4- LINK PREDICTION
&
GRAPH RECONSTRUCTION

LINK PREDICTION

- Do you know why Facebook "People you may know" is so frighteningly accurate?
- How youtube/Spotify/amazon recommend you the right item?
- =>Link prediction

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)

LINK PREDICTION

- Link prediction based on network properties:
  - Local: High clustering (friends of my friends will become my friends)
  - Global: Two unrelated hubs more likely to have links that unrelated small nodes
  - Meso-scale organisation: two nodes in the same community…
- Link prediction can also be based on node properties
  - Combining with usual machine learning, outside of the scope of this course

SIMILARITY INDICES UNSUPERVISED

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

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

PREDICTION

- How to predict links based on Common Neighbors?
- For each pair of unconnected nodes, compute CN
- =>Ordered list of pairs from more probable to less probable
- 5 most probable for a node? Take top 5 among non-already neighbors

JACCARD COEFFICIENT

- Used in many applications:
  - Measure of similarity of sets of different sizes

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

- Intuition:
  - Two people who know only the same 3 people, but 1 not shared:
    - =>high probability
  - Two people who know 1000 people, only 3 in commons
    - =>Lower probability

HUB PROMOTED

- Intuition:
  - One person do “everything as” the other one
  - One person know 1000 people and the other one 3
    - =>higher probability than
  - Two people who know 1000 people, only 3 in commons

$$\mathrm{HP}(x, y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{min(|\Gamma(x)|, |\Gamma(y)|)}$$

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

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

RESSOURCE ALLOCATION

- Similar to Adamic Adam, penalize more higher degrees

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

MANY OTHER SCORES

Sorenson Index

$$\mathrm{SI}(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| + |\Gamma(y)|}$$

Salton Cosine Similarity

$$\mathrm{SC}(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{\sqrt{|\Gamma(x)| \cdot |\Gamma(y)|}}$$

Hub Depressed

$$\mathrm{HD}(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{max(|\Gamma(x)|, |\Gamma(y)|)}$$

Leicht-Holme-Nerman

$$\mathrm{LHN}(x,y) = \frac{|\Gamma(x) \cap \Gamma(y)|}{|\Gamma(x)| \cdot |\Gamma(y)|}$$

PREFERENTIAL ATTACHMENT

- Preferential attachment:
  - Model of network growth based on the idea that **the rich get richer**
  - Every time a node join the network, it creates a link with nodes with probability =current degree
  - Generates power law distribution of degrees
    - But, in my opinion (and others), very unrealistic networks

- Score not based on common neighbors

- Intuition: Two nodes with many neighbors more likely to have new ones than nodes with few neighbors

$$\mathrm{PA}(x,y) = |\Gamma(x)| \cdot |\Gamma(y)|$$

WHICH ONE IS BEST?

- Compute on many networks using AUC score (Explained later)

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**

[Lu 2010]

WHICH ONE IS BEST?

- Compute on many networks using AUC score (Explained later)

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**

[Lu 2010]

WHICH ONE IS BEST?

- Compute on many networks using AUC score (Explained later)

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**

[Lu 2010]

WHICH ONE IS BEST?

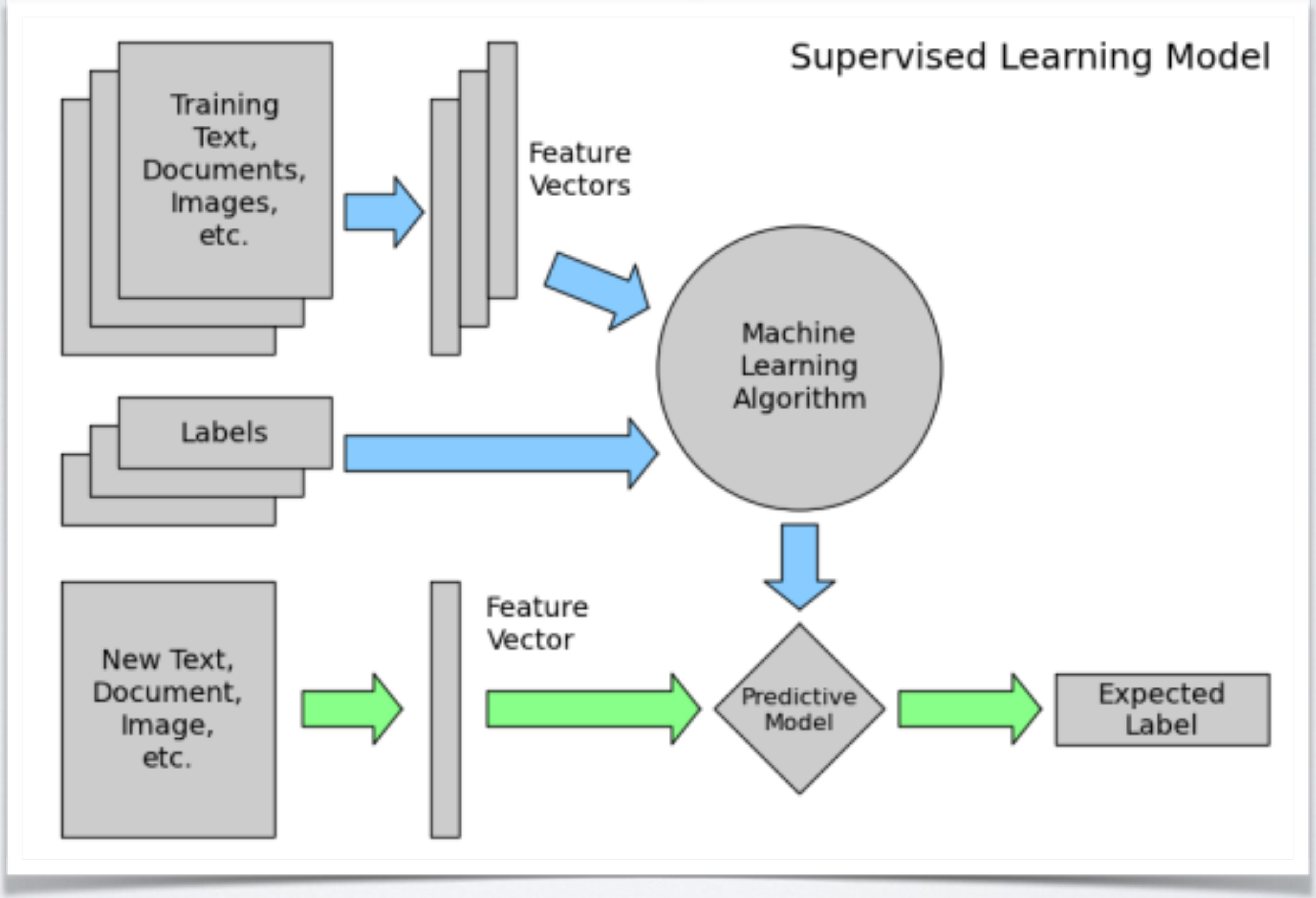
- All scores but PA are based on common neighbors
- =>No links between nodes at graph distance >2
- Inconsistent with observations
- =>We should combine PA and others

SIMILARITY INDICES
SUPERVISED

SUPERVISED MACHINE LEARNING

- Use Machine Learning algorithms to learn to predict something
- Takes **features** as input, provides prediction as output
- Two phases:
  - Training: show features + associated value
  - Testing: Try to predict value from features

SUPERVISED MACHINE LEARNING

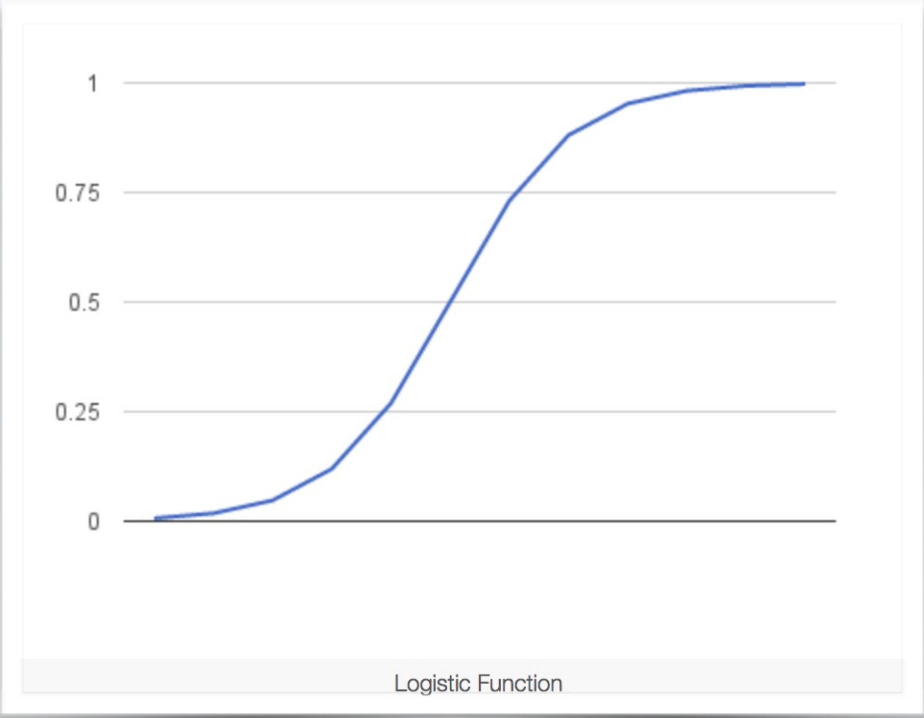


SUPERVISED MACHINE LEARNING

- Our features: similarity indices (CN, AA, PA, …)
  - One or, obviously, several
- Our 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
  - …

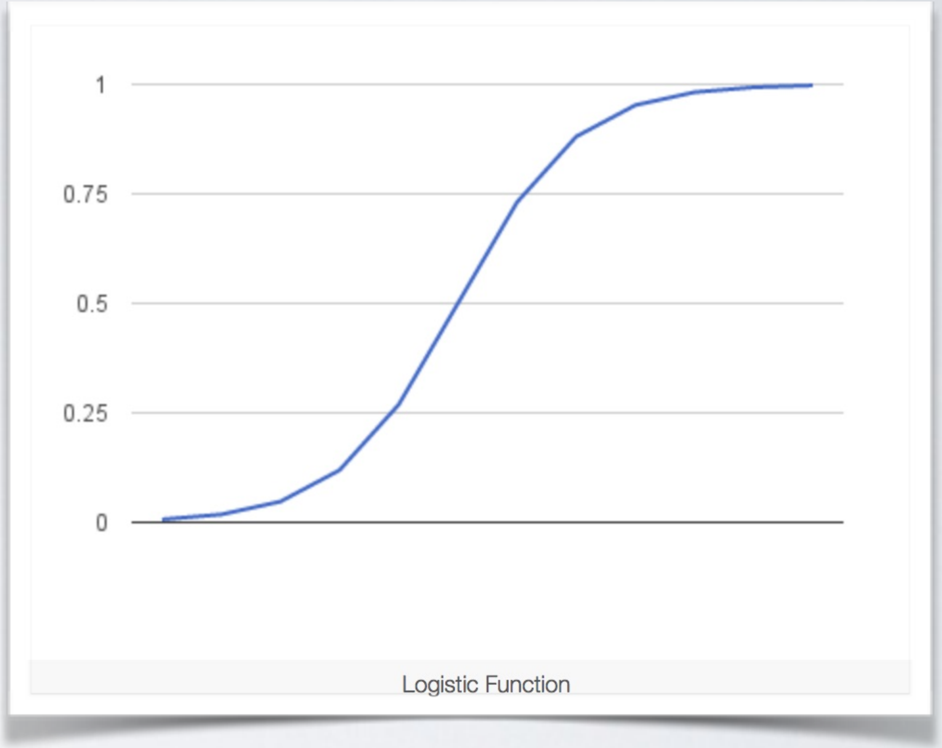
LOGISTIC CLASSIFICATION

- Very short introduction
- Value to predict:
  - 0 (no edge)
  - 1 (edge)
- Linear relations between variables
  - But constrained to [0-1] (unlike linear regression)



$$Ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

LOGISTIC CLASSIFICATION



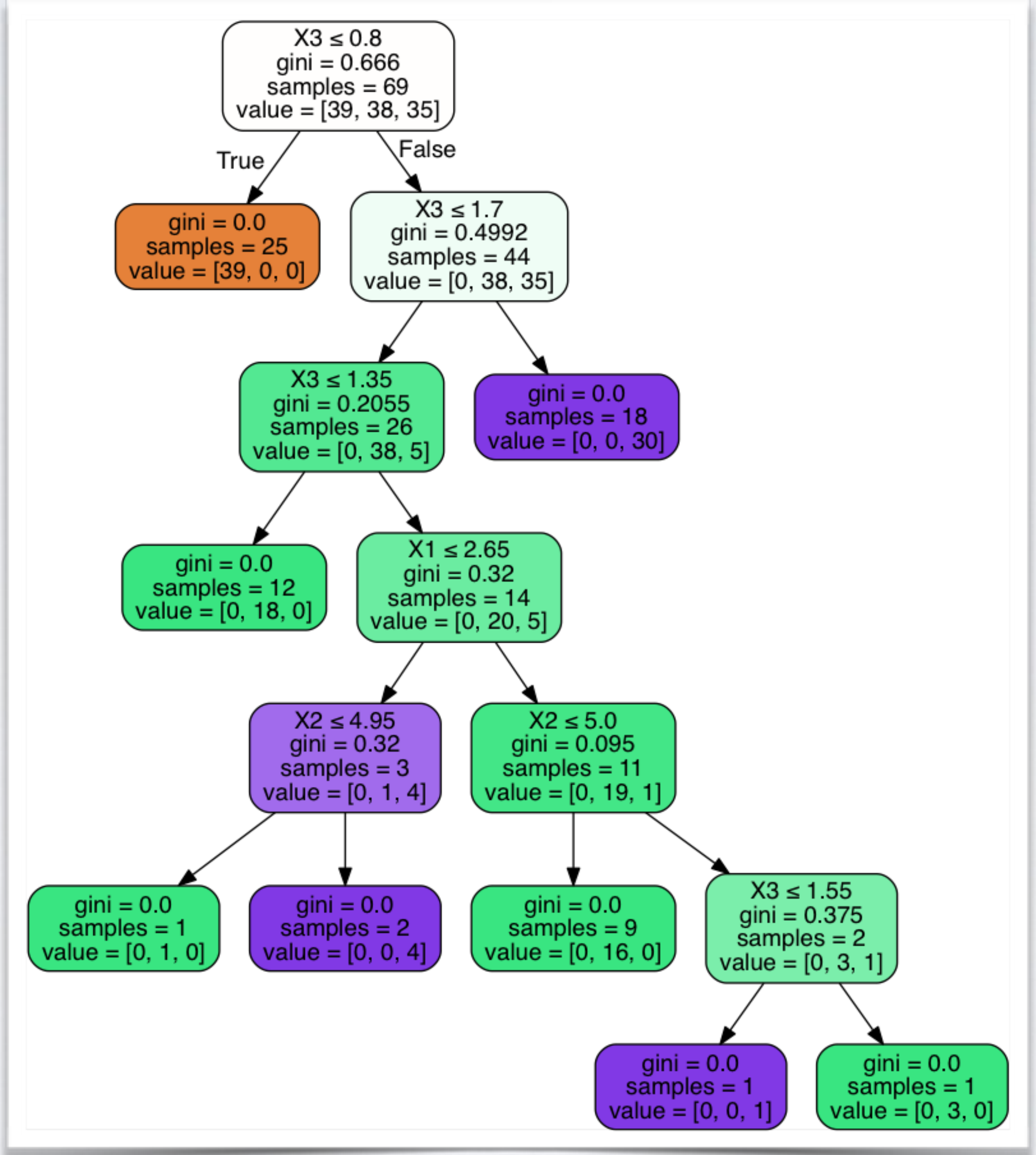
- Very short introduction
- Value to predict:
  - 0 (no edge)
  - 1 (edge)
- Linear relations between variables
  - But constrained to [0-1] (unlike linear regression)

Probability that Y=1

$$Ln\left(\frac{P}{1-P}\right) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \ldots + \beta_k X_k$$

DECISION TREES

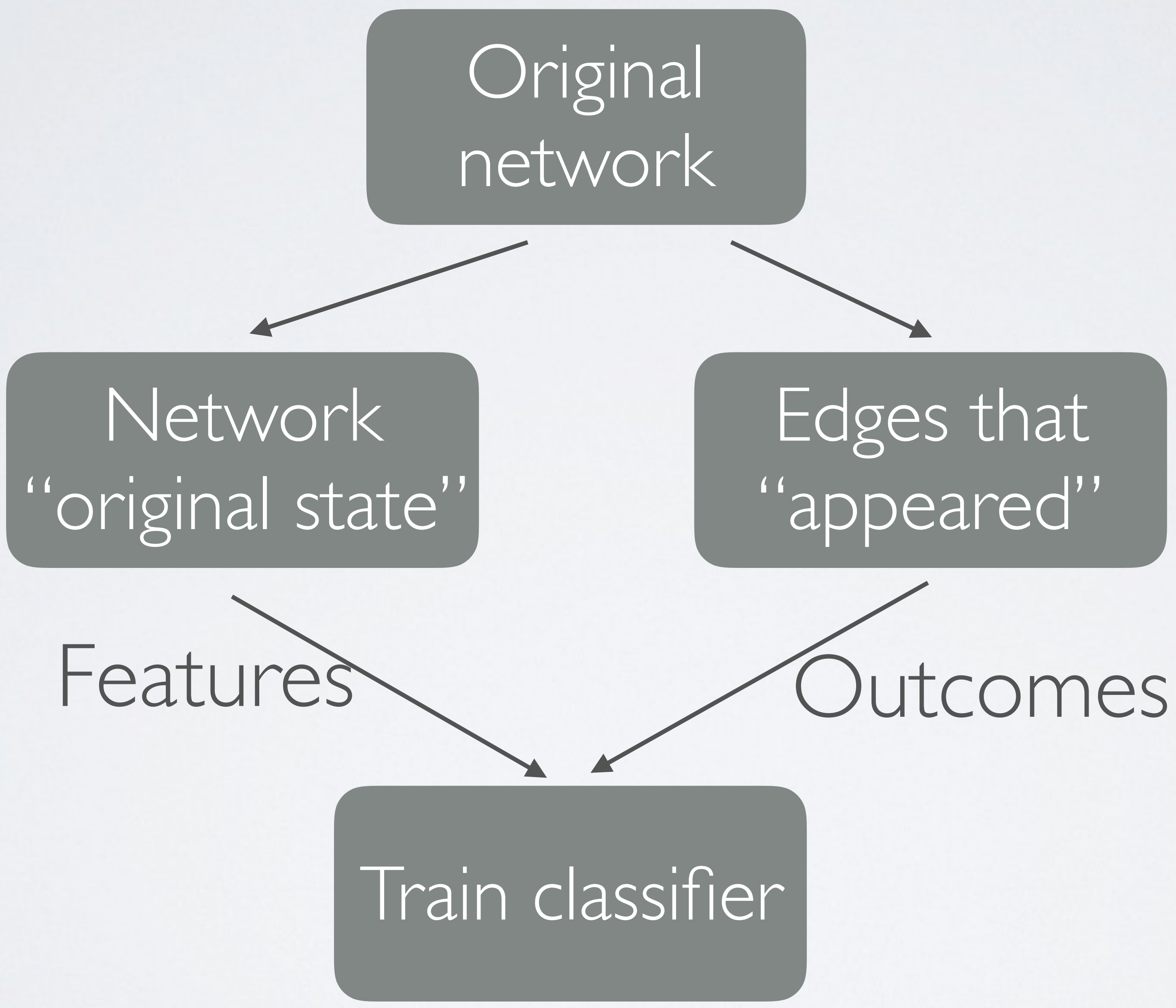
- Split recursively data in 2 to maximize homogeneity



CREATING TRAINING SET

- Problem: we need a training set: examples of correct link prediction
  - We have only a network with edges
- Solution:
  - 1)remove randomly edges
  - 2)Consider the resulting network as "original state"
  - 3)Consider the removed edges as edges to predict

CREATING TRAINING SET



CREATING TRAINING SET

- Example of possible outcomes with a decision tree:

- If CN <1
  - IF PA>1000 => Predict 1
  - ELSE => Predict 0

- ELSE
  - IF PA > 10000 => Predict 1
  - ELSE
    - IF AA > 10 => Predict 1
    - ELSE
    - IF JC < 0.2 => Predict 0
    - …

INTERPRETATION OF CLASSIFIER

- Classifier predict outcomes given features

- Two ways to obtain outcome:
  - Class => 0 or 1.
    - The classifier decides how many edges will appear
  - Probability
    - Distance to separating hyperplane
    - The most "certain" is the decision, the highest value
    - Same interpretation as scores such as CN or AA
    - We "decide" how many edges we want (Top k …)

- Results: slight but robust improvement over indexes alone

EVALUATION

- How to evaluate the quality of link prediction?
- 1)Create a train and test set
  - Artificially created or collected as a dynamic network
- 2)Use an appropriate evaluation measure
  - AUROC (or AUC) (Area Under the Receiver Operating Curve)
  - Average Precision
  - MAP (Mean Average Precision)

EVALUATION

- Creating the test dataset
  - Single network available:
    - Remove edges randomly
    - Removed edges are your test edges
  - Dynamic network (edges appear at a given date)
    - Choose a date to split data in 2
    - The first part is the training set
    - The second part is the test set
    - =>More realistic
    - =>Usually harder
    - =>Depends more on the chosen network

EVALUATION MEASURES

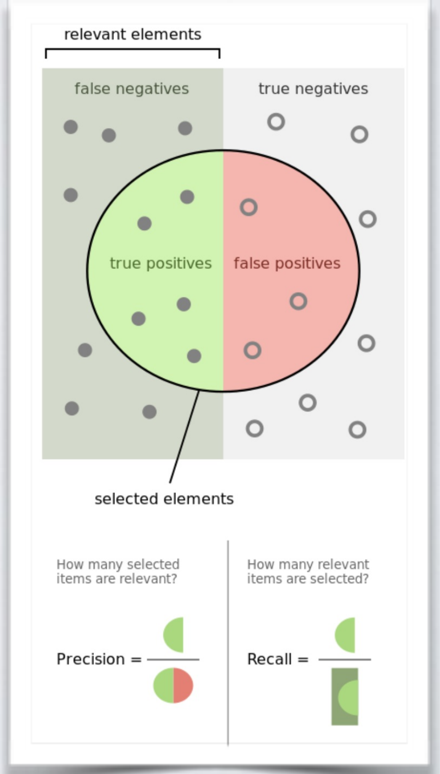
- Naive approach: Accuracy
- Simple, intuitive evaluation of a classifier:
  - Given a test set with positive and negative items, how many items correctly classified?
  - Problem: Our test set has only positive examples. We need to add negative examples (pairs of nodes without a new edge)
  - =>Result depends completely on the ratio between positive and negative examples
    - a *balanced* dataset is usually recommended, but useless for real applications
    - A dataset respecting the *density* is not adapted: the trivial solution “*always predict no edge*” has a very high score and is very hard to beat

AVERAGE PRECISION

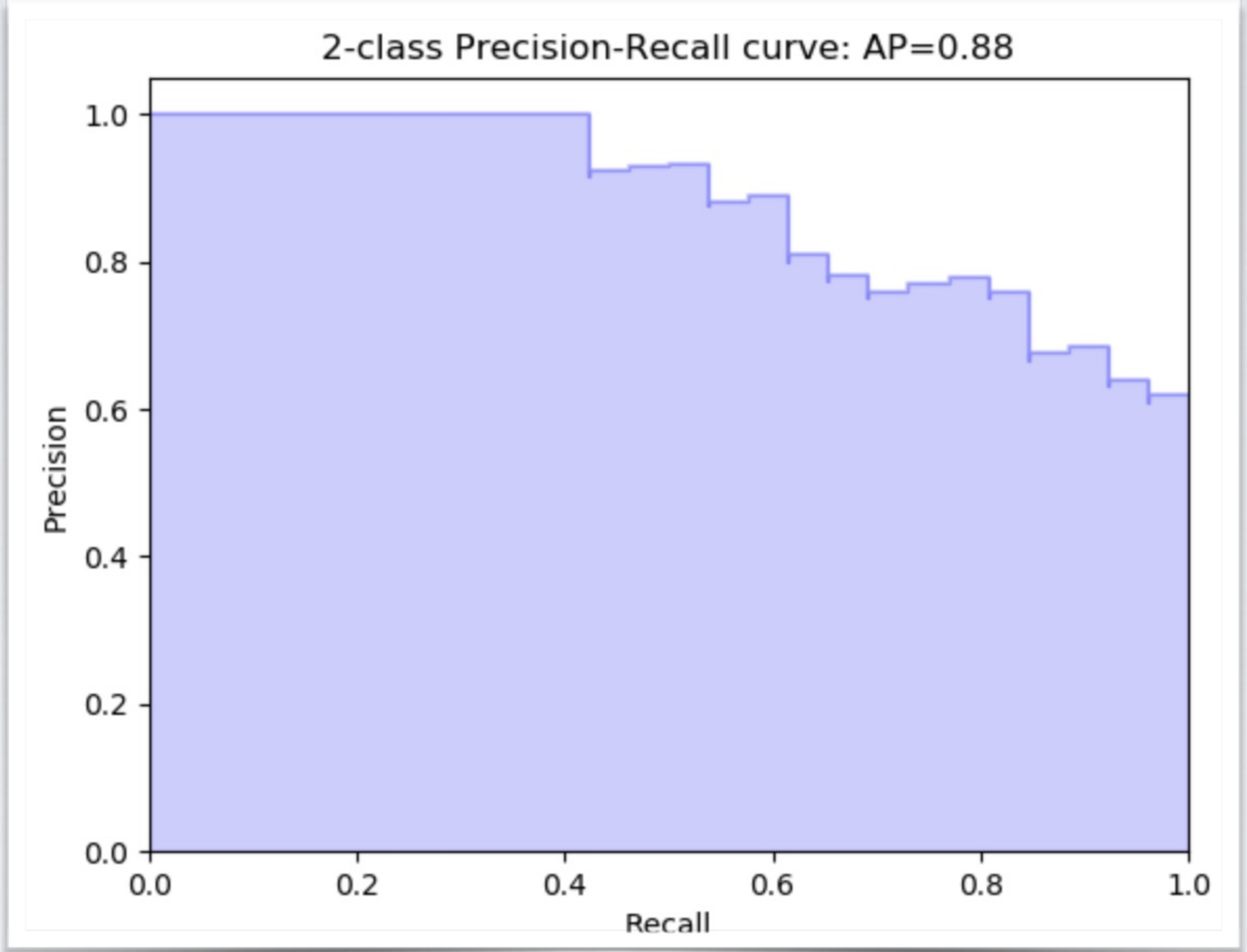
$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

- Better solution: Average Precision
- Let's define Precision and Recall:
- For a **desired** #of edges (top k)
  - 1 Value of Precision
  - 1 Value of Recall
- If we **increase** the number of desired edges:
  - Precision tend to decrease (decisions on harder cases)
  - Recall tend to increase (less missed edges)
  - => False negatives transformed in true positives or false negatives



AVERAGE PRECISION



Average precision:
Area Under the Precision/Recall Curve

AVERAGE PRECISION

- Input:
  - For each pair of node: score.
  - Rank by decreasing order.
  - Compute P/R for each value of desired edges
- Pros:
  - No need to arbitrarily decide # desired edges !
- Cons:
  - Result still depends on the ratio of really positive edges in the test set
  - =>Gives higher scores to solutions making less mistakes in the beginning

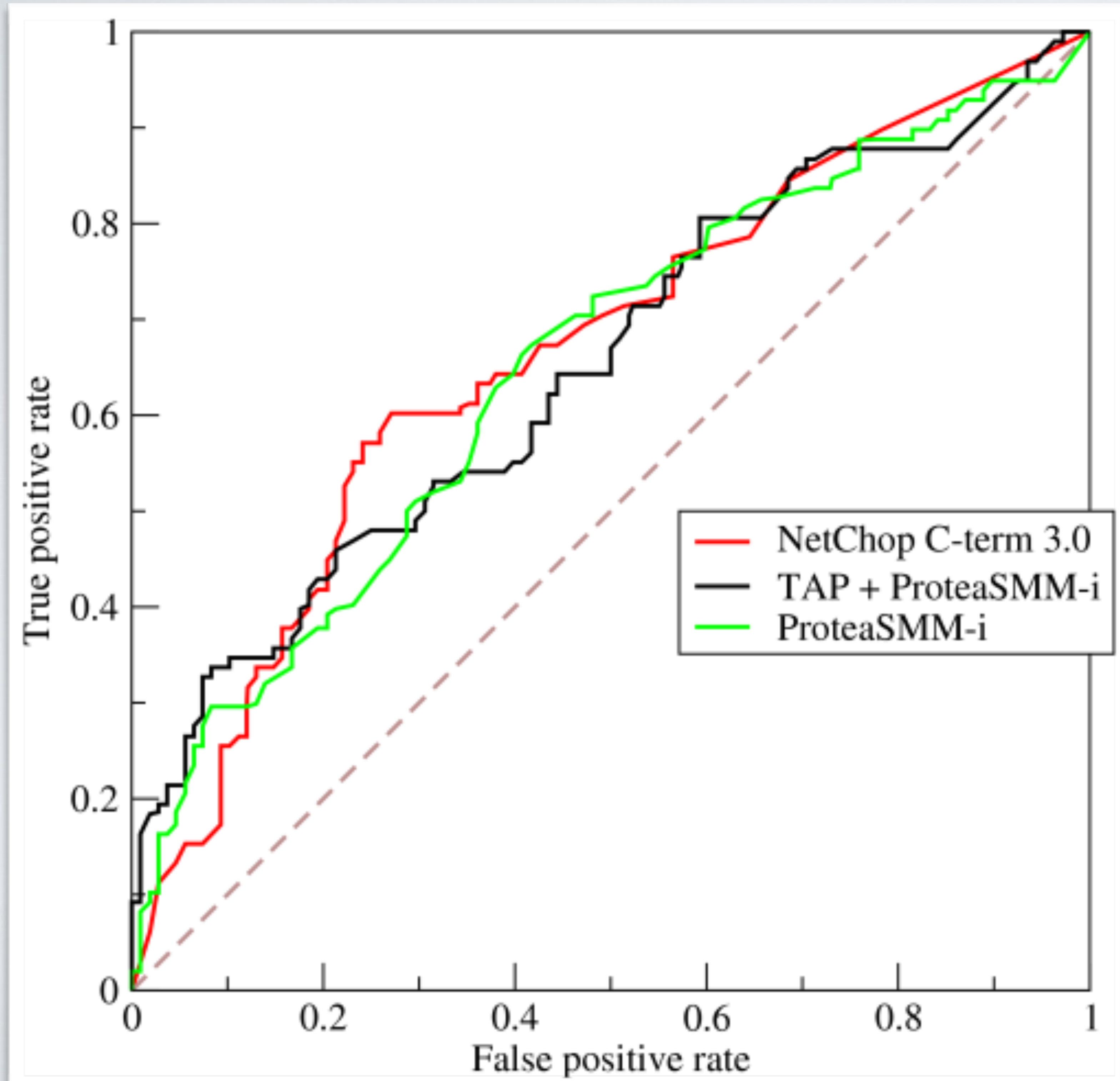
MEAN AVERAGE PRECISION

- MAP - Mean Average Precision: Variant of AP
  - Compute AP for each node separately
  - Take the average
- AP can be right for some types of nodes and not others
- MAP gives a different perspective
- Otherwise, same pros/cons

AUC - AUROC

- AUC: Area Under the Curve. Short (erroneous) name for AUROC (Area under the Receiver Operating Characteristic Curve)
- Similar idea than AP: Plot the relation between
  - False positives
  - True positives
- Take the area under the curve

AUC - AUROC



For each new TP,
How many FP added ?

AUC - AUROC

- Probabilistic interpretation:
  - If we pick a random positive example and a random negative example, probability that the positive one has a higher score
- Pros:
  - Mostly independent on the fraction of positive in the test set
- Cons:
  - Very high values, (env. 0.98), small relative improvements
  - Weigh equally all types of prediction (few links, many links) while, usually, we care more about predicting few edges

EVALUATION MEASURES

- Conclusion: Not one perfect measure, hard to say definitively that one method is *the best* without any doubt
- If not using AUC, favor a *small* fraction of positive cases, i.e. close to:
    - #edges to add / #pairs of nodes without edges
    - Note that this fraction is very low in large networks: e.g. 0.00001

OTHER METHODS: RANDOM WALKS

RANDOM WALKS

- Previous indices mostly depends on direct neighbors
- Idea: define a new index working at higher distance
- For a pair of nodes [u,v] at distance>1, compute the probability of reaching v from u after a random walk of distance k
- Problem: computationally costly. Wait for next class on graph embedding…

OTHER METHODS: COMMUNITY STRUCTURE

COMMUNITY STRUCTURE

- General idea:
  - 1)Compute community structure on the whole graph
  - 2)Assign high probability for 2 nodes in same community, low probability otherwise
- Results are not good enough alone
  - Combine with indices using supervised learning
  - Able to capture edges probability at distance >2

COMMUNITY STRUCTURE

- For InfoMap and Louvain:
  - Assign a score to each pair proportional to the change in the quality function
- For instance, Louvain optimize Modularity.
  - Each edge added between communities:
    - Decrease in the Modularity
  - Edge added inside community:
    - Increase in Modularity, depends on properties of the community and nodes

COMMUNITY STRUCTURE

- For SBM
- Reminder:
  - SBM assign each node to a community
  - For each pair of community, a probability of having an edge
- Probability of edge between pair:
  - Density between their respective communities
- If a Degree-Corrected SBM:
  - Probability also depends of degrees of nodes

GRAPH RECONSTRUCTION

GRAPH RECONSTRUCTION

- Use same indices, same methods, same evaluations as link prediction
- Difference
  - We do not split network in train and test: the whole network is used for **both**
  - It says how much the method captures the nature of the network organization

GRAPH RECONSTRUCTION

- Process (for instance, with AA index)
  - Compute AA for **all** pairs of nodes
  - Yield the ordered list of edges by AA
  - Evaluate by comparing with the original network, considering we want to predict **as many edges as originally in the network**
- Has several applications:
  - Identify possible errors in the collected networks (missing or non existing edges)
  - Generate variant of an observed network
  - Evaluate how well fitting is a model
  - …

GRAPH RECONSTRUCTION

- Interesting observation: Methods tend to be good **either** at graph reconstruction **or** at link prediction
- Classic problem of *overfitting* VS *generalization* in machine learning.
- If a method describes perfectly the current state, no need to correctly rank non-present edges.
  - The identity model has highest score at graph reconstruction, worst score at link prediction…

PRACTICALS

- (You'll have to use sklearn)
- 1)On your favorite network, predict edges according to Common Neighbors, Adamic Adar and Preferential Attachment
- 2)Compare manually the results and comment
- 3)Compare the results using AUC and AP (sklearn)
  - Need to remove 10% edges as test set, or use real dynamic network (Game of Thrones for instance)
- 4)Do the same using a classifier (sklearn)
  - Need to Remove 10% edges as training set
- 5)Advanced: Do the same using SBM