

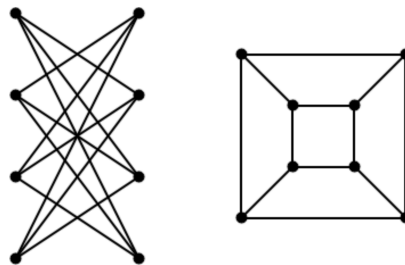
Network comparison by kernel functions

Network comparison aims to identify structural similarities and differences between networks, such as social networks from different communities, brain networks of healthy individuals and patients, or biological networks across species. Since networks often differ substantially in their size and structure, direct comparison is challenging.

Graph isomorphism

Determining whether two graphs are structurally identical is known as the *graph isomorphism* problem ([Köbler et al., 1993](#)). Consider two graphs $G_1(V_1, E_1)$ and $G_2(V_2, E_2)$ with the same numbers of nodes $n_1 = n_2$ and edges $m_1 = m_2$. Then, G_1 and G_2 are isomorphic if there exists a mapping $\sigma : V_1 \rightarrow V_2$ such that

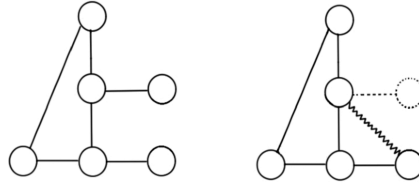
$$\{i, j\} \in E_1 \Leftrightarrow \{\sigma(i), \sigma(j)\} \in E_2.$$



Recall that the graph isomorphism problem belongs to the complexity class NP, but no polynomial-time algorithm is known for general graphs. Furthermore, graph isomorphism can only determine whether two networks are exactly the same, which is rarely the case in practice.

Graph edit distance

One way to measure the similarity between two networks is by *graph edit distance* ([Sanfeliu & Fu, 1983](#)). Given a set of graph edit operations, such as adding or removing nodes and edges, the graph edit distance is the minimum number of operations required to transform one graph into another.



Consider two graphs $G_1(V, E_1)$ and $G_2(V, E_2)$ on the same set of nodes V . Assuming that only edge additions and removals are allowed, their graph edit distance is simply the number of edges present in one graph but not the other, which can be computed in linear time $\mathcal{O}(m_1 + m_2)$.

$$D(G_1, G_2) = |E_1 \setminus E_2| + |E_2 \setminus E_1|$$

However, if the graphs have different sets of nodes V_1 and V_2 but of the same size $n = n_1 = n_2$, there are $n!$ possible matchings between their nodes (i.e., the *graph matching* problem). Furthermore, if the graphs have different numbers of nodes $n_1 \neq n_2$, as is usually the case for real-world networks, the problem becomes even harder, as node additions and removals must also be considered. Therefore, the graph edit distance is not an effective approach for comparing large networks.

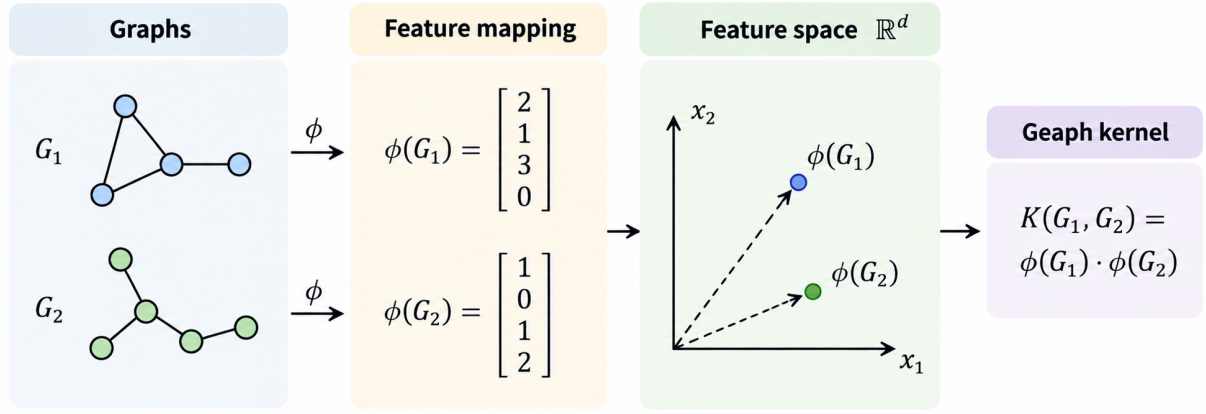
Graph kernel functions

An alternative is to compare networks based on their structural properties, without explicitly matching their nodes and edges. This is the main idea behind *graph kernels*, which measure the similarity between graphs by comparing their structural features, such as shortest paths or small subgraphs.

Formally, a graph kernel is a function

$$K(G_1, G_2) = \phi(G_1) \cdot \phi(G_2),$$

where ϕ maps each graph into some feature space $\mathbb{R}^d$ and $\cdot$ denotes the dot product. A valid graph kernel must be symmetric and positive semidefinite. This guarantees that the kernel corresponds to a dot product in a feature space.



Since kernel values depend on graph size and other factors, normalized kernels are often used

$$\widehat{K}(G_1, G_2) = \frac{K(G_1, G_2)}{\sqrt{K(G_1, G_1)K(G_2, G_2)}}$$

Note that the feature mapping ϕ need not be computed explicitly, as long as the dot product $K(G_1, G_2)$ can be computed directly. This is known as the *kernel trick*. Graph kernels are particularly useful in machine learning algorithms that rely on dot products, but can also be used directly to measure similarity between networks.

Weisfeiler-Lehman kernel

One of the most widely used graph kernels is the *Weisfeiler-Lehman kernel* ([Shervashidze et al., 2011](#)). It iteratively refines node colors based on the colors of their neighbors, allowing graphs to be compared by the frequencies of the resulting colors.

Initially, each node i is assigned the same color $c_i^{(0)} = c$. Then, at each iteration t , the color of a node is updated based on its previous color and the multiset of colors of its neighbors. Formally,

$$c_i^{(t)} = h \left(c_i^{(t-1)}, \{ \{ c_j^{(t-1)} \mid j \in \Gamma_i \} \} \right),$$

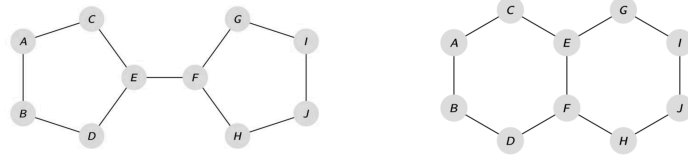
where h is an injective function (e.g., a hash function without collisions). The procedure is repeated for a fixed number of iterations, gradually capturing larger neighborhoods around each node.

Finally, the Weisfeiler-Lehman kernel compares two graphs by counting the occurrences of identical node colors across all iterations,

$$K(G_1, G_2) = \sum_t \sum_c n_c^{(t)}(G_1) n_c^{(t)}(G_2),$$

where $n_c^{(t)}(G)$ denotes the number of nodes with color c in graph G after iteration t . Thus, graphs with similar local neighborhood structures tend to have higher kernel similarity.

Note that if the frequencies of colors differ at any iteration, the graphs are not isomorphic. However, if the frequencies remain identical, they may or may not be isomorphic, since the Weisfeiler-Lehman test cannot distinguish all non-isomorphic graphs.



Graphlet sampling kernel

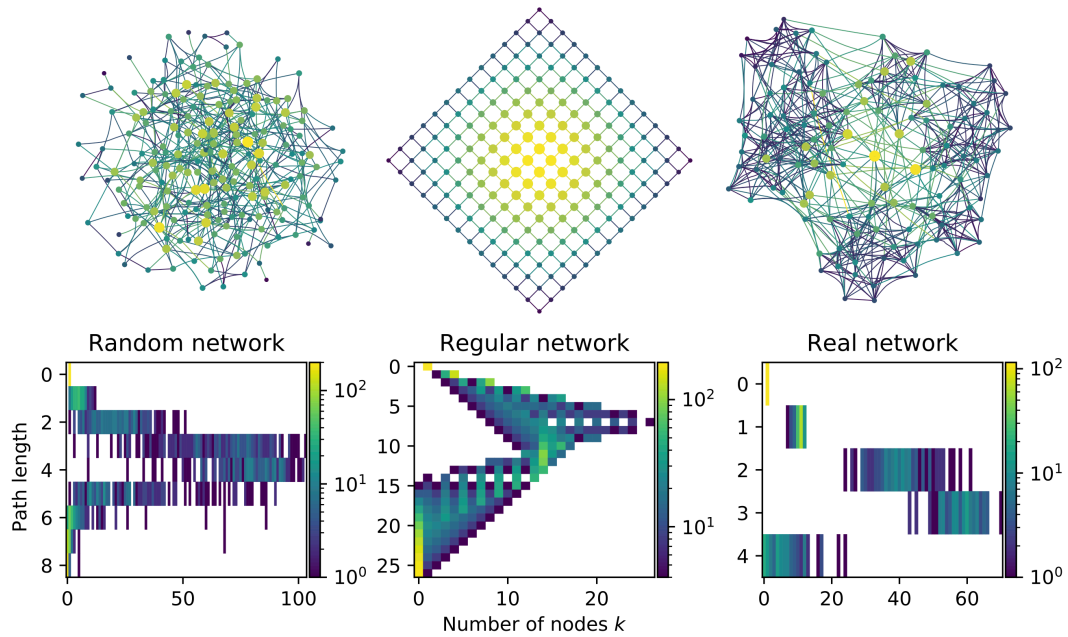
The *graphlet sampling kernel* compares graphs based on the frequencies of small induced subgraphs called *graphlets* ([Shervashidze et al., 2009](#)). Specifically, it counts the occurrences of graphlets of fixed size s and computes the dot product of their frequency vectors. Since enumerating all graphlets is computationally expensive $\mathcal{O}(n^s)$, the frequencies can instead be estimated by randomly sampling subsets of nodes.

Network comparison measures

Besides graph kernels, networks can also be compared using measures that directly quantify differences in their structure. These measures are often better suited for direct network comparison, whereas graph kernels are primarily designed for kernel-based machine learning. Unlike graph kernels, they do not necessarily correspond to dot products in a feature space and need not be positive semidefinite.

Portrait divergence

Portrait divergence measures the structural dissimilarity between two networks based on the distributions of distances between the nodes ([Bagrow & Bollt, 2019](#)). It relies on a matrix $B \in \mathbb{R}^{(d_{\max}+1) \times n}$ called *network portrait*, where $d_{\max}$ is the maximum distance or diameter of the network. The entry B_{dk} denotes the number of nodes in a network that have exactly k other nodes at distance d . The portraits of two networks are converted into probability distributions, and their difference is measured using the Jensen-Shannon divergence. Computing the network portrait requires finding the distances between all pairs of nodes, which can be done in quadratic time $\mathcal{O}(nm)$.



Graphlet agreement

Graphlet agreement compares networks based on the distributions of node positions, or *orbits*, defined by graphlet automorphisms ([Pržulj, 2007](#)). It measures the agreement between the corresponding orbit distributions and averages the individual agreements to obtain an overall similarity score between the networks.

