Machine Learning I

Bjoern Andres (Lectures)
Shengxian Zhao and Jerome Thiessat (Exercises)

Machine Learning for Computer Vision
TU Dresden

Winter Term 2021/2022

Contents.

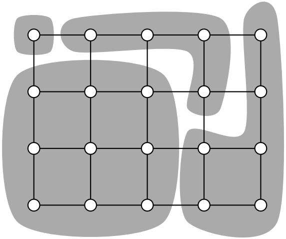
- This part of the course is about the problem of **decomposing (clustering)** a graph into **components (clusters)**, without knowing the number, size or any other property of the clusters.

Contents.

- This part of the course is about the problem of **decomposing (clustering)** a graph into **components (clusters)**, without knowing the number, size or any other property of the clusters.
- This generalizes the problem of partitioning a set. It specializes to the latter for complete graphs.

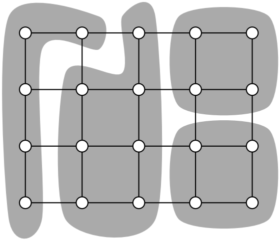
Contents.

- This part of the course is about the problem of **decomposing (clustering)** a graph into **components (clusters)**, without knowing the number, size or any other property of the clusters.
- This generalizes the problem of partitioning a set. It specializes to the latter for complete graphs.
- Analogously, the problem of decomposing a graph is introduced as an **unsupervised learning** problem w.r.t. **constrained data**.



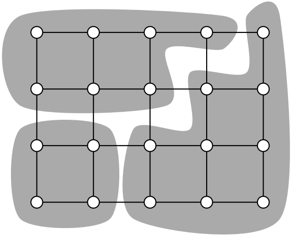
Decomposition of a graph $G = (V, E)$

Clustering

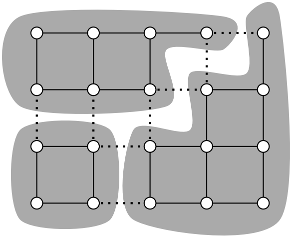


Decomposition of a graph $G = (V, E)$

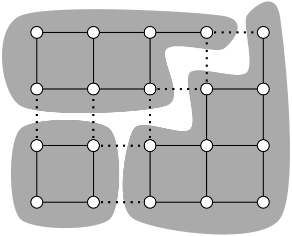
Clustering



Decomposition of a graph $G = (V, E)$

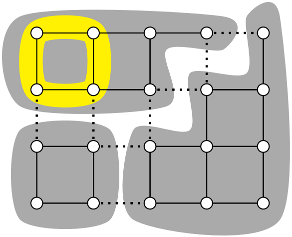


Decomposition of a graph $G = (V, E)$

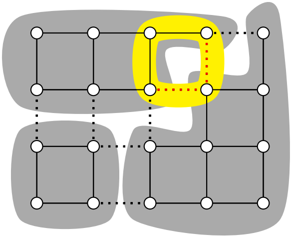


Multicut of a graph $G = (V, E)$

Clustering

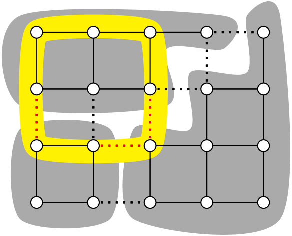


Multicut of a graph $G = (V, E)$

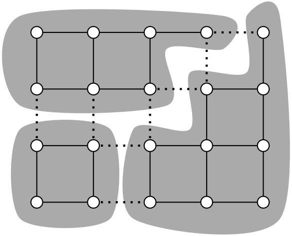


Multicut of a graph $G = (V, E)$

Clustering

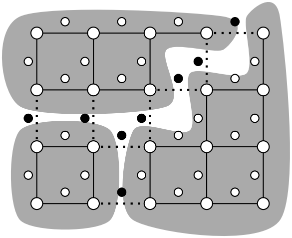


Multicut of a graph $G = (V, E)$



Multicut of a graph $G = (V, E)$

Clustering



Multicut of a graph $G = (V, E)$

Let $G = (A, E)$ be any graph.

Let $G = (A, E)$ be any graph.

Definition.

- A subgraph $G' = (A', E')$ of G is called a **component (cluster)** of G iff G' is non-empty, node-induced (i.e. $E' = E \cap \binom{A'}{2}$) and connected.

Let $G = (A, E)$ be any graph.

Definition.

- A subgraph $G' = (A', E')$ of G is called a **component (cluster)** of G iff G' is non-empty, node-induced (i.e. $E' = E \cap \binom{A'}{2}$) and connected.
- A partition Π of the node set A is called a **decomposition (clustering)** of G iff, for every $U \in \Pi$, the subgraph $(U, E \cap \binom{U}{2})$ of G induced by U is connected (and thus a component of G).

Let $G = (A, E)$ be any graph.

Definition.

- A subgraph $G' = (A', E')$ of G is called a **component (cluster)** of G iff G' is non-empty, node-induced (i.e. $E' = E \cap \binom{A'}{2}$) and connected.
- A partition Π of the node set A is called a **decomposition (clustering)** of G iff, for every $U \in \Pi$, the subgraph $(U, E \cap \binom{U}{2})$ of G induced by U is connected (and thus a component of G).
- We denote by D_G the set of all decompositions of G.

Definition.

- A subset $M \subseteq E$ of edges is called a **multicut** of G iff, for every cycle $C \subseteq E$ of G, we have $|C \cap M| \neq 1$.

Definition.

- A subset $M \subseteq E$ of edges is called a **multicut** of G iff, for every cycle $C \subseteq E$ of G, we have $|C \cap M| \neq 1$.
- We denote by M_G the set of all multicuts of G.

Definition.

- A subset $M \subseteq E$ of edges is called a **multicut** of G iff, for every cycle $C \subseteq E$ of G, we have $|C \cap M| \neq 1$.
- We denote by M_G the set of all multicuts of G.

Lemma.

- For any decomposition of a graph G, the set of those edges that straddle distinct components is a multicut of G. This multicut is said to be **induced** by the decomposition.

Definition.

- A subset $M \subseteq E$ of edges is called a **multicut** of G iff, for every cycle $C \subseteq E$ of G, we have $|C \cap M| \neq 1$.
- We denote by M_G the set of all multicuts of G.

Lemma.

- For any decomposition of a graph G, the set of those edges that straddle distinct components is a multicut of G. This multicut is said to be **induced** by the decomposition.
- The map from decompositions to induced multicuts is a **bijection** from D_G to M_G.

Remarks:

- The characteristic function $y\colon E \to \{0,1\}$ of a multicut $y^{-1}(1)$ decides, for every edge $\{a,b\} = e \in E$, whether the incident nodes a and b belong to the same component ($y_e = 0$) or distinct components ($y_e = 1$).

Remarks:

- The characteristic function $y \colon E \to \{0,1\}$ of a multicut $y^{-1}(1)$ decides, for every edge $\{a,b\} = e \in E$, whether the incident nodes a and b belong to the same component ($y_e = 0$) or distinct components ($y_e = 1$).
- By the definition of a multicut, these decisions are not necessarily independent.

Remarks:

- The characteristic function $y\colon E \to \{0,1\}$ of a multicut $y^{-1}(1)$ decides, for every edge $\{a,b\} = e \in E$, whether the incident nodes a and b belong to the same component ($y_e = 0$) or distinct components ($y_e = 1$).
- By the definition of a multicut, these decisions are not necessarily independent.

Lemma. For any $y\colon E \to \{0,1\}$, the set $y^{-1}(1)$ of those edges that are mapped to 1 is a multicut of G iff the following inequalities are satisfied:

$$\forall C \in \mathsf{cycles}(G)\ \forall e \in C\colon \quad y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{1}$$

Constrained Data

We reduce the problem of learning and inferring multicuts to the problem of learning and inferring decisions, by defining **constrained data** (S, X, x, Y) with

$$S = E \tag{2}$$

$$\mathcal{Y} = \left\{ y : E \to \{0,1\} \;\middle|\; \forall C \in \mathsf{cycles}(G)\ \forall e \in C\colon\ y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \right\} \tag{3}$$

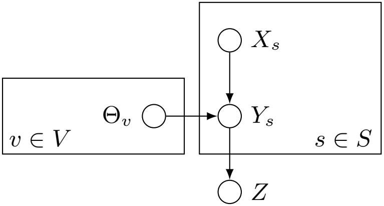
Familiy of functions

- We consider a finite, non-empty set V, called a set of **attributes**, and the **attribute space** $X = \mathbb{R}^V$

Familiy of functions

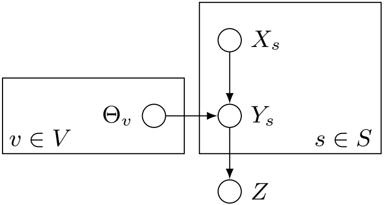
- We consider a finite, non-empty set V, called a set of **attributes**, and the **attribute space** $X = \mathbb{R}^V$
- We consider **linear functions**. Specifically, we consider $\Theta = \mathbb{R}^V$ and $f : \Theta \to \mathbb{R}^X$ such that

$$\forall \theta \in \Theta \; \forall \hat{x} \in \mathbb{R}^V : \quad f_\theta(\hat{x}) = \sum_{v \in V} \theta_v \, \hat{x}_v = \langle \theta, \hat{x} \rangle \ . \tag{4}$$



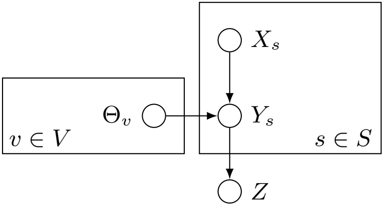
Random Variables

- For any $\{a, b\} = s \in S = E$, let X_s be a random variable whose value is a vector $x_s \in \mathbb{R}^V$, the **attribute vector** of s.



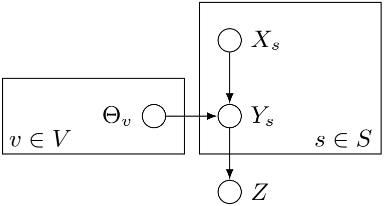
Random Variables

- For any $\{a, b\} = s \in S = E$, let X_s be a random variable whose value is a vector $x_s \in \mathbb{R}^V$, the **attribute vector** of s.
- For any $s \in S$, let Y_s be a random variable whose value is a binary number $y_s \in \{0, 1\}$, called the **decision** of joining $\{a, b\} = s$.



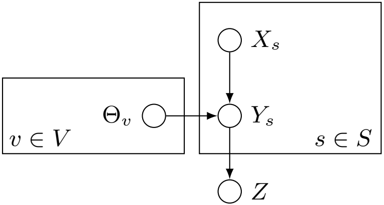
Random Variables

- For any $\{a, b\} = s \in S = E$, let X_s be a random variable whose value is a vector $x_s \in \mathbb{R}^V$, the **attribute vector** of s.
- For any $s \in S$, let Y_s be a random variable whose value is a binary number $y_s \in \{0, 1\}$, called the **decision** of joining $\{a, b\} = s$.
- For any $v \in V$, let Θ_v be a random variable whose value is a real number $\theta_v \in \mathbb{R}$, a **parameter** of the function we seek to learn.



Random Variables

- For any $\{a, b\} = s \in S = E$, let X_s be a random variable whose value is a vector $x_s \in \mathbb{R}^V$, the **attribute vector** of s.
- For any $s \in S$, let Y_s be a random variable whose value is a binary number $y_s \in \{0, 1\}$, called the **decision** of joining $\{a, b\} = s$.
- For any $v \in V$, let Θ_v be a random variable whose value is a real number $\theta_v \in \mathbb{R}$, a **parameter** of the function we seek to learn.
- Let Z be a random variable whose value is a subset $\mathcal{Z} \subseteq \{0, 1\}^S$ called the set of **feasible decisions**. For clustering, we are interested in $\mathcal{Z} = \mathcal{Y}$, the set characterizing multicuts of G.



Factorization

$$P(X, Y, Z, \Theta) = P(Z \mid Y) \prod_{s \in S} P(Y_s \mid X_s, \Theta) \prod_{v \in V} P(\Theta_v) \prod_{s \in S} P(X_s)$$

Factorization

- Supervised learning:

$$P(\Theta \mid X, Y, Z)$$

Factorization

- Supervised learning:

$$\begin{aligned}
P(\Theta \mid X, Y, Z) &= \frac{P(X, Y, Z, \Theta)}{P(X, Y, Z)} \\
&= \frac{P(Z \mid Y)\, P(Y \mid X, \Theta)\, P(X)\, P(\Theta)}{P(Z \mid X, Y)\, P(X, Y)} \\
&= \frac{P(Z \mid Y)\, P(Y \mid X, \Theta)\, P(X)\, P(\Theta)}{P(Z \mid Y)\, P(X, Y)} \\
&= \frac{P(Y \mid X, \Theta)\, P(X)\, P(\Theta)}{P(X, Y)} \\
&\propto P(Y \mid X, \Theta)\, P(\Theta) \\
&= \prod_{s \in S} P(Y_s \mid X_s, \Theta) \prod_{v \in V} P(\Theta_v)
\end{aligned}$$

Factorization

- Inference:

$$P(Y \mid X, Z, \theta)$$

Factorization

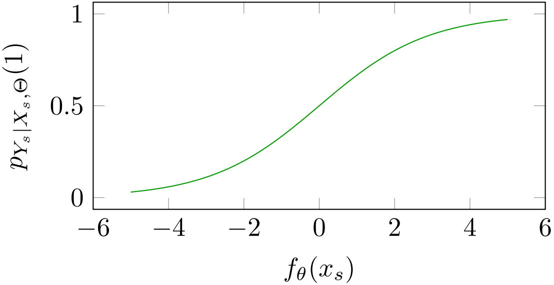
- Inference:

$$\begin{aligned}
P(Y \mid X, Z, \theta) &= \frac{P(X, Y, Z, \Theta)}{P(X, Z, \Theta)} \\
&= \frac{P(Z \mid Y)\, P(Y \mid X, \Theta)\, P(X)\, P(\Theta)}{P(X, Z, \Theta)} \\
&\propto P(Z \mid Y)\, P(Y \mid X, \Theta) \\
&= P(Z \mid Y) \prod_{s \in S} P(Y_s \mid X_s, \Theta)
\end{aligned}$$

Distributions

- **Logistic distribution**

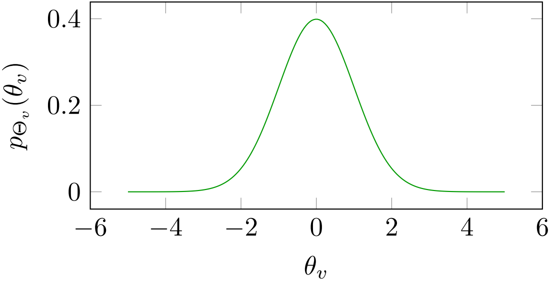
$$\forall s \in S: \qquad p_{Y_s|X_s,\Theta}(1) = \frac{1}{1 + 2^{-f_\theta(x_s)}} \tag{5}$$



Distributions

- **Normal distribution** with $\sigma \in \mathbb{R}^+$:

$$\forall v \in V : \qquad p_{\Theta_v}(\theta_v) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\theta_v^2/2\sigma^2} \tag{6}$$



Distributions

- **Uniform distribution on a subset**

$$\forall \mathcal{Z} \subseteq \{0,1\}^S \; \forall y \in \{0,1\}^S \quad p_{Z|Y}(\mathcal{Z}, y) \propto \begin{cases} 1 & \text{if } y \in \mathcal{Z} \\ 0 & \text{otherwise} \end{cases}$$

Note that $p_{Z|Y}(\mathcal{Y}, y)$ is non-zero iff the labeling $y \colon S \to \{0,1\}$ defines an multicut of G.

Lemma. Estimating maximally probable parameters θ, given attributes x and decisions y, i.e.,

$$\underset{\theta \in \mathbb{R}^V}{\operatorname{argmax}} \quad p_{\Theta|X,Y,Z}(\theta, x, y, \mathcal{Y})$$

is an l_2-regularized logistic regression problem.

Lemma. Estimating maximally probable parameters θ, given attributes x and decisions y, i.e.,

$$\underset{\theta \in \mathbb{R}^V}{\operatorname{argmax}} \quad p_{\Theta|X,Y,Z}(\theta, x, y, \mathcal{Y})$$

is an l_2-regularized logistic regression problem.

Proof. Analogous to the case of deciding, we obtain:

$$\begin{aligned} & \underset{\theta \in \mathbb{R}^V}{\operatorname{argmax}} \quad p_{\Theta|X,Y,Z}(\theta, x, y, \mathcal{Y}) \\ = & \underset{\theta \in \mathbb{R}^V}{\operatorname{argmin}} \quad \sum_{s \in S} \left(-y_s \, f_\theta(x_s) + \log \left(1 + 2^{f_\theta(x_s)} \right) \right) + \frac{\log e}{2\sigma^2} \|\theta\|_2^2 \ . \end{aligned}$$

Lemma. Estimating maximally probable decisions y, given attributes x, given the set of feasible decisions $\mathcal{Y}$, and given parameters θ, i.e.,

$$\underset{y \in \{0,1\}^S}{\operatorname{argmax}} \quad p_{Y|X,Z,\Theta}(y, x, \mathcal{Y}, \theta) \tag{7}$$

assumes the form of the **minimum cost multicut problem**:

$$\underset{y\colon E \to \{0,1\}}{\operatorname{argmin}} \quad \sum_{e \in E} (-\langle \theta, x_e \rangle)\, y_e \tag{8}$$

$$\text{subject to} \quad \forall C \in \mathsf{cycles}(G)\ \forall e \in C\colon \quad y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{9}$$

Lemma. Estimating maximally probable decisions y, given attributes x, given the set of feasible decisions $\mathcal{Y}$, and given parameters θ, i.e.,

$$\operatorname*{argmax}_{y \in \{0,1\}^S} \quad p_{Y|X,Z,\Theta}(y, x, \mathcal{Y}, \theta) \tag{7}$$

assumes the form of the **minimum cost multicut problem**:

$$\operatorname*{argmin}_{y\colon E \to \{0,1\}} \quad \sum_{e \in E} (-\langle \theta, x_e \rangle)\, y_e \tag{8}$$

$$\text{subject to} \quad \forall C \in \mathsf{cycles}(G)\ \forall e \in C\colon \quad y_e \leq \sum_{e' \in C \setminus \{e\}} y_{e'} \tag{9}$$

Theorem. The minimum cost multicut problem is NP-hard.

Bansal et al. (2004) reduce this problem to the k terminal cut problem whose NP-hardness is an important result Dahlhaus et al. (1994).

We will generalize the three **local search algorithms** we have defined for the set partition problem to the minimum cost multicut problem.

We will generalize the three **local search algorithms** we have defined for the set partition problem to the minimum cost multicut problem.

For simplicity, we define $c : E \to \mathbb{R}$ such that

$$\forall e \in S\colon \quad c_e = -\langle \theta, x_e \rangle \tag{10}$$

and write the (linear) cost function $\varphi : \{0,1\}^E \to \mathbb{R}$ such that

$$\forall y \in \{0,1\}^E\colon \quad \varphi(y) = \sum_{e \in E} c_e \, y_e \tag{11}$$

Greedy joining algorithm:

- The greedy joining algorithm is a local search algorithm that starts from any initial decomposition.

Greedy joining algorithm:

- The greedy joining algorithm is a local search algorithm that starts from any initial decomposition.
- It searches for decompositions with lower cost by joining pairs of **neighboring (!)** components recursively.

Greedy joining algorithm:

- The greedy joining algorithm is a local search algorithm that starts from any initial decomposition.
- It searches for decompositions with lower cost by joining pairs of **neighboring (!)** components recursively.
- As components can only grow and the number of components decreases by one in every step, one typically starts from the finest decomposition Π_0 of A into one-elementary components.

Definition. Let $G = (A, E)$ be any graph.

Definition. Let $G = (A, E)$ be any graph.

- For any disjoint sets $B, C \subseteq A$, the pair $\{B, C\}$ is called **neighboring** in G iff there exist nodes $b \in B$ and $c \in C$ such that $\{b, c\} \in E$.

Definition. Let $G = (A, E)$ be any graph.

- For any disjoint sets $B, C \subseteq A$, the pair $\{B, C\}$ is called **neighboring** in G iff there exist nodes $b \in B$ and $c \in C$ such that $\{b, c\} \in E$.
- For any decomposition Π of a graph $G = (A, E)$, we define

$$\mathcal{E}_\Pi = \left\{ \{B, C\} \in \binom{\Pi}{2} \;\middle|\; \exists b \in B \,\exists c \in C \colon \{b, c\} \in E \right\} \quad . \tag{12}$$

Definition. Let $G = (A, E)$ be any graph.

- For any disjoint sets $B, C \subseteq A$, the pair $\{B, C\}$ is called **neighboring** in G iff there exist nodes $b \in B$ and $c \in C$ such that $\{b, c\} \in E$.
- For any decomposition Π of a graph $G = (A, E)$, we define

$$\mathcal{E}_\Pi = \left\{ \{B, C\} \in \tbinom{\Pi}{2} \,\middle|\, \exists b \in B \, \exists c \in C \colon \{b, c\} \in E \right\} \ . \tag{12}$$

- For any decomposition Π of $G = (A, E)$ and any $\{B, C\} \in \mathcal{E}_\Pi$, let $\mathrm{join}_{BC}[\Pi]$ be the decomposition of G obtained by joining the sets B and C in Π, i.e.

$$\mathrm{join}_{BC}[\Pi] = (\Pi \setminus \{B, C\}) \cup \{B \cup C\} \ . \tag{13}$$

$\Pi' = \text{greedy-joining}(\Pi)$

choose $\{B,C\} \in \operatorname*{argmin}_{\{B',C'\} \in \mathcal{E}_\Pi} \varphi(y^{\text{join}_{B'C'}[\Pi]}) - \varphi(y^\Pi)$
if $\varphi(y^{\text{join}_{BC}[\Pi]}) - \varphi(y^\Pi) < 0$
    $\Pi' := \text{greedy-joining}(\text{join}_{BC}[\Pi])$
else
    $\Pi' := \Pi$

Greedy moving algorithm:

- The greedy moving algorithm is a local search algorithm that starts from any initial decomposition, e.g., the fixed point of greedy joining.

Greedy moving algorithm:

- The greedy moving algorithm is a local search algorithm that starts from any initial decomposition, e.g., the fixed point of greedy joining.
- It searches for decompositions with lower cost by recursively moving individual nodes from one component to a **neighboring!** component, possibly a new one.

Greedy moving algorithm:

- The greedy moving algorithm is a local search algorithm that starts from any initial decomposition, e.g., the fixed point of greedy joining.
- It searches for decompositions with lower cost by recursively moving individual nodes from one component to a **neighboring!** component, possibly a new one.
- When a **cut node** is moved out of a component or a node is moved to a new component, the number of components increases. When the last element is moved out of a component, the number of components decreases.

Definition. For any graph $G = (A, E)$ and any decomposition Π of G, the decomposition Π is called **coarsest** iff, for every $U \in \Pi$, the component $(U, E \cap \binom{U}{2})$ induced by U is maximal.

Definition. For any graph $G = (A, E)$ and any decomposition Π of G, the decomposition Π is called **coarsest** iff, for every $U \in \Pi$, the component $(U, E \cap \binom{U}{2})$ induced by U is maximal.

Lemma. For any graph G, the coarsest decomposition is unique. We denote it by Π^*_G.

Definition. For any graph $G = (A, E)$ and any decomposition Π of G, the decomposition Π is called **coarsest** iff, for every $U \in \Pi$, the component $(U, E \cap \binom{U}{2})$ induced by U is maximal.

Lemma. For any graph G, the coarsest decomposition is unique. We denote it by Π^*_G.

Definition. For any graph $G = (A, E)$, any decomposition Π of A and any $a \in A$, choose U_a to be the unique $U_a \in \Pi$ such that $a \in U_a$, and let

$$\mathcal{N}_a = \{\emptyset\} \cup \{W \in \Pi \mid a \notin W \ \wedge \ \exists w \in W \colon \{a, w\} \in E\} \tag{14}$$

$$G_a = \left(U_a \setminus \{a\},\, E \cap \binom{U_a \setminus \{a\}}{2}\right) \tag{15}$$

For any $U \in \mathcal{N}_a$, let $\mathsf{move}_{aU}[\Pi]$ the decomposition of A obtained by moving the node a to the set U, i.e.

$$\mathsf{move}_{aU}[\Pi] = \Pi \setminus \{U_a, U\} \cup \{U \cup \{a\}\} \cup \Pi^*_{G_a} \ . \tag{16}$$

$\Pi' = \mathsf{greedy\text{-}moving}(\Pi)$

choose $(a, U) \in \underset{a' \in A,\ U' \in \mathcal{N}_{a'}}{\operatorname{argmin}} \varphi(y^{\mathsf{move}_{a'U'}[\Pi]}) - \varphi(y^{\Pi})$

if $\varphi(y^{\mathsf{move}_{aU}[\Pi]}) - \varphi(y^{\Pi}) < 0$

$\quad \Pi' := \mathsf{greedy\text{-}moving}(\mathsf{move}_{aU}[\Pi])$

else

$\quad \Pi' := \Pi$

$\Pi' = \textsf{greedy-moving}(\Pi)$

choose $(a, U) \in \underset{a' \in A,\ U' \in \mathcal{N}_{a'}}{\operatorname{argmin}} \varphi(y^{\textsf{move}_{a'U'}[\Pi]}) - \varphi(y^{\Pi})$
if $\varphi(y^{\textsf{move}_{aU}[\Pi]}) - \varphi(y^{\Pi}) < 0$
    $\Pi' := \textsf{greedy-moving}(\textsf{move}_{aU}[\Pi])$
else
    $\Pi' := \Pi$

A generalization of this algorithm by means of the technique of Kernighan and Lin (1970) is analogous to the greedy moving algorithm for the set partition problem.

Summary.

- Learning and inferring decompositions (clusterings) of a graph is an unsupervised learning problem w.r.t. constrained data whose feasible labelings characterize the multicuts of the graph

Summary.

- Learning and inferring decompositions (clusterings) of a graph is an unsupervised learning problem w.r.t. constrained data whose feasible labelings characterize the multicuts of the graph
- The supervised learning problem can assume the form of l_2-regularized logistic regression where samples are pairs of neighboring nodes and decisions indicate whether these nodes are in the same or distinct components

Summary.

- Learning and inferring decompositions (clusterings) of a graph is an unsupervised learning problem w.r.t. constrained data whose feasible labelings characterize the multicuts of the graph
- The supervised learning problem can assume the form of l_2-regularized logistic regression where samples are pairs of neighboring nodes and decisions indicate whether these nodes are in the same or distinct components
- The inference problem assumes the form of the NP-hard minimum cost multicut problem

Summary.

- Learning and inferring decompositions (clusterings) of a graph is an unsupervised learning problem w.r.t. constrained data whose feasible labelings characterize the multicuts of the graph
- The supervised learning problem can assume the form of l_2-regularized logistic regression where samples are pairs of neighboring nodes and decisions indicate whether these nodes are in the same or distinct components
- The inference problem assumes the form of the NP-hard minimum cost multicut problem
- Local search algorithms for tackling this problem are greedy joining, greedy moving, and greedy moving using the technique of Kernighan and Lin.