# Active Semi-supervised Community Detection based on Pair-wise Constraints

程 建 军

兰 州 大 学

信 息 科 学 与 工 程 学 院

chengjianjun@lzu.edu.cn

# ☞ Contents

- Concepts about Community Detection
- Semi-supervised learning
- Must-link and Cannot-link
- Active Learning
- Proposed algorithms

# ☞ Concepts about community detection

## Complex network

- A **complex network** is an undirected and unweighted graph $G = (V, E)$, where $V$ represents the node set, and $E$ represents the edges set.

# ☞ Concepts about community detection

## Complex network

- A **complex network** is an undirected and unweighted graph $G = (V, E)$, where $V$ represents the node set, and $E$ represents the edges set.
- The node can be a people in the social network, a web page in the internet web graph, a paper in the citation network, *etc.*, and each edge in the network can represent the interaction or relationship between two people, can be a hyperlink between the web pages, a reference relationship between the papers, *etc.*

# ☞ Concepts about community detection

## Complex network

- A **complex network** is an undirected and unweighted graph $G = (V, E)$, where $V$ represents the node set, and $E$ represents the edges set.
- The node can be a people in the social network, a web page in the internet web graph, a paper in the citation network, *etc.*, and each edge in the network can represent the interaction or relationship between two people, can be a hyperlink between the web pages, a reference relationship between the papers, *etc.*
- A common property or characteristic of the complex network is its "**Community Structure**".

# ☞ Concepts about community detection

## Complex network

- A **complex network** is an undirected and unweighted graph $G = (V, E)$, where $V$ represents the node set, and $E$ represents the edges set.
- The node can be a people in the social network, a web page in the internet web graph, a paper in the citation network, *etc.*, and each edge in the network can represent the interaction or relationship between two people, can be a hyperlink between the web pages, a reference relationship between the papers, *etc.*
- A common property or characteristic of the complex network is its "**Community Structure**".
- A **Community** is a group of densely connected nodes in the network, such that the nodes within a group are much more connected to each other than to the rest of the network.

# ☞ Concepts about community detection

## Complex network

- A **complex network** is an undirected and unweighted graph $G = (V, E)$, where $V$ represents the node set, and $E$ represents the edges set.
- The node can be a people in the social network, a web page in the internet web graph, a paper in the citation network, *etc.*, and each edge in the network can represent the interaction or relationship between two people, can be a hyperlink between the web pages, a reference relationship between the papers, *etc.*
- A common property or characteristic of the complex network is its "**Community Structure**".
- A **Community** is a group of densely connected nodes in the network, such that the nodes within a group are much more connected to each other than to the rest of the network.
- Communities are of interest because they often correspond to functional units of the networks, and community detection may shed light on the structural and functional characteristics of the network.

## ☞ Semi-supervised learning

- Classification: supervised learning
- Clustering: unsupervised learning
- Semi-supervised learning: semi-supervised classification, semi-supervised clustering
- Two kinds of semi-supervised components: labeled data and must-link constraints, cannot-link constraints

## ☞ must-link constraints and cannot-link constraints

**Must-link constraints** $C_{ML}$: $\forall v_i, v_j \in V$, $(v_i, v_j) \in C_{ML}$ indicates the two nodes $v_i$ and $v_j$ must belong to the same community.

**Cannot-link constraints** $C_{CL}$: $\forall v_i, v_j \in V$, $(v_i, v_j) \in C_{CL}$ indicates the two nodes $v_i$ and $v_j$ cannot be classified into the same community, and they must be allocated into different communities.

# ☞ Active learning

- A strategy to select those data examples actively to annotate, such that the learner could achieve higher performance as compared with random selection
- its goal is to maximize the learner's ability at a minimum cost
- pool-based and stream-based
- the most informative data or the data with least certainty are selected to annotate by an **oracle**

# ☞ Proposed algorithms

- Semi-supervised community detection algorithm based on the must-link constraints and cannot-link constraints
  - Augment the must-link constraint set and Cannot-link constraint set using the transitive property of must-link constraints
  - Construct the skeleton of the initial community structure from cannot-link constraints and must-link constraints
  - Expand communities by greedily select the (*community,unclassified node*) pair from all the pairs, and insert the selected unclassified node into the corresponding community iteratively

## Random walk based algorithm to compute the similarity matrix

- A walker is placed in the network, at each time step, the walker is on a vertex and randomly jumps from the vertex to one of its neighbors following the edge between them

- Random walk based algorithm to compute the similarity matrix
  - A walker is placed in the network, at each time step, the walker is on a vertex and randomly jumps from the vertex to one of its neighbors following the edge between them
  - The walker tends to be trapped in a group of densely connected vertices corresponding to a community rather than walk across communities boundaries within limited number of steps

- Random walk based algorithm to compute the similarity matrix
  - A walker is placed in the network, at each time step, the walker is on a vertex and randomly jumps from the vertex to one of its neighbors following the edge between them
  - The walker tends to be trapped in a group of densely connected vertices corresponding to a community rather than walk across communities boundaries within limited number of steps
  - Can be used to compute the similarity matrix: in each walk, we keep track of the visited nodes in a se, the possibility that these nodes in the set belong in the same community is high, so the similarity values between each pair of the nodes in the set should be increased

- Random walk based algorithm to compute the similarity matrix
  - A walker is placed in the network, at each time step, the walker is on a vertex and randomly jumps from the vertex to one of its neighbors following the edge between them
  - The walker tends to be trapped in a group of densely connected vertices corresponding to a community rather than walk across communities boundaries within limited number of steps
  - Can be used to compute the similarity matrix: in each walk, we keep track of the visited nodes in a se, the possibility that these nodes in the set belong in the same community is high, so the similarity values between each pair of the nodes in the set should be increased
  - The algorithm in literatures increased the similarity values between all pairs of the nodes in the set with an unique increment, i.e., it treated all the nodes in the set equally, without considering the order of nodes been visited during the walk

- Random walk based algorithm to compute the similarity matrix
  - A walker is placed in the network, at each time step, the walker is on a vertex and randomly jumps from the vertex to one of its neighbors following the edge between them
  - The walker tends to be trapped in a group of densely connected vertices corresponding to a community rather than walk across communities boundaries within limited number of steps
  - Can be used to compute the similarity matrix: in each walk, we keep track of the visited nodes in a se, the possibility that these nodes in the set belong in the same community is high, so the similarity values between each pair of the nodes in the set should be increased
  - The algorithm in literatures increased the similarity values between all pairs of the nodes in the set with an unique increment, i.e., it treated all the nodes in the set equally, without considering the order of nodes been visited during the walk
  - The node sequence have been visited in each walk forms a path from the start node to the end node, intuitively, on that path, the start node is more similar with the nodes been reached earlier than with the nodes been visited later, i.e., the similarities between nodes have some relationships with the order of nodes been visited

- Random walk based algorithm to compute the similarity matrix
  - A walker is placed in the network, at each time step, the walker is on a vertex and randomly jumps from the vertex to one of its neighbors following the edge between them
  - The walker tends to be trapped in a group of densely connected vertices corresponding to a community rather than walk across communities boundaries within limited number of steps
  - Can be used to compute the similarity matrix: in each walk, we keep track of the visited nodes in a se, the possibility that these nodes in the set belong in the same community is high, so the similarity values between each pair of the nodes in the set should be increased
  - The algorithm in literatures increased the similarity values between all pairs of the nodes in the set with an unique increment, i.e., it treated all the nodes in the set equally, without considering the order of nodes been visited during the walk
  - The node sequence have been visited in each walk forms a path from the start node to the end node, intuitively, on that path, the start node is more similar with the nodes been reached earlier than with the nodes been visited later, i.e., the similarities between nodes have some relationships with the order of nodes been visited
  - We alter the similarity computation strategy of this method, add the consideration of nodes' order been visited during the random walk

- Active learning strategy to generate the must-link constraints and cannot-link constraints
  - Every node compete with all its neighbors to be the candidate of the selection according to their degrees: $\forall v \in V$, $u \leftarrow \arg\max_{n \in N(v)}(degree(n))$; if $degree(u) > degree(v)$, then the node $v$ votes for the node $u$, otherwise, the node $v$ vote for itself

- Active learning strategy to generate the must-link constraints and cannot-link constraints
  - Every node compete with all its neighbors to be the candidate of the selection according to their degrees: $\forall v \in V$, $u \leftarrow \arg\max_{n \in N(v)}(degree(n))$; if $degree(u) > degree(v)$, then the node $v$ votes for the node $u$, otherwise, the node $v$ vote for itself
  - all the nodes with larger degree in local are extracted into the candidate set $SC$ of the selection from the network

- Active learning strategy to generate the must-link constraints and cannot-link constraints
  - Every node compete with all its neighbors to be the candidate of the selection according to their degrees: $\forall v \in V$, $u \leftarrow \arg\max_{n \in N(v)}(degree(n))$; if $degree(u) > degree(v)$, then the node $v$ votes for the node $u$, otherwise, the node $v$ vote for itself
  - all the nodes with larger degree in local are extracted into the candidate set $SC$ of the selection from the network
  - Partition the candidate set $SC$ into some clusters: every node in $SC$ is taken as a cluster initially, and for every two nodes $u$ and $v$ in $SC$, if the number of their shared neighbors is larger than half of the degree of either of them, we merge the two clusters in which $u$ and $v$ belongs respectively into one; this merge operation is repeated until some termination criteria are met

- Active learning strategy to generate the must-link constraints and cannot-link constraints
  - Every node compete with all its neighbors to be the candidate of the selection according to their degrees: $\forall v \in V$, $u \leftarrow \arg\max_{n \in N(v)}(degree(n))$; if $degree(u) > degree(v)$, then the node $v$ votes for the node $u$, otherwise, the node $v$ vote for itself
  - all the nodes with larger degree in local are extracted into the candidate set $SC$ of the selection from the network
  - Partition the candidate set $SC$ into some clusters: every node in $SC$ is taken as a cluster initially, and for every two nodes $u$ and $v$ in $SC$, if the number of their shared neighbors is larger than half of the degree of either of them, we merge the two clusters in which $u$ and $v$ belongs respectively into one; this merge operation is repeated until some termination criteria are met
  - From every cluster, the node with the largest degree and the node(s) with edge(s) connected with node(s) in other cluster(s) are selected into a set $S$

- Active learning strategy to generate the must-link constraints and cannot-link constraints
  - Every node compete with all its neighbors to be the candidate of the selection according to their degrees: $\forall v \in V$, $u \leftarrow \arg\max_{n \in N(v)}(degree(n))$; if $degree(u) > degree(v)$, then the node $v$ votes for the node $u$, otherwise, the node $v$ vote for itself
  - all the nodes with larger degree in local are extracted into the candiate set $SC$ of the selection from the network
  - Partition the candidate set $SC$ into some clusters: every node in $SC$ is taken as a cluster initially, and for every two nodes $u$ and $v$ in $SC$, if the number of their shared neighbors is larger than half of the degree of either of them, we merge the two clusters in which $u$ and $v$ belongs respectively into one; this merge operation is repeated until some termination criteria are met
  - From every cluster, the node with the largest degree and the node(s) with edge(s) connected with node(s) in other cluster(s) are selected into a set $S$
  - For each pair of nodes in $S$, we access a noiseless oracle to query the relationship between them

- Active learning strategy to generate the must-link constraints and cannot-link constraints
  - Every node compete with all its neighbors to be the candidate of the selection according to their degrees: $\forall v \in V$, $u \leftarrow \arg\max_{n \in N(v)}(degree(n))$; if $degree(u) > degree(v)$, then the node $v$ votes for the node $u$, otherwise, the node $v$ vote for itself
  - all the nodes with larger degree in local are extracted into the candidate set $SC$ of the selection from the network
  - Partition the candidate set $SC$ into some clusters: every node in $SC$ is taken as a cluster initially, and for every two nodes $u$ and $v$ in $SC$, if the number of their shared neighbors is larger than half of the degree of either of them, we merge the two clusters in which $u$ and $v$ belongs respectively into one; this merge operation is repeated until some termination criteria are met
  - From every cluster, the node with the largest degree and the node(s) with edge(s) connected with node(s) in other cluster(s) are selected into a set $S$
  - For each pair of nodes in $S$, we access a noiseless oracle to query the relationship between them

The most informative nodes and the nodes with least uncertainty are selected to access a noiseless oracle to query the relationship between them