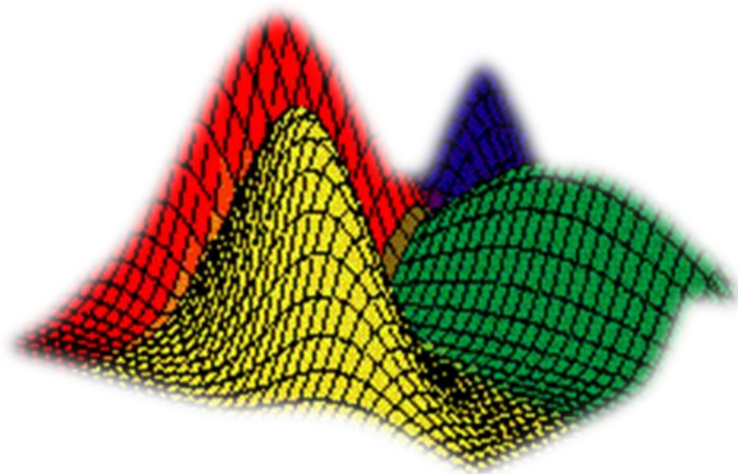


CSE 473: Pattern Recognition

Syntactic Pattern Recognition

Extension to Graph Matching Approach

- An *attributed graph* $G = \{N, P, R\}$ is a 3-tuple where
  - N is a set of nodes,
  - P is a set of properties of these nodes,
  - R is a set of relations between nodes.

Matching Through Attributed Graph

Assignment of nodes:

- Let $p_q^i(n)$ denote the value of the q'th property of node n of graph G_i.

Matching Through Attributed Graph

Assignment of nodes:

- Let $p_q^i(n)$ denote the value of the q'th property of node n of graph G_i.

• Nodes $n_1 \in N_1$ and $n_2 \in N_2$ are said to form an assignment (n_1, n_2) if

$$p_q^1(n_1) \sim p_q^2(n_2)$$

where "$\sim$" denotes similarity.

Matching Through Attributed Graph

Compatibility of assignments:

- Let $r_j^i(n_x, n_y)$ denote the j'th relation involving nodes $n_x, n_y \in N_i$.

- Two assignments (n_1, n_2) and (n_1', n_2') are considered *compatible* if

$$r_j^1(n_1, n_1') \sim r_j^2(n_2, n_2') \quad \forall j.$$

Matching Through Attributed Graph

Compatibility of assignments:

- Let $r_j^i(n_x, n_y)$ denote the j'th relation involving nodes $n_x, n_y \in N_i$.

- Two assignments (n_1, n_2) and (n_1', n_2') are considered *compatible* if

$$r_j^1(n_1, n_1') \sim r_j^2(n_2, n_2') \quad \forall j.$$

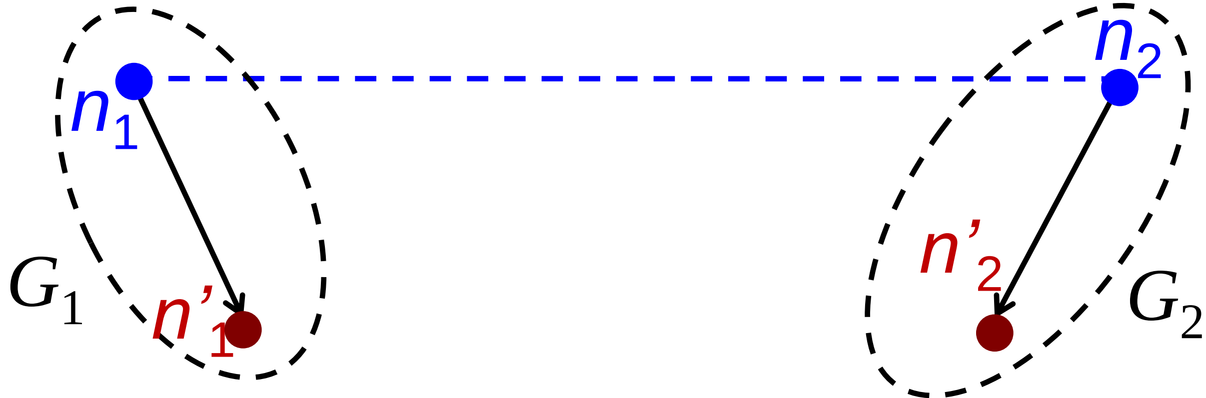
Matching Through Attributed Graph

Compatibility of assignments:

- Let $r_j^i(n_x, n_y)$ denote the j'th relation involving nodes $n_x, n_y \in N_i$.

- Two assignments (n_1, n_2) and (n_1', n_2') are considered *compatible* if

$$r_j^1(n_1, n_1') \sim r_j^2(n_2, n_2') \quad \forall j.$$



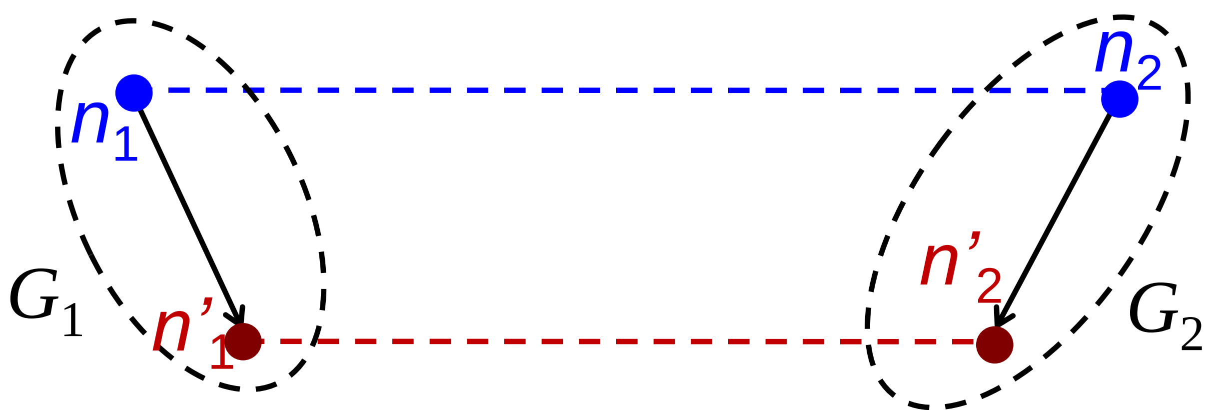
Matching Through Attributed Graph

Compatibility of assignments:

- Let $r^i_j(n_x, n_y)$ denote the j'th relation involving nodes $n_x, n_y \in N_i$.

- Two assignments (n_1, n_2) and (n'_1, n'_2) are considered *compatible* if

$$r^1_j(n_1, n'_1) \sim r^2_j(n_2, n'_2) \quad \forall j.$$



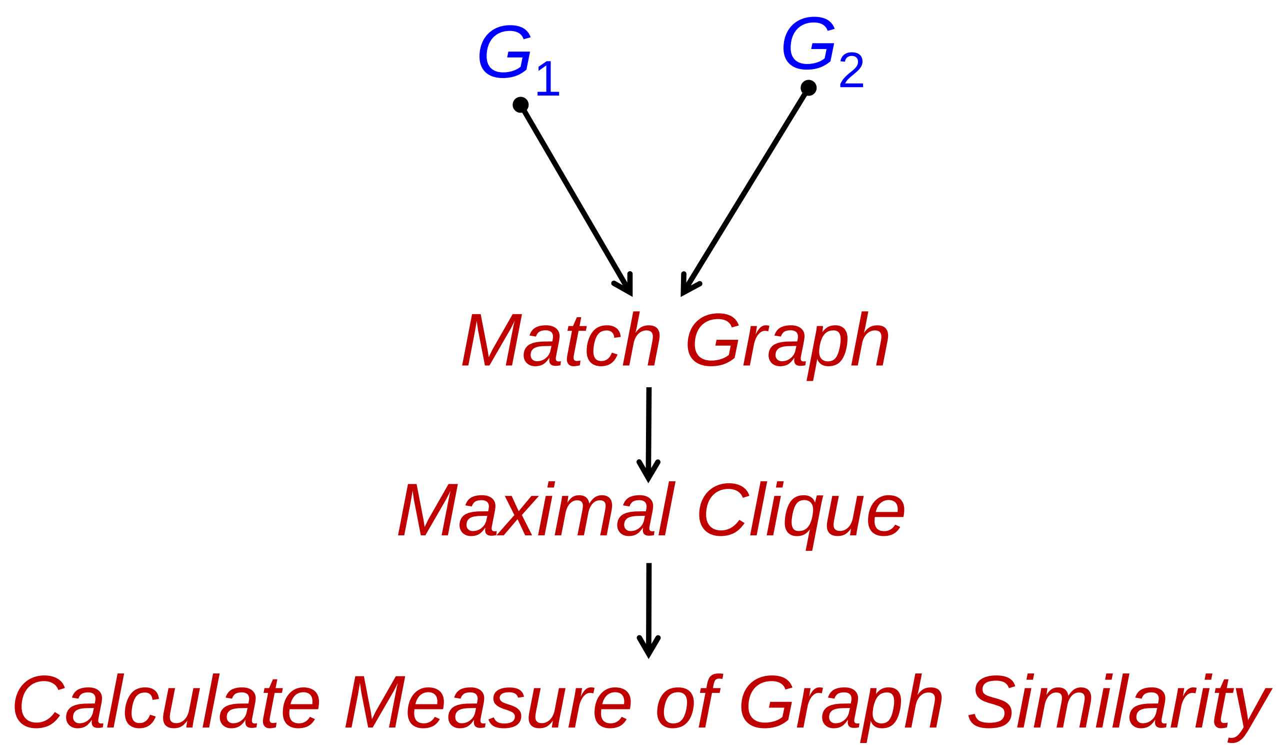
Matching Through Attributed Graph

Assignment and Isomorphism:

- Two attributed graphs G_1 and G_2 are isomorphic if there exists a set of 1:1 assignments of nodes in G_1 to nodes in G_2 such that all assignments are compatible.

Matching Through Attributed Graph

- A strategy for measuring the similarity between two attributed graphs is to find node pairings using the cliques of a match graph.



Matching Through Attributed Graph

- A *match graph* is formed from two graphs G_1 and G_2 as follows:
  - Nodes of the match graph are assignments from G_1 to G_2.
  - An edge in the match graph exists between two nodes if the corresponding assignments are compatible.

Matching Through Attributed Graph

- A *match graph* is formed from two graphs G_1 and G_2 as follows:
  - Nodes of the match graph are assignments from G_1 to G_2.
  - An edge in the match graph exists between two nodes if the corresponding assignments are compatible.
- A *clique* of a graph is a totally connected subgraph.
- A *maximal clique* is not included in any other clique.

Approaches in Matching Through Attributed Graph

- Steps
  - draw *attributed graphs* from the patterns
  - draw *match graphs* from the attributed graph
  - find the *maximum clique* from the match graph
  - calculate the *similarity*

Recursive Procedure to Find Cliques

Input:

- *X*: an initial clique (possibly empty)
- *Y*: the graph

Output:

- The set of all maximal cliques

Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y - \{y\}$)

Else return X

End

Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

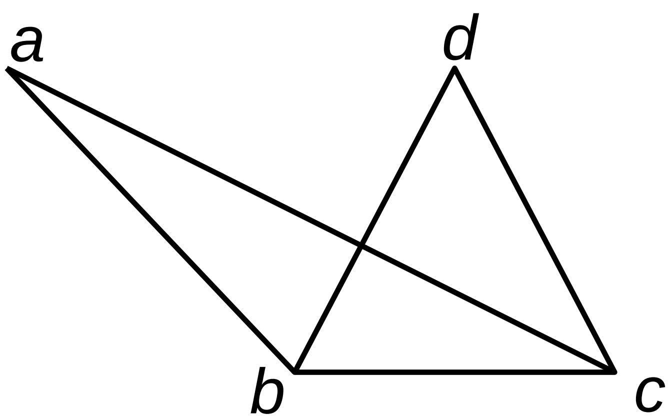
Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y-\{y\}$)

Else return X

End



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

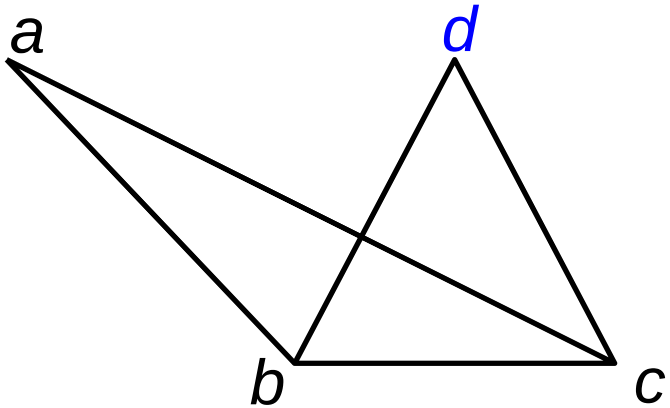
If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) $\cup$ *cliques* ($X, Y - \{y\}$)

Else return X

End

Find *clique* (d, Y)?



Recursive Procedure to Find Cliques

Procedure *clique* (*X*, *Y*)

Form *Y* – *X*;

If a node *y* in *Y* – *X* is connected to all nodes of *X*,

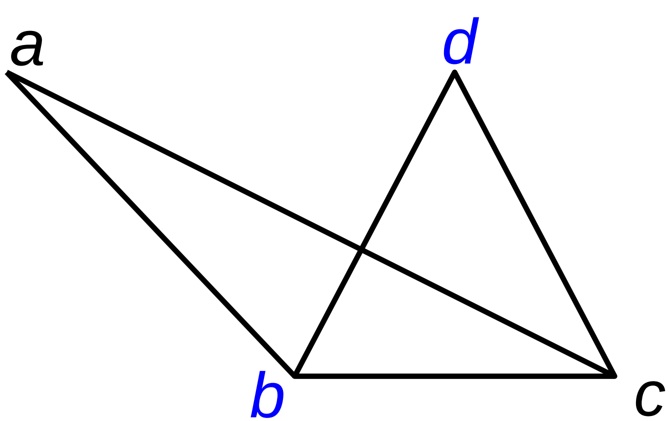
Then return *cliques* (*X* U {*y*},*Y*) U *cliques* (*X*,*Y*-{*y*})

Else return *X*

End

clique (*d*, *Y*) = *clique* ({*d*, *b*}, *Y*) U

clique (*d*, {*a*, *c*, *d*})



Recursive Procedure to Find Cliques

Procedure *clique* (*X*, *Y*)

Form *Y* – *X*;

If a node *y* in *Y* – *X* is connected to all nodes of *X*,

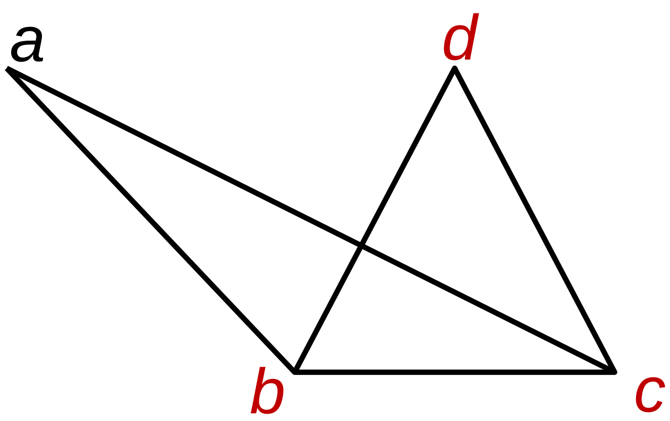
Then return *cliques* (*X* U {*y*},*Y*) U *cliques* (*X*,*Y*-{*y*})

Else return *X*

End

clique (*d*, *Y*) = *clique* ({*d*, *b*}, *Y*) U *clique* (*d*, {*a*, *c*, *d*})

clique ({*d*, *b*}, *Y*) = *clique* ({*d*, *b*, *c*}, *Y*) U *clique* ({*d*, *b*}, {*a*, *b*, *d*})



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

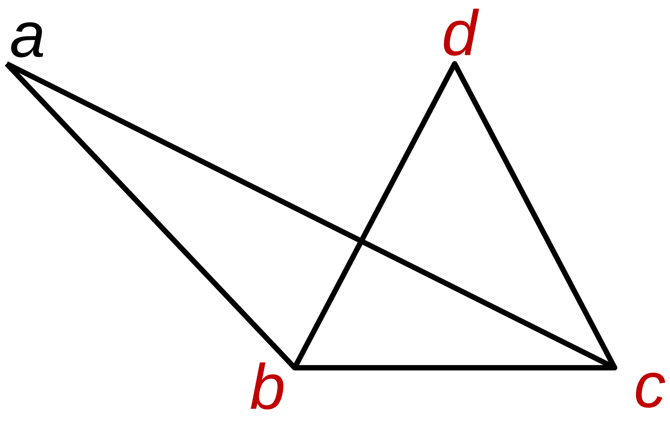
Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y-\{y\}$)

Else return X

End



clique (d, Y) = *clique* ($\{d, b\}$, Y) U *clique* (d, $\{a, c, d\}$)

clique ($\{d, b\}$, Y) = *clique* ($\{d, b, c\}$, Y) U *clique* ($\{d, b\}$, $\{a, b, d\}$)

$\{d, b\}$

Recursive Procedure to Find Cliques

Procedure *clique* (*X*, *Y*)

Form *Y* – *X*;

If a node *y* in *Y* – *X* is connected to all nodes of *X*,

Then return *cliques* (*X* U {*y*},*Y*) U *cliques* (*X*,*Y*-{*y*})

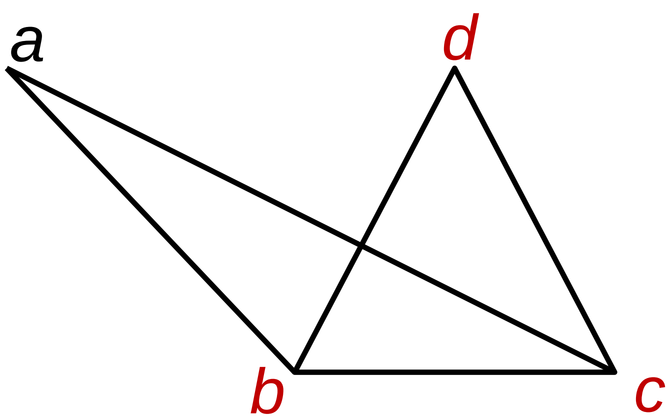
Else return *X*

End

clique (*d*, *Y*) = *clique* ({*d*, *b*}, *Y*) U *clique* (*d*, {*a*, *c*, *d*})

clique ({*d*, *b*}, *Y*) = *clique* ({*d*, *b*, *c*}, *Y*) U {*d*, *b*}

{*d*, *b*, *c*}



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

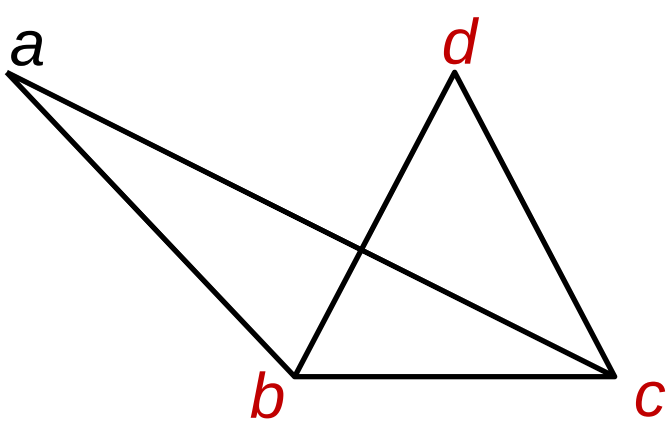
Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y - \{y\}$)

Else return X

End

clique (d, Y) = *clique* ($\{d, b\}$, Y) U *clique* (d, $\{a, c, d\}$)

clique ($\{d, b\}$, Y) = $\{d, b, c\} \cup \{d, b\}$



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y – X$;

If a node y in $Y – X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y-\{y\}$)

Else return X

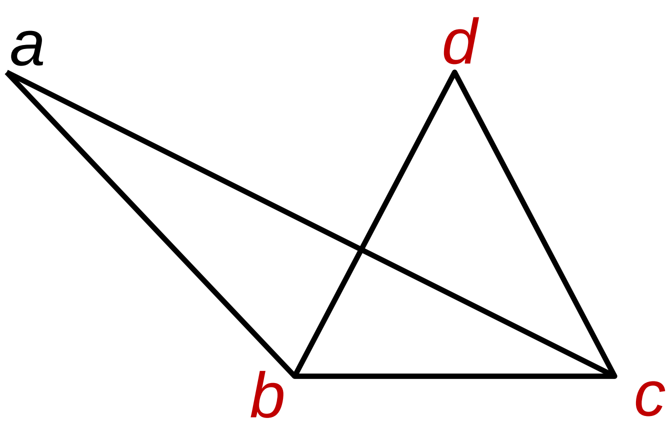
End

clique (d, Y) = *clique* ($\{d, b\}$, Y) U

clique (d, $\{a, c, d\}$)

clique ($\{d, b\}$, Y) = $\{d, b, c\}$ U $\{d, b\}$

= $\{d, b, c\}$



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

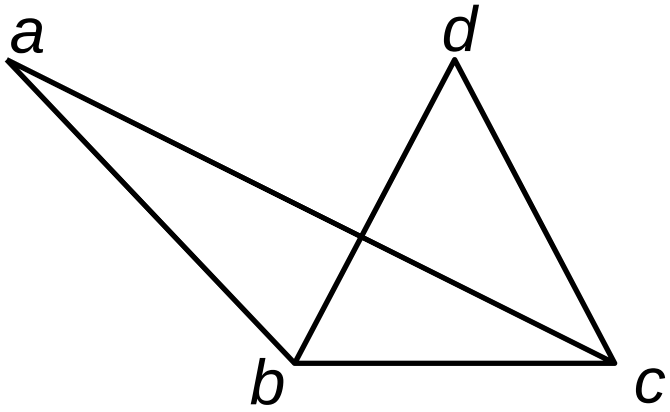
If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y - \{y\}$)

Else return X

End

clique (d, Y) = {d, b, c} U {d, b} U *clique* (d, {a, c, d})



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) $\cup$ *cliques* ($X, Y - \{y\}$)

Else return X

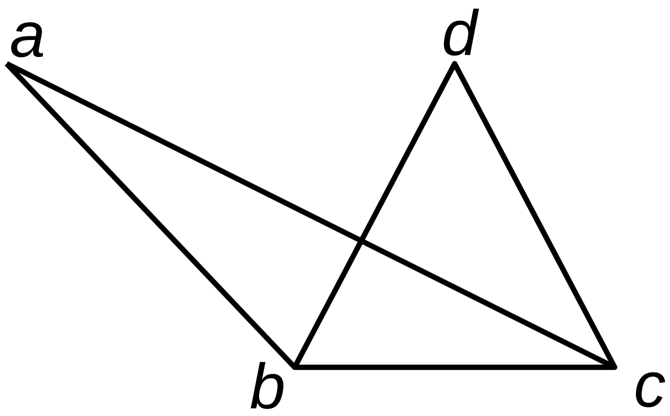
End

clique (d, Y) = $\{d, b, c\} \cup \{d, b\} \cup$ *clique* (d, $\{a, c, d\}$)

clique (d, $\{a, c, d\}$)

= *clique* ($\{d, c\}$, $\{a, c, d\}$)

$\cup$ *clique* (d, $\{a, d\}$)



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y - \{y\}$)

Else return X

End

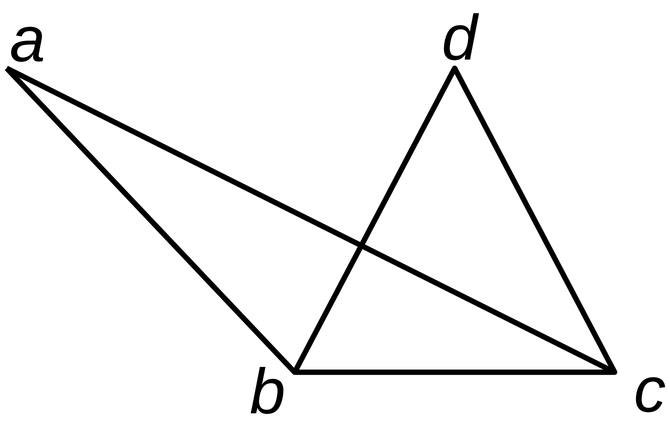
clique (d, Y) = $\{d, b, c\} \cup \{d, b\} \cup$ *clique* (d, $\{a, c, d\}$)

clique (d, $\{a, c, d\}$)

= *clique* ($\{d, c\}$, $\{a, c, d\}$)

U *clique* (d, $\{a, d\}$)

$= \{d, c\} \cup \{d\}$



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

If a node y in $Y - X$ is connected to all nodes of X,

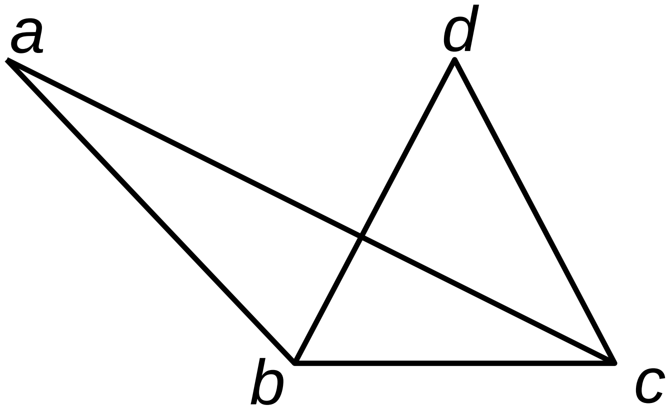
Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y - \{y\}$)

Else return X

End

clique (d, Y) = $\{d, b, c\} \cup \{d, b\}$

$\cup \{d, c\} \cup \{d\}$



Recursive Procedure to Find Cliques

Procedure *clique* (X, Y)

Form $Y - X$;

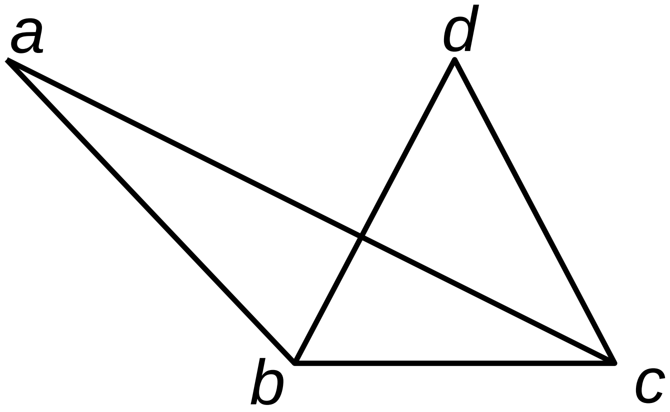
If a node y in $Y - X$ is connected to all nodes of X,

Then return *cliques* ($X \cup \{y\}, Y$) U *cliques* ($X, Y - \{y\}$)

Else return X

End

The maximal clique is = {*d, b, c*}



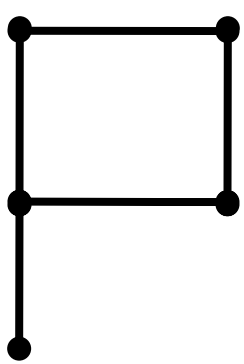
Example: Pattern Matching Using Attributed Graph

P T

Example: Pattern Matching Using Attributed Graph

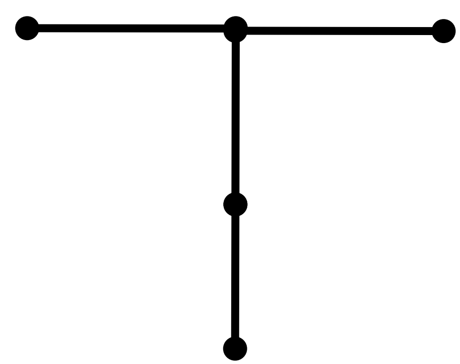
P

Reference



T

Reference



Example: Pattern Matching Using Attributed Graph

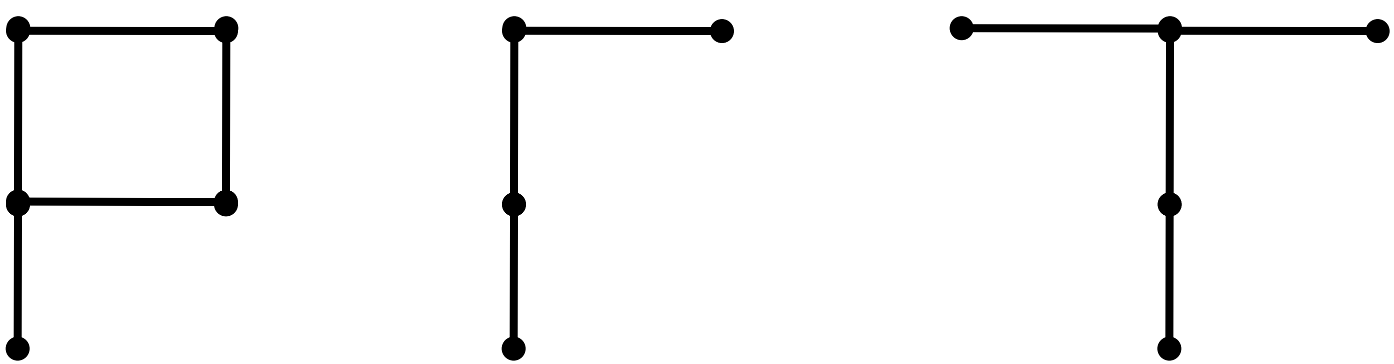
P

T

Reference

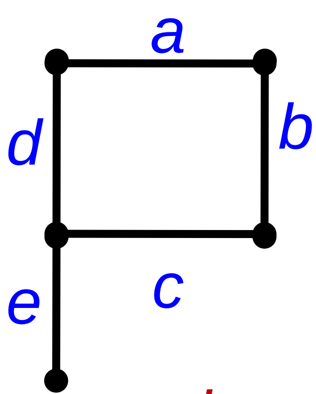
Test

Reference

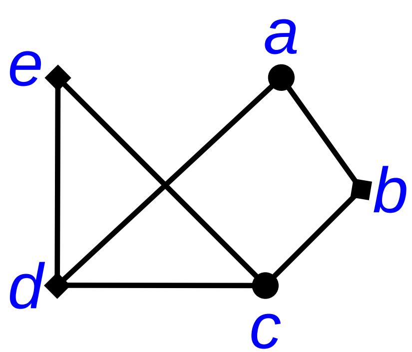


Example: Pattern Matching Using Attributed Graph

Find Attributed Graph for all patterns



I_1



AG_{I_1}

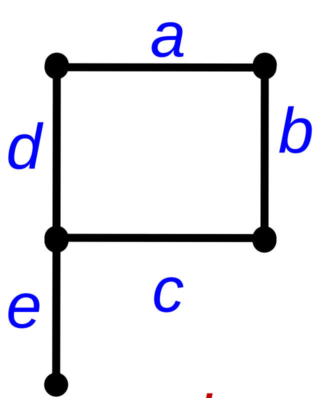
● Horizontal line

◊ Vertical line

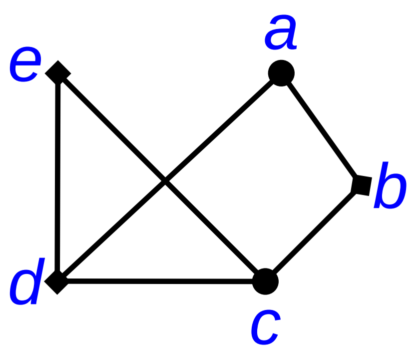
Relation: —— connected

Example: Pattern Matching Using Attributed Graph

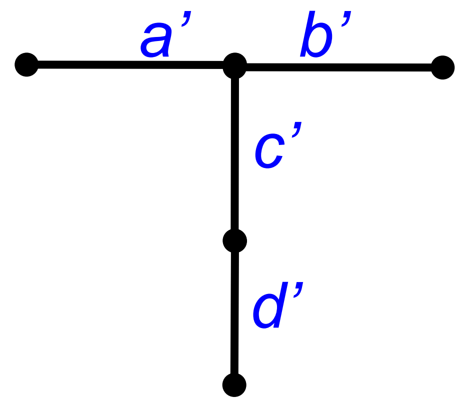
Find Attributed Graph for all patterns



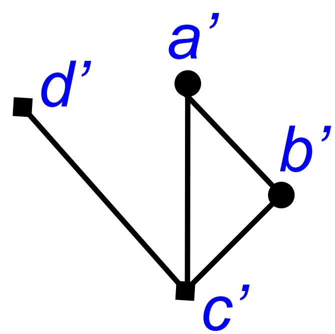
l_1



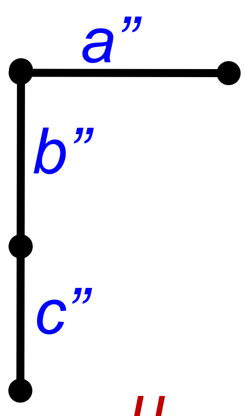
AG_{l_1}



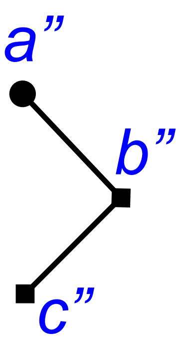
l_2



AG_{l_2}



u



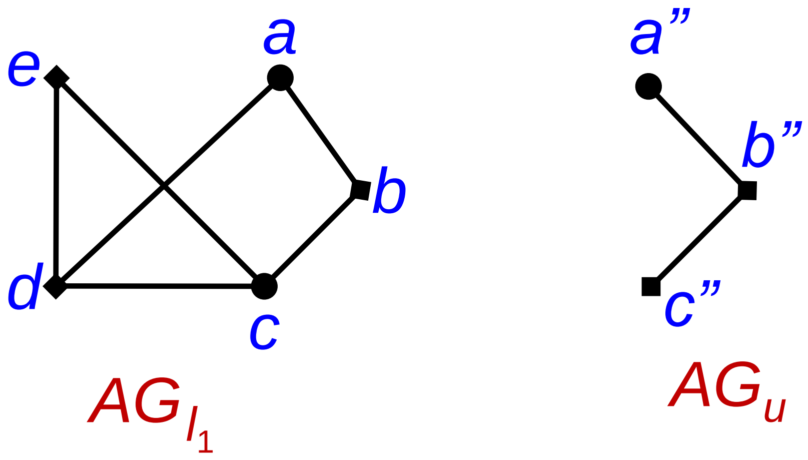
AG_u

- ● Horizontal line
- ◊ Vertical line

Relation: —— connected

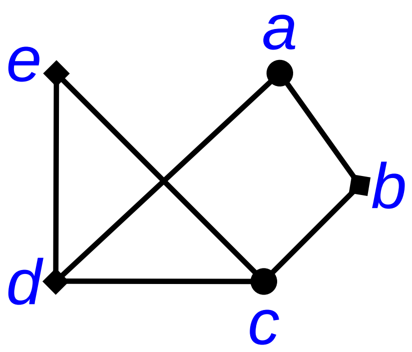
Example: Pattern Matching Using Attributed Graph

Find Matched Graph between AGl_1 and AG_u

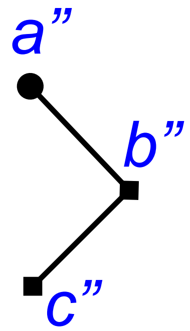


Example: Pattern Matching Using Attributed Graph

Find Matched Graph between AGl_1 and AG_u



AG_{l_1}



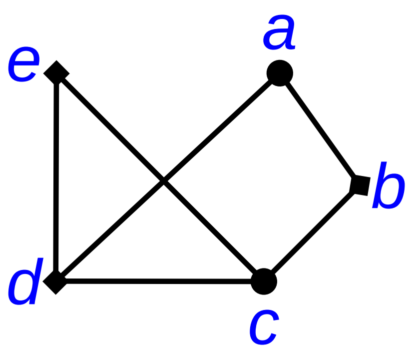
AG_u

Find all assignments between AGl_1 and AG_u:

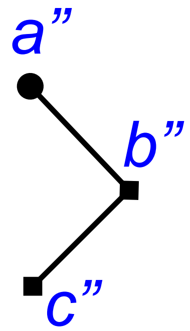
(*a*, *a*”), (*b*, *b*”), (*b*, *c*”), (*c*, *a*”), (*d*, *b*”), (*d*, *c*”), (*e*, *b*”), (*e*, *c*”)

Example: Pattern Matching Using Attributed Graph

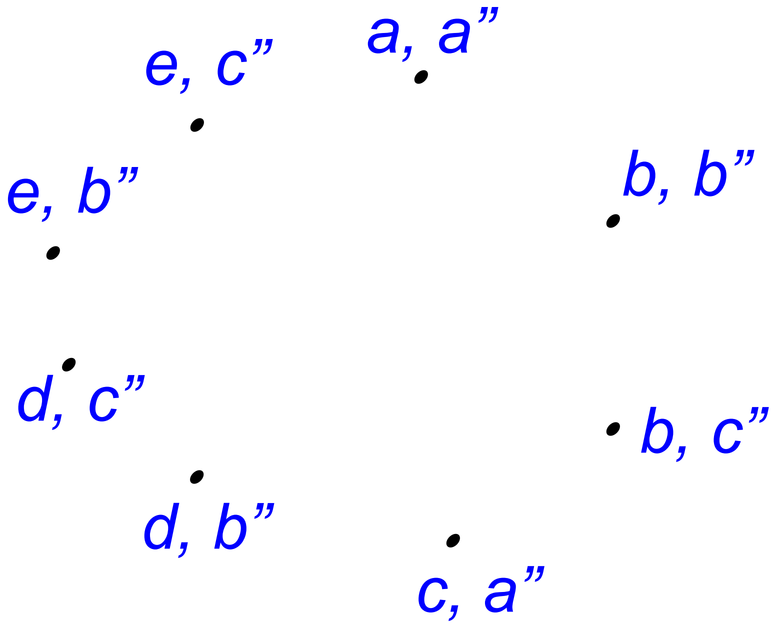
Find Matched Graph between AGl_1 and AG_u



AG_{l_1}



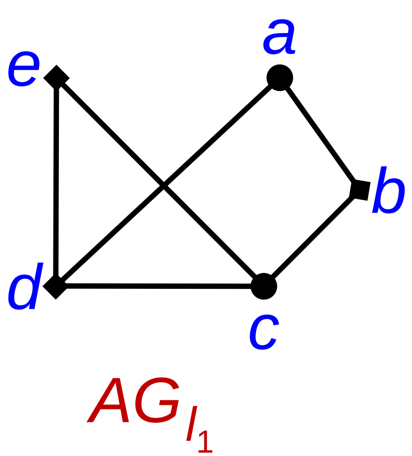
AG_u

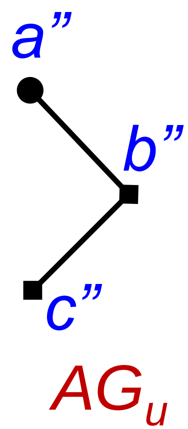


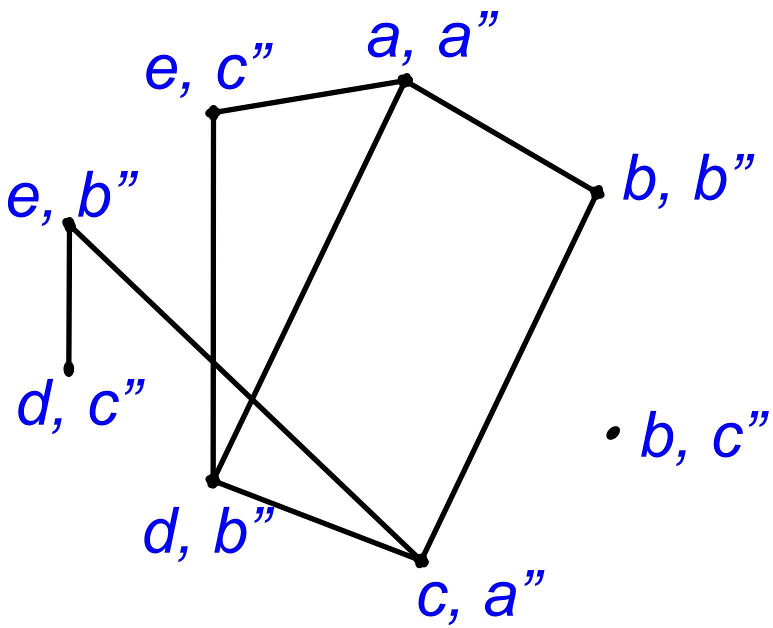
Find all assignments between AGl_1 and AG_u

Example: Pattern Matching Using Attributed Graph

Find Matched Graph between AGl_1 and AG_u



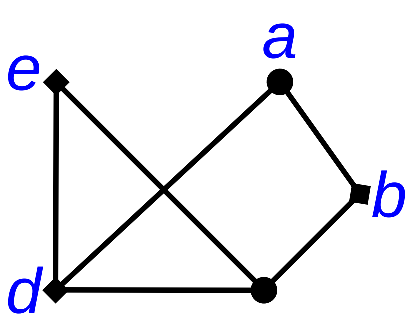




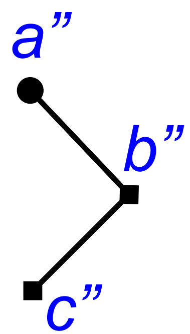
Connect the compatible assignments

Example: Pattern Matching Using Attributed Graph

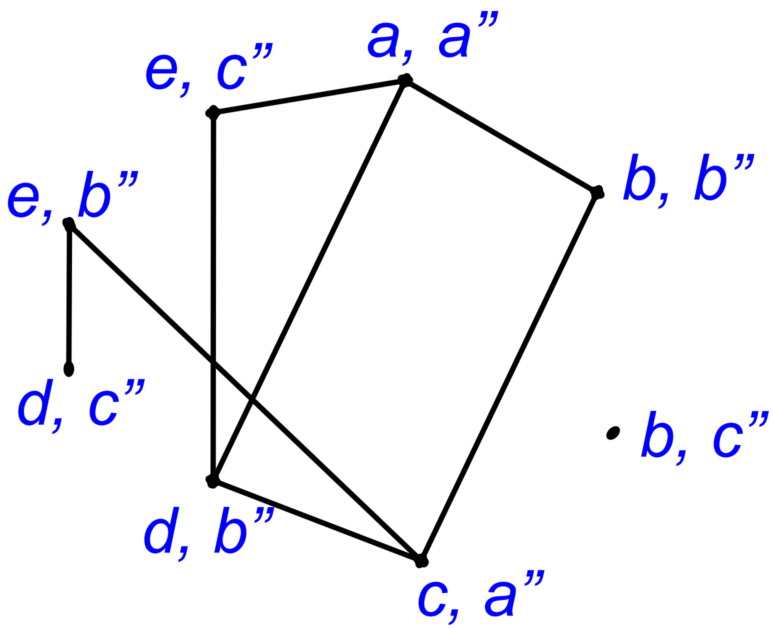
Find Matched Graph between AGl_1 and AG_u

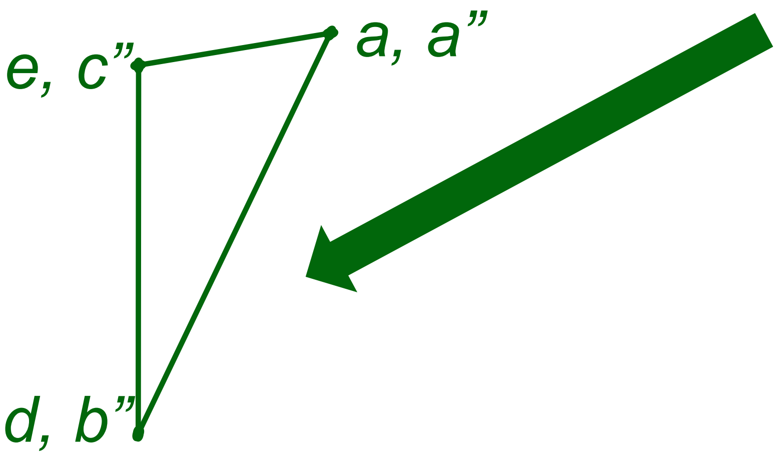


AG_{l_1}



AG_u

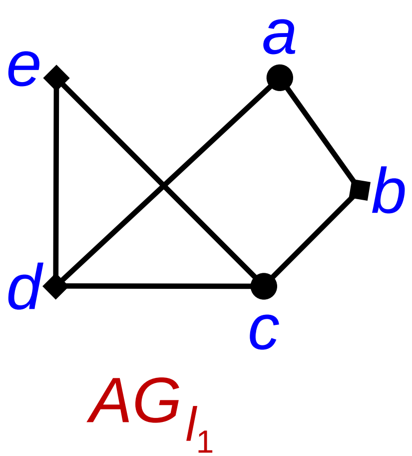


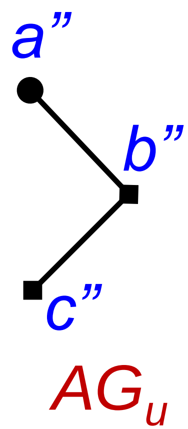


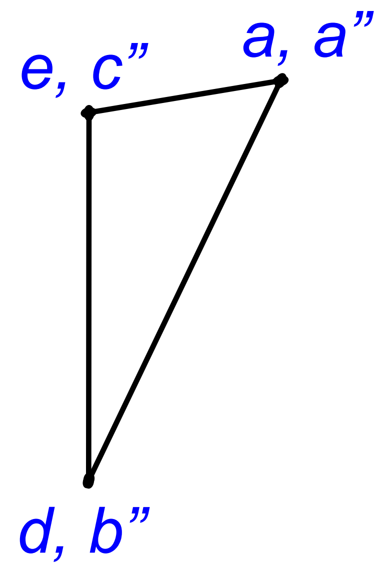
Find the maximal clique:
{(*a*, *a*”),(*d*, *b*”), (*e*, *c*”)}

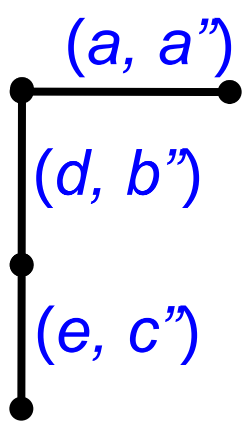
Example: Pattern Matching Using Attributed Graph

Find Matched Graph between AGl_1 and AG_u





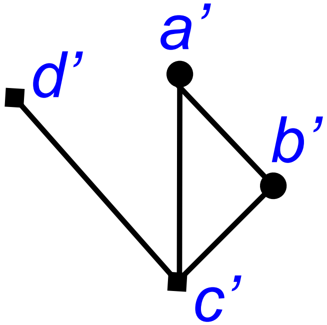




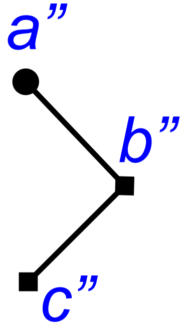
Visual Representation

Example: Pattern Matching Using Attributed Graph

Find Matched Graph between AGl_1 and AG_u



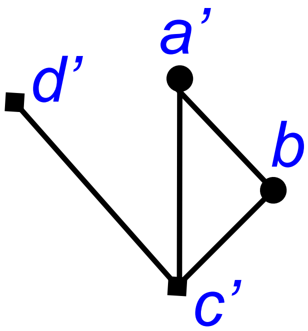
AG_{l_2}



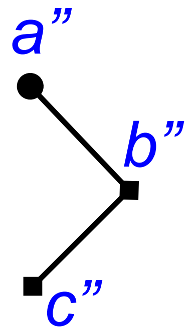
AG_u

Example: Pattern Matching Using Attributed Graph

Find Matched Graph between AGl_1 and AG_u



AG_{l_2}



AG_u

Find all assignments between AGl_2 and AG_u:

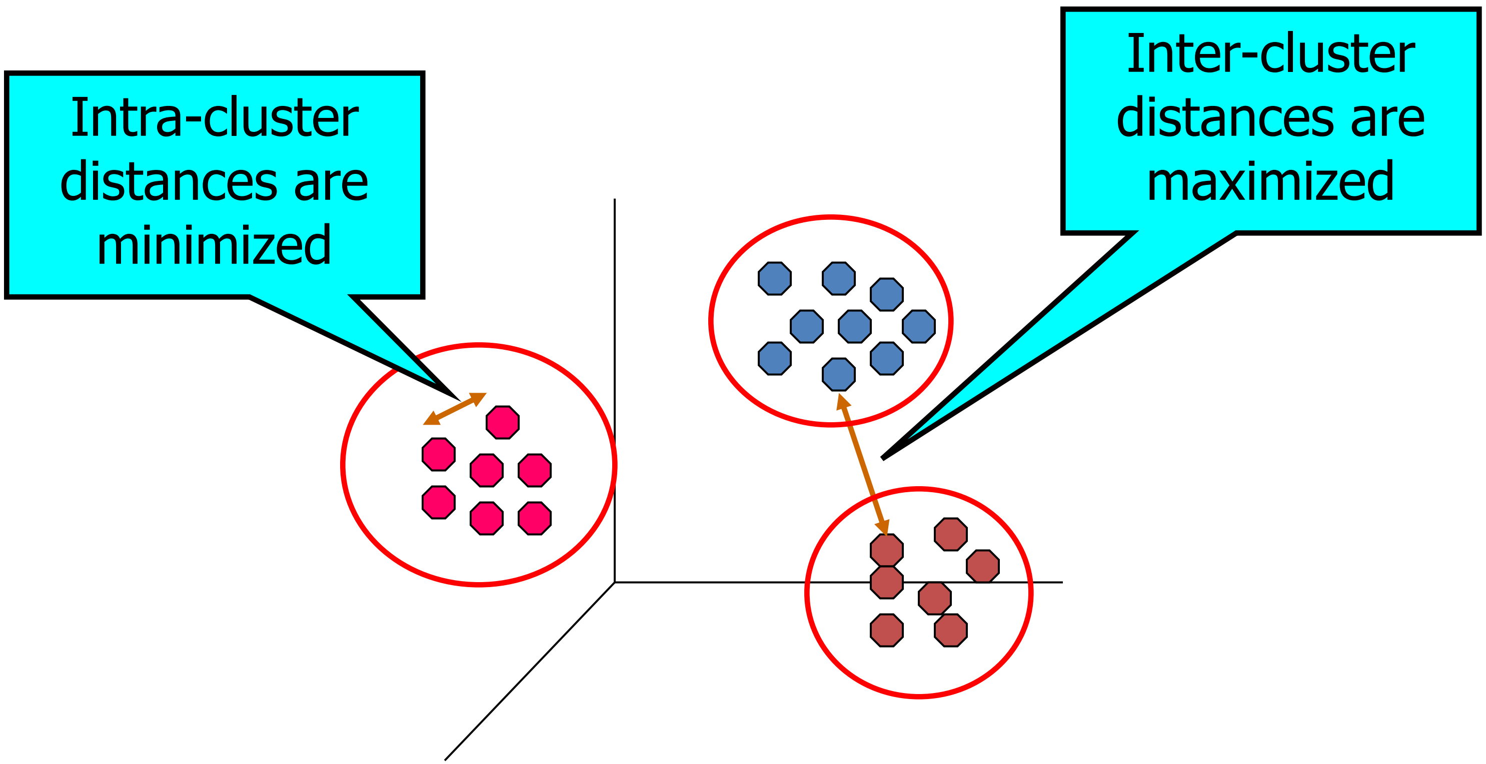
(*a*', *a*"), (*b*', *a*"), (*c*', *b*"), (*c*', *c*"), (*d*', *b*"), (*d*', *c*")

Unsupervised Learning:

Clustering

What is Cluster Analysis?

- Finding groups of objects such that the objects in a group will be similar (or related) to one another and different from (or unrelated to) the objects in other groups



Applications

- **Understanding**
  - Biological taxonomy
  - Group related documents for browsing
  - Group genes and proteins that have similar functionality
  - Group stocks with similar price fluctuations
- **Summarization**
  - Reduce the size of large data sets
- **Data Compression**
  - Vector quantization
- **Finding nearest neighbor**

Applications .. .

- **Hypothesis generation**
  - To infer some hypothesis
- **Hypothesis testing**
  - To verify an existing hypothesis
  - Example: ‘big companies invest overseas’
- **Prediction based on groups**
  - Predict unknown patterns
-
-
-

What is not Cluster Analysis?

- Supervised classification
  - Have class label information
- Simple segmentation
  - Dividing students into different registration groups alphabetically, by last name
- Results of a query
  - Groupings are a result of an external specification

Clustering Basis

- Basic Concepts

  a clustering criterion must first be adopted.

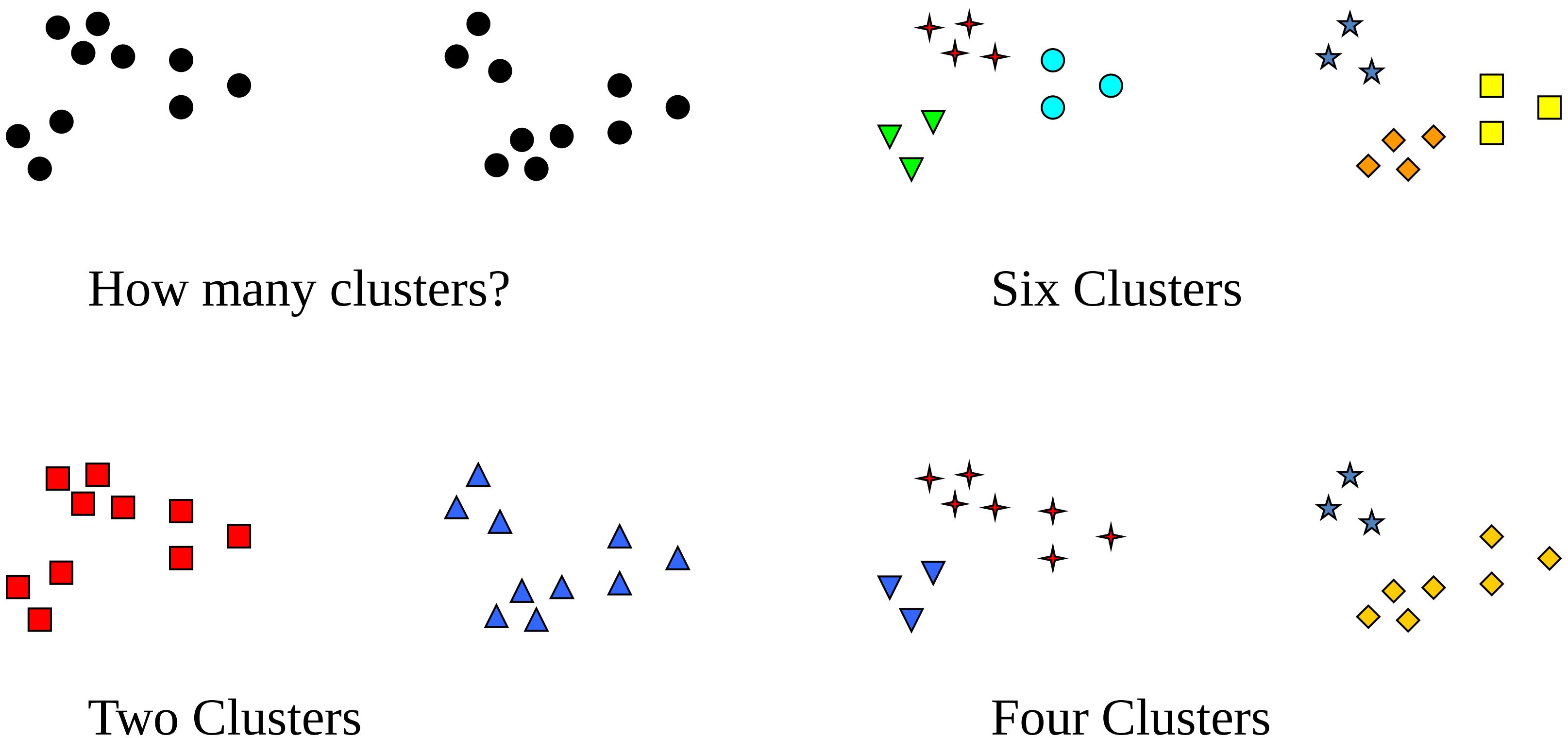
  Different criteria lead to different clusters.

Notion of a Cluster can be Ambiguous

- Depending on the similarity measure, the clustering criterion and the clustering algorithm, different clusters may result. **Subjectivity** is a reality to live with from now on.

Notion of a Cluster can be Ambiguous

– Depending on the similarity measure, the clustering criterion and the clustering algorithm, different clusters may result. **Subjectivity** is a reality to live with from now on.



Notion of a Cluster can be Ambiguous

- Let these animals to be clustered
  - Blue shark, sheep, cat, Dog, Lizard, sparrow, viper, seagull, gold fish, frog, red mullet

– A real example

Blue shark, sheep, cat, dog

Lizard, sparrow, viper, seagull, gold fish, frog, red mullet

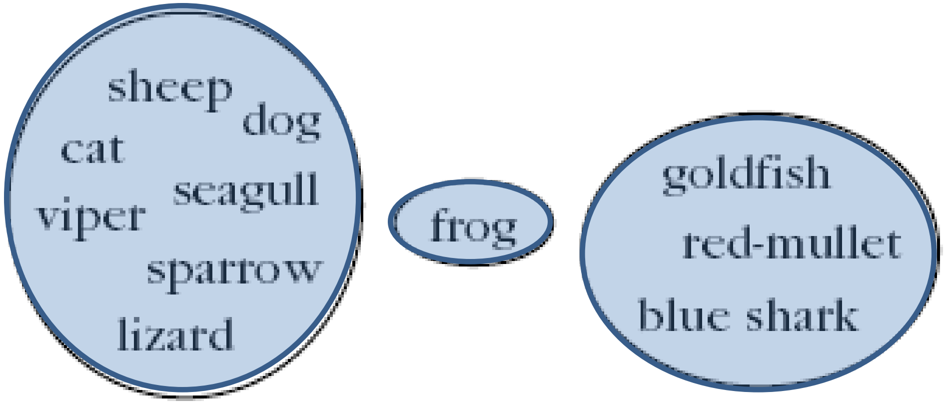
1. Two clusters
2. Clustering criterion: How mammals bear their progeny

Gold fish, red mullet, blue shark

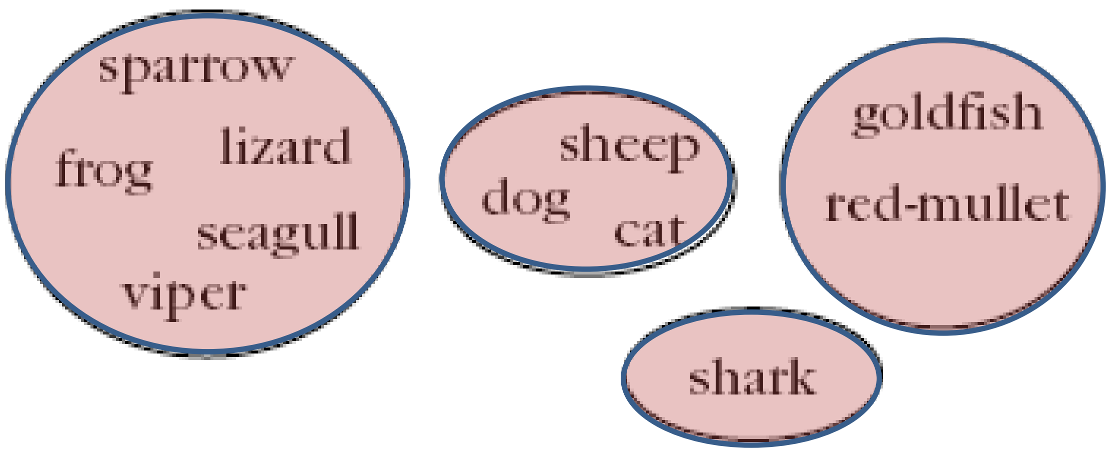
Sheep, sparrow, dog, cat, seagull, lizard, frog, viper

1. Two clusters
2. Clustering criterion: Existence of lungs

– A real example



1. Three clusters
2. Clustering criterion: Their living environment



1. Four clusters
2. Clustering criterion: Bear progeny and Existence of lungs

Clustering Task Stages

- Feature Selection: Information rich features-Parsimony
- Proximity Measure: This quantifies the term similar or dissimilar.
- Clustering Criterion: This consists of a cost function or some type of rules.
- Clustering Algorithm: This consists of the set of **steps** followed to reveal the structure, based on the similarity measure and the adopted criterion.
- Validation of the results.
- Interpretation of the results.

Types of Features

- With respect to their <u>domain</u>
  - Continuous (the domain is a continuous subset of $\Re$).
  - Discrete (the domain is a finite discrete set).
    - *Binary* or *dichotomous* (the domain consists of two possible values).

- With respect to the <u>relative significance of the values they take</u>
  - Nominal (the values code states, e.g., the home district of an individual).
  - Ordinal (the values are meaningfully ordered, e.g., the rating of the services of a hotel (poor, good, very good, excellent)).
  - Interval-scaled (the difference of two values is meaningful but their ratio is meaningless, e.g., temperature).
  - Ratio-scaled (the ratio of two values is meaningful, e.g., weight).

Clustering Definitions

- Hard Clustering: Each point belongs to a single cluster
  - Let $X = \{\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N\}$
  - An m-clustering R of X, is defined as the partition of X into m sets (clusters), $C_1, C_2, \ldots, C_m$, so that
    - $C_i \neq \emptyset, i = 1,2,\ldots,m$
    - $\bigcup_{i=1}^{m} C_i = X$
    - $C_i \cap C_j = \emptyset, \; i \neq j, \; i, j = 1,2,\ldots,m$
- In addition, data in C_i are more similar to each other and less similar to the data in the rest of the clusters.

Clustering Definitions

– Fuzzy clustering: Each point belongs to all clusters up to some degree.

A fuzzy clustering of X into m clusters is characterized by m functions

- u_j is the representative of *j*th cluster
- $u_j : \underline{x} \rightarrow [0,1], \quad j = 1,2,\ldots,m$

Clustering Definitions

These are known as membership functions.
Thus, each $\underline{x}_i$ belongs to any cluster "up to some degree", depending on the value of

$$u_j(\underline{x}_i), \;\; j = 1,2,\ldots,m$$

$$\boxed{\begin{array}{l} u_j(\underline{x}_i) \text{ close to } 1 \Rightarrow \text{high grade of} \\ \text{membership of } \underline{x}_i \text{ to cluster } j. \\ \\ u_j(\underline{x}_i) \text{ close to } 0 \Rightarrow \\ \text{low grade of membership.} \end{array}}$$

Clustering Definitions

– Fuzzy clustering: Each point belongs to all clusters up to some degree.

A fuzzy clustering of X into m clusters is characterized by m functions

- u_j is the representative of jth cluster
- $u_j : \underline{x} \rightarrow [0,1], \quad j = 1,2,\ldots,m$
- $\sum_{j=1}^{m} u_j(\underline{x}_i) = 1, \; i = 1,2,\ldots,N$
- $0 < \sum_{i=1}^{N} u_j(\underline{x}_i) < N, \;\; j = 1,2,\ldots,m$

Proximity Measures

- *Between vectors*
  - Dissimilarity measure (between vectors of X) is a function

$$d: \ X \times X \longrightarrow \Re$$

with the following properties

- $\exists d_0 \in \Re: \ -\infty < d_0 \leq d(\underline{x}, \underline{y}) < +\infty, \ \ \forall \underline{x}, \underline{y} \in X$
- $d(\underline{x}, \underline{x}) = d_0, \ \ \forall \underline{x} \in X$
- $d(\underline{x}, \underline{y}) = d(\underline{y}, \underline{x}), \ \ \forall \underline{x}, \underline{y} \in X$

Proximity Measures

If, in addition

- $d(\underline{x},\underline{y}) = d_0 \ if \ \ and \ \ only \ if \ \ \underline{x} = \underline{y}$
- $d(\underline{x},\underline{z}) \leq d(\underline{x},\underline{y}) + d(\underline{y},\underline{z}), \ \ \forall \underline{x},\underline{y},\underline{z} \in X$

  (triangular inequality)

d is called a metric dissimilarity measure.

Proximity Measures

– Similarity measure (between vectors of X) is a function

$$s: \quad X \times X \longrightarrow \Re$$

with the following properties

- $\exists s_0 \in \Re: \quad -\infty < s(\underline{x}, \underline{y}) \le s_0 < +\infty, \quad \forall \underline{x}, \underline{y} \in X$
- $s(\underline{x}, \underline{x}) = s_0, \quad \forall \underline{x} \in X$
- $s(\underline{x}, \underline{y}) = s(\underline{y}, \underline{x}), \quad \forall \underline{x}, \underline{y} \in X$

Proximity Measures

If, in addition

- $s(\underline{x}, \underline{y}) = s_0 \ if \ \ and \ only \ if \ \ \underline{x} = \underline{y}$
- $s(\underline{x}, \underline{y})s(\underline{y}, \underline{z}) \leq [s(\underline{x}, \underline{y}) + s(\underline{y}, \underline{z})]s(\underline{x}, \underline{z}), \ \ \forall \underline{x}, \underline{y}, \underline{z} \in X$

s is called a metric similarity measure.

Proximity Measures

- ***Between sets***

  Let $D_i \subset X$, $i = 1, \ldots, k$ and $U = \{D_1, \ldots, D_k\}$

  A proximity measure $\wp$ on U is a function

$$\wp : U \times U \longrightarrow \Re$$

Proximity Measures Between Points/Vectors

- Real-valued vectors
  - Dissimilarity measures (DMs)

    - *Weighted l_p metric DMs*

$$d_p(\underline{x}, \underline{y}) = \left(\sum_{i=1}^{l} w_i \,|\, x_i - y_i \,|^p\right)^{1/p}$$

Interesting instances are obtained for

– p=1 (*weighted Manhattan* norm)

– p=2 (*weighted Euclidean* norm)

– $p=\infty$ ($d_\infty(\underline{x},\underline{y})=\max_{1\le i\le l} w_i|x_i-y_i|$)

Proximity Measures Between Vectors

– Similarity measures

- *Inner product*

$$s_{inner}(\underline{x}, \underline{y}) = \underline{x}^T \underline{y} = \sum_{i=1}^{l} x_i y_i$$

- *Tanimoto measure*

$$s_T(\underline{x}, \underline{y}) = \frac{\underline{x}^T \underline{y}}{\|\underline{x}\|^2 + \|\underline{y}\|^2 - \underline{x}^T \underline{y}}$$

Proximity Measures Between Discrete-Valued Vectors

- Let $F=\{0,1,\ldots,k\text{-}1\}$ be a set of symbols and $X=\{\underline{x}_1,\ldots,\underline{x}_N\} \subset F^l$
- Let $A(\underline{x},\underline{y})=[a_{ij}]$, $i, j=0,1,\ldots,k\text{-}1$, where a_{ij} is the number of places where $\underline{x}$ has the i-th symbol and $\underline{y}$ has the j-th symbol.

Example: *l*=6, *k* =3

$$\boldsymbol{x} = [0,1,2,1,2,1]^T$$
$$\boldsymbol{y} = [1,0,2,1,0,1]^T$$

$$A(\boldsymbol{x},\boldsymbol{y}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

NOTE: $$\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} a_{ij} = l$$

Proximity Measures Between Discrete-Valued Vectors

- Several proximity measures can be expressed as combinations of the elements of $A(\underline{x},\underline{y})$.
  - Dissimilarity measures:
    - The Hamming distance (number of places where $\underline{x}$ and $\underline{y}$ differ)

$$d_H(\underline{x},\underline{y}) = \sum_{i=0}^{k-1} \sum_{\substack{j=0 \\ j \neq i}}^{k-1} a_{ij}$$

Proximity Measures Between Discrete-Valued Vectors

- Several proximity measures can be expressed as combinations of the elements of $A(\underline{x},\underline{y})$.

  - Dissimilarity measures:
    - The Hamming distance (number of places where $\underline{x}$ and $\underline{y}$ differ)

$$d_H(\underline{x},\underline{y}) = \sum_{i=0}^{k-1} \sum_{\substack{j=0 \\ j \neq i}}^{k-1} a_{ij}$$

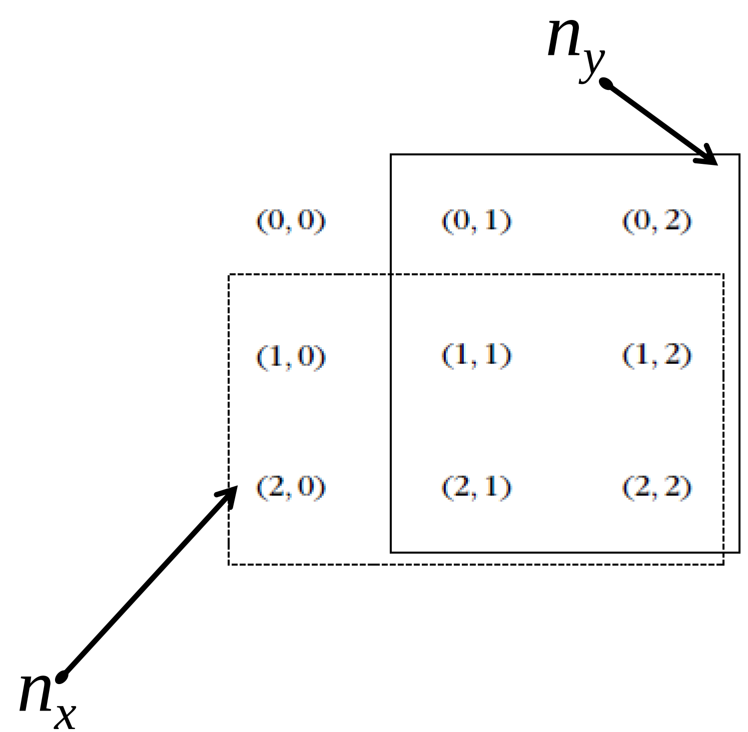
$$\boldsymbol{x} = [0, 1, 2, 1, 2, 1]^T$$
$$\boldsymbol{y} = [1, 0, 2, 1, 0, 1]^T$$

$$A(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Proximity Measures Between Discrete-Valued Vectors

– Similarity measures:

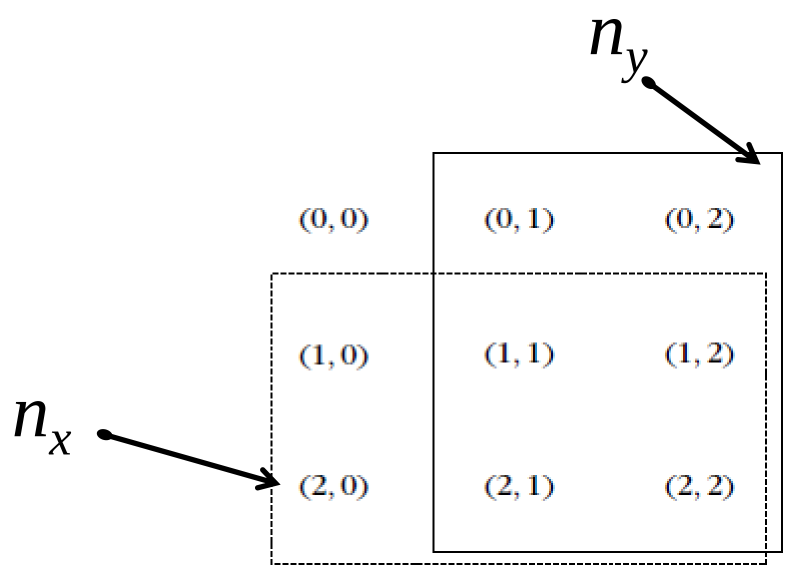
$$A(x, y) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



Proximity Measures Between Discrete-Valued Vectors

– Similarity measures:

$$A(\mathbf{x},\mathbf{y}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



- Tanimoto measure :

$$s_T(\underline{x},\underline{y}) = \frac{\sum_{i=1}^{k-1} a_{ii}}{n_x + n_y - \sum_{i=1}^{k-1}\sum_{j=1}^{k-1} a_{ij}}$$

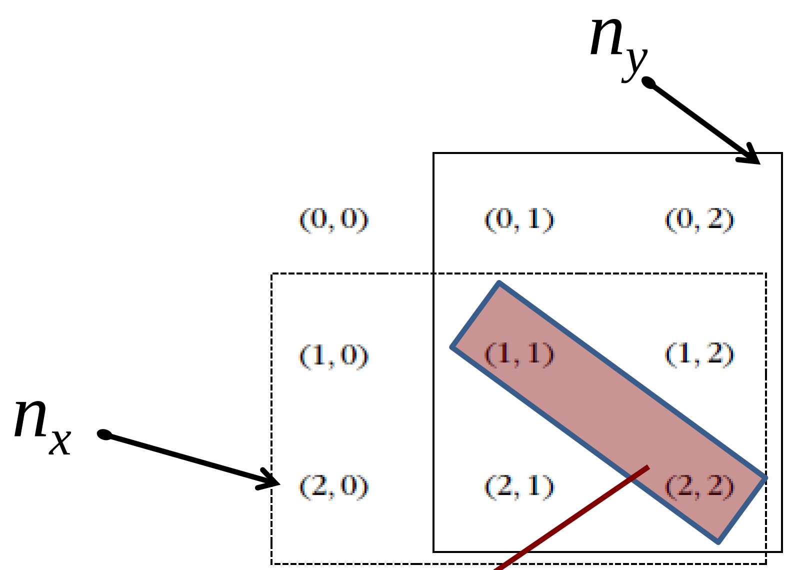
where

$$n_x = \sum_{i=1}^{k-1}\sum_{j=0}^{k-1} a_{ij}, \quad n_y = \sum_{i=0}^{k-1}\sum_{j=1}^{k-1} a_{ij},$$

Proximity Measures Between Discrete-Valued Vectors

– Similarity measures:

$$A(\boldsymbol{x}, \boldsymbol{y}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$



- Tanimoto measure :

$$s_T(\underline{x}, \underline{y}) = \frac{\sum_{i=1}^{k-1} a_{ii}}{n_x + n_y - \sum_{i=1}^{k-1} \sum_{j=1}^{k-1} a_{ij}}$$

where

$$n_x = \sum_{i=1}^{k-1} \sum_{j=0}^{k-1} a_{ij}, \quad n_y = \sum_{i=0}^{k-1} \sum_{j=1}^{k-1} a_{ij},$$