Instructor: Sael Lee
CS549 Spring – Computational Biology

LECTURE 21: GRAPH KERNELS

Resources:
  - Also their slides presented in *ICML2004*
How to define a valid kernel function $K(G_j, G_j)$, between two graphs $G_j$ and $G_j$.

- $K(G_j, G_j)$ should provide relationship (similarity / dissimilarity / correlation etc.) measure for between two graphs.
- $K(G_j, G_j)$ should be able to be applied in kernel based machine learning methods such that it provide optimal classification / clustering performance.

We will look at graph kernels that states similarity between kernels.
Marginalized Kernels

- Assume **hidden variables** $h$ (ex: walk of a graph) and make use of the probability distribution of **visible variables** $x, x'$ (structured data ex: Graph) and hidden variables

**Marginalized Kernels**: Expectation of the joint kernel over all possible values of $h$ and $h'$

$$K(x, x') = \sum_h \sum_{h'} K_z(z, z') p(h|x)p(h|x')$$

posterior probability $p(h|x)$ can be interpreted as a **feature extractor** that extracts informative features for classification from $x$
• A graph $G=(V, E, l)$,
  • $V$ is the set of vertices,
  • $E \subseteq (V \times V)$ is the set of undirected edges (Changed to directed for random walk), and
  • $l : V, E \rightarrow \Sigma$ is a function that assigns labels from an alphabet $\Sigma$ to nodes in the graph.

Changing undirected graph to directed graph

• ’s’ and ’d’ denote single and double bonds, respectively.
• Kernel assumes a directed graph, undirected edges are replaced by directed edges
Hidden variable: Random Walks on Graphs

- Hidden variable \( h = (h_1, ..., h_l) \) associated with graph \( G \) is a sequence of natural numbers from 1 to \(|G|\).

- \( h \) is generated by a random walk
  
  1-st step) \( h_1 \) is sampled from the prior probability distribution \( p_s(h) \).

  i-th step) \( h_i \) sampled subject to the transition probability \( p_t(h_i|h_{i-1}) \)
  and with walk termination probability \( p_q(h_{i-1}) \):

  \[
  \sum_{j=1}^{|G|} p_t(j|i) + p_q(i) = 1.
  \]

- Posterior probability for the walk \( h : p(h|G) \)

  \[
  p(h|G) = p_s(h_1) \prod_{i=2}^l p_t(h_i|h_{i-1}) p_q(h_{l}),
  \]

  where \( l \) is the length of \( h \)

- traversed labels are listed: \( v h_1 e h_1 h_2 v h_2 e h_2 h_3 v h_3 \cdots \)
Define joint kernel

Assume that two kernel functions are readily defined:

- \( K(v; v') \): Kernel between vertex labels
- \( K(e; e') \): Kernel between edge labels,

Constrain both kernels to be nonnegative

\[
K(v; v') \geq 0; \quad K(e; e') \geq 0
\]

Example of the vertex label kernels

- Dirac kernel: For Discrete labels
  \[
  K(v, v') = \delta(v = v'),
  \]
- Gaussian kernel: For Real value labels
  \[
  K(v, v') = \exp(- \| v - v' \|^2 / 2\sigma^2)
  \]

Joint Kernel

\[
K_z(z, z') = \begin{cases} 
0 & (\ell \neq \ell') \\
K(v_{h_1}, v'_{h_1}) \prod_{i=2}^{\ell} K(e_{h_{i-1}h_i}, e'_{h'_{i-1}h'_{i}}) \times \\
& K(v_{h_{\ell}}, v'_{h'_{\ell}}) & (\ell = \ell')
\end{cases}
\]

where \( z = (G, h) \).
**Computing Joint Kernel**

\[ K(G, G') = \sum_{\ell=1}^{\infty} \sum_{h} \sum_{h'} \left( P_s(h_1) \prod_{i=2}^{\ell} p_t(h_i|h_{i-1})p_q(h_1) \times p_s'(h_1') \prod_{j=2}^{\ell} p_t'(h_j'|h_{j-1}')p_q'(h_1') \right) \times \]

\[ K(v_{h_1}, v'_{h_1'}) \prod_{k=2}^{\ell} K(e_{h_{k-1}h_k}, e'_{h'_{k-1}h'_{k}}) K(v_{h_k}, v'_{h_k'}) , \]

\[ K(G, G') = \sum_{h_1, h'_1} s(h_1, h'_1) \lim_{L \to \infty} \sum_{\ell=1}^{L} r_\ell(h_1, h'_1) \]

\[ r_\ell(h_1, h'_1) := \left( \sum_{h_2, h'_2} t(h_2, h'_2, h_1, h'_1) \left( \sum_{h_3, h'_3} t(h_3, h'_3, h_2, h'_2) \times \sum_{h_\ell, h'_\ell} t(h_\ell, h'_\ell, h_{\ell-1}, h'_{\ell-1}) q(h_\ell, h'_\ell) \right) \right) \cdots \]

Where

\[ \sum_h := \sum_{h_{1=1}}^{G} \cdots \sum_{h_{\ell=1}}^{G} . \]

The straightforward enumeration is **impossible**, because \( \ell \) spans from 1 to infinity.
Restate this problem in **recursive form**

**Equilibrium equation:**

$$R_{\infty}(h_1, h'_1) = r_1(h_1, h'_1) + \sum_{i, j} t(i, j, h_1, h'_1) R_{\infty}(i, j)$$
The computation of the marginalized kernel finally comes down to iteratively solving for

\[
R_L(h_1, h'_1) = r_1(h_1, h'_1) + \sum_{k=2}^{T} r_k(h_1, h'_1)
\]

\[
r_k(h_1, h'_1) = \sum_{i,j} t(i, j, h_1, h'_1) r_{k-1}(i, j).
\]

starting from

\[
R_1(h_1, h'_1) = r_1(h_1, h'_1) := q(h_1, h'_1)
\]

\[
q(h_\ell, h'_\ell) := p_q(h_\ell)p'_q(h'_\ell)
\]

and substituting the solutions into

\[
K(G, G') = \sum_{h_1, h'_1} s(h_1, h'_1) \lim_{L \to \infty} \sum_{\ell=1}^{L} r_\ell(h_1, h'_1)
\]

\[
s(h_1, h'_1) := p_s(h_1)p'_s(h'_1)K(v_{h_1}, v'_{h'_1})
\]

until convergence

Proof of convergence in Section 3.4 of Kashima et al., 2003
EXTENSION TO MARGINALIZED GRAPH KERNEL

Model: Marginalized Graph Kernel with **Dirac** joint kernel

**Approaches:**

1. **Size of product graph affects runtime of kernel computation**
   - The more node labels, the smaller the product graph
   - Trick: Introduce new artificial node labels

   **Iterative Label Enrichment:**
   **Morgan Index** *(1965)*

2. Focusing on non-tottering walks is a way to get closer to the path kernel

   **Reduce Tottering effect by**
   **Using 2nd Order Markov Random Walk** instead of 1st order
SIMPLIFIED MARGINALIZED GRAPH KERNEL

\( K \): Marginalized graph kernel

\[
K(x, x') = \sum \sum K_z(z, z') p(h|x)p(h'|x').
\]

\[
K_z(z, z') = \begin{cases} 
0 & \text{if } \ell \neq \ell' \\
K(v_{h_1}, v_{h'_1}) \prod_{i=2}^{\ell} K(e_{h_{i-1}h_i}, e'_{h'_{i-1}h'_i}) & \text{if } \ell = \ell'
\end{cases}
\]

where \( z = (G, h) \).

Simplified by
1) not using edge kernel defined
2) Using Dirac vertex kernel

\[
K(G, G') = \sum_{(h, h') \in V^* \times V'^*} p(h|G)p'(h'|G')K_L(l(h), l(h'))
\]

\( K_L \): Dirac kernel between labeled sequence

\[
K_L(l, l') = \begin{cases} 
1 & \text{if } l = l' \\
0 & \text{otherwise}
\end{cases}
\]

\[
p(v_1 \ldots v_n) = p_s(v_1) \prod_{i=2}^{n} p_t(v_i|v_{i-1}).
\]

\[
\begin{align*}
p_s(v) &= p_0(v)p_q(v), \\
p_t(u|v) &= \frac{1-p_q(v)}{p_q(v)} p_a(u|v)p_q(u).
\end{align*}
\]
Tensor product graph id defined as labeled graph $G_p = (V_p, E_p)$ with $V_p \subset V_1 \times V_2$ are pairs of vertices with identical labels $(v_1, v_2) \in V_p$ iff $l(v_1) = l(v_2)$ and edges connecting the vertices $(u_1, u_2) and (v_1, v_2)$ iff $(u_i, v_i) \in E_p, for \ i = 1, 2, \ldots l$

A function $\pi$ on the set of walks (paths) $H(G_p)$

$$\pi((u_1,v_1)(u_2,v_2)\ldots(u_n,v_n)) = \pi_s((u_1,v_1))\prod_{i=2}^{n}\pi_t((u_i,v_i)|(u_{i-1},v_{i-1})),$$

with

$$\pi_s(u_1,u_2) = p_s^{(1)}(u_1)p_s^{(2)}(u_2),$$

$$\pi_t((v_1,v_2)|(u_1,u_2)) = p_t^{(1)}(v_1|u_1)p_t^{(2)}(v_2|u_2).$$

$$A \otimes B =$$

$$\begin{bmatrix}
    a_{11}b_{11} & a_{11}b_{12} & \cdots & a_{1n}b_{11} & a_{1n}b_{12} & \cdots & a_{1n}b_{1q} \\
    a_{11}b_{21} & a_{11}b_{22} & \cdots & a_{1n}b_{21} & a_{1n}b_{22} & \cdots & a_{1n}b_{2q} \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
    a_{11}b_{p1} & a_{11}b_{p2} & \cdots & a_{1n}b_{p1} & a_{1n}b_{p2} & \cdots & a_{1n}b_{pq} \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
    a_{m1}b_{11} & a_{m1}b_{12} & \cdots & a_{mn}b_{11} & a_{mn}b_{12} & \cdots & a_{mn}b_{1q} \\
    a_{m1}b_{21} & a_{m1}b_{22} & \cdots & a_{mn}b_{21} & a_{mn}b_{22} & \cdots & a_{mn}b_{2q} \\
    \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\
    a_{m1}b_{p1} & a_{m1}b_{p2} & \cdots & a_{mn}b_{p1} & a_{mn}b_{p2} & \cdots & a_{mn}b_{pq}
\end{bmatrix}.$$
**Simplified Marginalized Graph Kernel in Matrix Format**

\[ K(G_1, G_2) = \sum_{(h_1, h_2) \in V_1^* \times V_2^*} p_1(h_1 | G_1) p_2(h_2 | G_1) K_L(l(h_1), l(h_2)) \]

\[ K(G_1, G_2) = \sum_{h \in H(G)} \pi(h). \]

\[ \sum_{h \in H(G), |h|=n} \pi(h) = \pi_s^\top \Pi_t^n 1, \]
Problems:
• The computation of graph kernels is time-consuming.
• Need to increase the relevance of the features used to compare graphs.

Expected outcome:
• The computation of graph kernels is time-consuming.
• Need to increase the relevance of the features used to compare graphs.
Enrichment with vertex connectivity properties
→ extended connectivity descriptor:

Algorithm:
- $M_0(v) = 1, \forall v$
- $M_t(v) = \sum_{\text{neig}(v)} M_{t-1}(u)$
- $l_t(v) = l(v) \circ M_t(v)$
$M_n$: vector of labels in graph
Given adjacency matrix $A$ and setting $M_0 = 1$
$M_{n+1} = (A + I)M_n$

⇒ Family of kernels $K_n$:
\[
\begin{cases}
\text{decreased kernel complexity } (\mathcal{O}(|G_1 \times G_2|^3)) \\
\text{(potential) increased kernel expressivity}
\end{cases}
\]
PREVENTING TOTTERING

Tottering

A **tottering walk** is a walk $w = v_1 \ldots v_n$ with $v_i = v_i + 2$ for some $i$.

- A walk can visit the same cycle of nodes all over again
- Kernel measures similarity in terms of common walks
- Hence a small structural similarity can cause a huge kernel value
- Focusing on non-tottering walks is a way to get closer to the path kernel (e.g., equivalent on trees).
PREVENTING TOTTERING CONT.

- Tottering path: \( h = (v_1, \ldots, v_n) \), \( \exists i: v_{i+2} = v_i \).

\[
\begin{align*}
\text{Path: } h &= (v_1 \rightarrow v_2 \rightarrow v_3) \\
\text{l(h)} &= C - C - C \\
\text{h}(v_1 \rightarrow v_2 \rightarrow v_1)
\end{align*}
\]

\( \Rightarrow \) preventing totters \( \Leftrightarrow \) filtering blue path.
PREVENTING TOTTERING CONT.

- Motivation:

  - Every path of $G_2$ can be matched to a tottering path of $G_1$
  - $\Rightarrow$ Compounds are considered as identical
PREVENTING TOTTERING CONT.

- **Motivation:**

- ![Graphs showing different lengths and tottering paths]
  - Length 1
  - Length 2, no tottering
  - Only “real” chemical path are matched
  - ⇒ Compounds are now seen as different

- **Solution:** increase the order of the random walk model:

  \[ p_G(h) = p_s(v_1)p_t(v_2|v_1) \prod_{i=3}^{n} p_t(v_i|v_{i-2}, v_{i-1}) \]
The function is still a valid kernel but the implementation described for the first order Markov random walk cannot be directly used anymore.

=> Instead of explicitly working with 2nd Order Markov Random walk, transform the original graph $G$ to $G'$ such that $G'$ contains the look ahead information.
Transformation: \[ G = (V, E, l) \Rightarrow G' = (V', E', l') \] where:

- \[ V' = V \cup E \]
- \[ E' = \{ (v, (v, t)) \mid v \in V, (v, t) \in E \} \cup \{ ((u, v), (v, t)) \mid (u, v), (v, t) \in E, u \neq t \} \]
Transformation: $G = (V, E, l) \Rightarrow G' = (V', E', l')$ where:

- $V' = V \cup E$

- $E' = \{ (v, (v, t)) | v \in V, (v, t) \in E \} \cup \{ ((u, v), (v, t)) | (u, v), (v, t) \in E, u \neq t \}$
Transformation: \( G = (V, E, l) \Rightarrow G' = (V', E', l') \) where:

- \( V' = V \cup E \)

- \( E' = \{ (v, (v, t)) \mid v \in V, (v, t) \in E \} \)
  \( \cup \{ ((u, v), (v, t)) \mid (u, v), (v, t) \in E, u \neq t \} \)
Original Graph

Corresponding directed graph $G = (V,E,I)$

Transformed Graph

Labels in the transformed graph
Consider: 
\[ H_0(G) = \{\text{Non tottering paths of } G\} \]
\[ H_1(G') = \{\text{Paths of } G' \text{ starting from a node } v \in V \} \]

Theorem: \( p' \) factorizes as

\[ p'(h') = p'_s(v'_1) \prod_{i=2}^{n} p'_t(v'_i|v'_{i-1}) \]

- \( p'_s(v') = p_s(v') \)
- \( p'_t(u'|v') = \begin{cases} 
  p_t(u'|v) & \text{if } v' \in V \text{ and } u' = (v', u) \in E \\
  p_t(u|v, w) & \text{if } v' = (v, w) \text{ and } u' = (w, u) \in E 
\end{cases} \)

Corollary:

- graph transformation
- original graph kernel \( \Rightarrow \) tottering paths removed
Consider:

\[
H_0(G) = \{ \text{Non tottering paths of } G \} \\
H_1(G') = \{ \text{Paths of } G' \text{ starting from a node } v \in V \}
\]

The mapping \( f : H_0(G) \rightarrow H_1(G') \) defined by

\[
h = (v_1, ..., v_n) \mapsto h' = (v'_1, ..., v'_n)
\]

such that

\[
\begin{cases}
v'_1 = v_1 \\
v'_i = (v_{i-1}, v_i)
\end{cases}
\]

establishes a bijection between \( H_0(G) \) and \( H_1(G') \).

Let \( p' \) be the image of \( p_G \) by \( f \):

\[
\forall h' \in H_1(G'), \quad p'(h') := p_G \left( f^{-1}(h') \right)
\]
• **Bijection** (or bijective function or one-to-one correspondence) is a function giving an **exact** pairing of the elements of two sets.

• **Bijective function** $f: X \rightarrow Y$ is a one to one and onto mapping of a set $X$ to a set $Y$.

![Venn diagram](attachment:image.png)  

A bijection composed of an injection (left) and a surjection (right).

Theorem 1. \( f \) is a **Bijective function** between \( H_0(G) \) and \( H_1(G') \), and for any path \( h \in H_0(G) \) we have

\[
f : H_0(G) \rightarrow H_1(G')
\]

\[
\begin{align*}
  l(h|G) &= l'(f(h)|G') \\
p(h|G) &= p'(f(h)|G')
\end{align*}
\]

Corollary 1. For any two graphs \( G_1 \) and \( G_2 \), the marginalized graph kernel can be expressed in terms of the transformed graphs \( G'_1 \) and \( G'_2 \) by:

\[
K(G_1, G_2) = \sum_{(h'_1, h'_2) \in (\Sigma'_1)^* \times (\Sigma'_2)^*} p'_1(h'_1)p'_2(h'_2) K_L(l'_1(h'_1)l'_2(h'_2))
\]