tree has a single path only.

We output FI: E

Steps of Building Conditional FP-Trees

1. Find the paths containing on focus item.

2. **Read the tree** to determine the new counts of the items along those paths.

   Build a new header.

3. **Read again the tree**. Insert the paths into the conditional FP-Tree according to the new order.

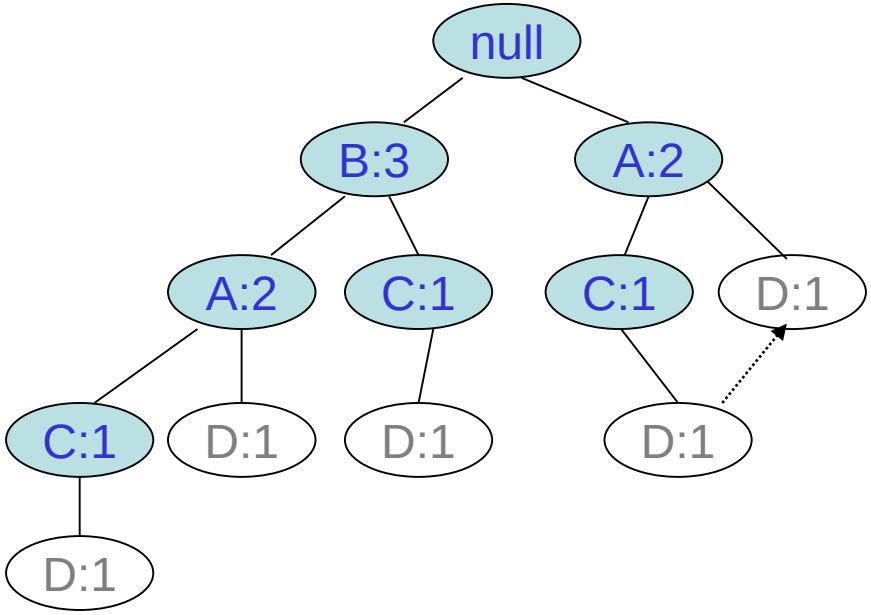
Base of Recursion

- We continue recursively on the conditional FP-Tree.

- Base case of recursion: when the tree is just a single path.
  - Then, we just produce all the subsets of the items on this path concatenated with the corresponding suffix.

Suffix D

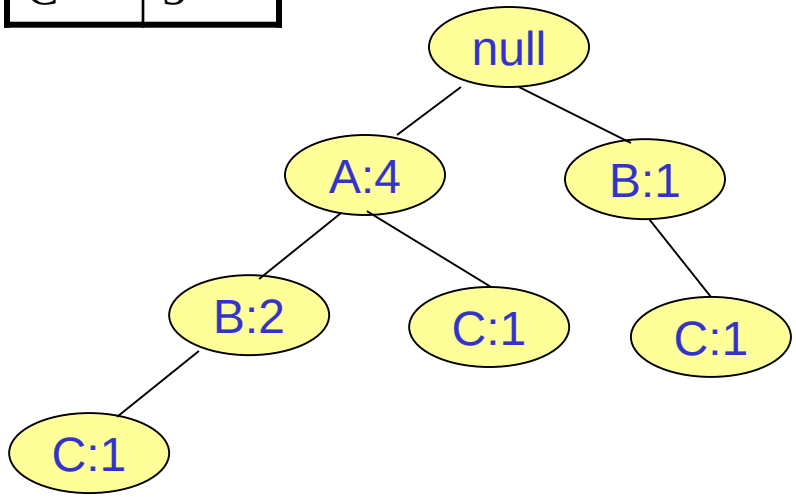
Observe how the shape of the tree changes.



(New) Header table

A	4
B	3
C	3

Conditional FP-Tree for suffix D



The set of paths ending in D.

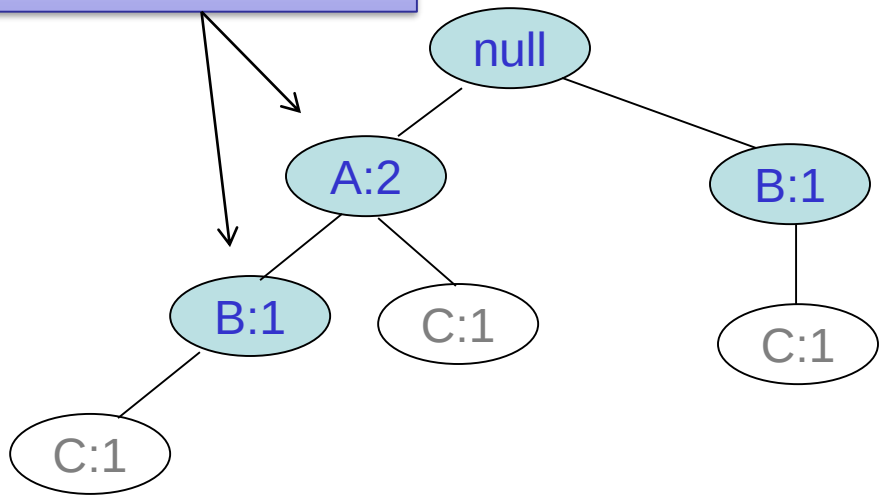
Insert each path (after truncating D) into a new tree.

We continue recursively.
Base of recursion: When the tree has a single path only.

FI: D

We update the counts, so that each count is equal to the sum of the leafs of the subtree.

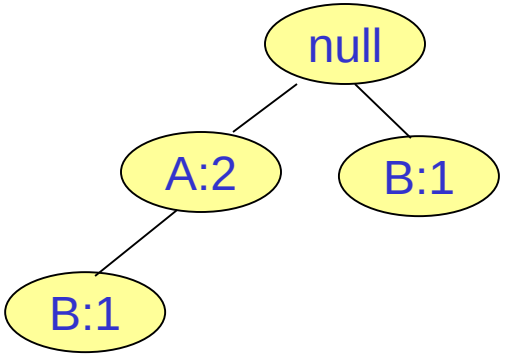
Suffix CD



(New) Header table

A	2
B	2

Conditional FP-Tree for suffix CD



The set of paths, from the D-conditional FP-Tree, ending in C.

Insert each path (after truncating C) into a new tree.

We continue recursively.
Base of recursion: When the tree has a single path only.

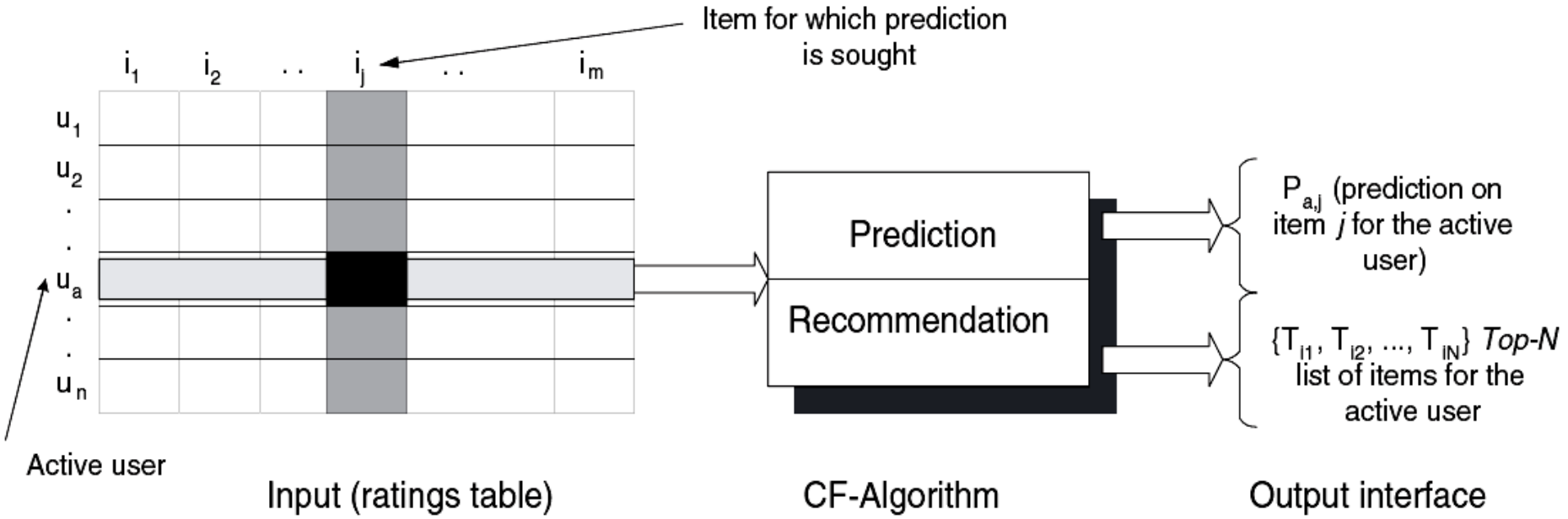
FI: CD

Also…

- Also carefully review:
  - Generating rules (association2.pdf)
  - Evaluation of association patterns (association5.pdf)
  - mining of sequences (association5.pdf)

Recommender Systems

Collaborative Filtering Process



Ratings in a hashtable - Example

```
critics = {
    'Lisa Rose':  {'Lady in the Water': 2.5,
                   'Snakes on a Plane': 3.5,
                   'Just my Luck': 3.0,
                   'Superman Returns': 3.5,
                   'You, Me and Dupree': 2.5,
                   'The Night Listener': 3.0},

    'Gene Seymour': {'Lady in the Water': 3.0,
                   'Snakes on a Plane': 3.5,
                   'Just my Luck': 1.5,
                   'Superman Returns': 5.0,
                   'The Night Listener': 3.0,
                   'You, Me and Dupree': 3.5},

      ...
```

User Similarity: Pearson Correlation

- Consider two user vectors of ratings:

  x=[1, 2, 3, 5, 8] and **y**=[.11, .12, .13, .15, .18]

- First **centralize** by subtracting their mean, then compute cosine similarity.
- m_x = (1 + 2 + 3 + 5 + 8)/5 = 3.8
- m_y = (.11 + .12 + .13 + .15 + .18)/5 = .138

x'=[1-3.8, 2-3.8, 3-3.8, 5-3.8, 8-3.8] =

[−2.8, −1.8, −0.8, 1.2, 4.2]

y'=[.11-.138, .12-.138, .13-.138, .15-.138, .18-.138] =

[−0.028, −0.018, −0.008, 0.012, 0.042]

$$sim_{x,y} = \cos\theta = \frac{\mathbf{x}'\cdot\mathbf{y}'}{\|\mathbf{x}'\|\|\mathbf{y}'\|} = \frac{0.308}{\sqrt{30.8}\sqrt{0.00308}} = 1$$

as we intuitively expect. Octave: a*b / (sqrt(a*a) * sqrt(b*b))

This is called **Pearson Correlation Coefficient**.

Pearson Correlation Formula

$$\mathbf{x} = [x_1, \ldots, x_m]$$
$$\mathbf{y} = [y_1, \ldots, y_m]$$

- The formula more detailed is:

$$sim_{\mathbf{x},\mathbf{y}} = \frac{\sum_{i=1}^{m} (x_i - \bar{x}) \cdot (y_i - \bar{y})}{\sqrt{\sum_{i=1}^{m} (x_i - \bar{x})^2} \cdot \sqrt{\sum_{i=1}^{m} (y_i - \bar{y})^2}}$$

Only the *i*'s for which both **x** and **y** have known ratings participate

How to use similarity?

- To recommend item *i* to user *u*, predict how *u* would have rated *i* by computing a **weighted average** of the ratings that other users have given to *i*.

- The weights are the similarities.
  - The more similar a user *v* is to *u*, the bigger the weight for *v*'s rating of *i*.

$$\hat{r}_{u,i} = \frac{\sum_{v \in U_i} r_{v,i} \cdot sim_{v,u}}{\sum_{v \in U_i} sim_{v,u}}$$

U_i is the set of users who have rated *i*.

This is using "user-user" similarities (only positive ones are considered)

Recommendations using item-item similarities

$$\hat{r}_{u,i} = \frac{\sum_{j \in I_u} r_{u,j} \cdot sim_{i,j}}{\sum_{j \in I_u} sim_{i,j}}$$

I_u is the set of items rated by user u.

This is using "item-item" similarities

Recommender Systems II

(Algorithm from Netflix Prize)

Evaluating Rec. Systems

- Consider root-mean-square error (RMSE).

$$\sqrt{\frac{\sum_{u,i} e_{u,i}^2}{N}} = \sqrt{\frac{\sum_{u,i} \left(r_{u,i} - \hat{r}_{u,i}\right)^2}{N}}$$

- How to minimize it?

We will use gradient descent to minimize each term:

$$e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2$$

Mean and biases: Baseline

First what will the prediction be?

$$\hat{r}_{u,i} = \mu + b_u + b_i$$

b_u and b_i capture the deviations of user u and item i from the average.

E.g., suppose that we want a baseline predictor for the rating of movie **Titanic** by user **Joe**.

Now, say that the average rating over all movies, μ, is 3.7 stars.

Furthermore, **Titanic** is better than an average movie, so it tends to be rated 0.5 stars above the average, i.e. b_{Titanic}=0.5

On the other hand, **Joe** is a critical user, who tends to rate 0.3 stars lower than the average, i.e. b_{Joe}= -0.3.

Thus, the baseline predictor for **Titanic's** rating by **Joe** would be 3.9 stars by calculating 3.7−0.3+0.5.

Then use gradient descent to minimize $e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2$

Mean and biases:Gradient

$$e_{u,i}^2 = \left(r_{u,i} - \hat{r}_{u,i}\right)^2 = \left(r_{u,i} - \mu - b_u - b_i\right)^2$$

We apply regularization to fight overfitting. So, **minimize**:

$$f\left(b_u, b_i\right) = \left(r_{u,i} - \mu - b_u - b_i\right)^2 + \lambda_1 b_u^2 + \lambda_2 b_i^2$$

Regularization: The last two terms are added to penalize big bu and bi values, so that bu and bi don't become too important for the particular dataset and thus have overfitting.
It's penalty for being too good for one dataset and suffering for another.

$$\frac{\partial f}{\partial b_u} =$$

$$-2\left(r_{u,i} - \mu - b_u - b_i\right) + 2\lambda_1 b_u =$$

$$-2e_{u,i} + 2\lambda_1 b_u =$$

$$-2\left(e_{u,i} - \lambda_1 b_u\right)$$

$$\frac{\partial f}{\partial b_i} =$$

$$-2\left(r_{u,i} - \mu - b_u - b_i\right) + 2\lambda_2 b_i =$$

$$-2e_{u,i} + 2\lambda_2 b_i =$$

$$-2\left(e_{u,i} - \lambda_2 b_i\right)$$

Mean and biases: Gradient Descent

Iterate over each $r_{u,i}$, and apply gradient descent update rules:

$$b_u \leftarrow b_u + \gamma\left(e_{u,i} - \lambda_1 b_u\right)$$
$$b_i \leftarrow b_i + \gamma\left(e_{u,i} - \lambda_2 b_i\right)$$

e.g.

$$\lambda_1 = 0.02$$
$$\lambda_2 = 0.02$$
$$\gamma = 0.005$$

are good, but cross-validation can be used to find better values.

30 iterations are good enough.

Document Retrieval

Vector space model

- Documents are treated as a "bag" of words or terms.
  - Each document is represented as a vector.

- TF: (normalized) **term frequency**

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$$

word *i* **document *j*** **size of vocabulary**

Retrieval in the vector space model

- Query **q** is represented in the same way as a document.
  - Weight of terms in **q** computed in the same way as for regular document.

- Relevance of **d** to **q**: Compare the similarity of query **q** and document **d**.
  - The bigger the cosine the smaller the angle and the higher the similarity

The sum has as many non-zero terms as there are words in the intersection of q and d_i.

Dot product

$$\cos(\vec{q},\vec{d}) = \frac{\vec{q}\bullet\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

We only need to retrieve from the inverted index the documents that contain at least one word of the query.

Documents that don't have some word in common with the query have a similarity of 0 with the query, so they don't need to be considered.

Document Frequency

term	df_t
calpurnia	1
animal	100
sunday	1,000
fly	10,000
under	100,000
the	1,000,000

- Suppose query is: calpurnia animal
- An appearance of "calpurnia" should be much more important than an appearance of "animal"

TF-IDF term weighting scheme

- The most well known weighting scheme
  - TF: (normalized) **term frequency**
  - IDF: **inverse document frequency**.

$$tf_{ij} = \frac{f_{ij}}{\max\{f_{1j}, f_{2j}, \ldots, f_{|V|j}\}}$$

N: total number of docs

df_i: the number of docs that t_i appears.

$$idf_i = \log \frac{N}{df_i}$$

- The final TF-IDF term weight is:

$$w_{ij} = tf_{ij} \times idf_j$$

Each document will be a vector of such numbers.

Cluster Analysis

Euclidean Distance

- When all the attributes are continuous we can use the Euclidean Distance

$$dist = \sqrt{\sum_{k=1}^{n} (p_k - q_k)^2}$$

Where n is the number of dimensions (attributes) and p_k and q_k are, respectively, the k^{th} attributes (components) or data objects p and q.

- Attribute scaling is necessary, if scales differ
  - E.g. weight, salary have different scales

Minkowski Distance

- Minkowski Distance is a generalization of Euclidean Distance

$$dist = \sqrt[r]{\sum_{k=1}^{n} | p_k - q_k |^r}$$

Where r is a parameter, n is the number of dimensions (attributes) and p_k and q_k are, respectively, the kth attributes (components) or data objects **p** and **q**.

Examples

- $r = 1$. City block (Manhattan, taxicab, L_1 norm) distance.
- $r = 2$. Euclidean distance
- $r \rightarrow \infty$. "supremum" (L_{max} norm, L_∞ norm) distance.
  - This is the maximum difference between any component of the vectors

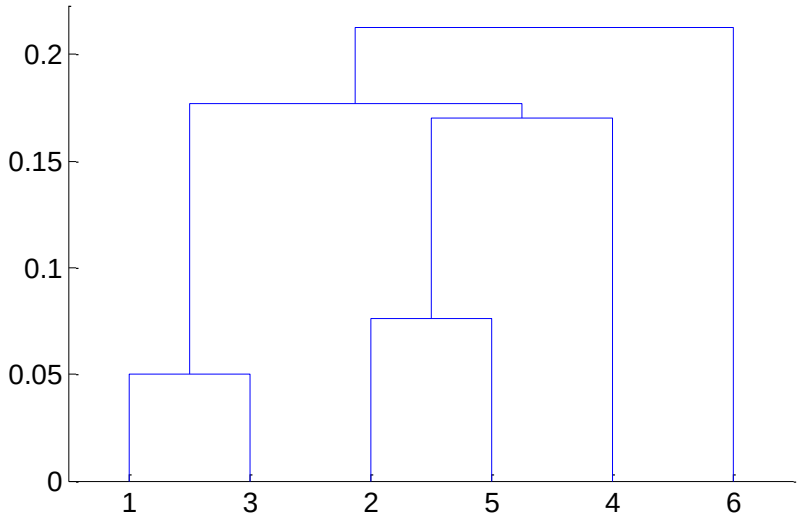
K-means Clustering

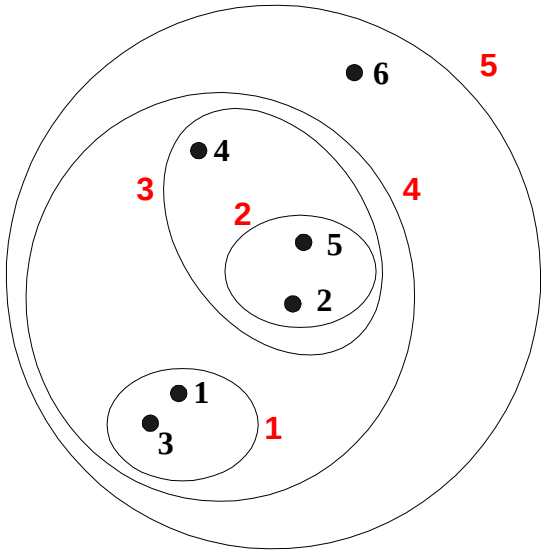
- Each cluster is associated with a centroid (center point)
- Each point is assigned to the cluster with the closest centroid
- Number of clusters, *K*, must be specified
- Basic algorithm is very simple

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

Hierarchical Clustering

- Produces a set of nested clusters organized as a hierarchical tree





Hierarchical Clustering

Algorithm

Let each data point be a cluster

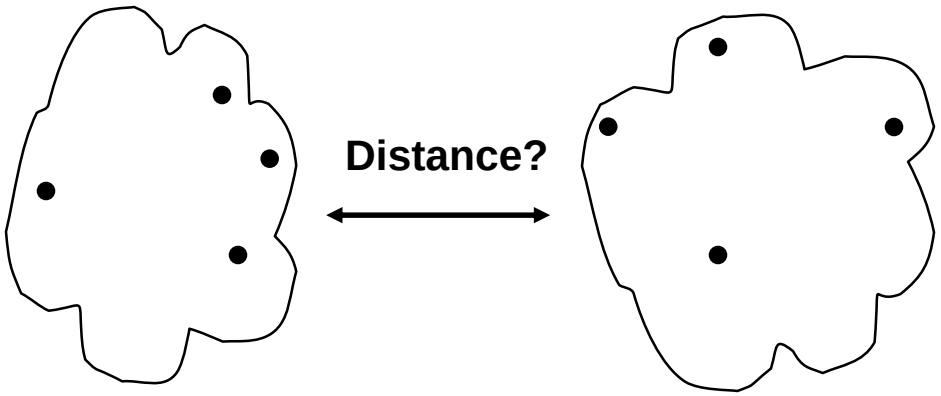
Repeat

Merge **the two closest** clusters

Until only a single cluster remains

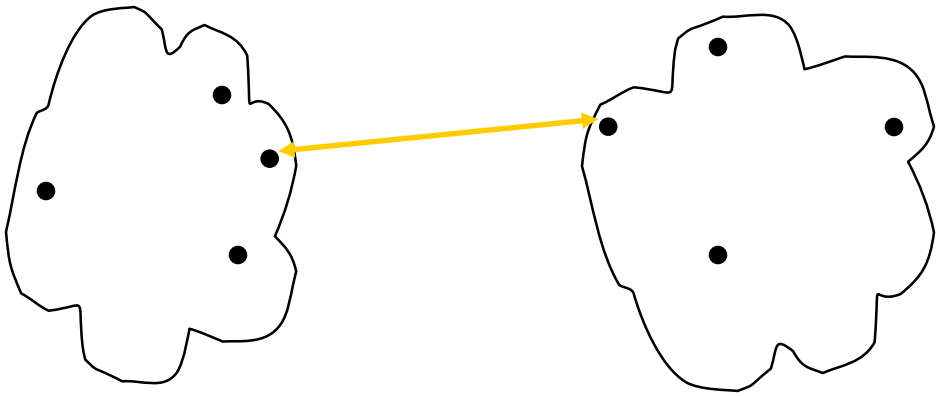
- Key operation is the computation of the proximity of two clusters.

First Define Inter-Cluster Similarity



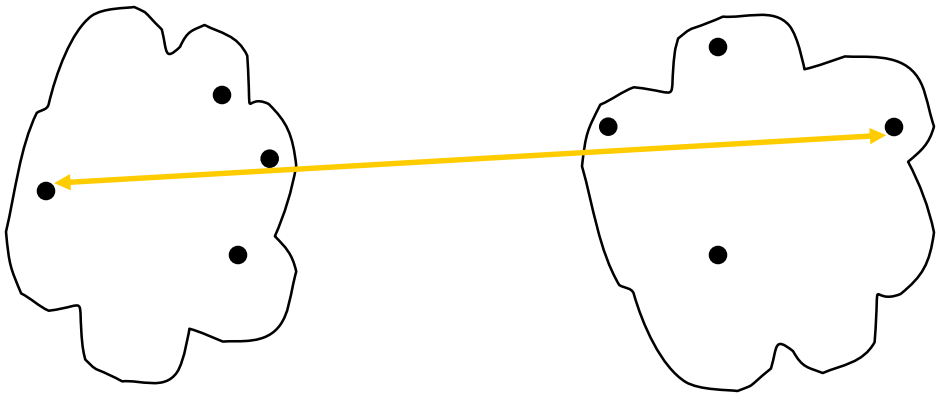
- **MIN**
- **MAX**
- **Group Average**

How to Define Inter-Cluster Similarity



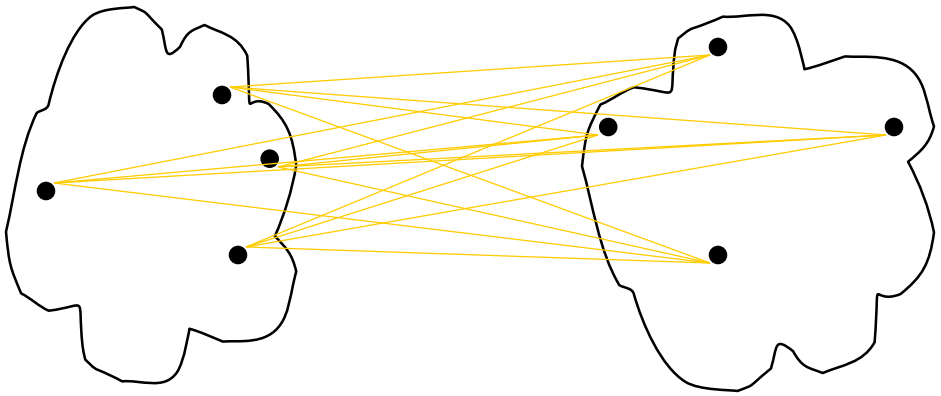
- **MIN**
- **MAX**
- **Group Average**

How to Define Inter-Cluster Similarity



- **MIN**
- **MAX**
- **Group Average**

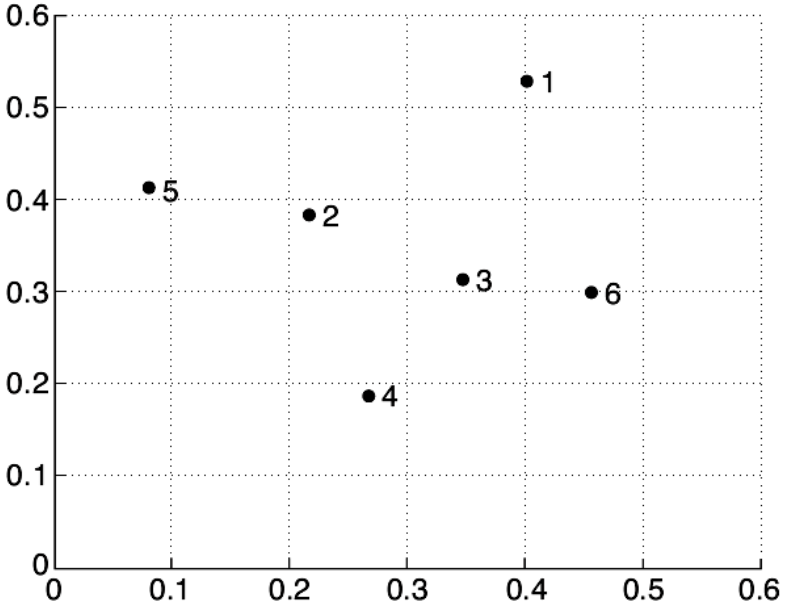
How to Define Inter-Cluster Similarity



- **MIN**
- **MAX**
- **Group Average**

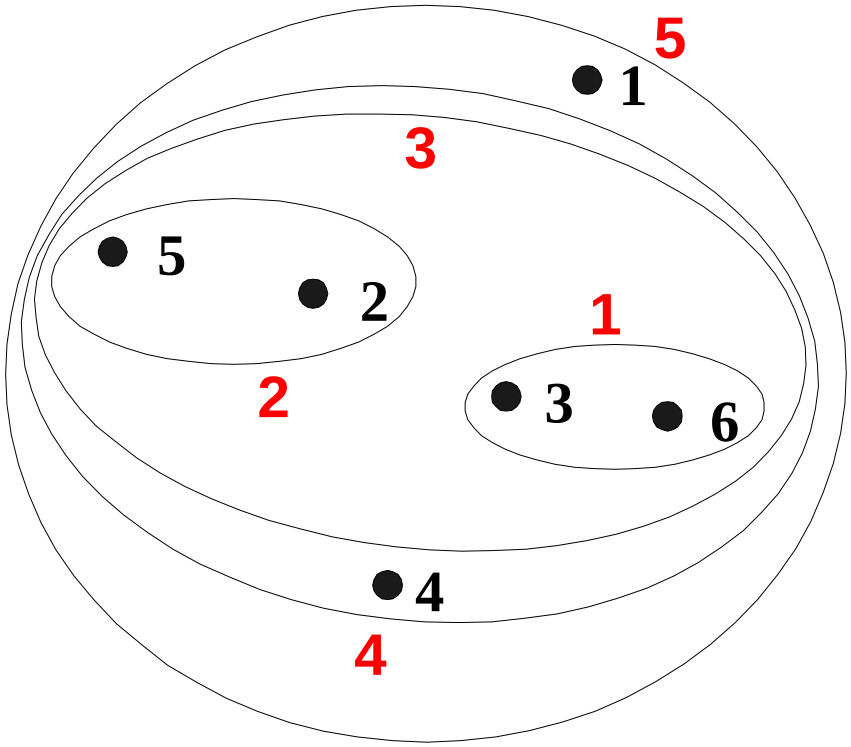
Cluster Similarity: MIN

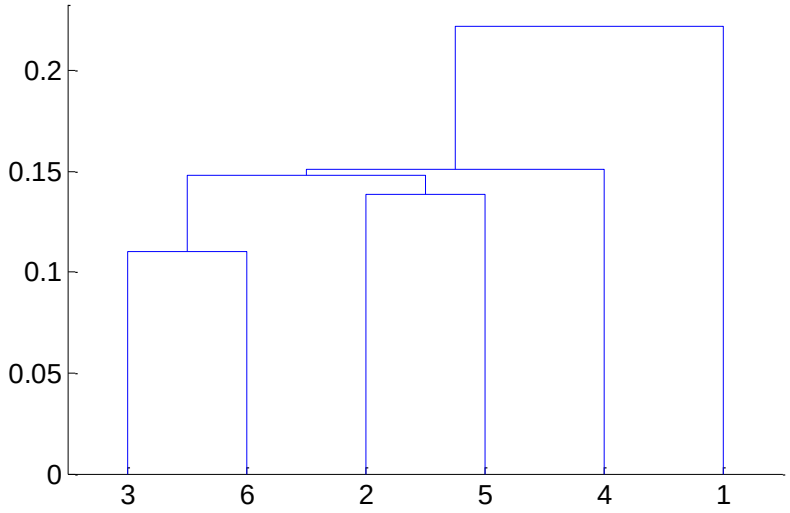
- Similarity of two clusters is based on the two most similar (closest) points in the different clusters
  - Determined by one pair of points



	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MIN



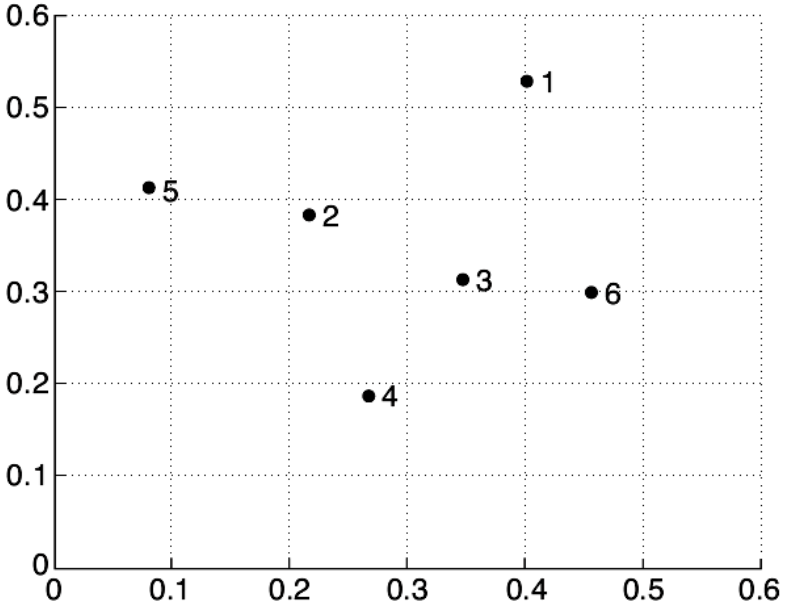


Nested Clusters

Dendrogram

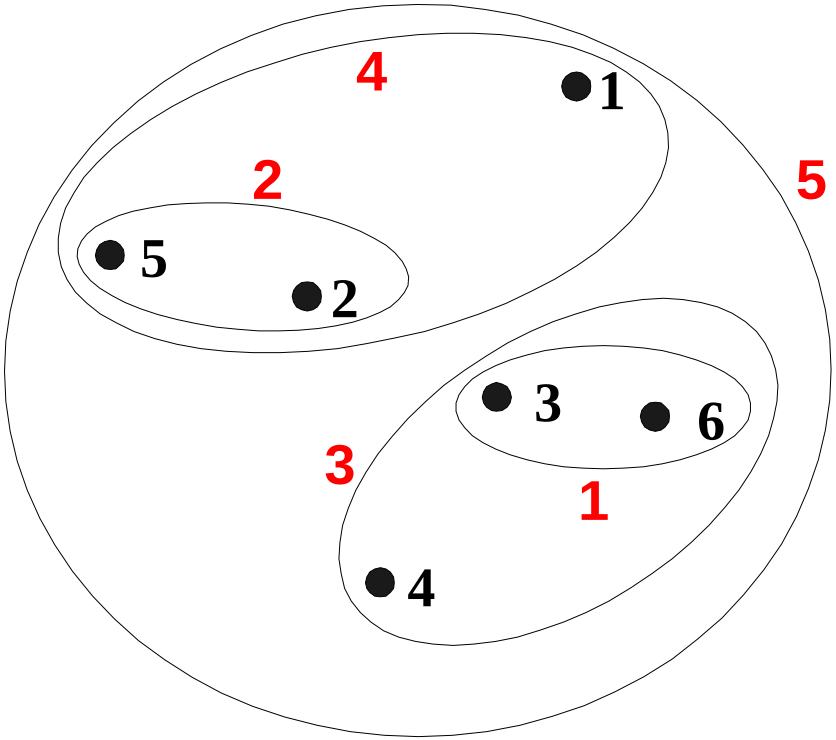
Cluster Similarity: MAX

- Similarity of two clusters is based on the two least similar (most distant) points in the different clusters
  - Determined by all pairs of points in the two clusters

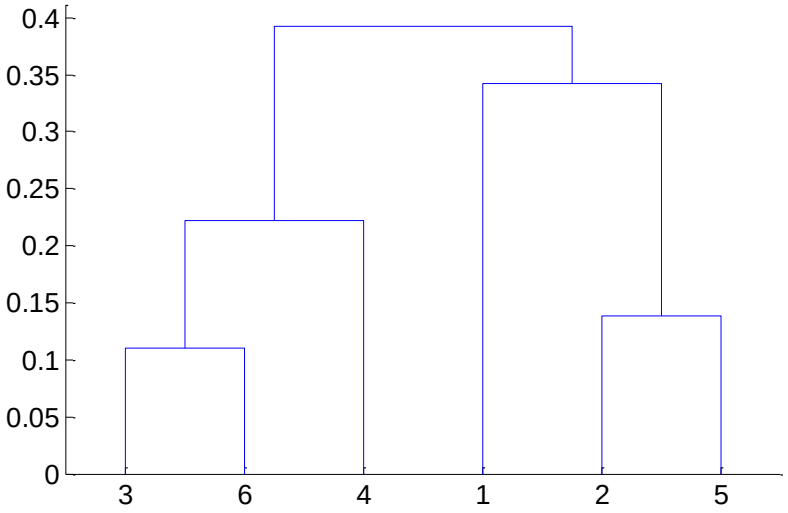


	p1	p2	p3	p4	p5	p6
p1	0.00	0.24	0.22	0.37	0.34	0.23
p2	0.24	0.00	0.15	0.20	0.14	0.25
p3	0.22	0.15	0.00	0.15	0.28	0.11
p4	0.37	0.20	0.15	0.00	0.29	0.22
p5	0.34	0.14	0.28	0.29	0.00	0.39
p6	0.23	0.25	0.11	0.22	0.39	0.00

Hierarchical Clustering: MAX



Nested Clusters



Dendrogram