Network Science Cheatsheet

UNIVERSITÉ DE LYON

Made by
Remy Cazabet

Machine Learning on Graphs

This class is about Machine Learning on graphs **without** using graph embedding and Graph Convolutional Networks. For those techniques, see the next class.

Machine Learning

Machine learning(ML) involves computers discovering how they can perform tasks without being explicitly programmed to do so. It involves computers learning from data provided so that they carry out certain tasks[a]. It is a subset of **Artificial Intelligence**.

[a] `https://en.wikipedia.org/wiki/Machine_learning`

Unsupervised ML

In unsupervised learning, the machine is presented with the data, but without any examples. It should infer automatically the rules, the organization of the data.
The best known example of unsupervised ML is the **clustering** task. Community detection is a type of clustering on networks, that we have already discuss, so this class will rather focus on Supervised ML.

Supervised ML

Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs.
After seeing enough examples structured as: input → output, the machine generalize a knowledge such as for any (unseen) input, it will predict an output.
Examples: Given properties of an apartment, predict its energy consumption or price. Given a picture, recognize objects in it. Given a patient profile, predict effect of a drug. etc.

Link Prediction

Link prediction is a typical machine learning task on networks. Example of applications are the prediction that an edge might appear in the future (e.g., in dynamic networks), how likely it is that a link that exist is missing in the network (e.g., missing synonyms in wiktionary, missing gene-disease interactions, etc.), or for recommendation: profile recommendation in social medias, content recommendation in online retailers or service providers (Netflix, Spotify, YouTube, etc.).
Link prediction can be based only on the network structure, or on a mixture of network structure and node properties.

Link Prediction: intuition

How likely it is for an edge to appear between nodes can depend on:

- Local factors: Friends of my friends being my friends, edges might appear to friends of my friends
- Nodes properties/attributes: Nodes with high degrees are statistically more likely to bond than nodes with few edges. Age, political opinions, spatial proximity might be correlated with edge existence (assortativiy, spatial networks, etc.)
- Meso-scale structure: Nodes belonging to the same (automatically discovered) communities might be more likely to connect, for instance.

Link Prediction: Heuristics

A fist approach to predict edges is not based on machine learning, but consists in defining **heuristics** based on different features:

- Local factors: Friends of my friends being my friends, edges might appear to friends of my friends
- Nodes properties/attributes: Nodes with high degrees are statistically more likely to bond than nodes with few edges. Age, political opinions, spatial proximity might be correlated with edge existence (assortativiy, spatial networks, etc.)
- Meso-scale structure: Nodes belonging to the same (automatically discovered) communities might be more likely to connect, for instance.

Heuristic: Common neighbors (CN)

Hypothesis: the **number** of common neighbors is an indicator of the probability to connect. Based on the "friend of my friend are my friend", transitivity principle.

$$CN(u,v) = |N(u) \cap N(v)|$$

Heuristic: Jaccard Coefficient (JC)

Hypothesis: The **fraction** of common neighbors might be more relevant than the raw number of common neighbors.

$$JC(u,v) = \frac{|N(u) \cap N(v)|}{|N(u) \cup N(v)|}$$

Heuristic: Hub Promoted (HP)

Hypothesis: the relation can be asymmetric: if one of the nodes has a high fraction of its neighbors in common with the other, it might be likely to connect with it, even if the contrary is not true (e.g., all of my friends follow celebrity x, so I'm likely to follow x, even if x has a large number of followers who are not my friend)

$$JC(u,v) = \frac{|N(u) \cap N(v)|}{\min(|N(u)|, |N(v)|}$$

Heuristic: Adamic Adar (AA)

Hypothesis: all neighbors are not worth the same, neighbors with a small degree are more significant than hubs (e.g., you and me having two close friends in common is more significant than following the same two celebrities with millions of followers.)

$$AA(u,v) = \sum_{w \in N(u) \cap N(v)} \frac{1}{\log(k_w)}$$

Heuristic: Resource Allocation (RA)

Hypothesis: Similar to AA, but penalize more higher degrees.

$$AA(u,v) = \sum_{w \in N(u) \cap N(v)} \frac{1}{k_w}$$

Heuristic: Preferential Attachment (PA)

Hypothesis: As known from the Configuration Model, nodes with higher degrees are more likely to be connected.

$$PA(u,v) = k_u k_v$$

Note that unlike previous ones, this heuristic gives non-zero scores to nodes at distances larger than 2.

Heuristic: Other scores

Several other scores based on similar ideas have been proposed in the literature[a](Sorenson Index, Salton Cosine Similarity, Hub Depressed, Leicht-Holme-Nerman, etc.)
Which heuristic is the most appropriate for link prediction does not have a straightforward answer.

[a]Zhou, Lü, and Zhang 2009.

Heuristic: Distances

Another family of heuristics to assess node likelihood to connect consists in using their network proximity as a proxy. Much as nodes which are close in geographical space tends to connect with higher probability, nodes located close in the topology of the network are usually more likely to connect. It can be seen as an extension of the principle of common neighbors to nodes at farther distance.
Methods to measure the distance between nodes can consist[a] in using the shortest-path distance, the probability to reach a node from another on a random walk, or the number of paths of a chosen length l between them.

[a]Lichtenwalter, Lussier, and Chawla 2010.

Heuristic: Community Structure

The community structure detected by an algorithm can be used as a heuristic for link prediction. The exact way to rank edges from more likely to less likely depends on the community detection algorithm used[a]:
For methods optimizing a global quality function, such as Modularity or Infomap Information compression, the *score* between each pair of nodes is proportional to the gain (or loss) in the global score if this edge were added. For instance, with Modularity, adding edges inside communities will tend to increase the global score (edges considered likely), while edges between communities will decrease it (unlikely edges).
For methods based on Stochastic Block Models, the score associated to a node is directly yielded by the model: the block matrix can be interpreted as describing the probability of an edge to exist between any two nodes belonging to a particular pair of blocks. In degree-corrected SBM, edge probabilities depends both of their blocks and of their degrees.

[a]Ghasemian, Hosseinmardi, and Clauset 2019.

Heuristic: Spatial Networks

If a spatial model (Gravity, Radiation, etc.) has been fit on a spatial graph, then the probability of any edge appearing is given by the model, much as for SBM.
The logic is the same for any kind of node properties, although, if there is no natural notion of distance of distance for these properties, it is more efficient to let the machine learning algorithm learn the best way to combine them.

From Heuristics to supervised ML

Heuristics can directly be used for link prediction, but, since they capture different types of properties (local/meso/global), using only one of them means missing the information brought by the others.
Supervised Machine Learning is the most efficient way to combine them: since we do not know *a priori* how to combine them, and the optimal way to do so might depend on the network, we let the computer learn how to do it.

Training for link prediction

Training a ML algorithm requires to constitute a **training set**, i.e., a set of examples *input*→ *output*. For link predictions, an example is composed of:

- Input: Heuristics associated to the node pair Output: 1 or 0 (Respectively, an edge exists or not between those 2 nodes in the original graph.)

Since we want to predict edges that are *not yet* on the network, we start by removing t edges from the network, that will constitute **positive examples** for training. We sample t other node pairs that are **not** connected by an edge in the original graph: they are **negative examples**.
Our training set is therefore composed of a balanced(50% negative, 50% positive) set of $2t$ examples.

ML algorithm: Classifiers

Supervised Machine Learning is usually split between methods that predict numerical values (**Regression**), and those who predict to which category an element belongs, among several choices (**Classification**). Link prediction is usually made with classification algorithms(Two classes: Edge or Not-Edge).
A wide variety of such algorithms exist, from simple Linear Regression to deep neural networks. Implementations and description of algorithms can be found, for instance, in the popular scikit-learn library[a]. Good places to start are Logistic classifiers and Decision Trees.

[a]`https://scikit-learn.org/stable/supervised_learning.html`

Classifier results

A trained Classifier is a model which, presented a set of features, yields a probability to belong to a particular class. While in other applications, we consider the most likly class as the answer, in link prediction, most of the time, we are interested in the probability itself.
We use this probability to **rank** pairs of nodes, from the most to the least likely of being connected by an edge.

Evaluation in Machine Learning

In the Machine Learning scientific field, the evaluation of the effectiveness of algorithms is a central question. The evaluation is usually done as follows: The original set of observations is split in 2, the **training set** and the **evaluation set**. The model is trained on the training set, without access to the evaluation set, and the performance of the trained model is evaluated on the evaluation set.

Evaluation in Link prediction

Link prediction evaluation is a little special, since we do not start with a set of observations, but a single observation of a graph. We cannot really split the graph, so we consider pairs of nodes as observations, although this introduces biases: the graph on which we train is not the same as the one used for validation: they have different number of edges, some nodes have different degrees, etc.
Note also that the original graph is split in 3: the training set (sample of node pairs), the test set (sample of node pairs) and the original graph without edges of the training and test sets, used to compute the features/heuristics.

Balanced training

Training and Test sets must be composed of positive and negatives examples, respectively node pairs between which an edge must and must not be predicted. For link prediction, the problem is that the set of all node pairs is usually extremely unbalanced. Networks are sparse, thus it is common in large graphs to have only one edge every 10,000 node pairs.

- The balanced of the training set can be chosen freely: What we care is to have a well-trained model, whatever the means. It is usually balanced because it is more efficient and convenient.
- The test set on the contrary must respect the original balance between edges and non-edges: the method must be tested in real conditions, not on an artificially simplified problem.

Classification evaluation: Precision@k

Precision@k is defined as the fraction of positive examples among the k pairs of nodes of highest score according to the classifier.
The weakness of this approach is that the result depends on the chosen k.

Classification evaluation: Average precision

Average precision, also known as Area Under the Precision/Recall Curve, is defined as the average Precision@k for all k.

Classification evaluation: AUC/AUROC

Area Under the Receiver Operating Characteristic Curve (**AUROC**), often simply abbreviated as **AUC** (Area Under the Curve), is defined as the area under the curve defined with False Positives on the horizontal axis and True Positives on the vertical axis. It has an intuitive probabilistic interpretation: the AUROC score corresponds to the probability, if we take two pairs of nodes are random, one a positive example and the other a negative one, that the positive example is ranked higher than the negative one.
The score thus lies between 0 and 1, and a score of 0.5 corresponds to a random prediction.
The main advantage over previous scores is that in theory, its value does not depend on the balance of the validation test, which is technically difficult to ensure for large graphs.
AUC is currently the most used evaluation score, although some limitations have been raised[a].

[a] Yang, Lichtenwalter, and Chawla 2015.

Machine Learning for nodes

The other main application of ML on graphs is to predict some node properties, being numerical values(regression) or categories(classification).
Among applications, we can cite the filling of missing values (e.g., speed limits in a road network, categories of Wikipedia article, etc.), the prediction of unknown/hidden attributes (e.g., in marketing, knowing the genre, political opinion, age, salary, etc.), or the detection of particular nodes (spammers, bots, fake accounts, etc.)

Non-network approaches

If there are several properties on nodes, some of these properties can be used as input to predict another one as output. For instance individuals age, genre, and political opinions could be used to predict (efficiently or not) the revenue of these same individuals, if we can collect training examples.

Centrality as attributes

A simple approach to improve the prediction consists in integrating some network properties computed on the node in the prediction. For instance, additionally to age, genre and opinions, one could integrate the degree, betweenness, closeness, clustering coefficient, etc. of a node in the prediction of its revenue.

Neighbors attributes as ego attributes

Following the popular saying "Tell me who your friends are and I will tell you who you are", we can use the network to observe what are the most common features of the neighbors of a target node to predict its own features. For instance, the revenues of your neighbors in the graph might be useful to predict your own revenue. Furthermore, the most common political opinion, or the average age of the neighbors can also be a useful hint.
In practice, a simple way to do this consists[a] in computing, for each node feature, the average value of those features in the neighborhood of nodes. A simple ML model can then be used as if those properties were the nodes own properties.

[a] Bhagat, Cormode, and Muthukrishnan 2011.

Random Walk attributes estimation

A generalization of the previous approach consists in evaluating the distribution of attributes not only among the direct neighbors of the target nodes, but more generally among nodes that are close from it in the graph. A simple way to achieve this is to sample attributes using random walks. Several methods exist[a], for instance for a numerical attribute, the estimated value $\tilde{y_u}[c]$ for attribute c for node u can be expressed as the average value encountered by a random walk of distance t. More formally:

$$\tilde{y_u}[c] = \sum_{v \in V} p_{uv}^t v[c]$$

with p_{ij}^t the probability to encounter node v from node u after a random walk of distance t, and $v[c]$ the value of label c for node v

[a] Bhagat, Cormode, and Muthukrishnan 2011.

Going Further

Python Library: `scikit-learn`
Surveys: (Lichtenwalter, Lussier, and Chawla 2010) (Al Hasan et al. 2006) (Lü and Zhou 2011)
Model stacking: (Ghasemian, Hosseinmardi, Galstyan, et al. 2020)
Evaluation of link prediction: (Yang, Lichtenwalter, and Chawla 2015)
Node Classification: (Bhagat, Cormode, and Muthukrishnan 2011)
Underfit and Overfit of community-based link prediction: (Ghasemian, Hosseinmardi, and Clauset 2019)

References

[1] Mohammad Al Hasan et al. "Link prediction using supervised learning". In: *SDM06: workshop on link analysis, counter-terrorism and security*. Vol. 30. 2006, pp. 798–805.

[2] Smriti Bhagat, Graham Cormode, and S Muthukrishnan. "Node classification in social networks". In: *Social network data analytics*. Springer, 2011, pp. 115–148.

[3] Amir Ghasemian, Homa Hosseinmardi, and Aaron Clauset. "Evaluating overfit and underfit in models of network community structure". In: *IEEE Transactions on Knowledge and Data Engineering* (2019).

[4] Amir Ghasemian, Homa Hosseinmardi, Aram Galstyan, et al. "Stacking models for nearly optimal link prediction in complex networks". In: *Proceedings of the National Academy of Sciences* 117.38 (2020), pp. 23393–23400.

[5] Ryan N Lichtenwalter, Jake T Lussier, and Nitesh V Chawla. "New perspectives and methods in link prediction". In: *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*. 2010, pp. 243–252.

[6] Linyuan Lü and Tao Zhou. "Link prediction in complex networks: A survey". In: *Physica A: statistical mechanics and its applications* 390.6 (2011), pp. 1150–1170.

[7] Yang Yang, Ryan N Lichtenwalter, and Nitesh V Chawla. "Evaluating link prediction methods". In: *Knowledge and Information Systems* 45.3 (2015), pp. 751–782.

[8] Tao Zhou, Linyuan Lü, and Yi-Cheng Zhang. "Predicting missing links via local information". In: *The European Physical Journal B* 71.4 (2009), pp. 623–630.