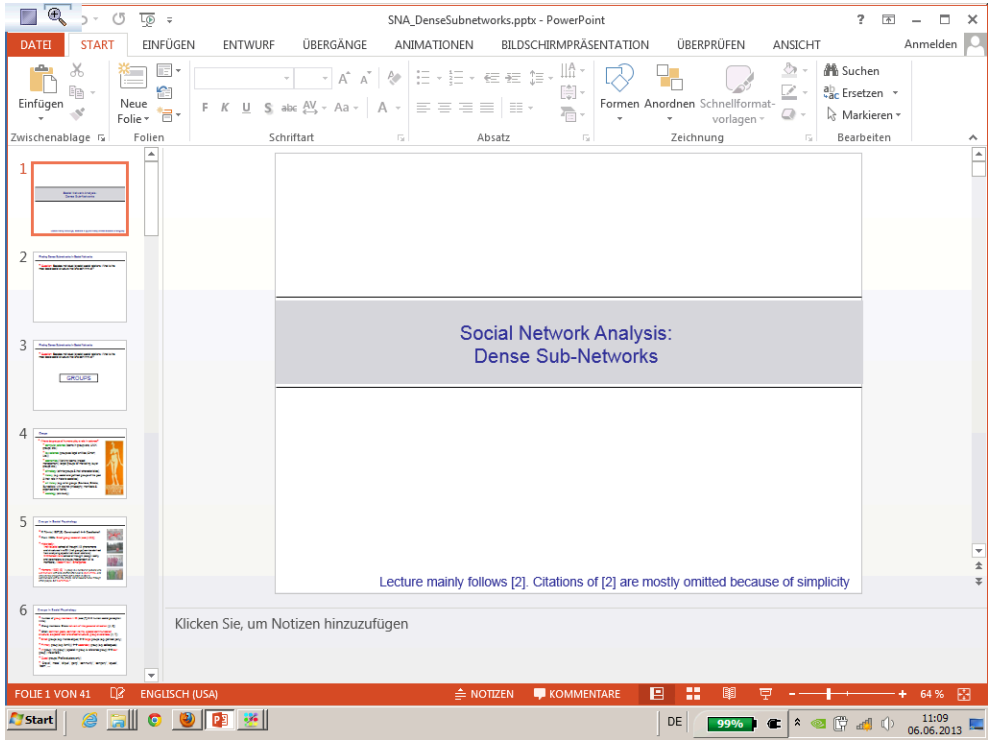
Script **generated by TTT**

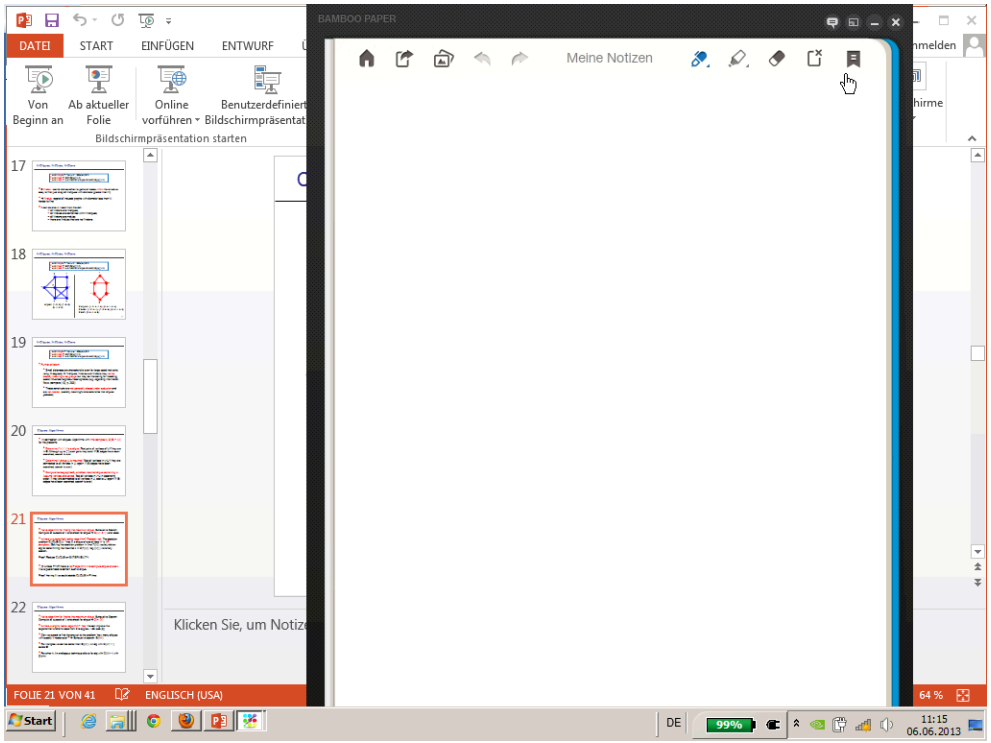
Title: profile1 (06.06.2013)

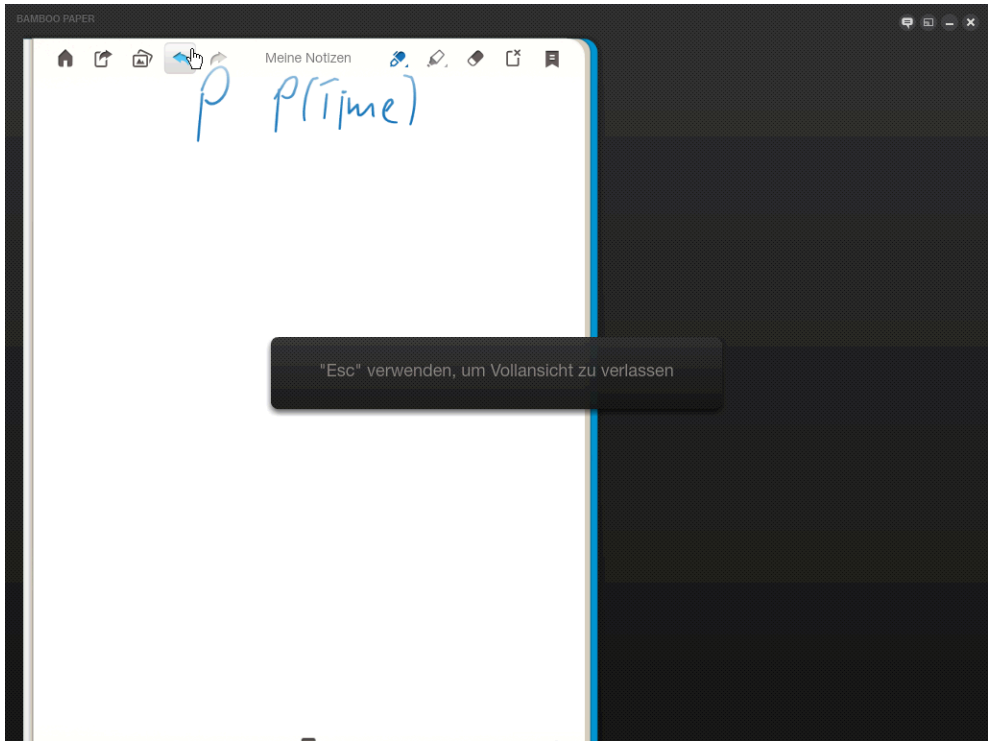
Date: Thu Jun 06 11:09:39 CEST 2013

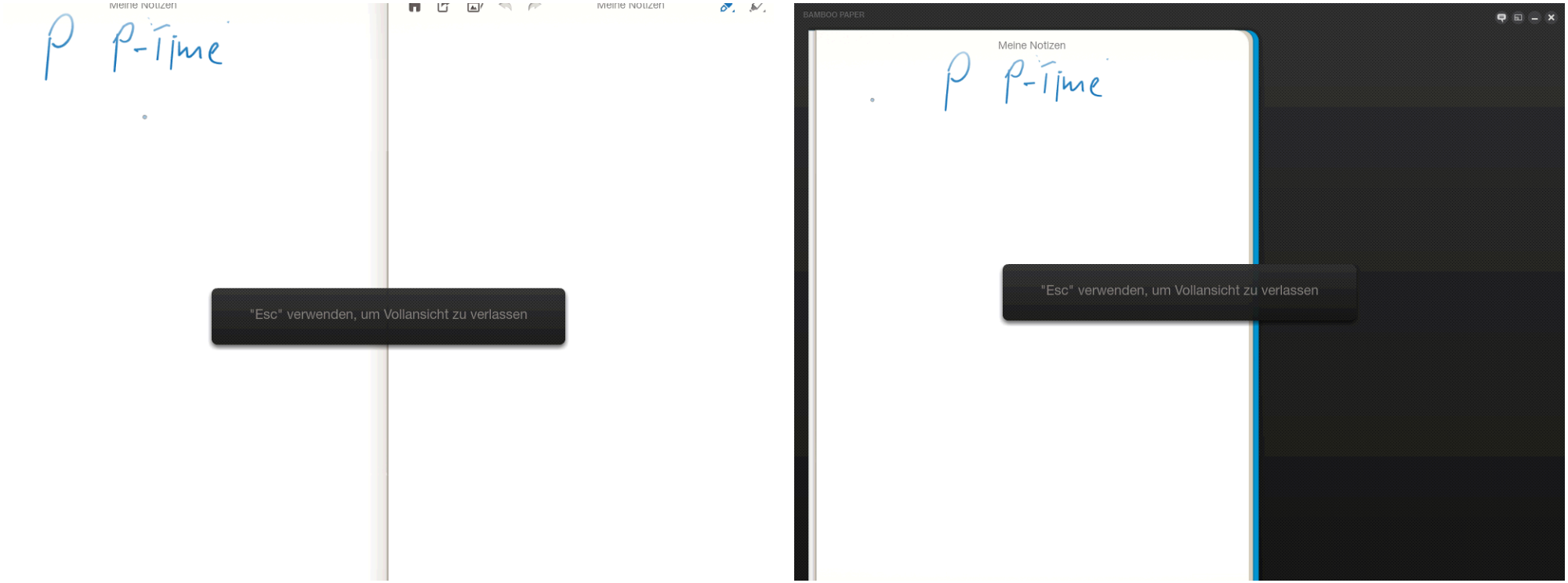
Duration: 93:54 min

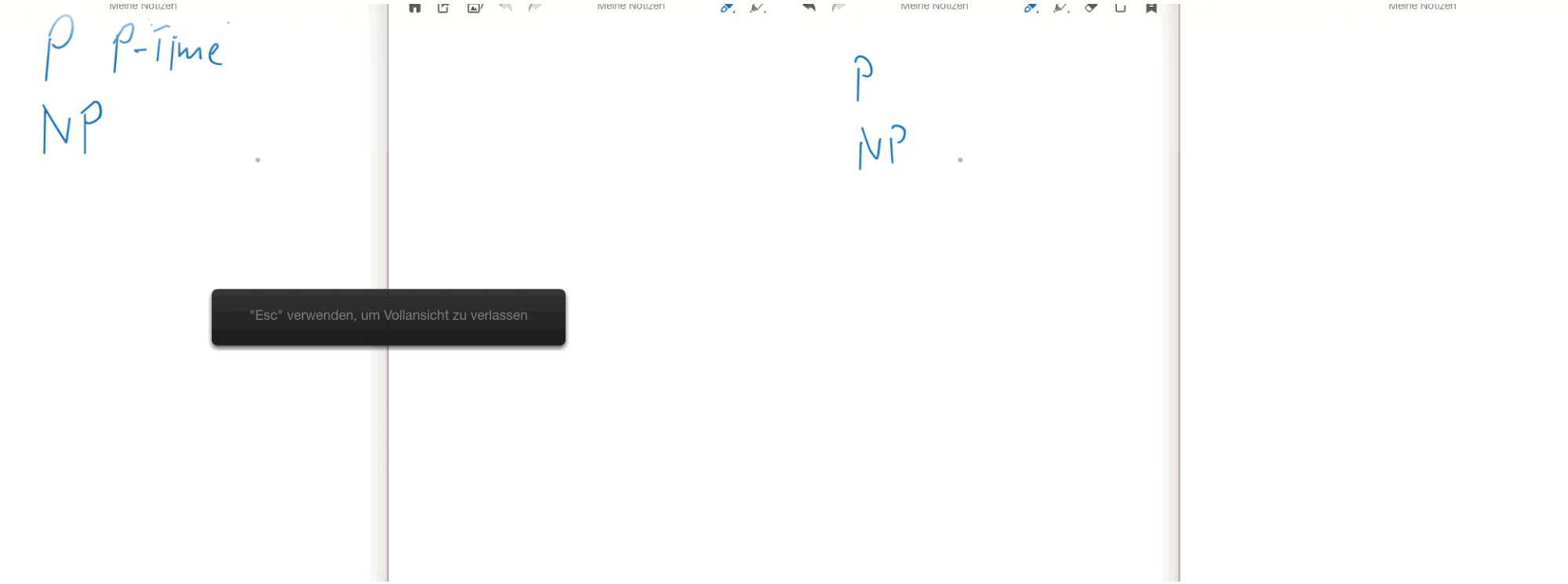
Pages: 81








Meine Notizen
P P-Time
"Esc" verwenden, um Vollansicht zu verlassen
BAMBOO PAPER
Meine Notizen
P P-Time
"Esc" verwenden, um Vollansicht zu verlassen


Meine Notizen
P P-Time
NP
"Esc" verwenden, um Vollansicht zu verlassen
Meine Notizen
P
NP
Meine Notizen

Meine Notizen

P

NP

P=NP?

Meine Notizen

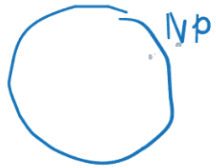
Meine Notizen

P

NP

P=NP?

P≠NP



Meine Notizen

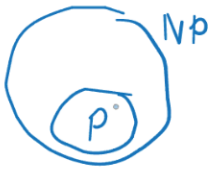
Meine Notizen

P

NP

P=NP?

P≠NP



Meine Notizen

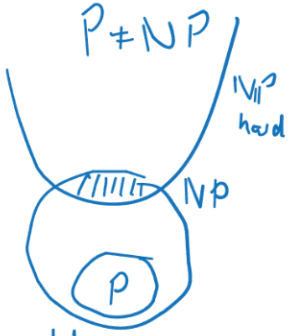
Meine Notizen

P

NP

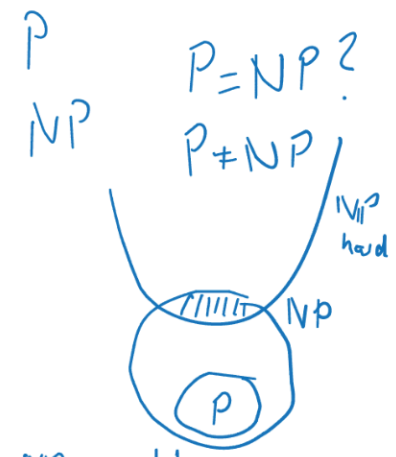
P=NP?

P≠NP



NP complete:

- NP hard
- NP



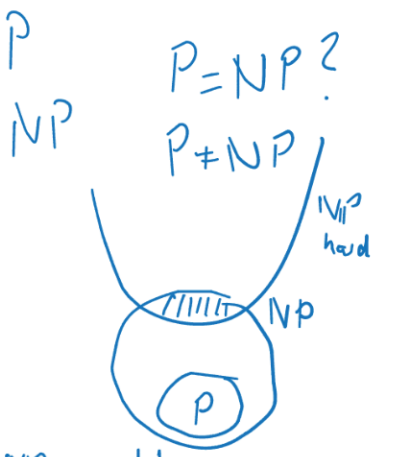
NP complete:

→ NP hard

∈ NP

NP

Meine Notizen



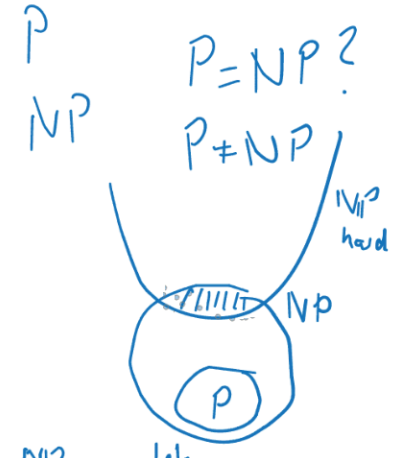
NP complete:

→ NP hard

∈ NP

L NP:

Meine Notizen



NP complete:

→ NP hard

∈ NP

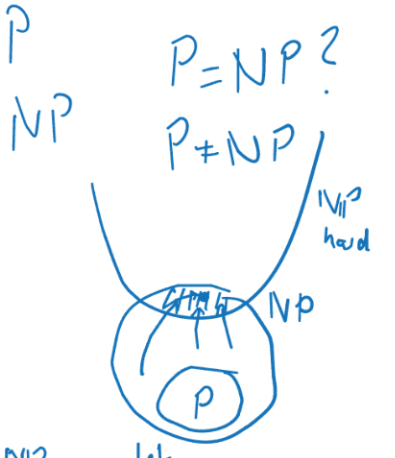
$L \in NP$ hard: $\forall L' \in NP : L' \leq_p L$

Meine Notizen

$\leq_p$: $\exists f \in P_{time}$

$\forall w \in L' : f(w) \in L$

Meine Notizen



NP complete:

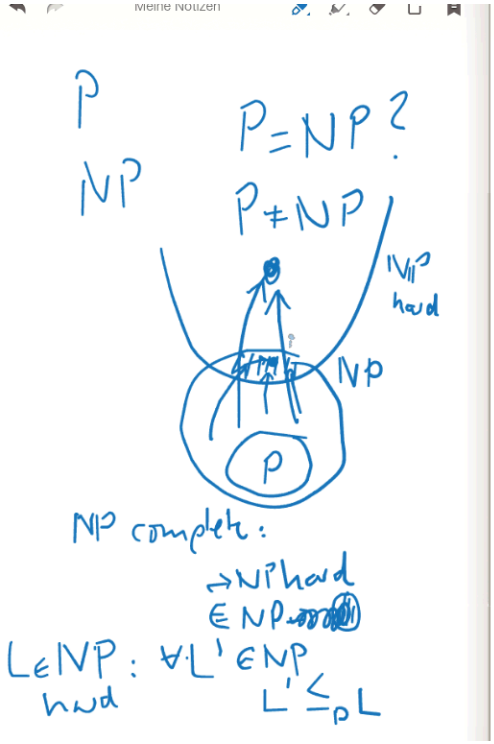
→ NP hard

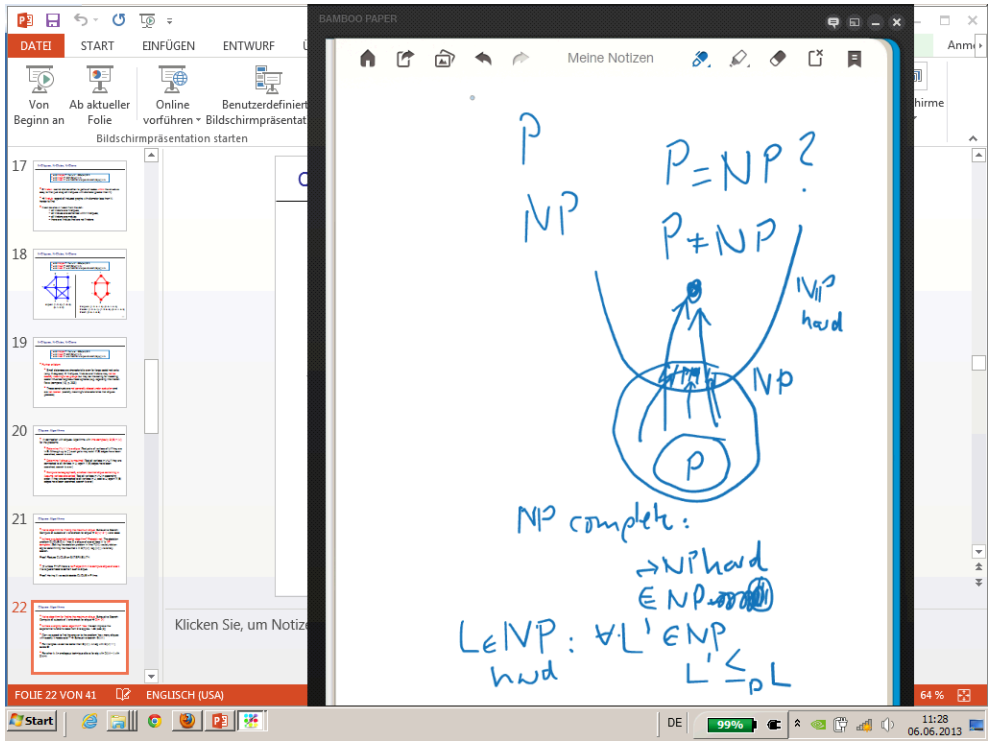
∈ NP

$L \in NP$ hard: $\forall L' \in NP : L' \leq_p L$

$\leq_p$: $\exists f \in P_{time}$

$\forall w \in L' : f(w) \in L$



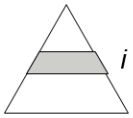


Cliques: Algorithms

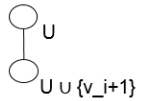
- Naive algorithm for finding the maximum clique: Exhaustive Search: Compute all subsets of V and check for clique → $O(n^2 2^n)$
- Is there a slightly better algorithm? Yes: We can improve the exponential function's base from 2 to approx. 1.38 (see [2])
- Can we expect to find the answer to the problem "how many cliques with exactly k nodes exist"? → Exhaustive search: $\Theta(|V|^k)$
- For triangles we can be better than $\Theta(|V|^3)$: An alg with $O(|V|^{2.376})$ exists ☺
- For other k: An analogous technique allows for alg with $O(|V|^{\beta(k)})$ with $\beta(k)<k$

Cliques: Enumeration

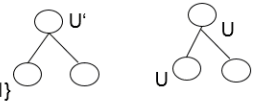
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in G[v_1,v_2,…,v_i}]
- level i+1: determine children of node U at level i: two cases :



(1) v_i+1 adjacent to all nodes in U (U⊆N(v_i+1):
→ U ∪ {v_i+1} is maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ U ∪ {v_i+1} is only child of U

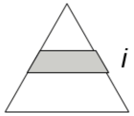


(2) ∃ vertex in U not adjacent to v_i+1 (U⊈N(v_i+1):
→ U itself is a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ if (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is maximal it is also a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U' at level i

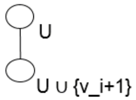


Cliques: Enumeration

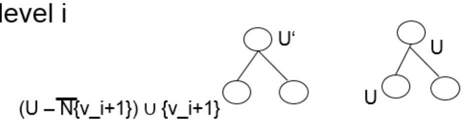
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in $G[v_1,v_2,\dots,v_i]$
- level i+1: determine children of node U at level i: two cases :



(1) v_{i+1} adjacent to all nodes in U ($U \subseteq N(v_{i+1})$):
→ $U \cup \{v_{i+1}\}$ is maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $U \cup \{v_{i+1}\}$ is only child of U

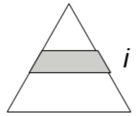


(2) ∃ vertex in U not adjacent to v_{i+1} ($U \not\subseteq N(v_{i+1})$):
→ U itself is a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ if $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is maximal it is also a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i

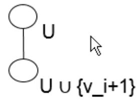


Cliques: Enumeration

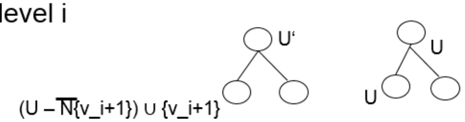
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in $G[v_1,v_2,\dots,v_i]$
- level i+1: determine children of node U at level i: two cases :



(1) v_{i+1} adjacent to all nodes in U ($U \subseteq N(v_{i+1})$):
→ $U \cup \{v_{i+1}\}$ is maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $U \cup \{v_{i+1}\}$ is only child of U

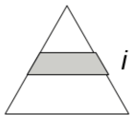


(2) ∃ vertex in U not adjacent to v_{i+1} ($U \not\subseteq N(v_{i+1})$):
→ U itself is a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ if $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is maximal it is also a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i

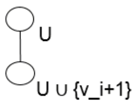


Cliques: Enumeration

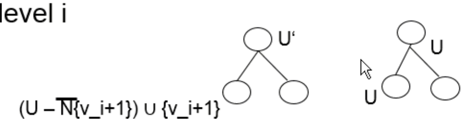
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in $G[v_1,v_2,\dots,v_i]$
- level i+1: determine children of node U at level i: two cases :



(1) v_{i+1} adjacent to all nodes in U ($U \subseteq N(v_{i+1})$):
→ $U \cup \{v_{i+1}\}$ is maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $U \cup \{v_{i+1}\}$ is only child of U

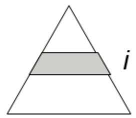


(2) ∃ vertex in U not adjacent to v_{i+1} ($U \not\subseteq N(v_{i+1})$):
→ U itself is a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ if $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is maximal it is also a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i

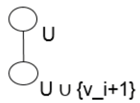


Cliques: Enumeration

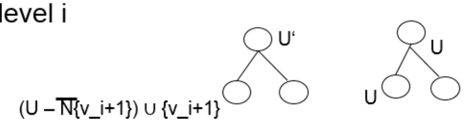
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in $G[v_1,v_2,\dots,v_i]$
- level i+1: determine children of node U at level i: two cases :



(1) v_{i+1} adjacent to all nodes in U ($U \subseteq N(v_{i+1})$):
→ $U \cup \{v_{i+1}\}$ is maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $U \cup \{v_{i+1}\}$ is only child of U

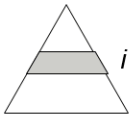


(2) ∃ vertex in U not adjacent to v_{i+1} ($U \not\subseteq N(v_{i+1})$):
→ U itself is a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ if $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is maximal it is also a maximal clique in $G[v_1,v_2,\dots,v_i, v_{i+1}]$
→ $(U - \overline{N}\{v_{i+1}\}) \cup \{v_{i+1}\}$ is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i

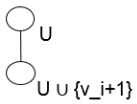


Cliques: Enumeration

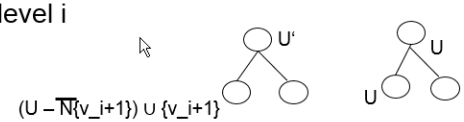
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in G[v_1,v_2,…,v_i}]
- level i+1: determine children of node U at level i: two cases :



(1) v_i+1 adjacent to all nodes in U (U⊆N(v_i+1):
→ U ∪ {v_i+1} is maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ U ∪ {v_i+1} is only child of U

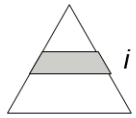


(2) ∃ vertex in U not adjacent to v_i+1 (U⊈N(v_i+1):
→ U itself is a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ if (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is maximal it is also a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i

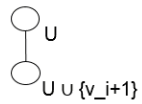


Cliques: Enumeration

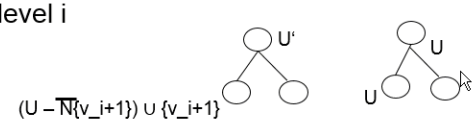
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in G[v_1,v_2,…,v_i}]
- level i+1: determine children of node U at level i: two cases :



(1) v_i+1 adjacent to all nodes in U (U⊆N(v_i+1):
→ U ∪ {v_i+1} is maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ U ∪ {v_i+1} is only child of U



(2) ∃ vertex in U not adjacent to v_i+1 (U⊈N(v_i+1):
→ U itself is a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ if (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is maximal it is also a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i



Cliques: Enumeration

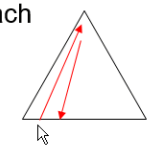
- → tree constructed in principle
- Algorithm: traverse and construct tree depth first, output leaves
- neccesary primitives:

-- Parent(U,i): for node U at level i determine lexicographically smallest clique in G[v_1,v_2,…,v_i, v_i-1}] : O(m+n)

-- LeftChild(U,i): either U or U ∪ {v_i+1} : O(m+n)

-- RightChild(U,i):
if case (1): no right child;
if case (2): if X:=(U – $\overline{N}${v_i+1}) ∪ {v_i+1} is maximal (takes O(m+n) time to determine) then X is right child if U = Parent(X, i+1) (takes O(m+n))

- longest path between two leaves passes over 2n-1 nodes; for each node: O(m+n) time → maximal delay: $O(n^3)$



Cliques: Enumeration

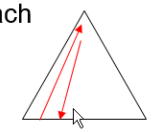
- → tree constructed in principle
- Algorithm: traverse and construct tree depth first, output leaves
- neccesary primitives:

-- Parent(U,i): for node U at level i determine lexicographically smallest clique in G[v_1,v_2,…,v_i, v_i-1}] : O(m+n)

-- LeftChild(U,i): either U or U ∪ {v_i+1} : O(m+n)

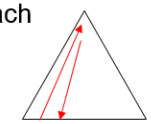
-- RightChild(U,i):
if case (1): no right child;
if case (2): if X:=(U – $\overline{N}${v_i+1}) ∪ {v_i+1} is maximal (takes O(m+n) time to determine) then X is right child if U = Parent(X, i+1) (takes O(m+n))

- longest path between two leaves passes over 2n-1 nodes; for each node: O(m+n) time → maximal delay: $O(n^3)$



Cliques: Enumeration

- → tree constructed in principle
- Algorithm: traverse and construct tree depth first, output leaves
- neccesary primitives:
  - -- Parent(U,i): for node U at level i determine lexicographically smallest clique in G[v_1,v_2,…,v_i, v_i-1}] : O(m+n)
  - -- LeftChild(U,i): either U or U ∪ {v_i+1} : O(m+n)
  - -- RightChild(U,i):
    if case (1): no right child;
    if case (2): if X:=(U – N̄{v_i+1}) ∪ {v_i+1} is maximal (takes O(m+n) time to determine) then X is right child if U = Parent(X, i+1) (takes O(m+n))
- longest path between two leaves passes over 2n-1 nodes; for each node: O(m+n) time → maximal delay: $O(n^3)$

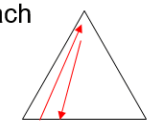


Cliques: Algorithms

- In connection with cliques: Algorithms with time complexity O(|E| + |V|) for the problems:
  - Determine if $U \subseteq V$ is a clique (Test pairs of vertices of U if they are in E. Although up to $\binom{|V|}{2}$ such pairs may exist: If |E| edges have been searched, search is over
  - Determine if clique U is maximal (Test all vertices in V-U if they are connected to all vertices in U; again: If |E| edges have been searched, search is over)
  - Compute lexicographically smallest maximal clique containing U: Assume vertices are sorted; Test all vertices in V-U in ascending order: if they are connected to all vertices in U, add to U; again: If |E| edges have been searched, search is over)

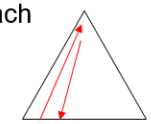
Cliques: Enumeration

- → tree constructed in principle
- Algorithm: traverse and construct tree depth first, output leaves
- neccesary primitives:
  - -- Parent(U,i): for node U at level i determine lexicographically smallest clique in G[v_1,v_2,…,v_i, v_i-1}] : O(m+n)
  - -- LeftChild(U,i): either U or U ∪ {v_i+1} : O(m+n)
  - -- RightChild(U,i):
    if case (1): no right child;
    if case (2): if X:=(U – N̄{v_i+1}) ∪ {v_i+1} is maximal (takes O(m+n) time to determine) then X is right child if U = Parent(X, i+1) (takes O(m+n))
- longest path between two leaves passes over 2n-1 nodes; for each node: O(m+n) time → maximal delay: $O(n^3)$



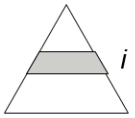
Cliques: Enumeration

- → tree constructed in principle
- Algorithm: traverse and construct tree depth first, output leaves
- neccesary primitives:
  - -- Parent(U,i): for node U at level i determine lexicographically smallest clique in G[v_1,v_2,…,v_i, v_i-1}] : O(m+n)
  - -- LeftChild(U,i): either U or U ∪ {v_i+1} : O(m+n)
  - -- RightChild(U,i):
    if case (1): no right child;
    if case (2): if X:=(U – N̄{v_i+1}) ∪ {v_i+1} is maximal (takes O(m+n) time to determine) then X is right child if U = Parent(X, i+1) (takes O(m+n))
- longest path between two leaves passes over 2n-1 nodes; for each node: O(m+n) time → maximal delay: $O(n^3)$

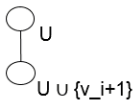


Cliques: Enumeration

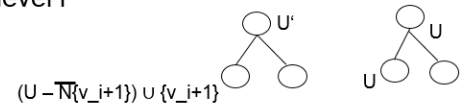
- Binary tree, n levels, leaves only at level n
- each level i ↔ vertex v_i
- nodes at level i: maximal cliques in G[v_1,v_2,…,v_i}]
- level i+1: determine children of node U at level i: two cases :



(1) v_i+1 adjacent to all nodes in U (U⊆N(v_i+1):
→ U ∪ {v_i+1} is maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ U ∪ {v_i+1} is only child of U



(2) ∃ vertex in U not adjacent to v_i+1 (U⊈N(v_i+1):
→ U itself is a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ if (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is maximal it is also a maximal clique in G[v_1,v_2,…,v_i, v_i+1}]
→ (U – $\overline{N}${v_i+1}) ∪ {v_i+1} is potentially child of many nodes → define it als child of lexiographically smallest clique (node) U‘ at level i



Plexes and Cores

- We have seen: Cliques are computationally hard but have certain desirable properties (closed under exclusion, nestedness) and perfectly fulfill requirements for groups
- We have seen: distance based relaxations (N-cliques, N-clubs, N-clans) do not have these properties, do not fulfill the requirements for groups extremely well
- → look for other relaxations of the clique concept
- “Allow clique members to miss some (up to N) ties to other members” → N-Plex
- A subset U ⊆ V is a N-Plex iff $\delta(G([U])) \geq |U| - N$

minimal degree

Plexes and Cores

- A N-Plex is maximal iff it is not strictly contained in another larger N-Plex; A N-Plex is a maximum N-Plex iff it has a maximum number of vertices among all N-Plexes in G
- N-Plexes are closed under exclusion; N-Plexes are nested → good candidates for social groups
- Furthermore: If the set of nodes V of G is an N-Plex we have: (Combinatorial Proof: see [2]):

  if N < (|V|+2) / 2 then diam(G) ≤ 2

→ If an N-Plex has a not too large “sloppyness” N “its” diameter is very small → Socially realistic

- → So far: wonderful but:

Plexes and Cores

- A N-Plex is maximal iff it is not strictly contained in another larger N-Plex; A N-Plex is a maximum N-Plex iff it has a maximum number of vertices among all N-Plexes in G
- N-Plexes are closed under exclusion; N-Plexes are nested → good candidates for social groups
- Furthermore: If the set of nodes V of G is an N-Plex we have: (Combinatorial Proof: see [2]):

  if N < (|V|+2) / 2 then diam(G) ≤ 2

→ If an N-Plex has a not too large “sloppyness” N “its” diameter is very small → Socially realistic

- → So far: wonderful but:

Plexes and Cores

- But the decision problem NPLEX(G,k,N): "Does G contain an N-Plex of size at least k?" is NP-complete for all N. Proof: Informal argument: CLIQUE(G,k)=NPLEX(G,k,1); formal: reduce CLIQUE(G,k) to N-PLEX (see [2]).
- Ok, so (instead of fixing how many edges can be missed at most): why not demand a minimum degree for every node in the structure?
- A subset $U \subseteq V$ is a N-Core iff $\delta(G([U])) \geq N$
- (Defs for maximal and maximum apply accordingly)
- If U is N-core, U is also a (|V|-N)-plex; Socially: If N is small compared to |U|, N-cores are not very meaningful

Plexes and Cores

- But the decision problem NPLEX(G,k,N): "Does G contain an N-Plex of size at least k?" is NP-complete for all N. Proof: Informal argument: CLIQUE(G,k)=NPLEX(G,k,1); formal: reduce CLIQUE(G,k) to N-PLEX (see [2]).
- Ok, so (instead of fixing how many edges can be missed at most): why not demand a minimum degree for every node in the structure?
- A subset $U \subseteq V$ is a N-Core iff $\delta(G([U])) \geq N$
- (Defs for maximal and maximum apply accordingly)
- If U is N-core, U is also a (|V|-N)-plex; Socially: If N is small compared to |U|, N-cores are not very meaningful

Plexes and Cores

- But the decision problem NPLEX(G,k,N): "Does G contain an N-Plex of size at least k?" is NP-complete for all N. Proof: Informal argument: CLIQUE(G,k)=NPLEX(G,k,1); formal: reduce CLIQUE(G,k) to N-PLEX (see [2]).
- Ok, so (instead of fixing how many edges can be missed at most): why not demand a minimum degree for every node in the structure?
- A subset $U \subseteq V$ is a N-Core iff $\delta(G([U])) \geq N$
- (Defs for maximal and maximum apply accordingly)
- If U is N-core, U is also a (|V|-N)-plex; Socially: If N is small compared to |U|, N-cores are not very meaningful

Plexes and Cores

- But the decision problem NPLEX(G,k,N): "Does G contain an N-Plex of size at least k?" is NP-complete for all N. Proof: Informal argument: CLIQUE(G,k)=NPLEX(G,k,1); formal: reduce CLIQUE(G,k) to N-PLEX (see [2]).
- Ok, so (instead of fixing how many edges can be missed at most): why not demand a minimum degree for every node in the structure?
- A subset $U \subseteq V$ is a N-Core iff $\delta(G([U])) \geq N$
- (Defs for maximal and maximum apply accordingly)
- If U is N-core, U is also a (|V|-N)-plex; Socially: If N is small compared to |U|, N-cores are not very meaningful

Plexes and Cores

- But the decision problem NPLEX(G,k,N): "Does G contain an N-Plex of size at least k?" is NP-complete for all N. Proof: Informal argument: CLIQUE(G,k)=NPLEX(G,k,1); formal: reduce CLIQUE(G,k) to N-PLEX (see [2]).
- Ok, so (instead of fixing how many edges can be missed at most): why not demand a minimum degree for every node in the structure?
- A subset $U \subseteq V$ is a N-Core iff $\delta(G([U])) \geq N$
- (Defs for maximal and maximum apply accordingly)
- If U is N-core, U is also a (|V|-N)-plex; Socially: If N is small compared to |U|, N-cores are not very meaningful

Plexes and Cores

- If U1 and U2 are N-cores, U1 U U2 is also an N-core; Socially: not very desirable; (math: → maximum N-core is unique)
- → If U1 and U2 are connected and maximal N-cores they are disjoint; Socially: not very desirable
- N-cores are not closed under exclusion (Example: cycle is 2-core but subset not) and are generally not nested
- N-cores need not be connected
- → N-cores do not seem to be good candidates for social groups
- But: We can easily compute the maximum N-core (see [2])

LS-Sets and Lambda Sets

- We will now look at some non-local concepts:
- LS-sets (Luccio-Sami-Sets): Formalize the paradigm „Intra-cluster coherence, inter-cluster decoherence": An LS-set is a „network region where internal ties are more significant than external ties" [2] (extreme version of paradigm: „strong alliance": complete component (disconnected clique))
- A subset $U \subseteq V$ is a LS-set iff all proper subsets of U share more ties with the network outside than U does:

$$U' \subset U \rightarrow |E(U', V-U')| > |E(U, V-U)|$$

- LS-sets have some interesting properties (e.g. no trival overlaps: if $u1 \cap u2 \neq \emptyset \rightarrow u1 \subseteq u2$ or $u2 \subseteq u1$; min degree of LS set is at least half the number of outgoing edges) and may be good candidates for social groups (overlap property can be a counter argument)
- LS-sets can be computed with reasonable complexity ([2])

LS-Sets and Lambda Sets

- We will now look at some non-local concepts:
- LS-sets (Luccio-Sami-Sets): Formalize the paradigm „Intra-cluster coherence, inter-cluster decoherence": An LS-set is a „network region where internal ties are more significant than external ties" [2] (extreme version of paradigm: „strong alliance": complete component (disconnected clique))
- A subset $U \subseteq V$ is a LS-set iff all proper subsets of U share more ties with the network outside than U does:

$$U' \subset U \rightarrow |E(U', V-U')| > |E(U, V-U)|$$

- LS-sets have some interesting properties (e.g. no trival overlaps: if $u1 \cap u2 \neq \emptyset \rightarrow u1 \subseteq u2$ or $u2 \subseteq u1$; min degree of LS set is at least half the number of outgoing edges) and may be good candidates for social groups (overlap property can be a counter argument)
- LS-sets can be computed with reasonable complexity ([2])

LS-Sets and Lambda Sets

- We will now look at some non-local concepts:
- LS-sets (Luccio-Sami-Sets): Formalize the paradigm „Intra-cluster coherence, inter-cluster decoherence“: An LS-set is a „network region where internal ties are more significant than external ties“ [2] (extreme version of paradigm: „strong alliance“: complete component (disconnected clique))
- A subset $U \subseteq V$ is a LS-set iff all proper subsets of U share more ties with the network outside than U does:

$$U' \subset U \rightarrow |E(U', V-U')| > |E(U, V-U)|$$

- LS-sets have some interesting properties (e.g. no trival overlaps: if $u1 \cap u2 \neq \emptyset \rightarrow u1 \subseteq u2$ or $u2 \subseteq u1$; min degree of LS set is at least half the number of outgoing edges) and may be good candidates for social groups (overlap property can be a counter argument)
- LS-sets can be computed with reasonable complexity ([2])

LS-Sets and Lambda Sets

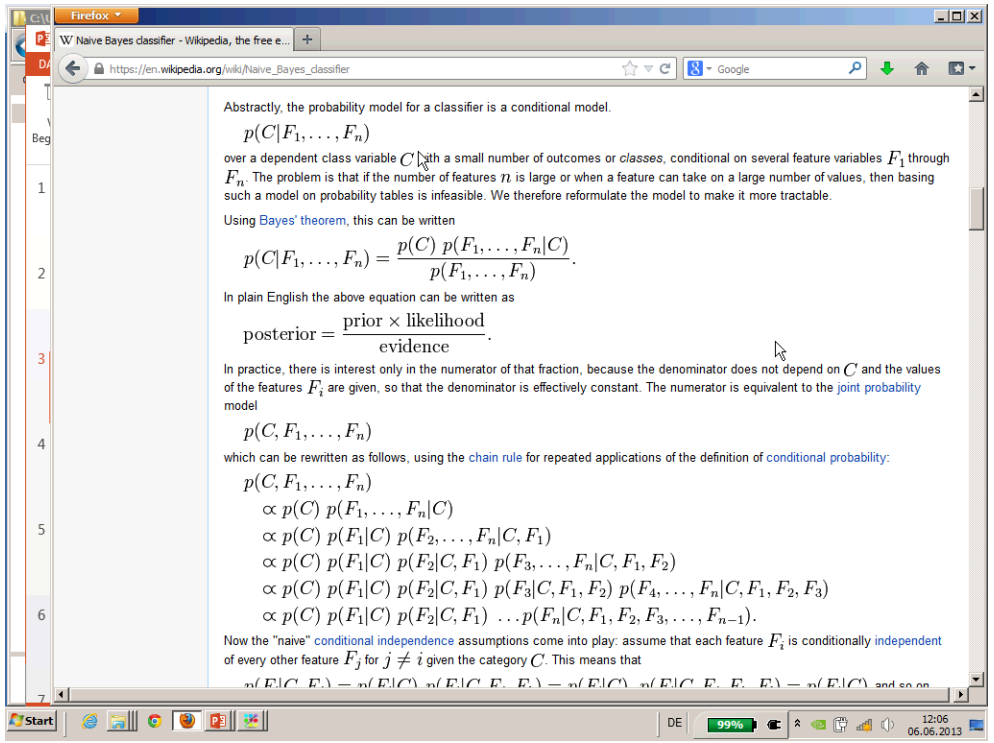
- Lambda Sets are another related concept: In a lambda set, members are connected to other members by more (edge disjoint) paths than to outside nodes
- Let $\lambda(u,v)$ denote the number of edge-disjoint paths between nodes u and v;
A subset $U \subseteq V$ is a Lambda set iff

$$\min_{u,v \in U} \lambda(u,v) > \max_{u \in U,\, v \in V-U} \lambda(u,v)$$

- Lambda sets can be computed in P time [2]

Recommended Reading

- minimal approach:
  - study the slides and mentally review the introduced concepts, definitions and connections
- standard approach:
  - minimal approach + read the corresponding parts of [2]
- interested students:
  - standard approach + read all of [2], studying also the proofs



Naive Bayes classifier - Wikipedia, the free e...

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Abstractly, the probability model for a classifier is a conditional model.

$$p(C|F_1,\dots,F_n)$$

over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, this can be written

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

In plain English the above equation can be written as

$$\mbox{posterior} = \frac{\mbox{prior} \times \mbox{likelihood}}{\mbox{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C, F_1, \dots, F_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} & p(C, F_1, \dots, F_n) \\ & \varpropto p(C)\ p(F_1,\dots,F_n|C) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2,\dots,F_n|C, F_1) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3,\dots,F_n|C, F_1, F_2) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4,\dots,F_n|C, F_1, F_2, F_3) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ \dots p(F_n|C, F_1, F_2, F_3,\dots,F_{n-1}). \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$ given the category C. This means that

$p(F_i|C, F_j) = p(F_i|C)$, $p(F_i|C, F_j, F_k) = p(F_i|C)$, $p(F_i|C, F_j, F_k, F_l) = p(F_i|C)$ and so on

DE 99% 12:07 06.06.2013

Naive Bayes classifier - Wikipedia, the free e...

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Abstractly, the probability model for a classifier is a conditional model.

$$p(C|F_1,\dots,F_n)$$

over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, this can be written

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

In plain English the above equation can be written as

$$\mbox{posterior} = \frac{\mbox{prior} \times \mbox{likelihood}}{\mbox{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C, F_1, \dots, F_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} & p(C, F_1, \dots, F_n) \\ & \varpropto p(C)\ p(F_1,\dots,F_n|C) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2,\dots,F_n|C, F_1) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3,\dots,F_n|C, F_1, F_2) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4,\dots,F_n|C, F_1, F_2, F_3) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ \dots p(F_n|C, F_1, F_2, F_3,\dots,F_{n-1}). \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$ given the category C. This means that

$p(F_i|C, F_j) = p(F_i|C)$, $p(F_i|C, F_j, F_k) = p(F_i|C)$, $p(F_i|C, F_j, F_k, F_l) = p(F_i|C)$ and so on

DE 99% 12:07 06.06.2013

Naive Bayes classifier - Wikipedia, the free e...

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Abstractly, the probability model for a classifier is a conditional model.

$$p(C|F_1,\dots,F_n)$$

over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, this can be written

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

In plain English the above equation can be written as

$$\mbox{posterior} = \frac{\mbox{prior} \times \mbox{likelihood}}{\mbox{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C, F_1, \dots, F_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} & p(C, F_1, \dots, F_n) \\ & \varpropto p(C)\ p(F_1,\dots,F_n|C) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2,\dots,F_n|C, F_1) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3,\dots,F_n|C, F_1, F_2) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4,\dots,F_n|C, F_1, F_2, F_3) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ \dots p(F_n|C, F_1, F_2, F_3,\dots,F_{n-1}). \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$ given the category C. This means that

$p(F_i|C, F_j) = p(F_i|C)$, $p(F_i|C, F_j, F_k) = p(F_i|C)$, $p(F_i|C, F_j, F_k, F_l) = p(F_i|C)$ and so on

DE 99% 12:07 06.06.2013

Naive Bayes classifier - Wikipedia, the free e...

https://en.wikipedia.org/wiki/Naive_Bayes_classifier

Abstractly, the probability model for a classifier is a conditional model.

$$p(C|F_1,\dots,F_n)$$

over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, this can be written

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

In plain English the above equation can be written as

$$\mbox{posterior} = \frac{\mbox{prior} \times \mbox{likelihood}}{\mbox{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C, F_1, \dots, F_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned} & p(C, F_1, \dots, F_n) \\ & \varpropto p(C)\ p(F_1,\dots,F_n|C) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2,\dots,F_n|C, F_1) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3,\dots,F_n|C, F_1, F_2) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4,\dots,F_n|C, F_1, F_2, F_3) \\ & \varpropto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ \dots p(F_n|C, F_1, F_2, F_3,\dots,F_{n-1}). \end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$ given the category C. This means that

$p(F_i|C, F_j) = p(F_i|C)$, $p(F_i|C, F_j, F_k) = p(F_i|C)$, $p(F_i|C, F_j, F_k, F_l) = p(F_i|C)$ and so on

DE 99% 12:08 06.06.2013

over a dependent class variable C with a small number of outcomes or *classes*, conditional on several feature variables F_1 through F_n. The problem is that if the number of features n is large or when a feature can take on a large number of values, then basing such a model on probability tables is infeasible. We therefore reformulate the model to make it more tractable.

Using Bayes' theorem, this can be written

$$p(C|F_1,\dots,F_n) = \frac{p(C)\ p(F_1,\dots,F_n|C)}{p(F_1,\dots,F_n)}.$$

In plain English the above equation can be written as

$$\text{posterior} = \frac{\text{prior} \times \text{likelihood}}{\text{evidence}}.$$

In practice, there is interest only in the numerator of that fraction, because the denominator does not depend on C and the values of the features F_i are given, so that the denominator is effectively constant. The numerator is equivalent to the joint probability model

$$p(C, F_1, \dots, F_n)$$

which can be rewritten as follows, using the chain rule for repeated applications of the definition of conditional probability:

$$\begin{aligned}
&p(C, F_1, \dots, F_n) \\
&\quad \propto p(C)\ p(F_1, \dots, F_n|C) \\
&\quad \propto p(C)\ p(F_1|C)\ p(F_2, \dots, F_n|C, F_1) \\
&\quad \propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3, \dots, F_n|C, F_1, F_2) \\
&\quad \propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ p(F_3|C, F_1, F_2)\ p(F_4, \dots, F_n|C, F_1, F_2, F_3) \\
&\quad \propto p(C)\ p(F_1|C)\ p(F_2|C, F_1)\ \dots p(F_n|C, F_1, F_2, F_3, \dots, F_{n-1}).
\end{aligned}$$

Now the "naive" conditional independence assumptions come into play: assume that each feature F_i is conditionally independent of every other feature F_j for $j \neq i$ given the category C. This means that

$$p(F_i|C, F_j) = p(F_i|C),\ p(F_i|C, F_j, F_k) = p(F_i|C)\ ,\ p(F_i|C, F_j, F_k, F_l) = p(F_i|C),$$ and so on, for $i \neq j, k, l$, and so the joint model can be expressed as

$$\begin{aligned}
p(C|F_1,\dots,F_n) &\propto p(C)\ p(F_1|C)\ p(F_2|C)\ p(F_3|C)\ \cdots \\
&\propto p(C) \prod_{i=1}^{n} p(F_i|C).
\end{aligned}$$

The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular. These assumptions lead to two distinct models, which are often confused.[4][5] When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

For example, suppose the training data contain a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class c, and let σ_c^2 be the variance of the values in x associated with class c. Then, the probability of some value given a class, $P(x = v|c)$, can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2. That is,

$$P(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}}\, e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features. In general, the distribution method is a better choice if there is a small amount of training data, or if the precise distribution of the data is known. The discretization method tends to do better if there is a large amount of training data because it will learn to fit the distribution of the data. Since naive Bayes is typically used when a large amount of data is available (as more computationally expensive models can generally achieve better accuracy), the discretization method is generally preferred over the distribution method.

Sample correction [edit]

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero.

Constructing a classifier from the probability model [edit]

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function classify defined as follows:

$$\text{classify}$$

The assumptions on distributions of features are called the *event model* of the Naive Bayes classifier. For discrete features like the ones encountered in document classification (include spam filtering), multinomial and Bernoulli distributions are popular. These assumptions lead to two distinct models, which are often confused.[4][5] When dealing with continuous data, a typical assumption is that the continuous values associated with each class are distributed according to a Gaussian distribution.

For example, suppose the training data contain a continuous attribute, x. We first segment the data by the class, and then compute the mean and variance of x in each class. Let μ_c be the mean of the values in x associated with class c, and let σ_c^2 be the variance of the values in x associated with class c. Then, the probability of some value given a class, $P(x = v|c)$, can be computed by plugging v into the equation for a Normal distribution parameterized by μ_c and σ_c^2. That is,

$$P(x = v|c) = \frac{1}{\sqrt{2\pi\sigma_c^2}}\, e^{-\frac{(v-\mu_c)^2}{2\sigma_c^2}}$$

Another common technique for handling continuous values is to use binning to discretize the feature values, to obtain a new set of Bernoulli-distributed features. In general, the distribution method is a better choice if there is a small amount of training data, or if the precise distribution of the data is known. The discretization method tends to do better if there is a large amount of training data because it will learn to fit the distribution of the data. Since naive Bayes is typically used when a large amount of data is available (as more computationally expensive models can generally achieve better accuracy), the discretization method is generally preferred over the distribution method.

Sample correction [edit]

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero.

Constructing a classifier from the probability model [edit]

The discussion so far has derived the independent feature model, that is, the naive Bayes probability model. The naive Bayes classifier combines this model with a decision rule. One common rule is to pick the hypothesis that is most probable; this is known as the *maximum a posteriori* or *MAP* decision rule. The corresponding classifier, a Bayes classifier, is the function classify defined as follows:

$$n$$

Examples [edit]

Sex classification [edit]

Problem: classify whether a given person is a male or a female based on the measured features. The features include height, weight, and foot size.

Training [edit]

Example training set below.

sex	height (feet)	weight (lbs)	foot size(inches)
male	6	180	12
male	5.92 (5'11")	190	11
male	5.58 (5'7")	170	12
male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

The classifier created from the training set using a Gaussian distribution assumption would be (given variances are sample variances):

sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. This prior probability distribution might be based on our knowledge of frequencies in the larger population, or on frequency in the training set.

Testing [edit]

male	5.92 (5'11")	165	10
female	5	100	6
female	5.5 (5'6")	150	8
female	5.42 (5'5")	130	7
female	5.75 (5'9")	150	9

The classifier created from the training set using a Gaussian distribution assumption would be (given variances are sample variances):

sex	mean (height)	variance (height)	mean (weight)	variance (weight)	mean (foot size)	variance (foot size)
male	5.855	3.5033e-02	176.25	1.2292e+02	11.25	9.1667e-01
female	5.4175	9.7225e-02	132.5	5.5833e+02	7.5	1.6667e+00

Let's say we have equiprobable classes so P(male)= P(female) = 0.5. This prior probability distribution might be based on our knowledge of frequencies in the larger population, or on frequency in the training set.

Testing [edit]

Below is a sample to be classified as a male or female.

sex	height (feet)	weight (lbs)	foot size(inches)
sample	6	130	8

We wish to determine which posterior is greater, male or female. For the classification as male the posterior is given by

$$posterior(male) = \frac{P(male)\,p(height|male)\,p(weight|male)\,p(footsize|male)}{evidence}$$

For the classification as female the posterior is given by

$$posterior(female) = \frac{P(female)\,p(height|female)\,p(weight|female)\,p(footsize|female)}{evidence}$$

The evidence (also termed normalizing constant) may be calculated since the sum of the posterior probabilities must equal one.

$$evidence = P(male)\,p(height|male)\,p(weight|male)\,p(footsize|male) + P(female)\,p(height|female)\,p(weight|female)\,p(footsize|female)$$

The evidence may be ignored since it is a positive constant. (That is, the evidence is the same for any sample.) We now determine the probability distribution for the sex of the sample.

$$P(male) = 0.5$$

$$p(\text{height}|\text{male}) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{-(6-\mu)^2}{2\sigma^2}\right) \approx 1.5789,$$

where $\mu = 5.855$ and $\sigma^2 = 3.5033e-02$ are the parameters of normal distribution which have been previously determined from the training set. Note that a value greater than 1 is OK here – it is a probability density rather than a probability, because height is a continuous variable.

$$p(\text{weight}|\text{male}) = 5.9881e-06$$

$$p(\text{foot size}|\text{male}) = 1.3112e-3$$

$$\text{posterior numerator (male)} = \text{their product} = 6.1984e-09$$

$$P(\text{female}) = 0.5$$

$$p(\text{height}|\text{female}) = 2.2346e-1$$

$$p(\text{weight}|\text{female}) = 1.6789e-2$$

$$p(\text{foot size}|\text{female}) = 2.8669e-1$$

$$\text{posterior numerator (female)} = \text{their product} = 5.3778e-04$$

Since posterior numerator is greater in the female case, we predict the sample is female.

Document Classification [edit]

Here is a worked example of naive Bayesian classification to the document classification problem. Consider the problem of classifying documents by their content, for example into spam and non-spam e-mails. Imagine that documents are drawn from a number of classes of documents which can be modelled as sets of words where the (independent) probability that the i-th word of a given document occurs in a document from class *C* can be written as

$$p(w_i|C)$$

(For this treatment, we simplify things further by assuming that words are randomly distributed in the document - that is, words are not dependent on the length of the document, position within the document with relation to other words, or other document-context.)

Then the probability that a given document *D* contains all of the words w_i, given a class *C*, is

$$p(D|C) = \prod_i p(w_i|C)$$

The question that we desire to answer is: "what is the probability that a given document *D* belongs to a given class *C*?" In other words, what is $p(C|D)$?

Now by definition

$$p(D|C) = \frac{p(D \cap C)}{p(C)}$$

and

$$p(C|D) = \frac{p(D \cap C)}{p(D)}$$

Bayes' theorem manipulates these into a statement of probability in terms of likelihood.

$$p(C|D) = \frac{p(C)}{p(D)}\,p(D|C)$$

Assume for the moment that there are only two mutually exclusive classes, *S* and ¬*S* (e.g. spam and not spam), such that every element (email) is in either one or the other;

$$p(D|S) = \prod_i p(w_i|S)$$

Graph Clustering

- Given directed, weighted graph G=(V,E,w); A graph clustering **C**={C_1, C_2, ..., C_k} is a partition of V into non-empty subsets C_i;
- Notations:
  - E(C_i,C_j): Set of edges in G from C_i to C_j;
  - E(**C**) = $U_{i=1,..,k}$E(C_i): Set of intra-cluster edges;
  - $\overline{E(\mathbf{C})}$ = E \ E(**C**) : Set of inter-cluster edges;
  - m(**C**) = |E(**C**)|; $\overline{m}$(**C**) = $|\overline{E(\mathbf{C})}|$;
  - G([C_i]): subgraph induced by C_i;
  - **C** with k=1: *1-clustering*; **C** with k=|V|: *singletons*; (both: *Trivial clustering);* **C** with k=2: *cut;*
  - **A**(G): Set of all possible clusterings on G;

Graph Clustering

- Given directed, weighted graph G=(V,E,w); A graph clustering **C**={C_1, C_2, ..., C_k} is a partition of V into non-empty subsets C_i;
- Notations:
  - E(C_i,C_j): Set of edges in G from C_i to C_j;
  - E(**C**) = $U_{i=1,..,k}$E(C_i): Set of intra-cluster edges;
  - $\overline{E(\mathbf{C})}$ = E \ E(**C**) : Set of inter-cluster edges;
  - m(**C**) = |E(**C**)|; $\overline{m}$(**C**) = $|\overline{E(\mathbf{C})}|$;
  - G([C_i]): subgraph induced by C_i;
  - **C** with k=1: *1-clustering*; **C** with k=|V|: *singletons*; (both: *Trivial clustering);* **C** with k=2: *cut;*
  - **A**(G): Set of all possible clusterings on G;

Graph Clustering

- Given directed, weighted graph G=(V,E,w); A graph clustering **C**={C_1, C_2, ..., C_k} is a partition of V into non-empty subsets C_i;
- Notations:
  - E(C_i,C_j): Set of edges in G from C_i to C_j;
  - E(**C**) = $U_{i=1,..,k}$E(C_i): Set of intra-cluster edges;
  - $\overline{E(\mathbf{C})}$ = E \ E(**C**) : Set of inter-cluster edges;
  - m(**C**) = |E(**C**)|; $\overline{m}$(**C**) = $|\overline{E(\mathbf{C})}|$;
  - G([C_i]): subgraph induced by C_i;
  - **C** with k=1: *1-clustering*; **C** with k=|V|: *singletons*; (both: *Trivial clustering);* **C** with k=2: *cut;*
  - **A**(G): Set of all possible clusterings on G;

Graph Clustering

- Given directed, weighted graph G=(V,E,w); A graph clustering **C**={C_1, C_2, ..., C_k} is a partition of V into non-empty subsets C_i;
- Notations:
  - E(C_i,C_j): Set of edges in G from C_i to C_j;
  - E(**C**) = $U_{i=1,..,k}$E(C_i): Set of intra-cluster edges;
  - $\overline{E(\mathbf{C})}$ = E \ E(**C**) : Set of inter-cluster edges;
  - m(**C**) = |E(**C**)|; $\overline{m}$(**C**) = $|\overline{E(\mathbf{C})}|$;
  - G([C_i]): subgraph induced by C_i;
  - **C** with k=1: *1-clustering*; **C** with k=|V|: *singletons*; (both: *Trivial clustering);* **C** with k=2: *cut;*
  - **A**(G): Set of all possible clusterings on G;

Graph Clustering

- Given directed, weighted graph G=(V,E,w); A graph clustering **C**={C_1, C_2, ..., C_k} is a partition of V into non-empty subsets C_i;
- Notations:
  - E(C_i,C_j): Set of edges in G from C_i to C_j;
  - E(**C**) = $U_{i=1,..,k}$E(C_i): Set of intra-cluster edges;
  - $\overline{E(\mathbf{C})}$ = E \ E(**C**) : Set of inter-cluster edges;
  - m(**C**) = |E(**C**)|; $\overline{m}$(**C**) = $|\overline{E(\mathbf{C})}|$;
  - G([C_i]): subgraph induced by C_i;
  - **C** with k=1: *1-clustering*; **C** with k=|V|: *singletons*; (both: *Trivial clustering);* **C** with k=2: *cut;*
  - **A**(G): Set of all possible clusterings on G;

Graph Clustering

- Given directed, weighted graph G=(V,E,w); A graph clustering **C**={C_1, C_2, ..., C_k} is a partition of V into non-empty subsets C_i;
- Notations:
  - E(C_i,C_j): Set of edges in G from C_i to C_j;
  - E(**C**) = $U_{i=1,..,k}$E(C_i): Set of intra-cluster edges;
  - $\overline{E(\mathbf{C})}$ = E \ E(**C**) : Set of inter-cluster edges;
  - m(**C**) = |E(**C**)|; $\overline{m}$(**C**) = $|\overline{E(\mathbf{C})}|$;
  - G([C_i]): subgraph induced by C_i;
  - **C** with k=1: *1-clustering*; **C** with k=|V|: *singletons*; (both: *Trivial clustering);* **C** with k=2: *cut;*
  - **A**(G): Set of all possible clusterings on G;

Graph Clustering

- Notations (continued):
  - Let **C_1**={C_1, C_2, ..., C_k} and **C_2**={C'_1, C'_2, ..., C'_l}: **C_1** ≤ **C_2** ↔ ∀i ∃j : C_i ⊆C'_j; **C_1:** *refinement of* **C_2; C_2:** *coarsening of* **C_1;**
  - *Chain* (comparable set) of clusterings: *hierarchy;*
  - Hierarchy is *total* ←→ Both trivial clusterings are contained;
  - Hierarchy that contains one clustering for each {1,2,...,|V|}: *complete;*
  - S(V): *(Proper) Cut function*: Cutting the node-set in two (non-empty) subsets: S(V) and V \ S(V);

Quality Measures for Clusterings

- Quality measure: Objective function **A**(G) → $\mathbb{R}$ that formalizes the clustering paradigm in a special way
- G= (V,E,w): Weight function w: E→ $\mathbb{R}^+$ is interpreted as "similarity" (higher weights correspond to more intense tie); also possible: negative weights = dissimilarity; or w: E → [0,1] or w: E → [-1,1] etc.
- Distinguish between no edge and edge with weight zero;
- Notation: w(E) = $\sum_{e \in E} w(e)$

Quality Measures for Clusterings

- Quality measure: Objective function **A**(G) → $\mathbb{R}$ that formalizes the clustering paradigm in a special way
- G= (V,E,w): Weight function w: E→ $\mathbb{R}^+$ is interpreted as "similarity" (higher weights correspond to more intense tie); also possible: negative weights = dissimilarity; or w: E → [0,1] or w: E → [-1,1] etc.
- Distinguish between no edge and edge with weight zero;
- Notation: $w(E) = \sum_{e \in E} w(e)$

6: Graph Clustering

- General framework for a quality index of a clustering:

$$index(\mathbf{C}) = \frac{f(\mathbf{C}) + g(\mathbf{C})}{\max\{f(\mathbf{C}') + g(\mathbf{C}') : \mathbf{C}' \in \mathbf{A}(G)\}}$$

- f : **A**(G) → $\mathbb{R}^+$ measures intra cluster density (coherence);
  g : **A**(G) → $\mathbb{R}^+$ measures inter cluster sparseness (decoherence);

Coverage

- First quality measure: Coverage

$$\gamma(\mathbf{C}) = \frac{w(E(\mathbf{C}))}{w(E)} = \frac{\sum_{e \in E(\mathbf{C})} w(e)}{\sum_{e \in E} w(e)}$$

- Thus: f = w(E(**C**)) and g = 0; → only accumulated intra cluster density is measured
- Maximum value 1 achieved for **C**={V} (1-clustering)
- A clustering has coverage γ(**C**)=1 iff $\overline{E(\mathbf{C})}$=∅ (clustering is union of connected components of G) or w($\overline{E(\mathbf{C})}$)=0
- **C** with k>1 can be transformed into **C**´ with k´<k and $\gamma(\mathbf{C}) \leq \gamma(\mathbf{C}')$ by merging two clusters in **C** → optimal non-trivial **C** is a minimum cut
- → „Monotonic" behavior of coverage → coverage is not a good sole quality index

Coverage

- First quality measure: Coverage

$$\gamma(\mathbf{C}) = \frac{w(E(\mathbf{C}))}{w(E)} = \frac{\sum_{e \in E(\mathbf{C})} w(e)}{\sum_{e \in E} w(e)}$$

- Thus: f = w(E(**C**)) and g = 0; → only accumulated intra cluster density is measured
- Maximum value 1 achieved for **C**={V} (1-clustering)
- A clustering has coverage γ(**C**)=1 iff $\overline{E(\mathbf{C})}$=∅ (clustering is union of connected components of G) or w($\overline{E(\mathbf{C})}$)=0
- **C** with k>1 can be transformed into **C**´ with k´<k and $\gamma(\mathbf{C}) \leq \gamma(\mathbf{C}')$ by merging two clusters in **C** → optimal non-trivial **C** is a minimum cut
- → „Monotonic" behavior of coverage → coverage is not a good sole quality index

Conductance

- Clustering paradigm reformulated: Clusters should be well connected (many edges need to be removed to make it unconnected); few inter cluster edges (ideally none)
- Conductance: Measure for bottlenecks (Bottleneck: Cut that separates V into roughly same size halves and “cuts across” relatively few edges)
- Let **C**={C_1, V \ C_1} be a cut. Conductance φ of **C** is defined as

$$\varphi(\mathbf{C}) = \begin{cases} 1 & \text{if } C\_1 \in \{\emptyset, V\} \\ 0 & \text{if } C\_1 \notin \{\emptyset, V\},\ w(\overline{E(\mathbf{C})})=0 \\ \dfrac{w(\overline{E(\mathbf{C})})}{\min(\sum_{e\in E(C\_1, V)} w(e), \sum_{e\in E(V\setminus C\_1, V)} w(e))} & \text{otherwise} \end{cases}$$

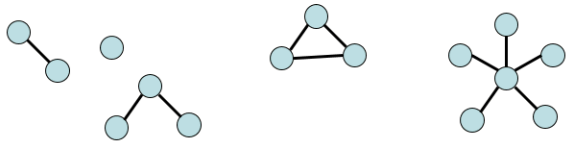
the smaller $\varphi(\mathbf{C})$, the more „bottlenecky“ is **C**

Conductance

- Conductance φ of G is defined as

$$\varphi(G) = \min_{C\_1 \subseteq V} \varphi(C\_1,\ V \setminus C\_1)$$

- Small conductance ←→”good cut possible”
- All unconnected graphs have conductance 0
- Theorem: If G is undirected and positively weighted, G has maximum conductance φ(G)=1 iff G is connected and has at most three nodes or is a star.

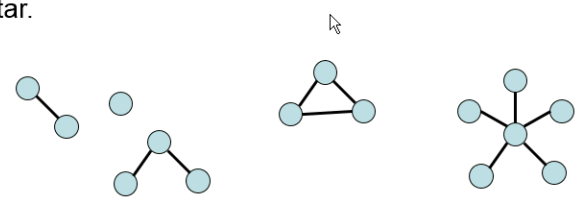


Conductance

- Conductance φ of G is defined as

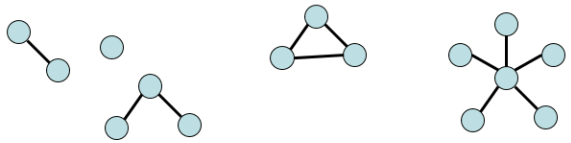
$$\varphi(G) = \min_{C\_1 \subseteq V} \varphi(C\_1,\ V \setminus C\_1)$$

- Small conductance ←→”good cut possible”
- All unconnected graphs have conductance 0
- Theorem: If G is undirected and positively weighted, G has maximum conductance φ(G)=1 iff G is connected and has at most three nodes or is a star.



Conductance

- Theorem: If G is undirected and positively weighted, G has maximum conductance φ(G)=1 iff G is connected and has at most three nodes or is a star. (Proof: see [1])



- Proof `←‘ $\sum_{e\in E(C\_1, V)} w(e) = w(E(C\_1)) + w(\overline{E(\mathbf{C})})$ →

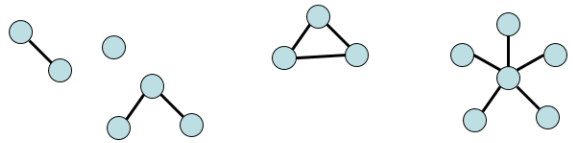
$$\frac{w(\overline{E(\mathbf{C})})}{\min(\sum_{e\in E(C\_1, V)} w(e), \sum_{e\in E(V\setminus C\_1, V)} w(e))} =$$

$$\frac{w(\overline{E(\mathbf{C})})}{w(\overline{E(\mathbf{C})}) + \min(w(E(C\_1)), w(E(V \setminus C\_1)))} = 1$$

$w(E(C\_1))$ =0 if star or at most 3 nodes

Conductance

- Theorem: If G is undirected and positively weighted, G has maximum conductance φ(G)=1 iff G is connected and has at most three nodes or is a star. (Proof: see [1])



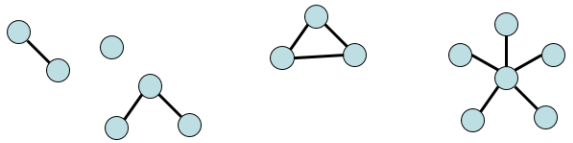
- Proof `←‘ $\sum_{e \in E(C\_1, V)} w(e) = w(E(C\_1)) + w(\overline{E(C)})$ →

$$\frac{w(\overline{E(C)})}{\min(\sum_{e \in E(C\_1, V)} w(e), \sum_{e \in E(V \setminus C\_1, V)} w(e))} =$$

$$\frac{w(\overline{E(C)})}{w(\overline{E(C)}) + \min(\underbrace{w(E(C\_1))}_{=0 \text{ if star or at most 3 nodes}}, w(E(V \setminus C\_1)))} = 1$$

Conductance

- Theorem: If G is undirected and positively weighted, G has maximum conductance φ(G)=1 iff G is connected and has at most three nodes or is a star. (Proof: see [1])



- Proof `←‘ $\sum_{e \in E(C\_1, V)} w(e) = w(E(C\_1)) + w(\overline{E(C)})$ →

$$\frac{w(\overline{E(C)})}{\min(\sum_{e \in E(C\_1, V)} w(e), \sum_{e \in E(V \setminus C\_1, V)} w(e))} =$$

$$\frac{w(\overline{E(C)})}{w(\overline{E(C)}) + \min(\underbrace{w(E(C\_1))}_{=0 \text{ if star or at most 3 nodes}}, w(E(V \setminus C\_1)))} = 1$$