Master MVA | **Scribed By: Fabien Pesquerel & David Robin**
Graphs in Machine Learning | **Lecture #1**
Instructor: Dr. Michal Valko | **Fall, 2019**

Contents

Slides available on Michal Valko's INRIA page
`http://researchers.lille.inria.fr/~valko/projects/courses/graphsML/20192020/mlgraphs1.pdf`

1 Introduction

1.1 Central Notions

Definition (Directed Graph)
A directed graph G consists of:

- A non-empty set V of elements called **vertices**
- A finite set E of *ordered pairs of distinct vertices* called **edges** or **arcs**

A directed graph will often be written $G = (V, E)$ or $G = (V, A)$ where $A \subseteq V^2$ refers to the set of arcs. A directed graph is also called a *digraph*. For an arc (u, v), the first vertex is called its **head** and the second its **tail**.

If the set E is considered to be a set of *unordered pairs* of distinct vertices, then G is called an undirected graph. In this case, elements of E are called edges.

For instance, social networks such as *Twitter*, *Mastodon* and *YouTube* can be represented with directed grahs. In those social networks, the set of vertices is the set of people and there is an arc between two people if and only if one is a follower or subscriber of another. Because one can follow someone who does not follow in return, this notion of *follower* is not symmetric and edges between people are indeed directed. Note that with the notion of *sharing content* we can start to think of a notion of *influence* where a person influences another if the content he/she posts is shared by the other person. We can even imagine weighting an arc by a measure of how much influence the head of the arc has on the tail of the said arc.

As another example, social networks such as *Facebook* and *Diaspora* can be represented with undirected graphs. In these social networks, relations are symmetric : both people agree to be friends before an edge is added between them, hence the undirected edges.

Remark. A graph is an abstract object and as such, it has intrinsic properties which are independant from the representation/visualisation of the graph, *e.g.*, connectedness, paths between points, etc. However we will often visualize graphs as embedded in an Euclidian space where each vertex is mapped to a point in $\mathbb{R}^n$ and each edge is mapped to a path between its endpoints (often enough, the straight line). This embedding can be inherent to the data that naturally occurs as points in a metric space or built to fit a model and, for instance, using a similarity function (see section 5).

1.2 Recommended readings

The recommended textbooks for the course are Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction* [1] and Masashi Sugiyama - *Statistical reinforcement learning: modern machine learning approaches* [2].

2 First example : Product placement

2.1 Problem statement

Define G an undirected graph whose nodes are people and edges are influential links between people. Having an edge from Alice (A) to Charlie (C) with weight 0.5 means that if one of them obtains a product, there is a 50% chance that it will influence the other to buy it as well.

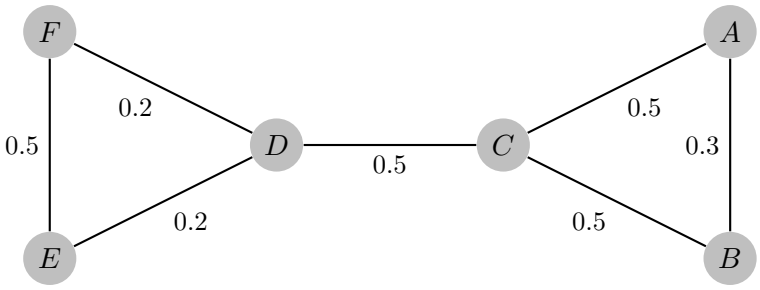

Figure 1: An example of weighted graph

A marketing team trying to improve the reach of the product chooses to send k free products to selected "influencers", so that more people end up buying the product thanks to their influence.

Who should get the free products ?

For instance, a marketing team could give a free product to A. A could influence C but not B to buy the product. C could then influence D but not B to buy the product. Finally, it could be that D influences no one. By giving one free product, propagation in the influence graph makes two people (C and D) buy the product.

2.2 Centrality heuristics

For a chosen set of people, we can run a cascading simulation and compute the average number of people who would buy the product. Choosing the optimal set of people can thus be done by enumeration of all possible choices. This algorithm is prohibitively expensive in practice, and an approximated solution would be acceptable if it were computable in a more reasonable time frame.

Intuitively, we understand easily that it is better to select nodes that are "central" in the graph, for some definition of centrality, because it will be easier to spread from a central node that from an isolated one. A first heuristic to solve the problem would hence be to sort the nodes by centrality and pick the top k. Common definitions of centrality include:

- the degree of a node
- the sum of weights of edges adjacent to a node
- the average distance to other nodes

For example, in the above graph (Figure 1), D and C have the highest degree, C is the node with the greatest sum of adjacent edges' weight, and C and D have the lowest average distance to other nodes.

2.3 Submodularity

Definition
A function $f : 2^A \mapsto \mathbb{R}$ on a discrete set A is **submodular** if for any $S \subseteq T \subseteq A$ and any $e \in A \setminus T$,

$$f(S \cup \{e\}) - f(S) \geq f(T \cup \{e\}) - f(T)$$

Intuitively, a function is submodular if the marginal increase diminishes with the size of the set. This is studied in economics as the law of diminishing marginal returns.

As an example, let A be a set of stuff to buy $A = \{\text{bread}, \text{apple}, \text{tomato}, \cdots\}$ and f the cost of buying such items. Let $S = \{\text{bread}\}$ and $T = \{\text{bread}, \text{tomato}\}$ and $e = \text{apple}$

$$\begin{aligned}
f(\{\text{bread}\}) &= c(\text{bakery}) + c(\text{bread}) \\
f(\{\text{bread}, \text{apple}\}) &= c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{apple}) \\
f(\{\text{bread}, \text{tomato}\}) &= c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) \\
f(\{\text{bread}, \text{tomato}, \text{apple}\}) &= c(\text{bakery}) + c(\text{bread}) + c(\text{market}) + c(\text{tomato}) + c(\text{apple})
\end{aligned}$$

$$f(S \cup \{e\}) - f(S) = c(\text{market}) + c(\text{apple}) \geq c(\text{apple}) = f(T \cup \{e\}) - f(T)$$

Adding an apple to the smaller set costs more because we have to pay both the cost for going to the market and the cost for the apple, whereas for the larger set we only pay the apple's cost because we only need to go to the market once.

2.4 Approximation algorithm : Greedy

Coming back to our optimization problem,

$$\underset{S \subseteq A, |S| \leq k}{\arg\max} \; f(S)$$

This problem is NP-hard in general. However, in our particular case, it is possible to obtain a $(1 - \frac{1}{e})$ approximation because f is non-negative, monotone, submodular.

Definition (Monotonicity)
A submodular function f is *monotone* if for every $T \subseteq S$, $f(T) \leq f(S)$.

The greedy[1] algorithm is the algorithm obtained by greedily selecting items one by one. It thus starts with $S_0 = \varnothing$ and computes the iterates $(S_i)_i$ defined by

$$S_{i+1} = S_i \cup \{ \arg\max_{e \in A \setminus S_i} f(S_i \cup \{e\}) \}$$

Let $S^* = \arg\max_{S \subseteq A, |S| \leq k} f(S)$ be the optimal solution, and let S_{greedy} be the solution obtained by greedily selecting k items. Then S_{greedy} satisfies

$$f(S_{\text{greedy}}) \geq \left(1 - \frac{1}{e}\right) \cdot f(S^*)$$

Lemma Let Ω be a discrete set, and $f : 2^\Omega \to \mathbb{R}$ a submodular function.
Take $X \subseteq \Omega$ and n distinct elements $(x_i)_{i \leq n} \in \Omega \setminus X$. Then

$$f(X \cup \{x_1, \cdots, x_n\}) \leq \sum_{i=1}^{n} f(X \cup \{x_i\}) - (n-1) f(X)$$

The proof of this result is done by induction with heredity: (by definition of submodularity)

$$f(X \cup \{x_1 \cdots x_n\} \leq f(X \cup \{x_n\}) + f(X \cup \{x_1 \cdots x_{n-1}\}) - f(X)$$

Proof (Greedy Algorithm Approximation): Let $(S_i)_{i \leq k}$ be the iterates computed by the greedy algorithm, hence $S_{\text{greedy}} = S_k$.
The optimality gap before the i^{th} step is

$$\begin{aligned}
f(S^*) - f(S_{i-1}) &\leq f(S^* \cup S_{i-1}) - f(S_{i-1}) && \text{by optimality of } S^* \text{ and monotonicity} \\
&\leq \sum_{a \in S^* \setminus S_{i-1}} f(\{a\} \cup S_{i-1}) - f(S_{i-1}) && \text{by the previous lemma} \\
&\leq \sum_{a \in S^* \setminus S_{i-1}} f(S_i) - f(S_{i-1}) && \text{by definition of the greedy alg.} \\
&\leq k \cdot (f(S_i) - f(S_{i-1}))
\end{aligned}$$

Therefore

$$\begin{aligned}
f(S^*) - f(S_i) &= f(S^*) - f(S_{i-1}) - (f(S_i) - f(S_{i-1})) \\
&\leq f(S^*) - f(S_{i-1}) - \frac{1}{k}(f(S^*) - f(S_{i-1})) \\
&= (1 - \frac{1}{k})(f(S^*) - f(S_{i-1}))
\end{aligned}$$

[1]Each time we write an *argmax*, we are aware that this is *a priori* a set of solution(s) but we solve collisions by choosing an element of the set of *argmax*.

By recurrence, $f(S^*) - f(S_i) \leq \left(1 - \frac{1}{k}\right)^i \cdot f(S^*)$.

Therefore, for k^{th} element, $f(S^*) - f(S_k) \leq \left(1 - \frac{1}{k}\right)^k \cdot f(S^*)$. The result follows by noticing $(1-\frac{1}{k})^k \xrightarrow[k\to\infty]{}{}^{+} e^{-1}$, which yields $f(S^*)-f(S_k) \leq \frac{1}{e} f(S^*)$. Any uniform upper bound on $\left(1 - \frac{1}{k}\right)^k$ would have been enough, but this one is tight.

Rearranging the terms, we get, $f(S_k) \geq \left(1 - \frac{1}{e}\right) \cdot f(S^*)$, which proves our claim. □

Numerically, we proved that a result found by the greedy algorithm is such that

$$f(S^*) \geq f(S_{greedy}) \geq \left(1 - \frac{1}{e}\right) \cdot f(S^*)$$

meaning that the greedily selected subset's score is at least 63% of the optimal value.

3 A more intricate example : Page Rank

3.1 Problem statement

Consider the directed graph whose nodes are web pages and edges are arcs (u, v) such that page u points to page v. We want to rank all web pages by importance, *i.e.* how many pages point to them. Ideally, we would like a page to be important only if important pages point to them, which makes the definition circular.

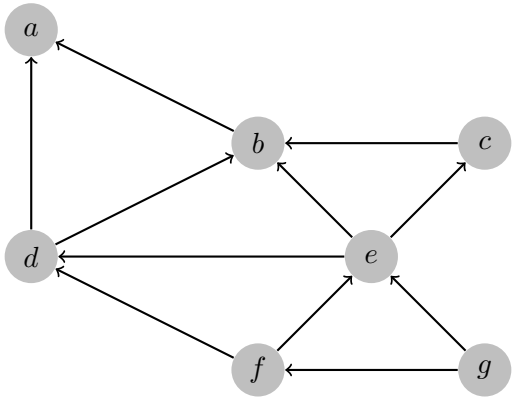

Figure 2: A tiny internet with a sink in node **a**

3.2 Proposed algorithm : Random surfer

Google's PageRank algorithm considers a "random surfer", a model of a user that just clicks on links at random, never hitting back. The position of this random surfer defines a Markov chain with transition matrix M, where $M_{i,j} = 1/k_i \cdot \mathbb{1}(\text{link from } i \text{ to } j)$ with k_i the number of links on page i. This matrix is stochastic (rows sum to 1).

This model has a problem : the random surfer always ends up stuck on "sink" pages, nodes with no outer edges (a in the example above). To tackle this issue, PageRank defines a probability p (15% in the original paper) that the user gets bored and starts on a random page. This means that it uses instead the "google" matrix $G = (1-p) \cdot M + p \cdot \frac{1}{n} 1_n 1_n^T$ as transition matrix.

We look for a distribution over nodes invariant by G's action : a stationnary distribution. That is to say a right eigenvector v with value 1, i.e. such that $Gv = v$. Because G is stochastic with positive entries, we know by Perron's theorem that it always exists and is unique

3.3 Solving at scale

The US patent for PageRank was granted in 2001. Google has since been indexing 10's of billions of web pages. At such a scale, it is mandatory to find the eigenvector in an efficient way. A simple way to do that is to use the *power method*. Start with $v_0 = (\delta_{i=0})_i$, and iterate $v_{t+1} = Gv_t$ until a fixed point is reached numerically.

The average outdegree of a node is only 7, so although G is dense, M is very sparse. Instead of performing an expensive matrix multiplication by G, one can multiply independently by M and $1_n 1_n^T$, which is much cheaper, and sum the outcomes.

4 Machine learning graphs in the wild

4.1 Inherent graphs : biological networks

Biological systems and their interactions often exhibit an inherent graph structure. This is the case for instance of molecules, or gene regulatory networks. This structure carries relevant information and may be used to improve understanding of such interactions, or perform predictions on future interactions. In such cases, graphs can be an object of study, and be given as input to a machine-learning algorithm in order to discover properties of the structure. An algorithm may also be required to produce output in the form of graphs, as would be the case for predictions.

A classical example of biological network is the protein-protein interactions network. Building this network and understanding it is a significant goal for the biological field as it allows to answer many questions in the field. For instance, if a researcher were to discover that a drug affecting a protein helps to cure some illness, he/she will first check the possible side effect of this drug by looking at how affecting the protein would propagate through the graph. Another example is in evolutionary biology where it was discovered that proteins with high degrees of connectedness are more likely to be essential for survival than proteins with lesser degrees.

4.2 Imposed graphs, abstractions

Sometimes the object of study is not naturally modeled by a graph structure, but imposing a graph structure on the data points, which can be learned or reconstructed, provides a framework to better analyse such data. Graphs in this case are often not the object of study itself, but rather an abstraction used as a tool to understand relationships between objects, and form the basis for more complicated algorithms that can act on the abstract graph rather than the raw data itself.

The method of construction of those graphs typically encodes structural properties of the data. The choice of which graph to use for each application is very often non-trivial, and the model chosen will have consequences on the outcome of algorithms, such as generating biases induced by the hypotheses made during graph construction.

4.3 Modeling graphs

Studying properties of graphs helps us to choose the right model for the right situation. The aim of a model is to reduce the number of parameters. We can thus fit a model on the data, then deal with the model to analyse the behavior of the graph in some test situation. There are many models of graph in the wild and it might be difficult to choose the right one. It is just as studying probabilities to fit the best model we can when it comes to statistics. Example of graphs model are the Erdös-Renyi graph, which models independent edges, Barabási-Albert graph, which models preferential attachment, and the Stochastic Blocks graph, which models communities.

5 Similarity graphs

On data which does not have an inherent graph structure, graphs can be used as an abstraction to get insight about the relationship between data points V. The general idea is to define a similarity $s : V \times V \to \mathbb{R}$ that will define a distance over the data points.

Remark. Often enough, the vertex set will be embedded in a metric space so that a natural notion of distance between vertices exist. This underlying metric is implicitly used in many cases where we want to define a similarity graph on a set of points belonging to the metric space. Finding such a metric can be difficult and is the topic of metric learning.

5.1 fully-connected graphs

The simplest form of similarity graph is a fully connected graph ($E = V \times V$), with the weight of each edge being the inverse similarity.

A default choice of similarity is the gaussian smoothing $s : (x, y) \mapsto e^{-\frac{1}{2\sigma^2}\|x-y\|^2}$

This approach has the drawback of creating a very dense graph. This is an issue first because the time complexity of graph algorithms typically have a dependency on the number of edges, which means they will be slower on such graphs. A less obvious drawback comes from the approximator

nature of similarity functions. The reason we are using similarity functions instead of a distance in the first place is because defining a meaningful distance is hard, and we instead use a similarity to distinguish elements "roughly similar" from elements "vastly different", but the precise value of the similarity does not carry much information, particularly for the low similarities.

5.2 ε-neighborhood graphs

In order to mitigate the drawbacks of fully-connected graphs, one can sparsify such a graph by deleting edges that correspond to low similarities (which is essentially equivalent to approximating all big distances by infinity).

This sparsification can be performed with a threshold ε, i.e. connecting two nodes in the graph only if their distance is smaller than ε. This is called an ε-neighborhood graph.

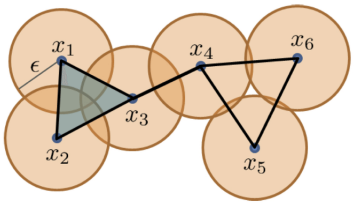

Figure 3: An ε-graph

In such a graph each edge has roughly the same scale ε. It is unclear whether adding precise weights on existing edges would bring additional information, which is why ε-neighborhood graphs are often unweighted.

The degree of freedom here is the threshold ε. A common heuristic to choose a meaningful ε is to first build the fully-connected graph and then choose ε as the length of the longest edge in a minimum spanning tree. This of course ensures connectivity of the built ε-graph. One problem with that heuristic is that it is not robust to outliers, in the sense that any point too far from the others would dramatically increase the scale of ε, thus making the ε-neighborhood graph a very dense graph, if not fully-connected.

As a matter of fact, Penrose proved in 1999 that, given a n-sampling from the uniform distribution in $[0,1]^d$, the choice of $\varepsilon = \left(\frac{\log N}{N}\right)^{\frac{1}{d}}$ is a sufficient condition for the ε-neighborhood graph to be connected, where N is the number of nodes and d the dimension of the space.

5.3 k-nearest neighbors neighborhood graphs

ε-neighborhood graphs can be highly unbalanced, in the sense that many points may have a very high degree, while others have only few neighbors. A more balanced graph can be obtained by instead selecting for each node exactly k neighbors, the k closest, or k most similar samples in the set of points[2]. With this framework, it is possible to build at least three different graphs: one directed graph and two unoriented ones. The unoriented graphs can be derived from the directed one.

We can add an oriented edge between x and y if y is a k-NN of x.

We can add an unoriented edge between two points x and y if x is a k-NN of y **or** y is a k-NN of x. This graph can be obtained from the directed one by removing the directional information from this graph.

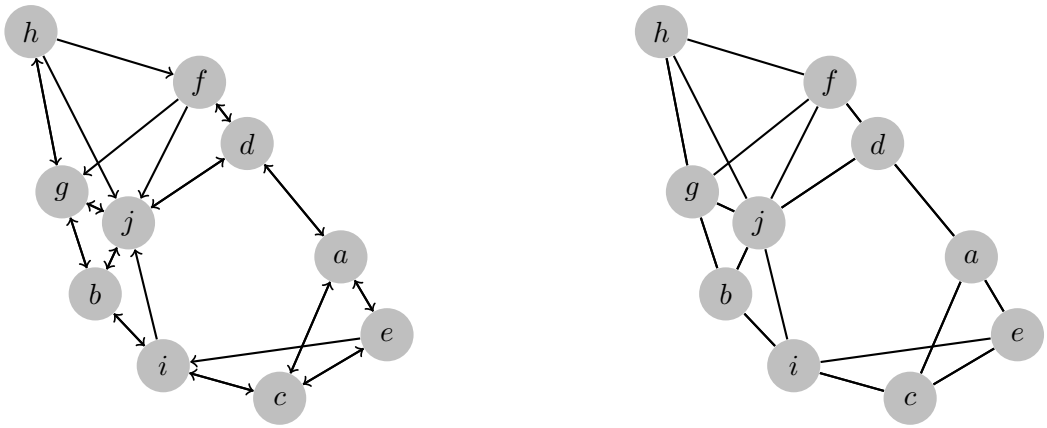

Figure 4: Directed 3-NN graph and its associated **OR** undirected graph

We can add an unoriented edge between two points x and y if x is a k-NN of y **and** y is a k-NN of x.

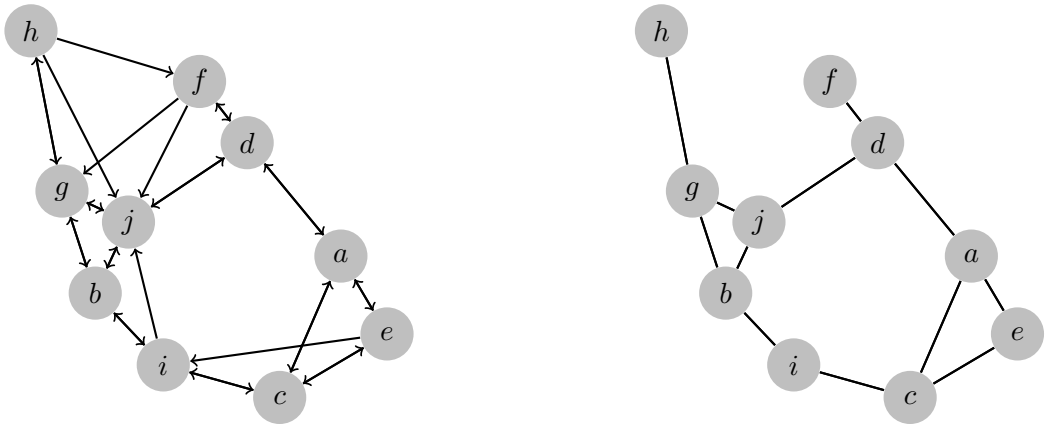

Figure 5: Directed 3-NN graph and its associated **AND** undirected graph

[2]By convention, a point is not considered a neighbor of itself

This graph is a subgraph of the first one and can be constructed from the second one by removing arcs that are not paired with their opposite. This is called *mutual* k-NN. One advantage of this mutual k-NN construction is that it can make appear clusters of points in the form of different connected components since this construction tends to not connect regions with different density. However this behavior might also be unwanted as it can isolate points even if not pertinent for the model. That is why larger k are necessary when dealing with the mutual k-NN construction.

The degree of freedom of a chosen model is the number k of nearest neighbors we consider. The case $k = n-1$, where n is the number of points, corresponds to the fully-connected case. A common heuristic is to choose $k \approx \log n$, as suggested by some asymptotic behaviors[3]. Furthermore, $\sqrt{n}$ is a common heuristic for the upper bound we consider on k.

5.4 Choosing the correct model

Choosing the right model can be cumbersome and there is little theoretical underpinning on how to choose a model, and no more on how to choose parameters that define a model. All heuristics and all similarity functions are very data-dependent and there is a field of research, metric learning, that aims to make those heuristics more reliable.

Furthermore, building a similarity graph can be a bottleneck since it is possible that the construction procedure does not scale well, neither with the number of data points, nor with the dimension of the underlying space. Processing the graph and addressing combinatorial problems can also be untractable and we must develop approximation methods. Approximation methods include:

- Approximating the problem
- Approximating the graph (graph sparsification methods)
- Finding approximate solution (see the greedy algorithm of section 2.4 for example)

References

[1] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[2] Masashi Sugiyama. *Statistical reinforcement learning: modern machine learning approaches*. Chapman and Hall/CRC, 2015.

[3] For instance, if X is a random variable *i.i.d* from the data $X_1, \cdots, X_n$ and if $\frac{k}{n} \to 0$, then $\|X_{(k)}(X) - X\| \to 0$ almost surely. Where $X_{(k)}(X)$ denotes the k-nearest neighbor of X in the dataset. If $k = \log n$, then $\frac{k}{n} \to 0$, thus making this choice of k a reasonable one.