RANDOM MODELS

RANDOM MODELS

- Given an observed graph **G,** how to create a random network with similar properties?
  - Erdős-Rényi model (ER) => Keep number of nodes and edges. Distribute edges randomly
  - Configuration model => Keep the exact same degree distribution (i.e., same #nodes, same #edges, same # of nodes with a given degree

RANDOM MODELS

Network	Degree distribution	Path length	Clustering coefficient
Real world networks	broad	short	large
Regular lattices (grid)	constant	long	large
ER random networks	Poissonian	short	small
Configuration Model	Custom, can be broad	short	small

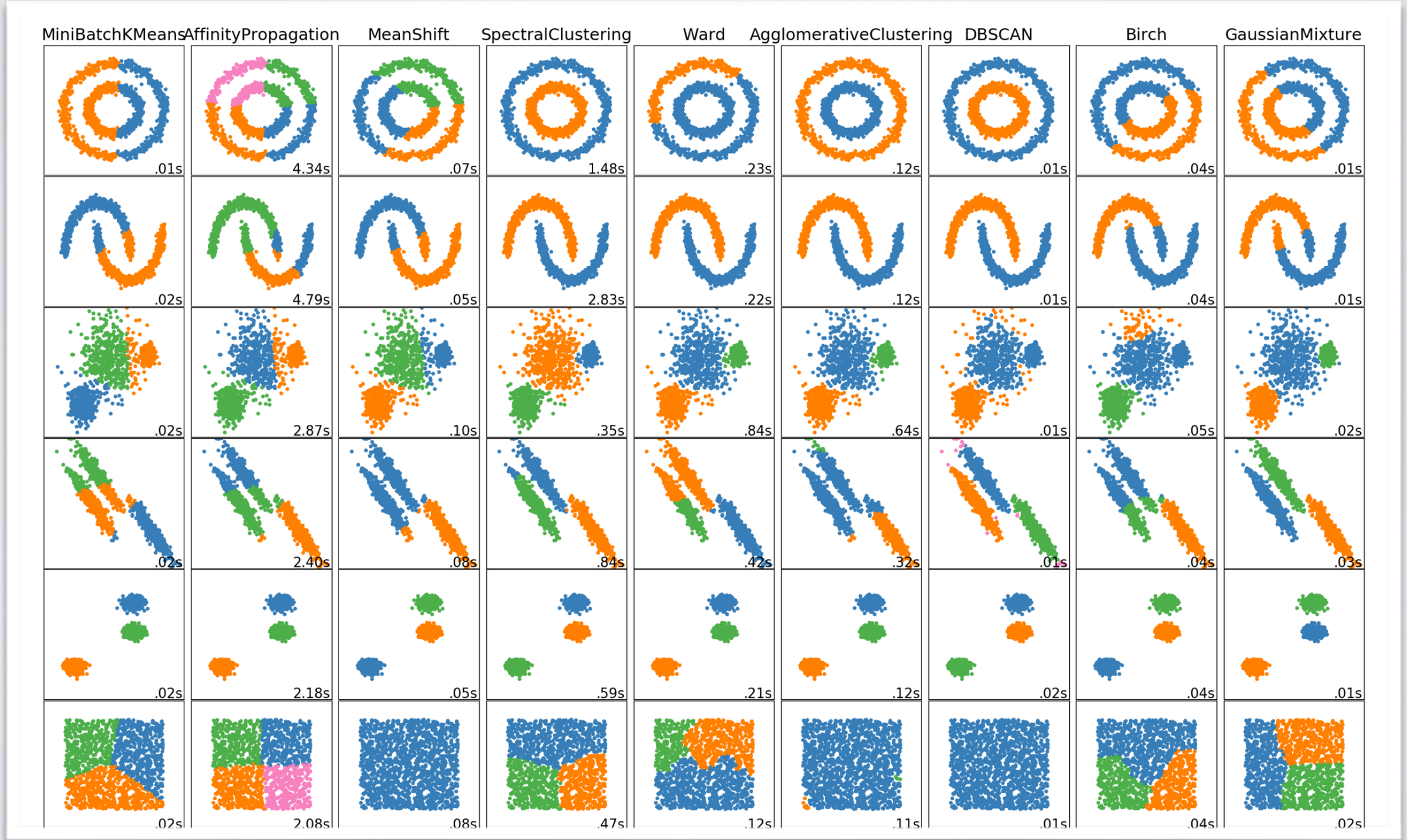
COMMUNITY DETECTION

COMMUNITY DETECTION

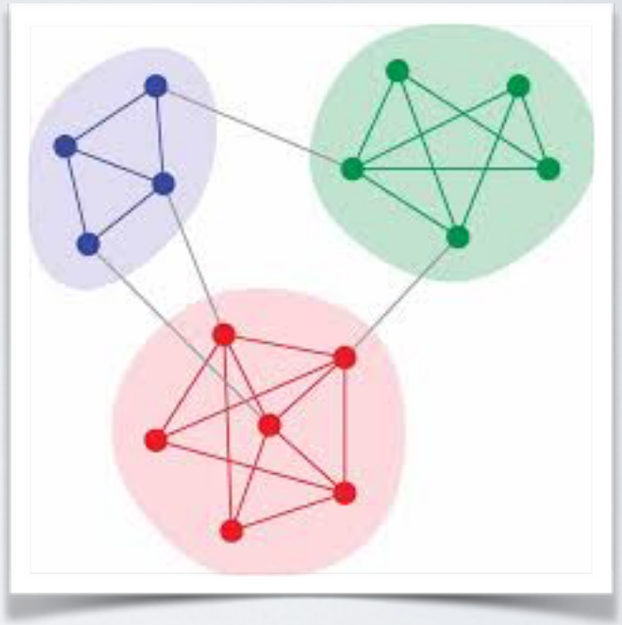
- Community detection is equivalent to "clustering" in unstructured data

- Clustering: unsupervised machine learning
  - Find groups of elements that are similar to each other
    - People based on DNA, apartments based on characteristics, etc.
  - Hundreds of methods published since 1950 (k-means)
  - Problem: what does "similar to each other" means ?

COMMUNITY DETECTION



COMMUNITY DETECTION



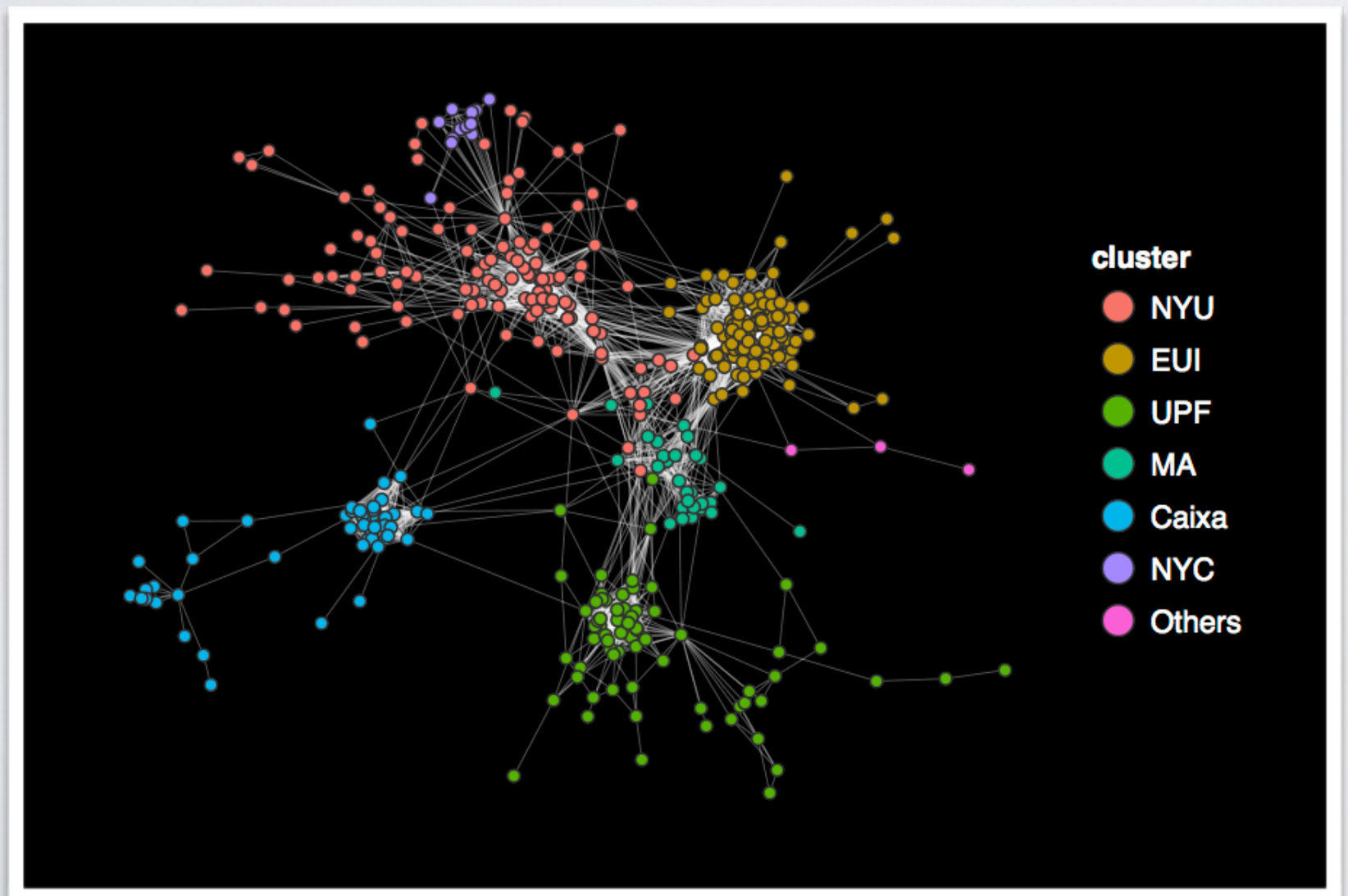
- Community detection:
  - Find groups of nodes that are:
    - Strongly connected to each other
    - Weakly connected to the rest of the network
    - Ideal form: each community is 1)A clique, 2) A separate connected component
  - No formal definition
  - Hundreds of methods published since 2003

WHY COMMUNITY DETECTION ?

- One of the key properties of complex networks was
  - High clustering coefficient
  - (friends of my friends are my friends)
- Different from random networks. How to explain it ? Evenly distributed ?
  - Watts strogatz (spatial structure?)
  - Forest fire, copy mechanism ?
- => In real networks, presence of dense groups: communities
  - Small, dense (random) networks have high density.
  - Large networks could be interpreted as aggregation of smaller, denser networks, with much fewer edges between them

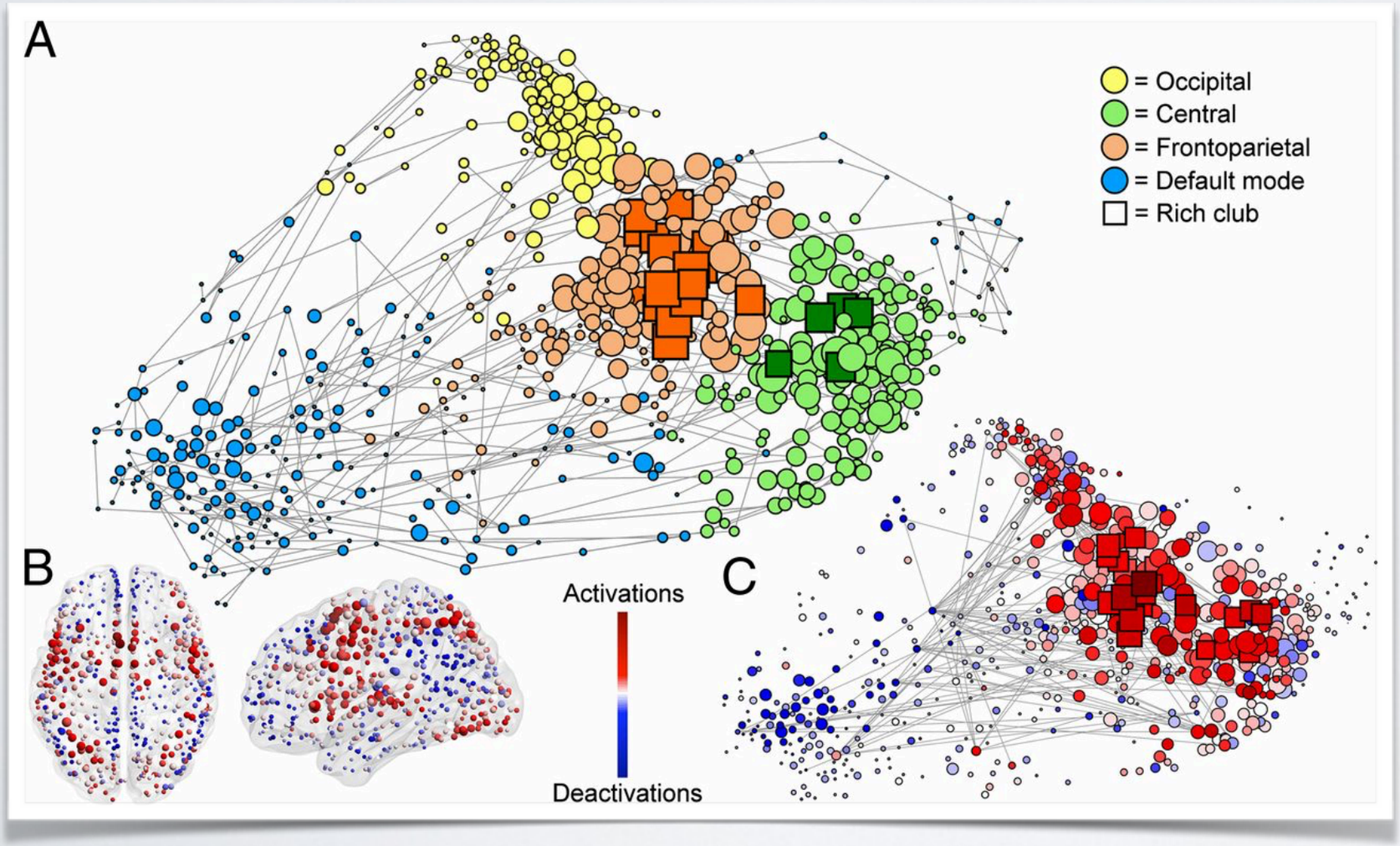
COMMUNITY STRUCTURE IN REAL GRAPHS

- If you plot the graph of your facebook friends, it looks like this



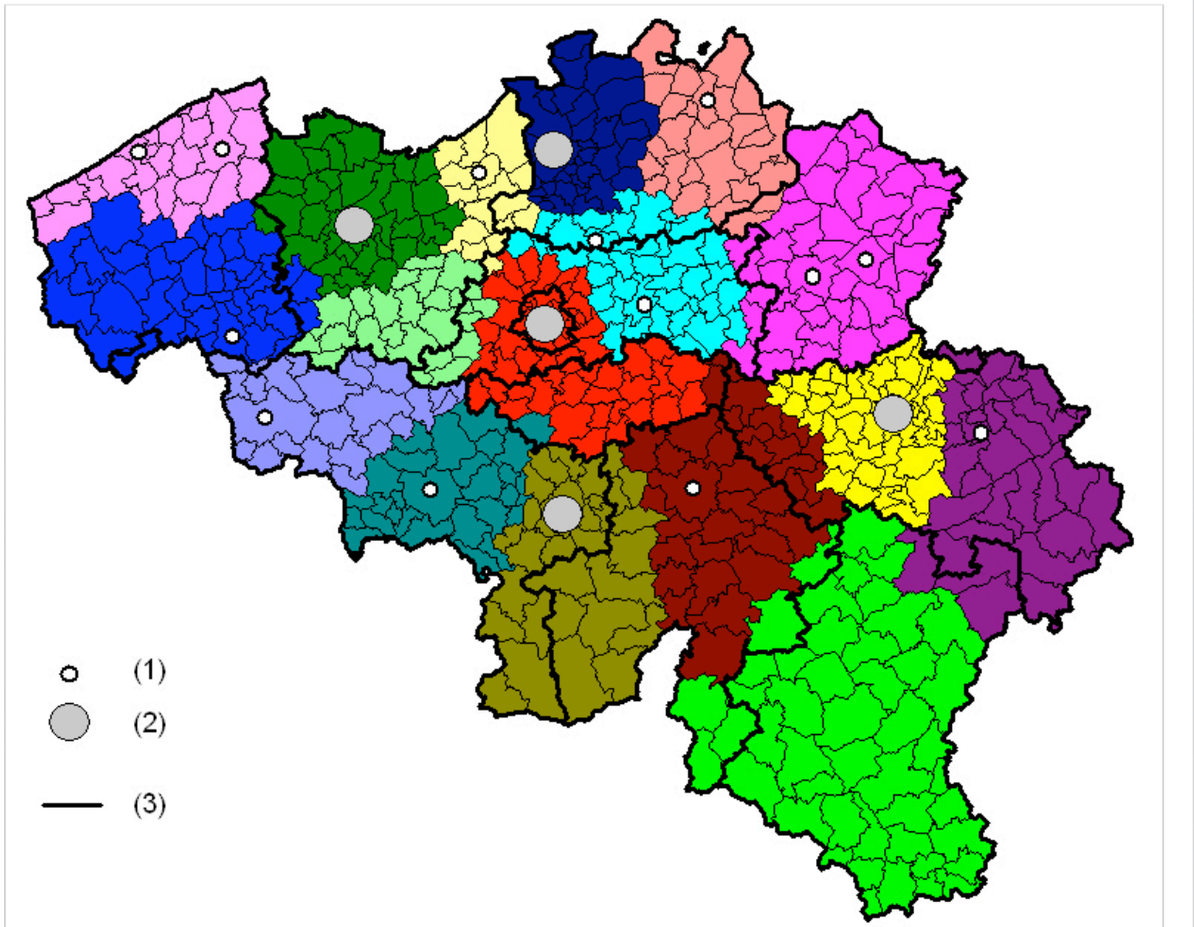
COMMUNITY STRUCTURE IN REAL GRAPHS

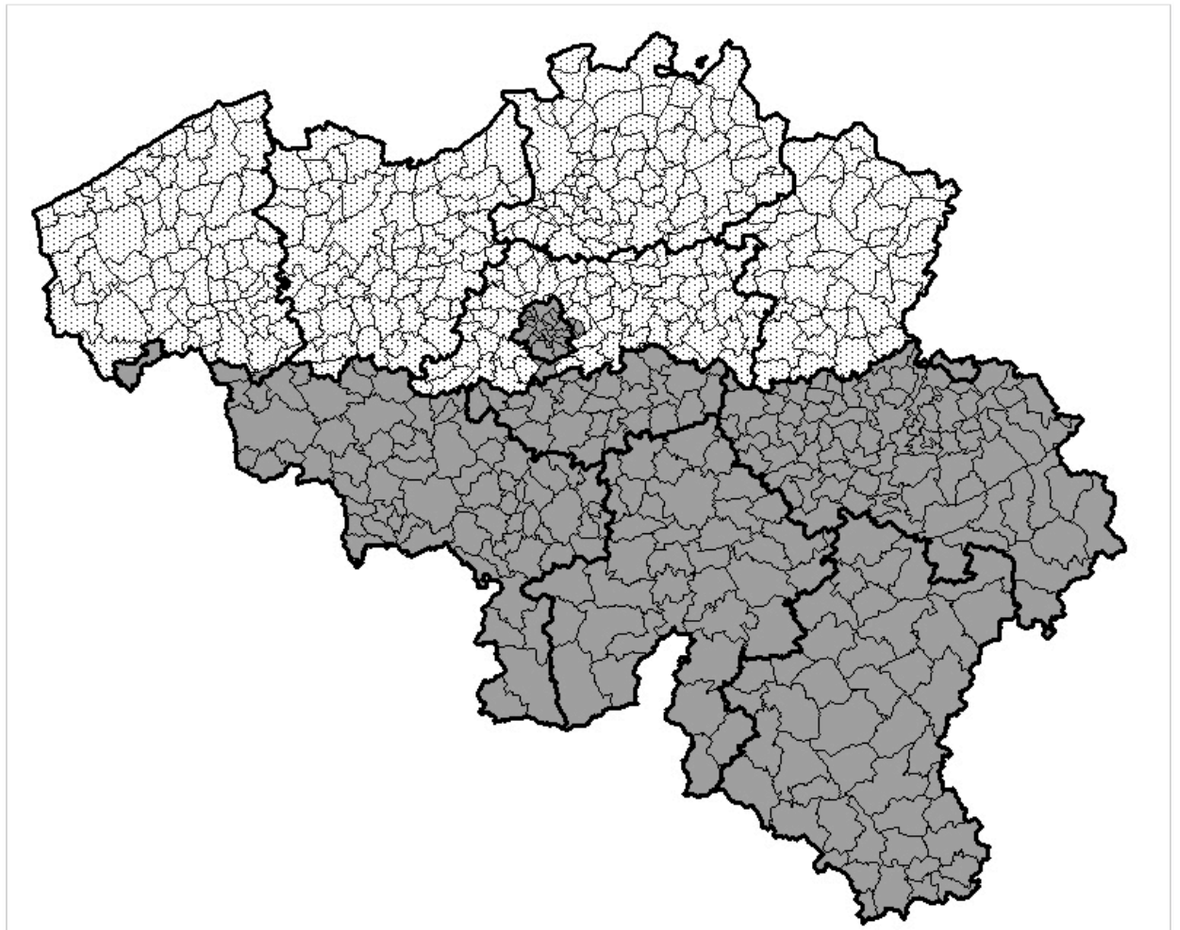
- Connections in the brain ?



COMMUNITY STRUCTURE IN REAL GRAPHS

- Phone call communications in Belgium ?

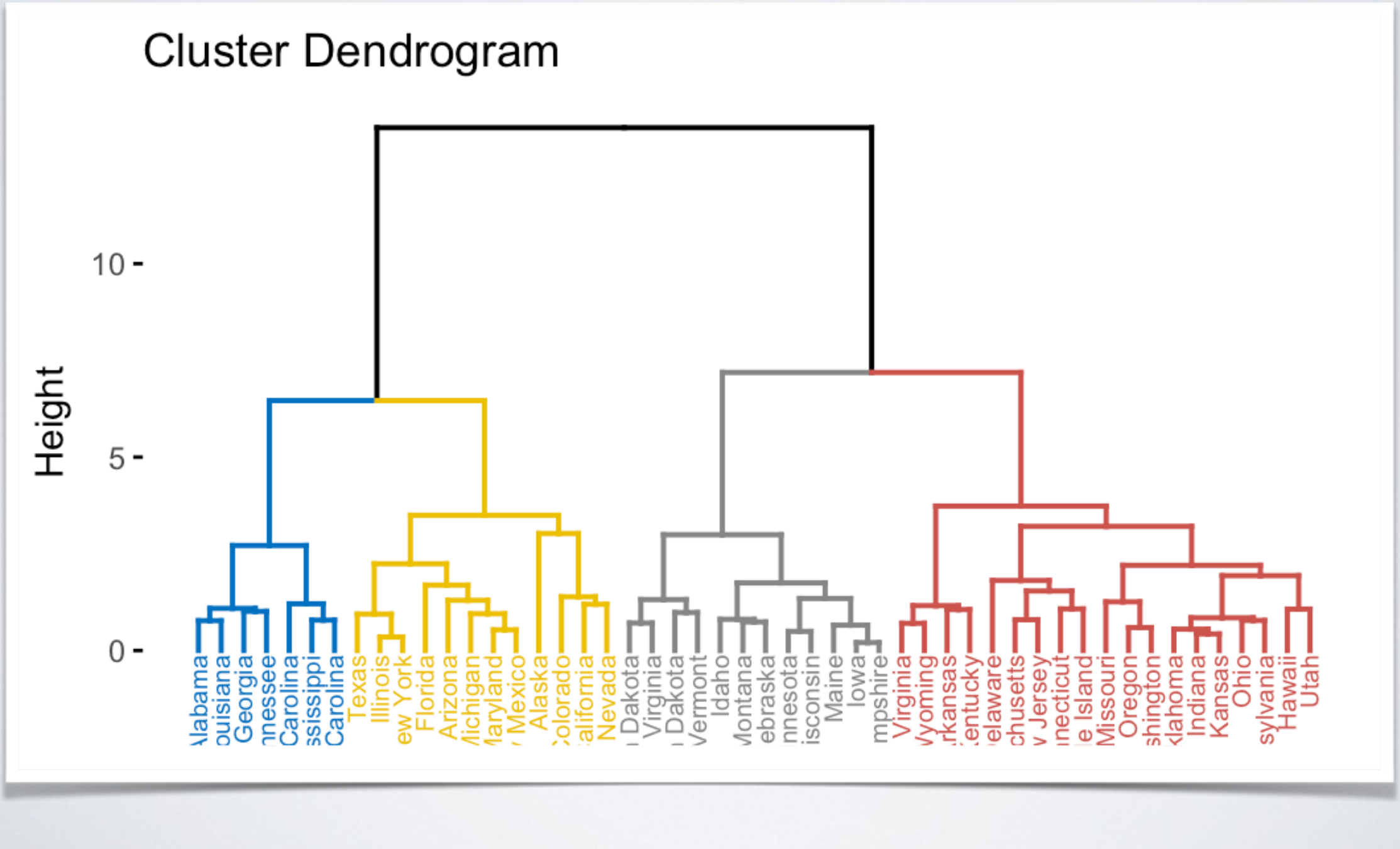




FIRST METHOD BY GIRVAN & NEWMAN

- 1)Compute the betweenness of all edges
- 2)Remove the edge of highest betweenness
- 3)Repeat until all edges have been removed
  - Connected components are communities
- => It is called a *divisive* method
- =>What you obtain is a dendrogram
- How to cut this dendrogram at the *best* level ?

FIRST METHOD BY GIRVAN & NEWMAN



FIRST METHOD BY GIRVAN & NEWMAN

- Introduction of the **Modularity**

- The modularity is computed for a partition of a graph
  - (each node belongs to one and only one community)

- It compares :
  - The **observed** *fraction of edges inside communities*
  - To the **expected** *fraction of edges inside communities* in a random network
    - Random network: Configuration model

MODULARITY

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

Original formulation

MODULARITY

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

Sum over all pairs of nodes

MODULARITY

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

1 if in same community

MODULARITY

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

1 if there is an edge between them

MODULARITY

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w)$$

Probability of an edge in a random network

MODULARITY

$$Q = \frac{1}{(2m)} \sum_{vw} \left[A_{vw} - \frac{k_v k_w}{(2m)} \right] \delta(c_v, c_w) = \sum_{i=1}^{c} (e_{ii} - a_i^2)$$

$$\sum_{i=1}^{c} (e_{ii} - a_i^2)$$

Sum for each community **Expected internal edges**

Internal edges

$$a_i = \frac{k_i}{2m} = \sum_j e_{ij}$$

$$e_{ij} = \sum_{vw} \frac{A_{vw}}{2m} 1_{v \in c_i} 1_{w \in c_j}$$

MACHINE LEARNING ON GRAPHS

MACHINE LEARNING

- Examples of supervised machine learning
  - Given properties of an apartment, predict its energy consumption
  - Given a picture, recognize objects in it
  - Given a student profile, predict its success
  - Given a criminal profile, predict its probability of recidivism
  - Given past values and collected news, predict market fluctuations
  - Given a patient profile, predict effect of a drug
  - Given a fingerprint/face, recognize the user
  - …

SUPERVISED MACHINE LEARNING I: LINK PREDICTION

LINK PREDICTION

- Do you know why Facebook "People you may know" is so accurate?
- How youtube/Spotify/amazon recommend you the right item?
- =>Link prediction
  - More generally, recommendation, but link prediction is a popular way to do it

LINK PREDICTION

- Observed network: current state

- Link prediction: What edge
  - Might appear in the future (*future link* prediction)
  - Might have been missed (*missing link* prediction)

HEURISTICS

- Network science experts can design **heuristics** to predict where new edge might appear/be missing

- Principle: design a score based on network topology f(v1,v2) which, given two nodes, express their likeliness of being connected (if they aren't already)
  - Common neighbors
  - Jaccard coefficient
  - Hub promoted
  - Adamic Adar
  - Ressource allocation
  - Community based

Zhou, T., Lü, L., & Zhang, Y. C. (2009). Predicting missing links via local information. *The European Physical Journal B*, *71*(4), 623-630.

COMMON NEIGHBORS

- "Friends of my friends are my friends"
- High clustering in most networks
- =>The more friends in common, the highest probability to become friends

$$\mathrm{CN}(x, y) = |\Gamma(x) \cap \Gamma(y)|$$

$\Gamma(x) =$ Neighbors of x

PREDICTION

- How to predict links based on Common Neighbors (CN)?
- For each pair of unconnected nodes, compute CN
- =>Ordered list of pairs from more likely to less likely
- The *k* first edges of the list are the *k* most likely to appear

JACCARD COEFFICIENT

- Used in many applications:
  - Measure of similarity of sets of different sizes
- Intuition:
  - Two people who have 4 friends, 3 in common:
    - =>high probability
  - Two people who know 1000 people, only 3 in commons
    - =>Lower probability

$$\mathrm{JC}(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x) \cup \Gamma(y)|}$$

ADAMIC ADAR

- Intuition:
  - For previous scores: all common nodes are worth the same
  - For AA:
    - A common node with ONLY them in common is worth the most
    - A common node connected to everyone is worth the less
    - The higher the size of its neighborhood, the lesser its value

$$\mathrm{AA}(x,y) = \sum_{z \in \Gamma(x) \cap \Gamma(y)} \frac{1}{log|\Gamma(z)|}$$

OTHER SCORES

- Communities
  - Two nodes are in the same community =1, otherwise 0
  - Two nodes are in the same community, score=density of the community
- Distance based:
  - Length of the shortest path
  - Number of paths of length *l* between the nodes
- *Etc.*

WHICH ONE IS BEST?

- Compute on many networks using AUC score

Indices	PPI	NS	Grid	PB	INT	USAir
CN	0.889	**0.933**	**0.590**	0.925	**0.559**	0.937
Salton	0.869	0.911	0.585	0.874	0.552	0.898
Jaccard	0.888	**0.933**	**0.590**	0.882	**0.559**	0.901
Sørensen	0.888	**0.933**	**0.590**	0.881	**0.559**	0.902
HPI	0.868	0.911	0.585	0.852	0.552	0.857
HDI	0.888	**0.933**	**0.590**	0.877	**0.559**	0.895
LHN1	0.866	0.911	0.585	0.772	0.552	0.758
PA	0.828	0.623	0.446	0.907	0.464	0.886
AA	0.888	0.932	**0.590**	0.922	**0.559**	0.925
RA	**0.890**	**0.933**	**0.590**	**0.931**	**0.559**	**0.955**

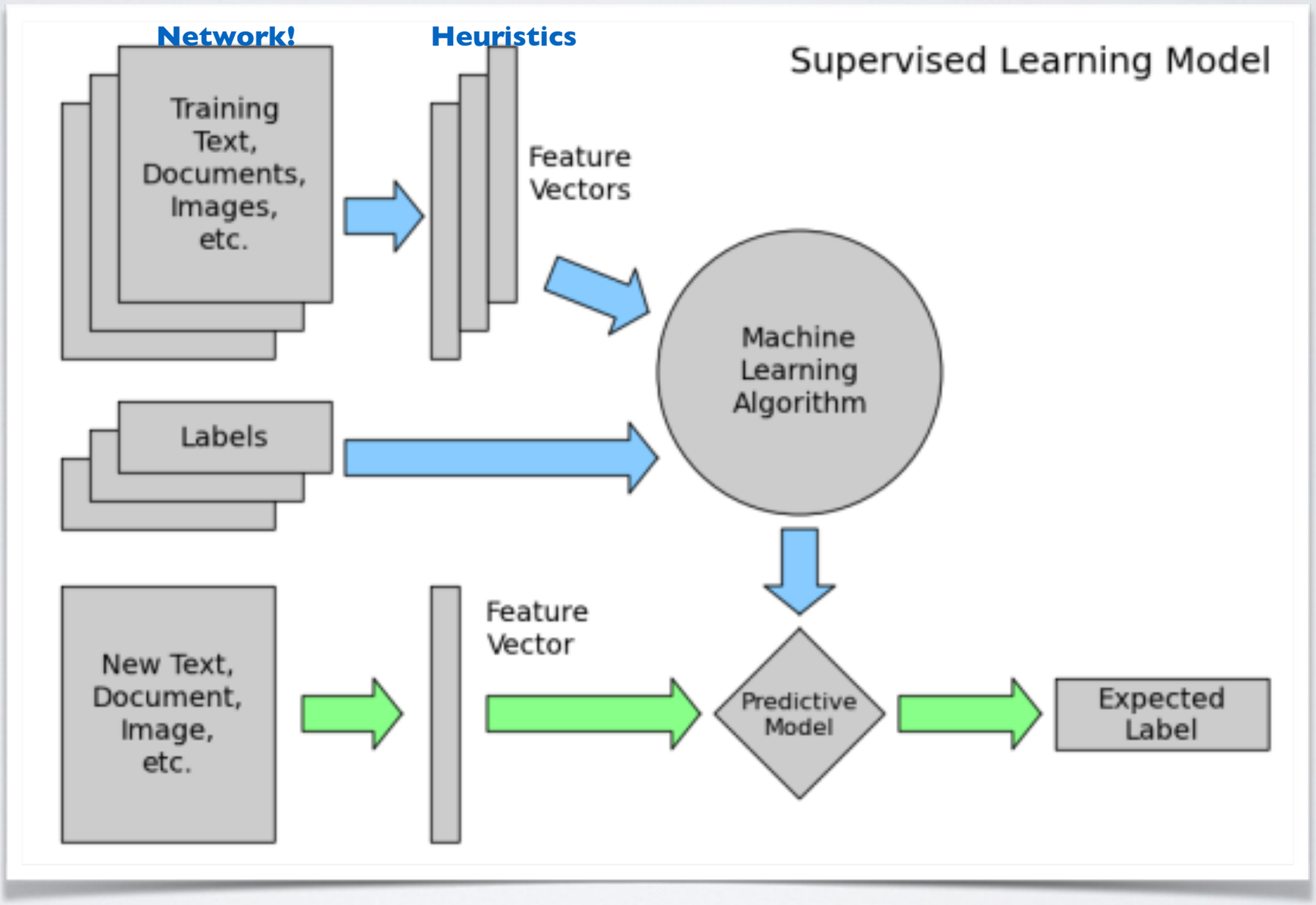
Zhou, T., Lü, L., & Zhang, Y. C. (2009). Predicting missing links via local information. *The European Physical Journal B*, *71*(4), 623-630.

SUPERVISED MACHINE LEARNING

- Use Machine Learning algorithms to **learn** how to combine heuristics for optimizing predictions

- Two phases:
  - Training: show features + associated value
  - Testing: Try to predict value from features

SUPERVISED MACHINE LEARNING



SUPERVISED MACHINE LEARNING

- Our features: similarity indices (CN, AA, PA, …)
  - One (limited interest) or, obviously, several
  - One could add nodes attributes if known (age, salary, etc.)

- Our label/value to predict: *Link* or *No link* (2 **classes**)

- These types of ML algorithms are called **classifiers**
  - Logistic Classifier
  - Decision Tree Classifier
  - Neural networks Classifier
  - …

SUPERVISED MACHINE LEARNING

- Scores of methods, very different in their mechanisms, but same **input** and **output**

```
#lm = linear_model.LinearRegression()
#lm = linear_model.ElasticNet()
#lm = ensemble.GradientBoostingRegressor()
#lm = ensemble.RandomForestRegressor()
lm = MLPRegressor(hidden_layer_sizes=(3,3,3))



lm.fit(X_train,y_train)
```

NODE CLASSIFICATION

Bhagat, S., Cormode, G., & Muthukrishnan, S. (2011). Node classification in social networks. In *Social network data analytics* (pp. 115-148). Springer, Boston, MA.

NODE CLASSIFICATION

- For the node classification task, we want to predict the class/ category (or numerical value) of some nodes
  - Missing values in a dataset
  - Learn to predict, in a social network, individuals':
    - Political position, opinion on a given topic, possible security threat, …
    - Interests, tastes, etc.
    - Age, genre, sexual orientation, language spoken, salary, etc.
    - Fake accounts, spammers, bots, malicious accounts, etc.
    - …
  - Wikipedia article category, types of road in an urban network, etc.

NODE CLASSIFICATION

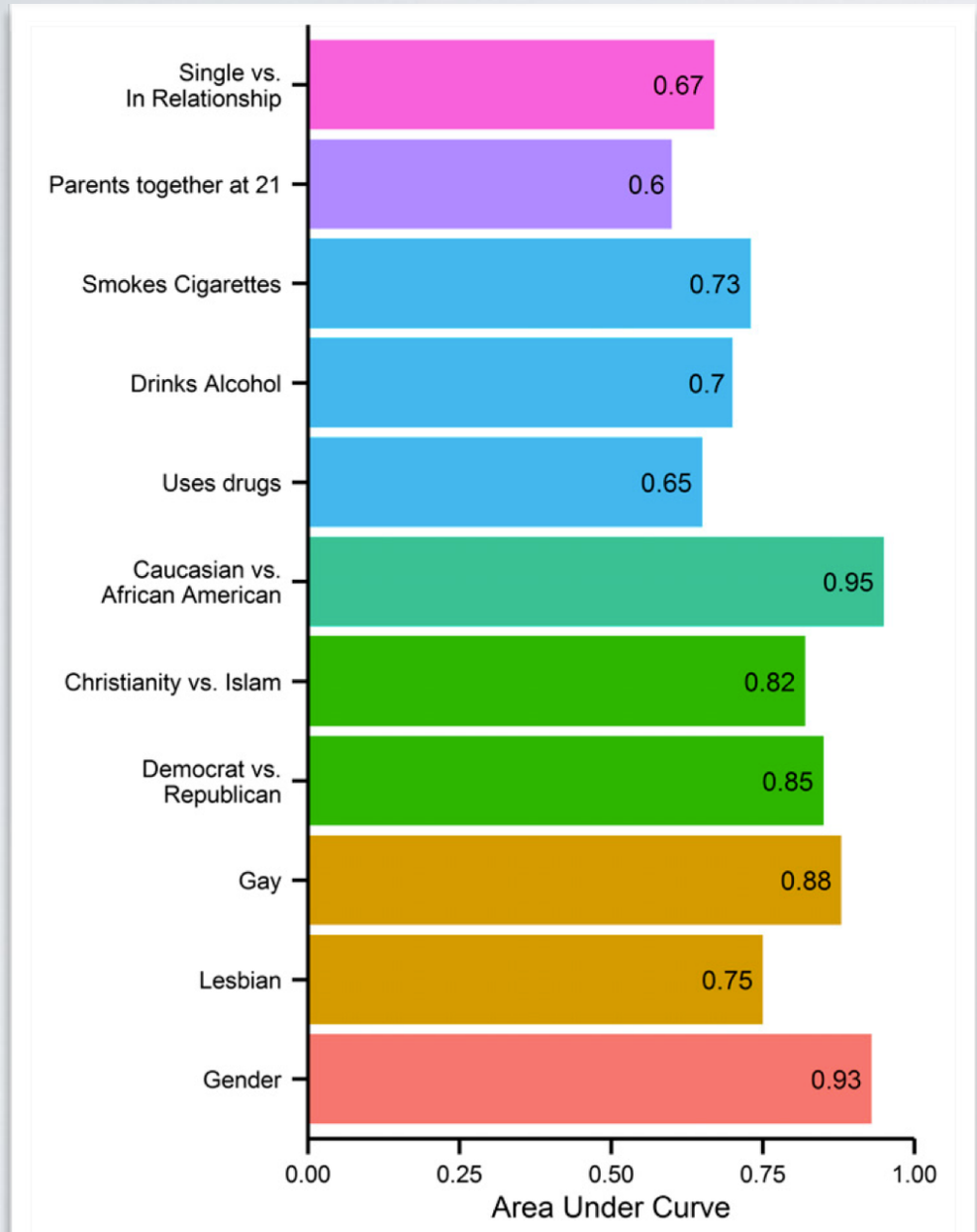


Fig. 2. Prediction accuracy of classification for dichotomous/dichotomized attributes expressed by the AUC.

Example of risks

Kosinski, M., Stillwell, D., & Graepel, T. (2013). Private traits and attributes are predictable from digital records of human behavior. *Proceedings of the National Academy of Sciences*, *110*(15), 5802-5805.

NODE FEATURES

- Non-network approach: Use a classification algorithm based on features of the node itself (age, salary, etc.)
- The network structure can be integrated using node centralities: Degree, clustering coefficient, betweenness, etc.
- But we can do much better:
  - "Tell me who your friends are, and I will tell you who you are"

NEIGHBORHOOD BASED CLASSIFICATION

- Classification based on the distribution of features in the neighborhood

- For each node, compute the distribution of labels in its neighborhood (vectors of length *m*, with *m* the set of all possible labels)
  - Pick the most frequent
    - e.g., political opinions
  - Train a classifier on this distribution
    - e.g., distribution of age, language in the neighborhoods to recognize bots (unexpectedly random)

NEIGHBORHOOD BASED CLASSIFICATION

- More advanced methods exist based on random walks, Graph embedding, Graph Neural Networks…

- Idea: take into account not only the first neighbors, and automatically detect how strongly each node depends on its neighbors

EXERCICES

1. Check the notebook examples on the class webpage.
2. Compute classic centralities with networkx
3. Create a randomized version of a network (you can use the **double_edge_swap** or **configuration_model** function of networkx, for instance)
    1. Compare the clustering coefficient and average shortest paths in the original and random graphs
    2. Compare centralities of the same nodes in both graphs
4. Other possible exercices:
    1. Using the website wallet explorer (https://www.walletexplorer.com), you can obtain the class of several clusters (Exchanges, Gambling, Pools, Services). Train a classifier to predict the class of other clusters
    2. Compute the sum of bitcoins received and sent by each cluster. Estimate the number of bitcoins that should be exchanged between them if exchanges were random (expected wieight). Compare real values with this reference to find the most "unlikely" transactions in the network.