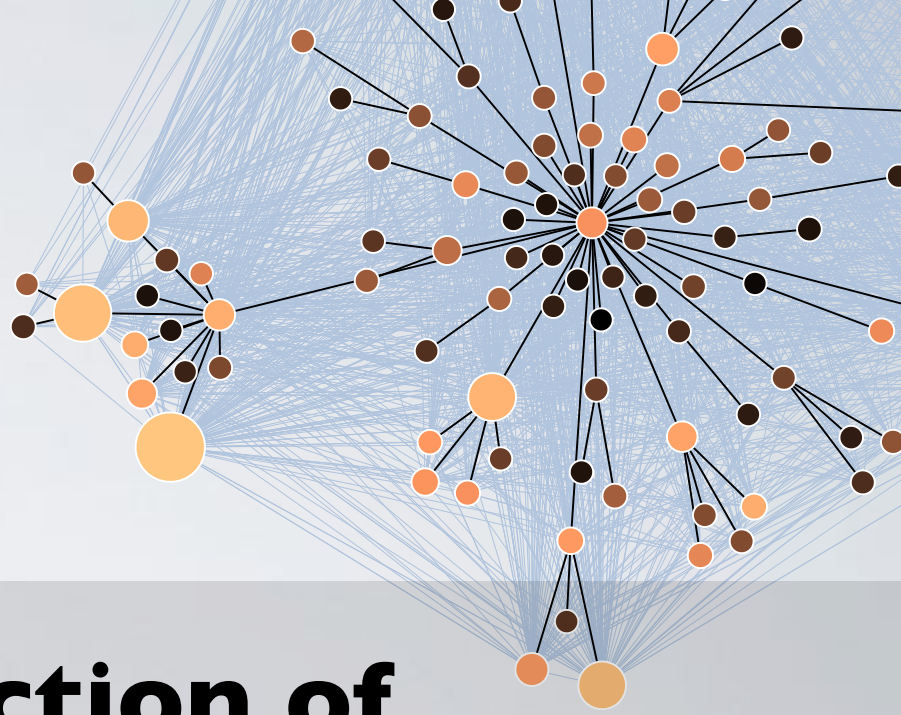


Nearly-optimal prediction of missing links in networks

joint work with Amir Ghasemian (Yale), Homa Hosseinmardi (Penn),
Aram Galstyan (USC), and Edo Airoldi (Harvard/Temple)

Aaron Clauset
@aaronclauset
Computer Science Dept. & BioFrontiers Institute
University of Colorado, Boulder
External Faculty, Santa Fe Institute

© 3 Sept 2021

talk outline

- what is link prediction
- all the data, all the methods
- a diversity of errors
- learning to combine methods
- nearly-optimal link prediction

what is link prediction?

most real-world networks are *incomplete*

what is link prediction?

most real-world networks are *incomplete*

- social networks
  - all are sampled
  - some edges hidden intentionally
  - some simply unobserved/able
- biological networks
  - observed by expensive experiments
  - edges must often be inferred
- communication networks
  - different types of interactions
  - missing all off-platform interactions

what is link prediction?

most real-world networks are *incomplete*

link prediction helps fill in the missing connections

- useful for comparing models
  which model is better at out-of-sample predictions?

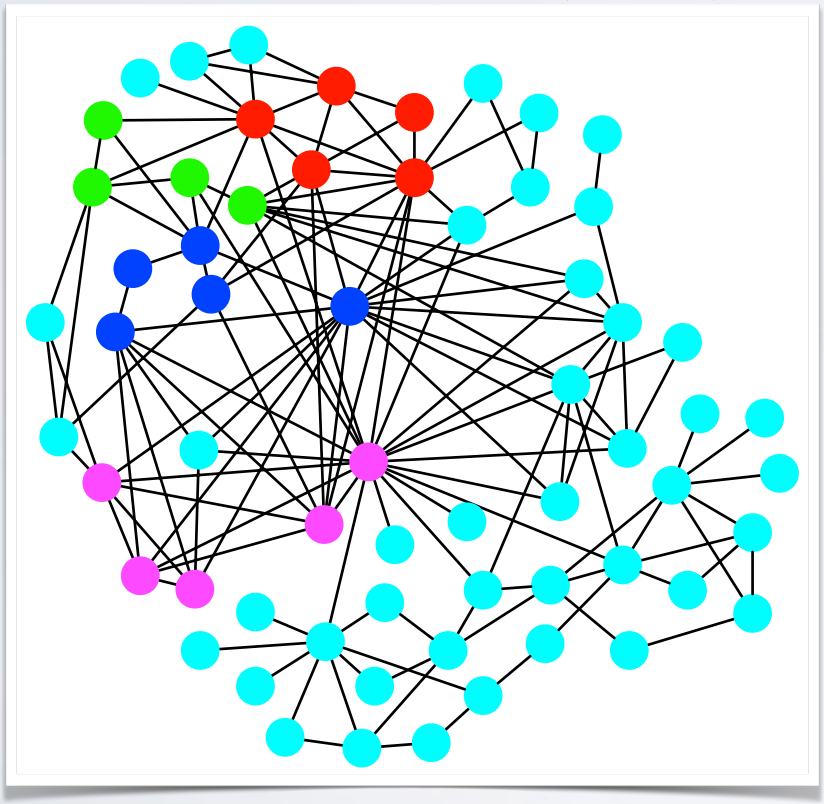
- useful for marshaling resources
  which connections should we sample next?

- useful for predicting the future (dynamic networks)
  which connections will form next?

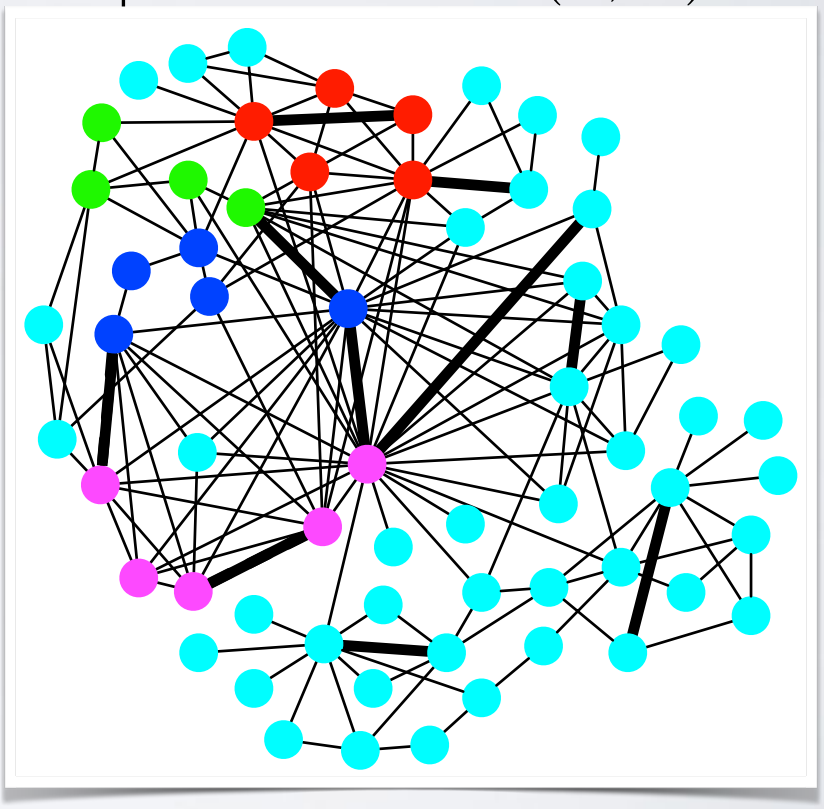
what is link prediction?

e.g.: predict associations among *9/11 hijackers*

observed $G' = (V, E')$



predicted $G = (V, E)$



* hijackers network made from data in Krebs, *Connections* **24** (2002)

predicting missing links

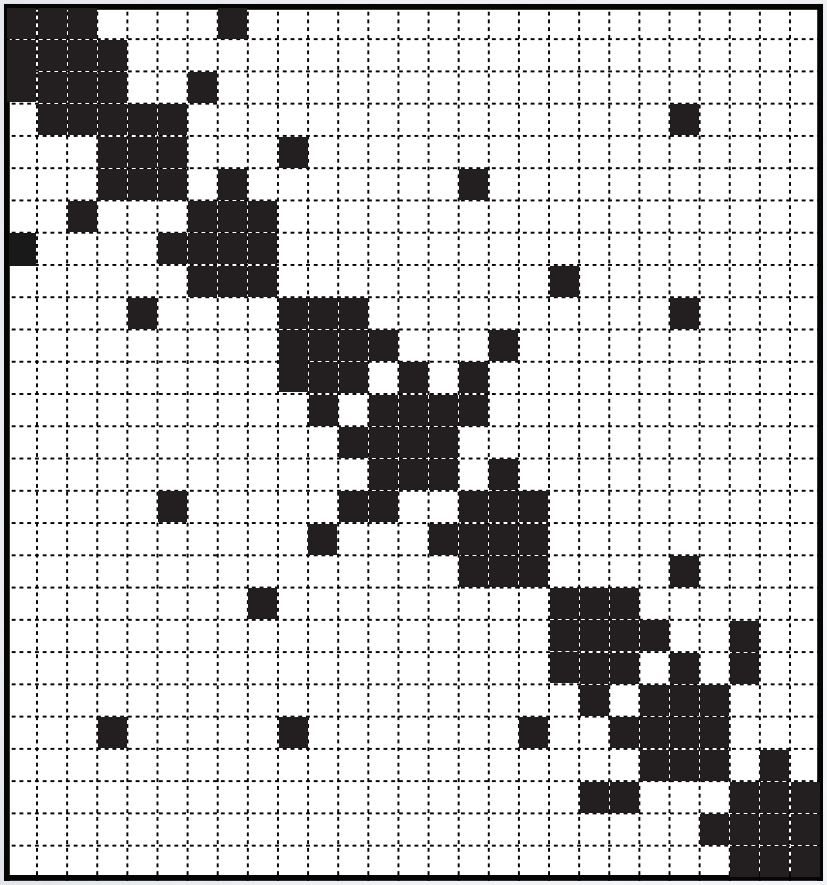
- the general setting :
  - a *latent* network $G = (V, E)$, and
  - an *observed* network $G' = (V, E')$ where $E' \subset E$
  - given G' :
    - learn how observed links E' correlate with $X = E - E'$
    - within $(V \times V) - E'$, predict X

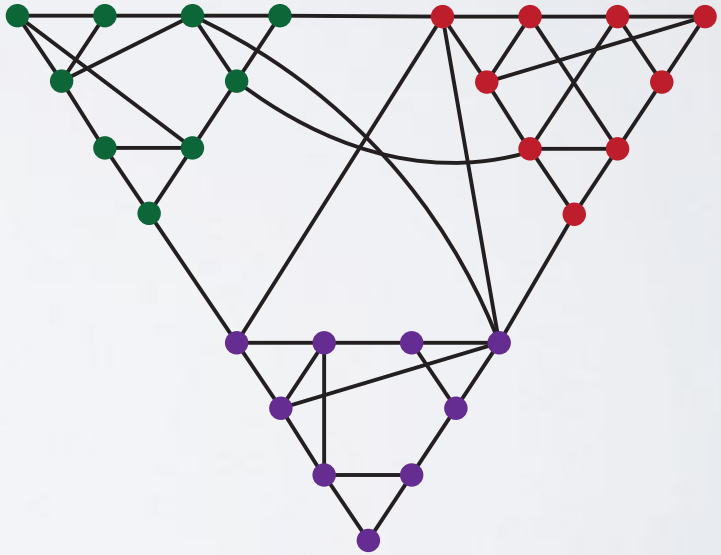
* we assume all nodes are known, and just edges are missing. predicting missing nodes is a harder problem because nodes are correlated bundles of edges

predicting missing links

for example:

latent $G = (V, E)$

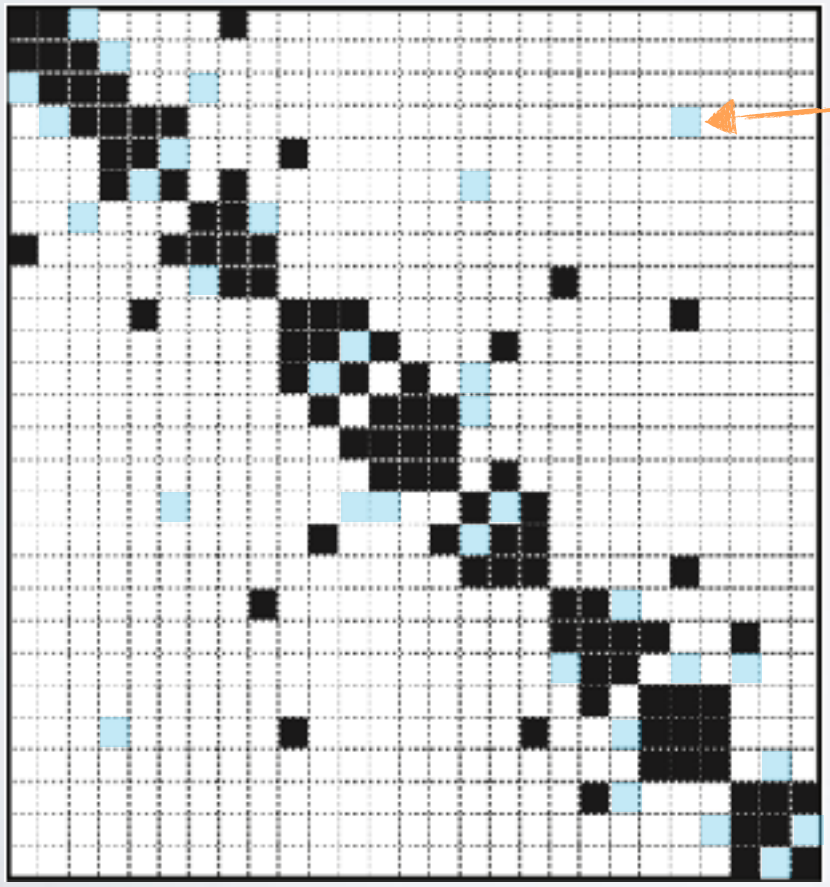




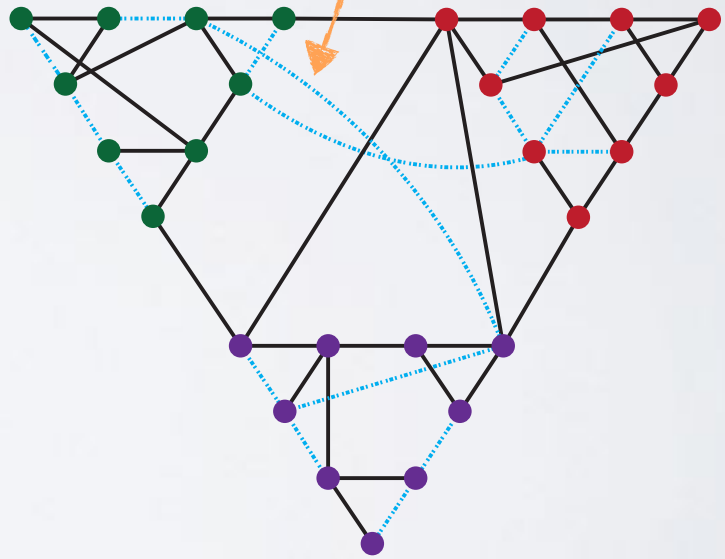
predicting missing links

for example:

observed $G' = (V, E')$



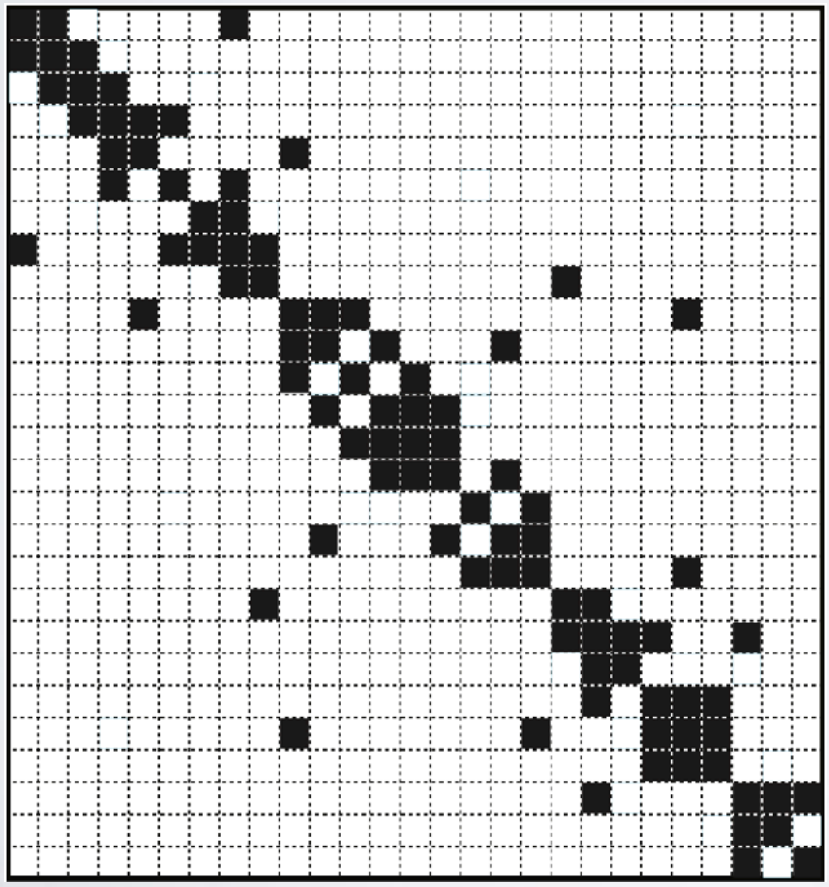
missing links $X = E - E'$



predicting missing links

for example:

observed $G' = (V, E')$



prediction task:

for every  in G'

guess  or 

non-edge

missing link $\in X$

predicting missing links

- the general setting :
  - a *latent* network $G = (V, E)$, and
  - an *observed* network $G' = (V, E')$ where $E' \subset E$
  - given G' :
    - learn how observed links E' correlate with $X = E - E'$
    - within $\underbrace{(V \times V) - E'}$, predict $\underbrace{X}$

haystack : $O(n^2)$ all needles : $O(n)$

baseline accuracy : $O(1/n)$

* baseline: assume the network is sparse, with mean degree $\langle k \rangle = O(1)$ and guessing uniformly at random

predicting missing links

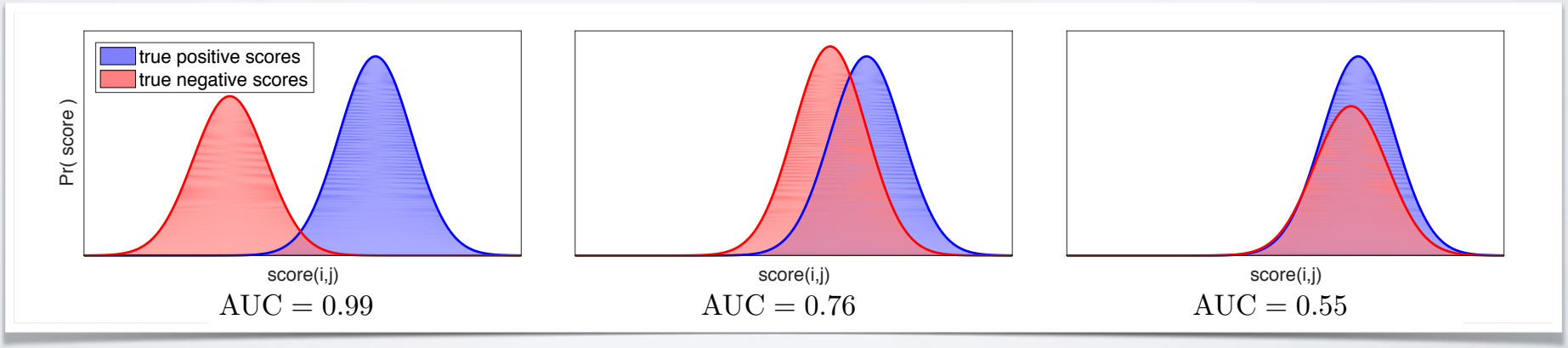
- by other names :
  - recommendation algorithms [bipartite networks, with node+edge attributes]
  - matrix completion
- all good approaches, slightly different problem formulations

in networks, we can only learn from the observed connections

* recommendation algorithms and matrix completion have literatures far too large to summarize here
* no metadata (node attributes) allowed for learning here

predicting missing links

- in practice :
  - define a *score* function: $\text{score}(.) : i, j \rightarrow \mathbb{R}$
  - apply to all possible missing links $(V \times V) - E'$
  - sort candidates by *score*, and predict top *k*
  - measure accuracy by AUC
    - given a TP, TN pair, probability of $\text{score}(\text{TP}) > \text{score}(\text{TN})$



* AUC provides a use-case-agnostic measure of class distinguishability; precision-recall provides a related measure, useful for specific use-cases, e.g., 1st page of Google search results

predicting missing links

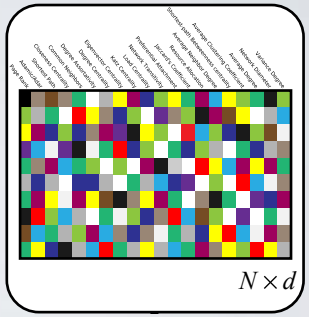
- in practice : huge number of link prediction methods

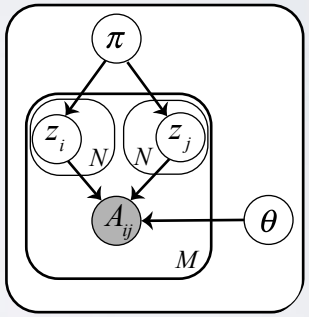
  - **topological features**
    degree, common neighbors, Jaccard, short paths, etc.

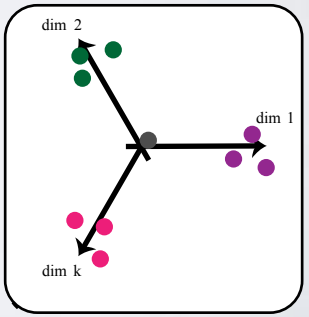
  - **model-based methods**
    probabilistic models, stochastic block models, modularity, Infomap, etc.

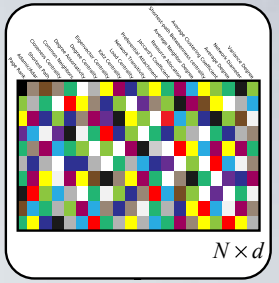
  - **embedding methods**
    spectral embedding, latent spaces, neural embeddings, node2vec, etc.

  - **and more**









predicting missing links

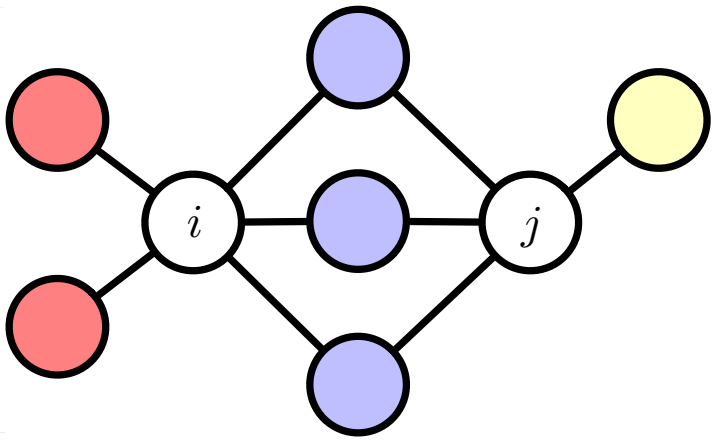
- examples of "topological" score functions:

$$\text{Jaccard}(i,j) = \frac{|\nu(i) \cap \nu(j)|}{|\nu(i) \cup \nu(j)|}$$

$\nu(i)$: set of neighbors of i

$$\text{Degree-product}(i,j) = k_i \times k_j$$

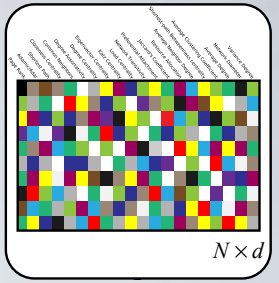
k_i : num. of neighbors of i



$\text{Jaccard}(i,j)$

more predictive than

$\text{Degree-product}(i.j)$



predicting missing links

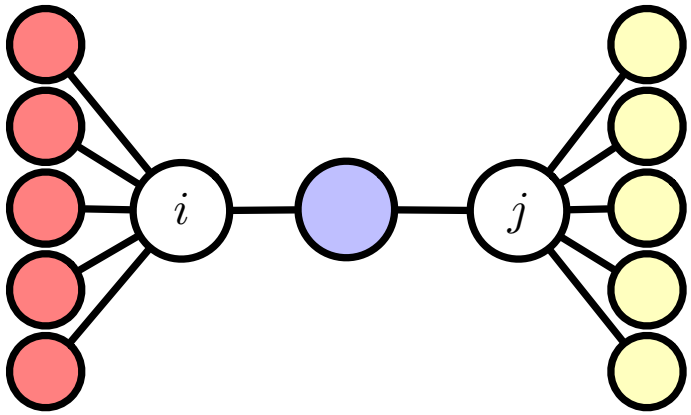
- examples of "topological" score functions:

$$\text{Jaccard}(i,j) = \frac{|\nu(i) \cap \nu(j)|}{|\nu(i) \cup \nu(j)|}$$

$\nu(i)$: set of neighbors of i

$$\text{Degree-product}(i,j) = k_i \times k_j$$

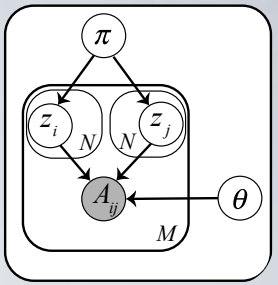
k_i : num. of neighbors of i



$\text{Degree-product}(i.j)$

more predictive than

$\text{Jaccard}(i,j)$

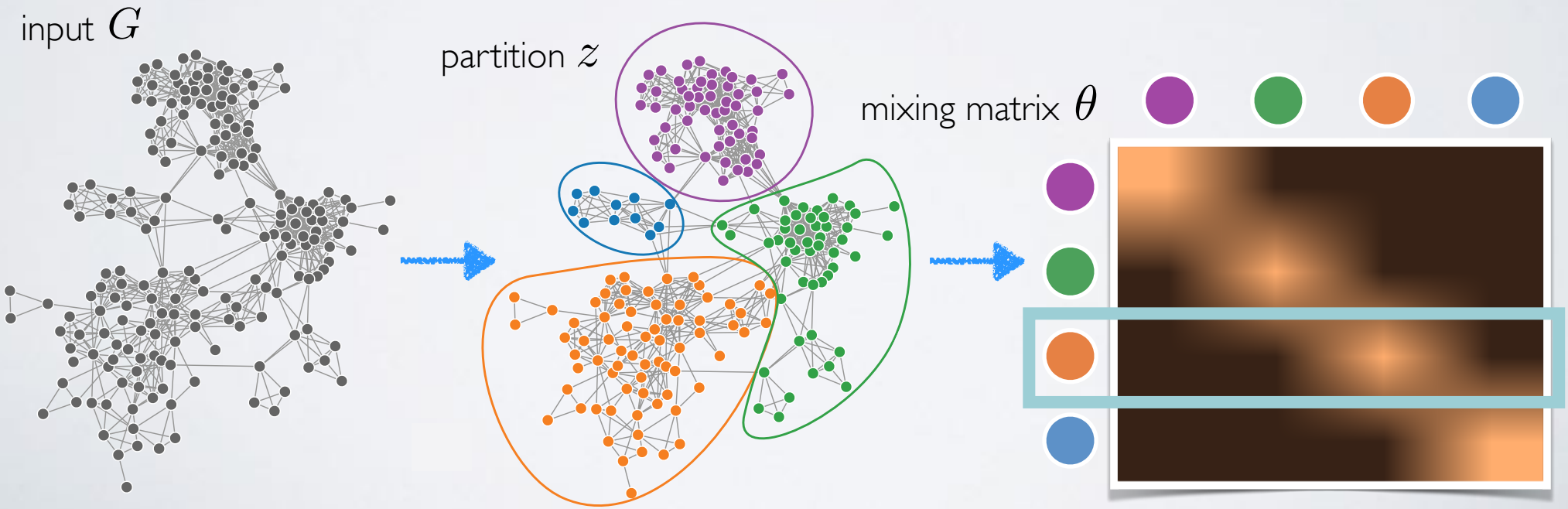


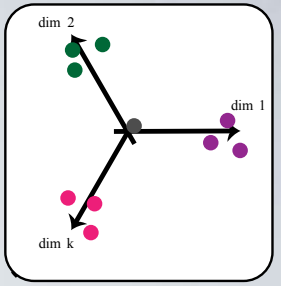
predicting missing links

- example of "mode-based" score functions:

  stochastic block model: probabilistic generative network model

  $\mathrm{SBM}(i,j) = \Pr(A_{ij} = 1 \mid z, \theta)$ [score is probability under learned model]



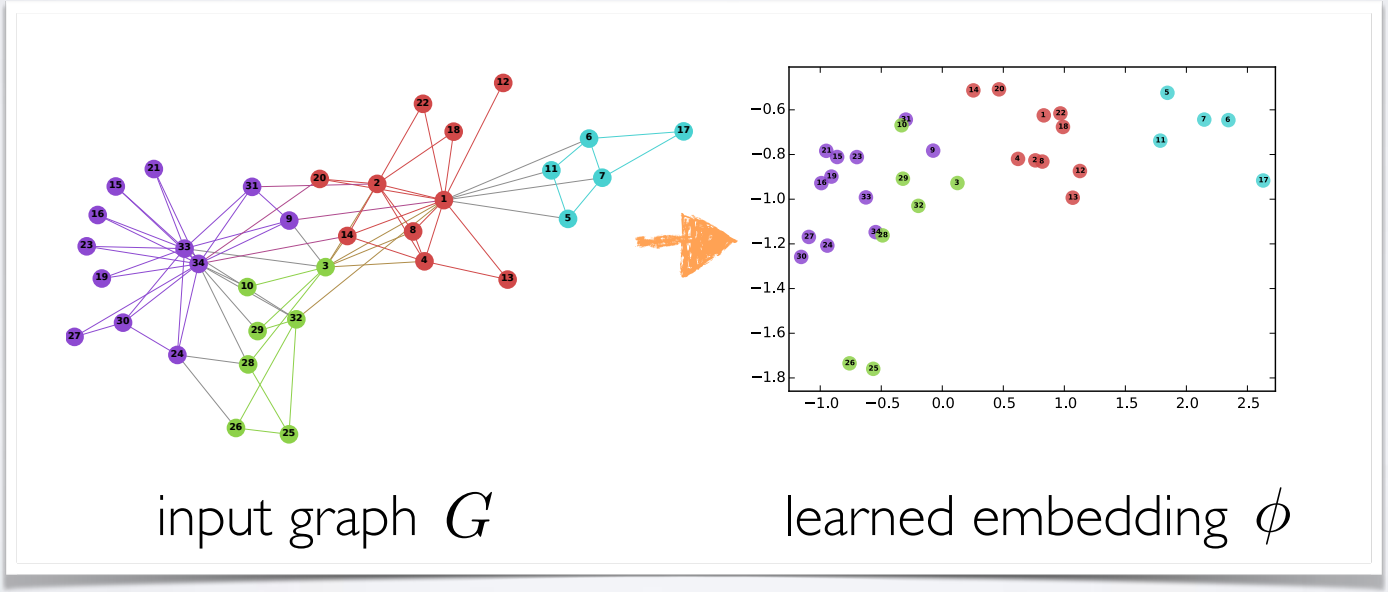


predicting missing links

- example of "embedding-based" score functions:

  learn a latent space embedding of nodes ϕ

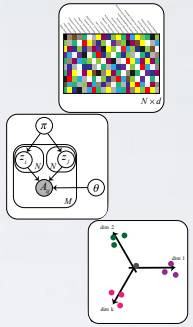
  $\text{DeepWalk}(i, j) = d(i, j \mid \phi)^{-1}$ [score is *closeness* in learned embedding]



* Perozzi et al., "DeepWalk: Online Learning of Social Representations." *KDD* 2014

predicting missing links

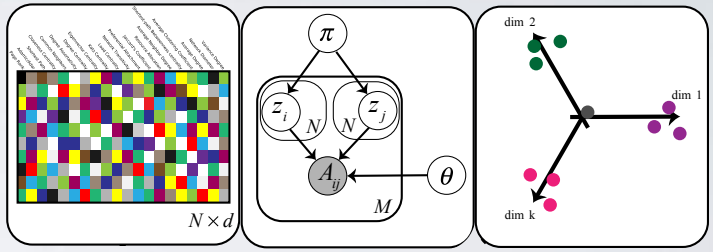
- in practice : huge number of link prediction methods
  - **topological features**
  - **model-based methods**
  - **embedding methods**

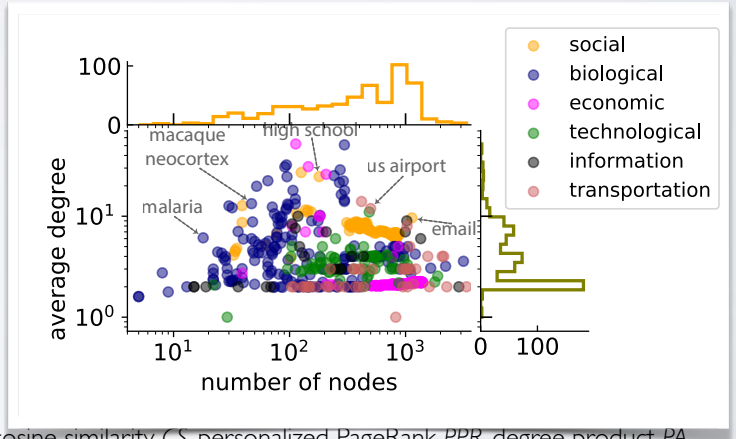


- ***all*** *published link prediction methods work well*
- are they all learning the same thing?
- is there one method that is *best overall*?
- or, does predictability vary *across* network types & settings?
- just how predictable *are* missing edges?

all the methods, all the data

- *methods* : 203 features of a candidate pair i, j
  - 42 topological features
  - 11 probabilistic network models
  - 150 features from graph embeddings
- *data* : 550 structurally diverse networks
  - from icon.colorado.edu
  - social, biological, economic, technological, information, & transportation





* **topological features**: node count *N*, edge count *OE*, degree distribution variance *VD*, common neighbors *CN*, shortest path *SP*, cosine similarity *CS*, personalized PageRank *PPR*, degree product *PA*, Jaccard coefficient *JC*, Adamic-Adar index *AA*, resource allocation index *RA*, dot product of low-rank approx *dLRA*, mean neighbor degree *AND*, betweenness centrality *SPBC*, closeness centrality *CC*, PageRank *PR*, and load centrality *LC*.

* **probabilistic models**: modularity *Q*, multi-resolution modularity *Q-MR*, msg. passing modularity *Q-MP*, Bayesian Newman-Reinert *B-NR (SBM)*, Bayesian Newman Reinert *B-NR (DC-SBM)*, Bayesian Hayashi-Konishi-Kawamoto *B-HKK (SBM)*, corrected int. class. likelihood *cICL-HKK (SBM)*, *Infomap*, min. descr. length *MDL (SBM)*, and min. descr. length *MDL (DC-SBM)*, *S-NB*

* **embedding features**: {hadamard product, dot product, L2 distance} x 128-dimension DeepWalk and 16-dim variational graph auto-encoder

* **network corpus**: we omit 24 of original 572 networks because they are too small to support cross-validation

all the methods, all the data 

- which method is *best*? do methods all learn the same things?

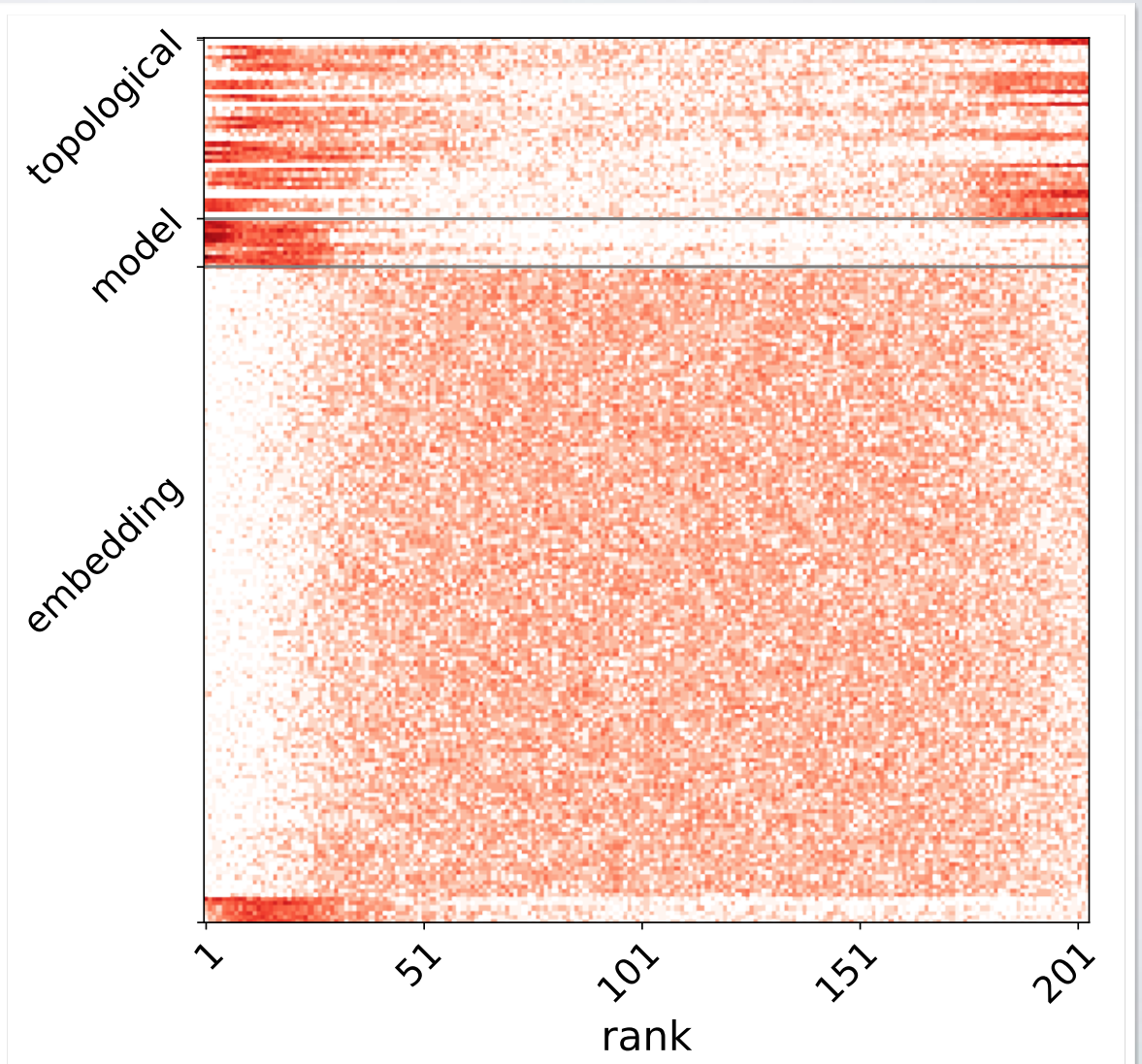
- experiment : for 550 networks, simulate missing data by removing 20% of edges, train random forest over all predictors
- consider results in 2 ways:
  - how often is each method the r^{th} most *important*?
  - how *important* is each each method for each domain?

* Gini "importance" of a predictor within the trained random forest

all the methods, all the data 🔥

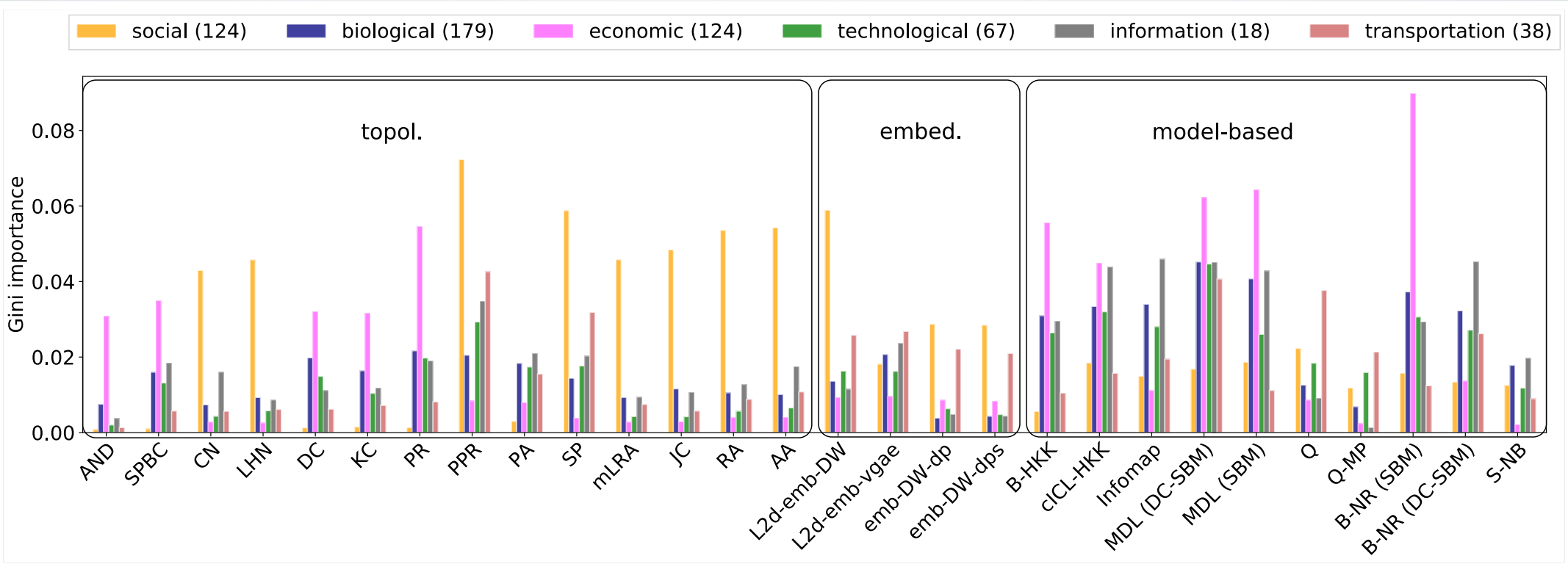
- how often is each method the r^{th} most *important*?

- *no method is best or worst on all inputs* 😮
- broad variability



all the methods, all the data 🔥

- how *important* is each each method for each domain?
- *no method is best on a given domain, or across domains* 😮



a diversity of errors

how can we make sense of this behavior?

- *No Free Lunch* theorem*— no method can be *best* on all input graphs
- different methods capture different aspects of structure
- performance varies by domain (soc, bio, etc.)

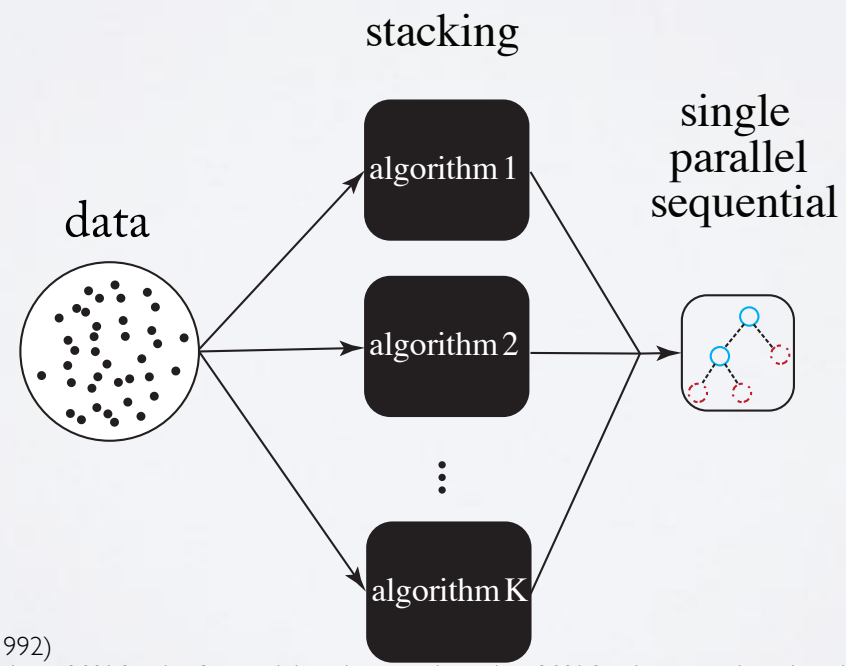
- methods make a *diversity of errors*
- let's *exploit* this diversity via *meta-learning* to *combine* methods to make *strictly better* predictions than any individual method

* Wolpert and Macready, "No Free Lunch Theorems for Optimization." *IEEE Transactions on Evolutionary Computation* **1** (1997)
* Peel et al. "The ground truth about metadata and community detection in networks." *Science Advances* **3** (2017)

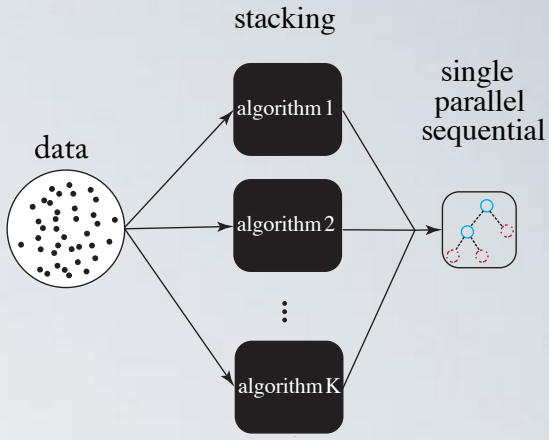
learning to combine methods

learning to combine methods

- here, via *model stacking* [a la Netflix Prize]
  - *learn* which algorithm to apply to a given test case x_{ij}, based on its attributes $f(x_{ij})$
  - *leverages* the independence of the errors each method makes, to be better than any component method



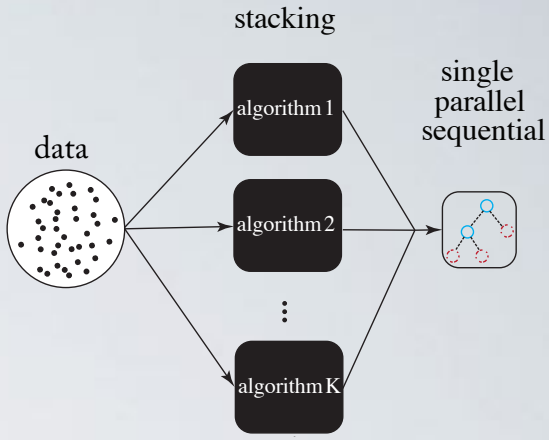
* stacked generalization a la Wolpert, *Neural Networks* **5** (1992)
* this supervised technique has two training phases; we hold out 20% for the first, training phase, and another 20% for the second, testing phase



learning to combine methods

- here, via *model stacking* [a la Netflix Prize]
  - direct method to average point estimates from multiple models
  - learning process:
    1. split data into hold-out / test / train
    2. fit / evaluate each predictor on same training / test split
    3. learn predictor weights $\vec{w}$ that minimize hold-out error
  - constructs an optimal *predictive distribution* over the data
  - more accurate than, e.g., Bayesian model averaging, when true data generating model is unknown

* under mild assumptions, stacking selects the optimal sets over *K* models, and produces a consistent estimate of the posterior score
* this setting is the $\mathcal{M}$-open case, where the true model is unknown and we cannot approximate it (for whatever reason), see Yao et al. *Bayesian Analysis* **13** (2018)
* Bayesian model averaging works well in the $\mathcal{M}$-closed setting, where true model is in the set, and we have good model priors for all models; without that, it only selects a single model out of the set



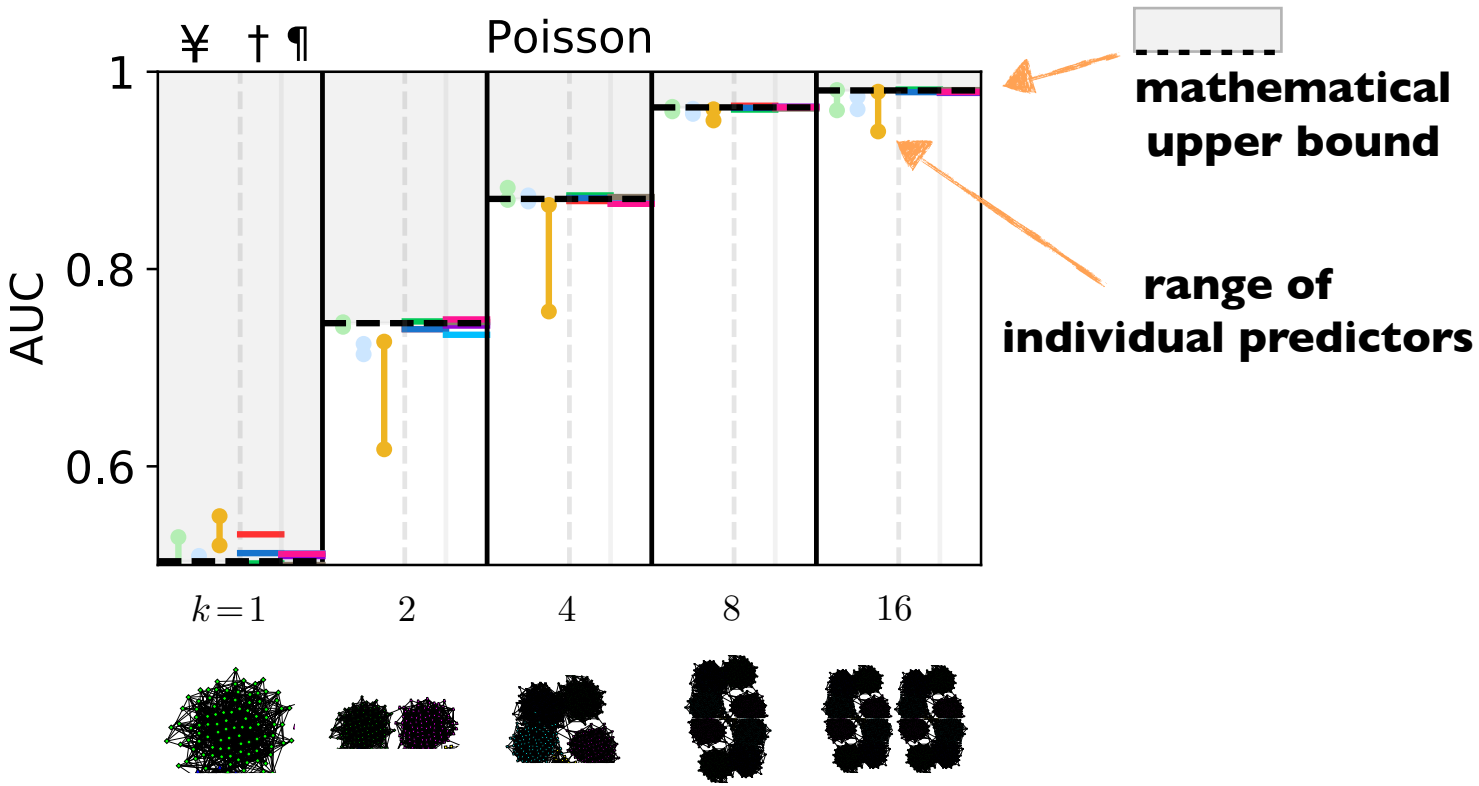
learning to combine methods

- here, via *model stacking* [a la Netflix Prize]
  - *claim: this method produces nearly-optimal link predictions*
  - testing this claim:
    - synthetic data with known structure
    - real-world data from large, diverse corpus
    - accuracy saturation as more features are included?
    - do individual features satisfy AdaBoost theorem?

* stacked generalization a la Wolpert, *Neural Networks* **5** (1992)
* this supervised technique has two training phases; we hold out 20% for the first, training phase, and another 20% for the second, testing phase

on synthetic data

- warm-up (1 dimension) : low-variance in connectivity, sharp communities, and vary number of communities
- score by AUC on 20% holdout & compare w/ *mathematical upper bound*



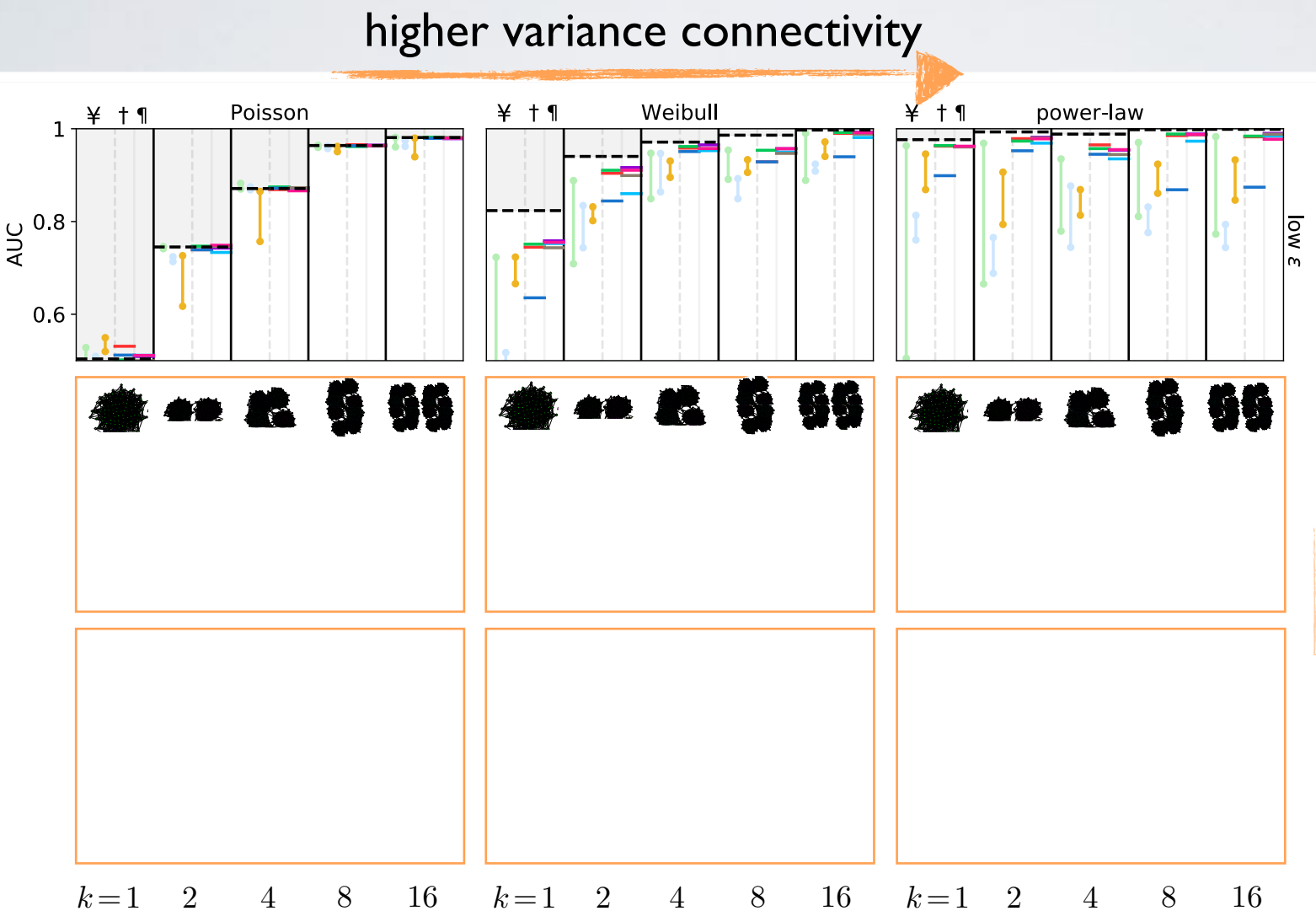
¥ individual predictors
- 5 model-based candidates
- embedding techniques
- top 5 topols.

† within-family stacking
- all topol. predictors
- all model-based
- all embed. predictors

¶ across-family stacking
- all topol & model predictors
- all topol. & embed. predictors
- all model, & embed. predictors
- all topol., model, & embed. pred

on synthetic data

- vary by 3 dimensions : variance in connectivity, fuzziness of structure, and number of communities
- score by AUC on 20% holdout & compare w/ *mathematical upper bound*



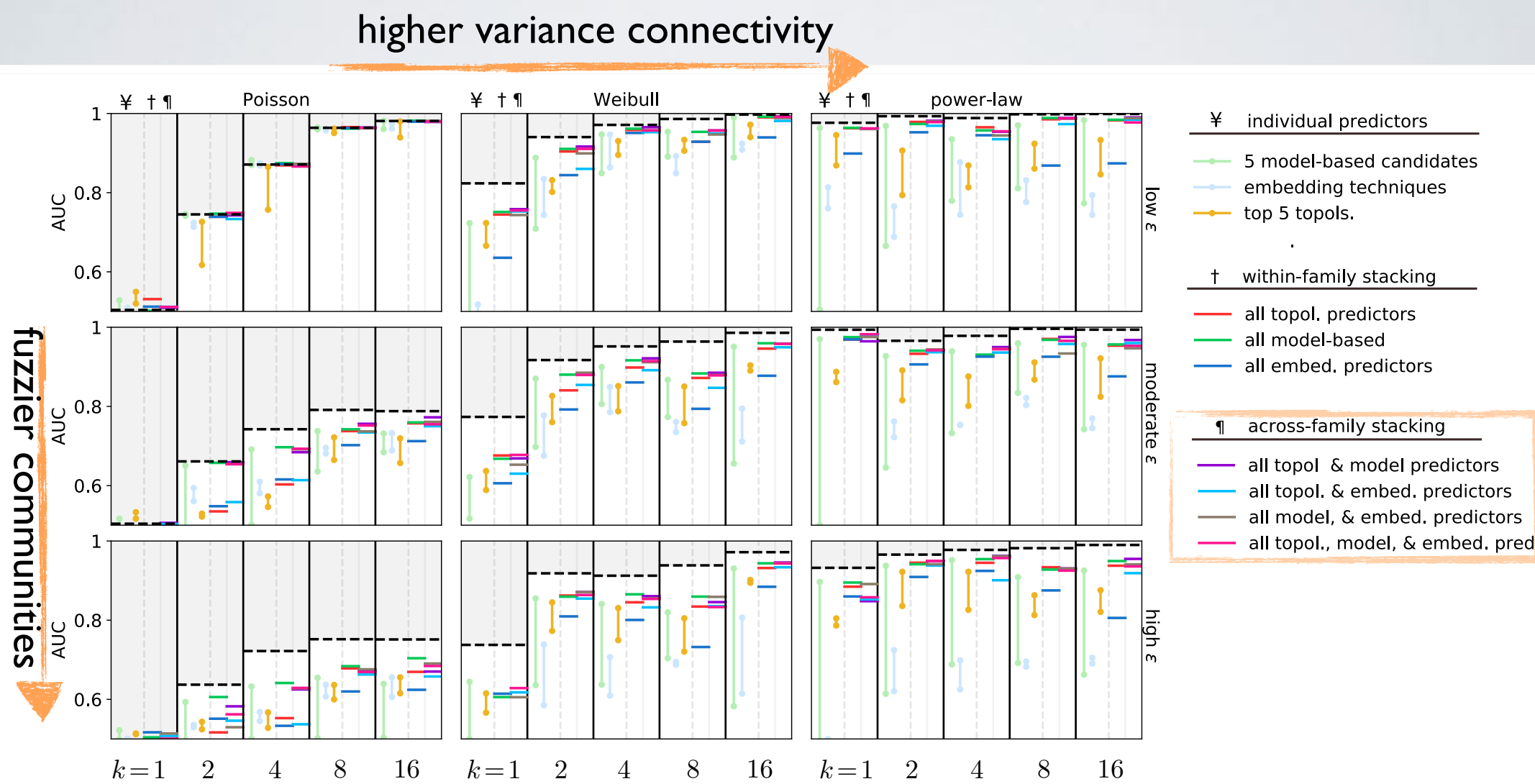
¥ individual predictors
- 5 model-based candidates
- embedding techniques
- top 5 topols.

† within-family stacking
- all topol. predictors
- all model-based
- all embed. predictors

¶ across-family stacking
- all topol & model predictors
- all topol. & embed. predictors
- all model, & embed. predictors
- all topol., model, & embed. pred

on synthetic data

- vary by 3 dimensions : variance in connectivity, fuzziness of structure, and number of communities
- score by AUC on 20% holdout & compare w/ *mathematical upper bound*



on synthetic data

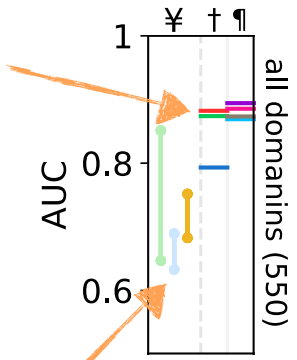
- vary by 3 dimensions : variance in connectivity, fuzziness of structure, and number of communities
- score by AUC on 20% holdout & compare w/ *mathematical upper bound*

- fuzzier communities → harder link prediction
- more homogeneous degrees → harder link prediction
- less modular structure → harder link prediction
- performance differences → mismatch of model to data's structure
- **across methods, *stacking* best or nearly best**
- embedding methods (naively) perform poorly (overfitting?)
- supervised methods perform better than unsupervised (unsurprising)

on real data

- 550 structurally diverse networks from the CommunityFitNet corpus
- score by AUC on 20% holdout

stacked generalization nearly always the best, across domains



embeddings nearly always worst, across domains

¥ individual predictors
- 5 model-based candidates
- embedding techniques
- top 5 topols.

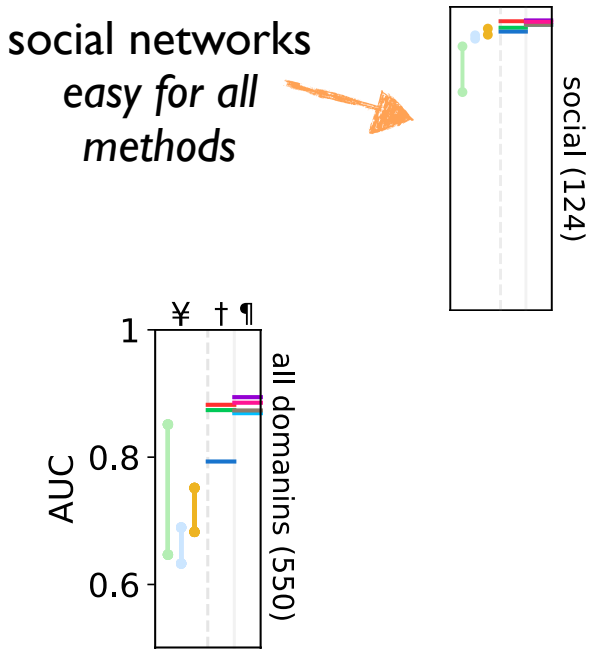
† within-family stacking
- all topol. predictors
- all model-based
- all embed. predictors

¶ across-family stacking
- all topol & model predictors
- all topol. & embed. predictors
- all model, & embed. predictors
- all topol., model, & embed. pred

* CommunityFitNet corpus : github.com/AGhasemian/CommunityFitNet

on real data

- 550 structurally diverse networks from the CommunityFitNet corpus
- score by AUC on 20% holdout



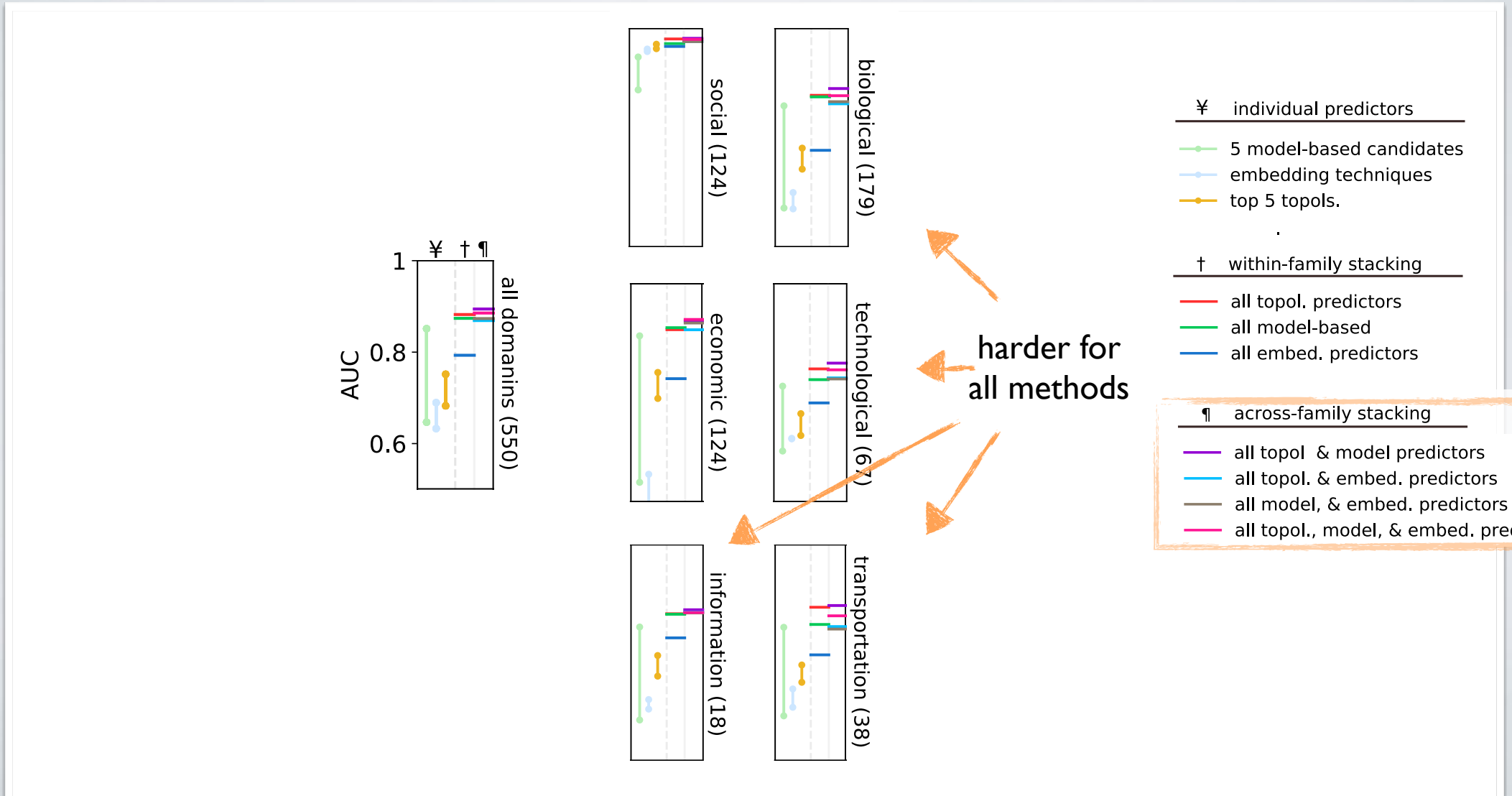
¥ individual predictors
- 5 model-based candidates
- embedding techniques
- top 5 topols.

† within-family stacking
- all topol. predictors
- all model-based
- all embed. predictors

¶ across-family stacking
- all topol & model predictors
- all topol. & embed. predictors
- all model, & embed. predictors
- all topol., model, & embed. pred

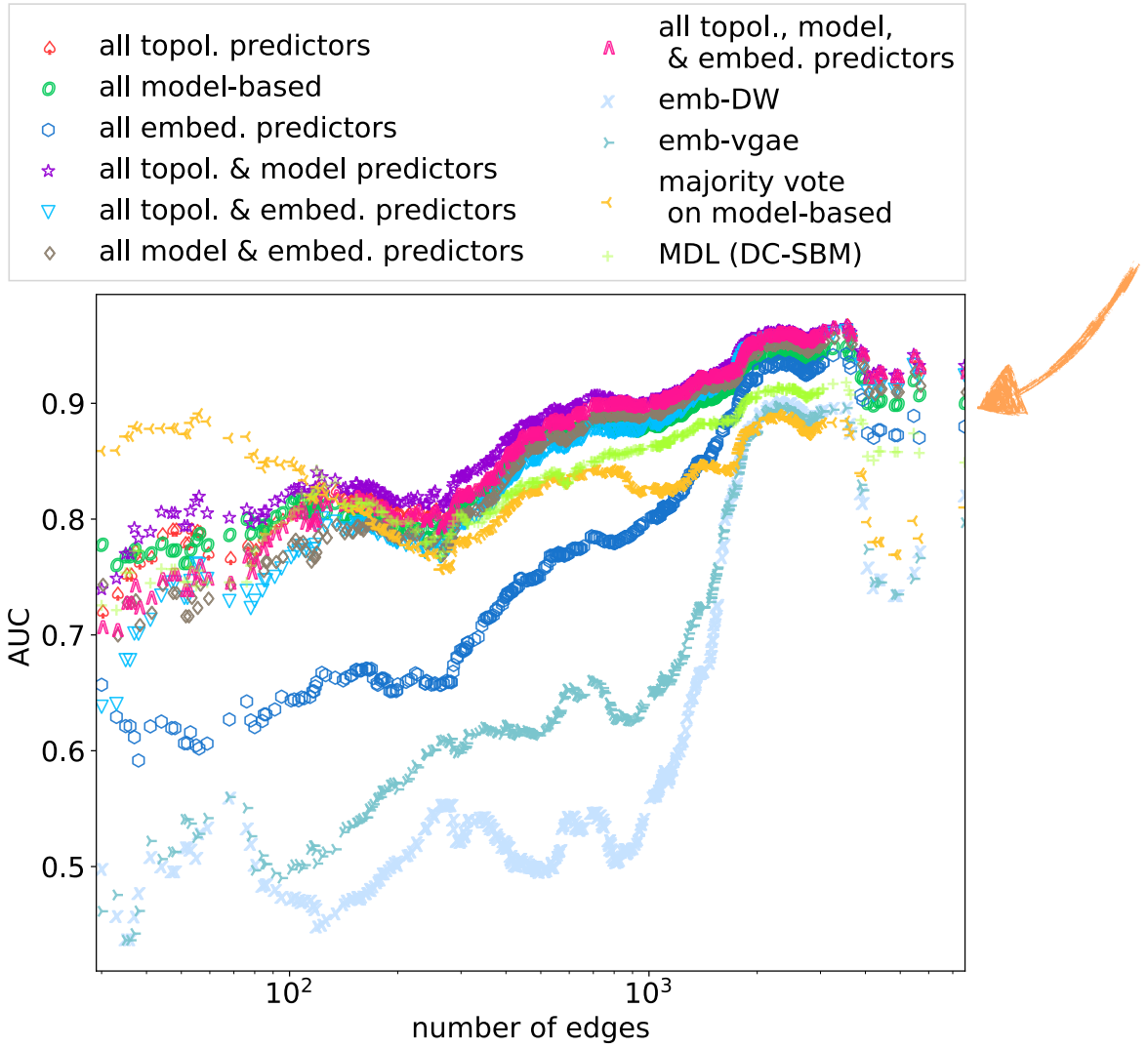
on real data

- 550 structurally diverse networks from the CommunityFitNet corpus
- score by AUC on 20% holdout



on real data

- 550 structurally diverse networks from the CommunityFitNet corpus
- score by AUC on 20% holdout
- all results, segregated by network size M → bigger networks = better results



on real data

- 550 structurally diverse networks from the CommunityFitNet corpus
- score by AUC on 20% holdout

- **across methods and domains, *stacking* best or nearly best**
- stacking advantage grows with network size
- predicting missing links in social networks is *easy* (all methods quite good)
- hard domains → biological, technological, transportation, information

on real data

- 550 structurally diverse networks from the CommunityFitNet corpus
- score by AUC on 20% holdout

- **across methods and domains, *stacking* best or nearly best**
- stacking advantage grows with network size
- predicting missing links in social networks is *easy* (all methods quite good)
- hard domains → biological, technological, transportation, information
- only 25-30 features (models, topol.) sufficient for near-optimality
- state-of-the-art accuracy
- can train to optimize AUC or P/R

algorithm	AUC	precision	recall
mean model-based	0.75 ± 0.16	0.15 ± 0.24	0.24 ± 0.25
mean indiv. topol.	0.61 ± 0.14	0.09 ± 0.2	0.23 ± 0.27
mean indiv. topol. & model	0.64 ± 0.15	0.1 ± 0.21	0.23 ± 0.27
emd-DW	0.63 ± 0.23	0.11 ± 0.17	0.3 ± 0.29
emb-vgae	0.69 ± 0.19	0.15 ± 0.21	0.25 ± 0.23
all topol.	0.88 ± 0.1	0.31 ± 0.33	0.35 ± 0.29
all model-based	0.87 ± 0.1	0.25 ± 0.28	0.29 ± 0.22
all embed.	0.79 ± 0.14	0.18 ± 0.23	0.27 ± 0.23
all topol. & model	0.89 ± 0.09	0.31 ± 0.34	0.34 ± 0.28
all topol. & embed.	0.87 ± 0.11	0.27 ± 0.31	0.33 ± 0.25
all model & embed.	0.87 ± 0.11	0.23 ± 0.26	0.3 ± 0.23
all topol., model & embed.	0.89 ± 0.1	0.28 ± 0.31	0.34 ± 0.26

* mean AUC over 550 networks

conclusions and outlook

conclusions and outlook

- nearly all networks are incomplete — link prediction helps fill in the missing details
- but, no link predictor is best across all networks (No Free Lunch)
- predictors make independent errors — *meta-learning* opportunity

---

- *stacked generalization* combines many methods across settings — nearly always better than individual predictors + can incorporate new predictors as they emerge
- hits theoretical optimum across messy synthetic data — real-world performance possibly *nearly optimal*
- missing links in social networks *inherently* easy to recover; biological networks substantially harder

Stacking Models for Nearly Optimal Link Prediction in Complex Networks

Amir Ghasemian[a,b,c,*], Homa Hosseinmardi[b], Aram Galstyan[b], Edoardo M. Airoldi[c,d], and Aaron Clauset[a,e,f,*]

PNAS **117**(38), 23393 (2020)

Dr. Amir Ghasemian
(Yale)

Dr. Homa Hosseinmardi
(Penn)

Prof. Aram Galstyan
(Southern California)

Prof. Edo Airoldi
(Harvard / Temple)

Data & code:

👉 https://github.com/Aghasemian/OptimalLinkPrediction

Acknowledgements: Brendan Tracey (DeepMind),
David Wolpert (Santa Fe),
Cris Moore (Santa Fe)

Funding support: