Introduction to Kernels (part III) Application to Graphs, Molecular Structures, Chemistry

Liva Ralaivola

liva@ics.uci.edu

School of Information and Computer Science

Institute for Genomics and Bioinformatics

Outline

- Machine learning and graphs
- Quick reminders on graphs
- Naive/stupid kernels
- Convolution kernels
- Path/walk graph kernels
- Graph kernels based on powers of adjacency matrices
- Marginalized kernels (next session)
- Depth-first search kernels (next session)

Machine learning and graphs

- Problems
  - text processing problems (words viewed as vertices of graphs)
  - natural language processing (sentences viewed as trees)
  - digital image interpretation (different areas of one image are interpreted as vertices of a graphs)
  - classification of chemical compounds (toxicity/non toxicity, activity/inactivity, etc.)

\# methods

many $\longleftarrow$ few

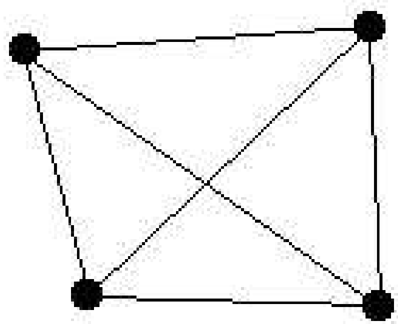
$$\mathcal{X} = \quad \mathbb{N}^d, \mathbb{R}^d \quad \cdots \quad \text{sequences} \quad \cdots \quad \text{graphs}$$

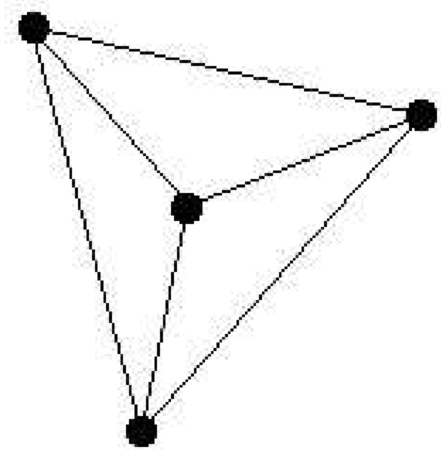
"easy" $\longrightarrow$ hard

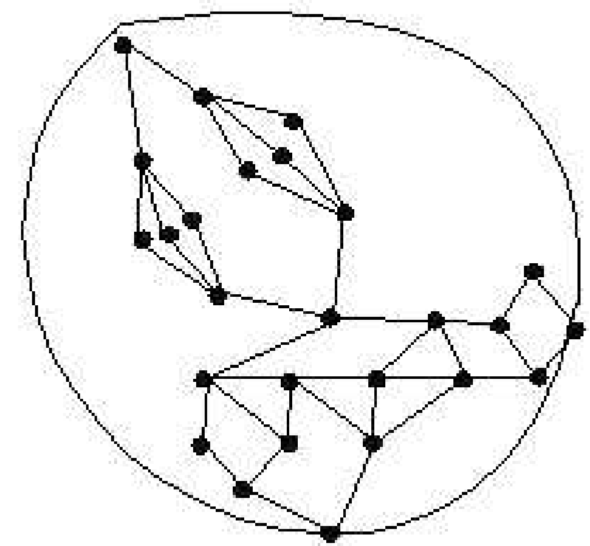
difficulty

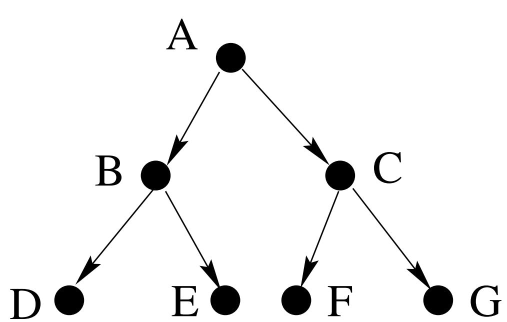
Quick reminders on graphs (1/5)

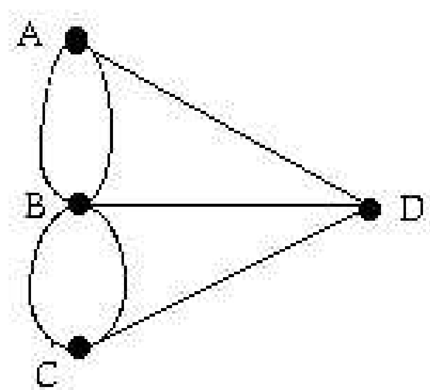
- Examples

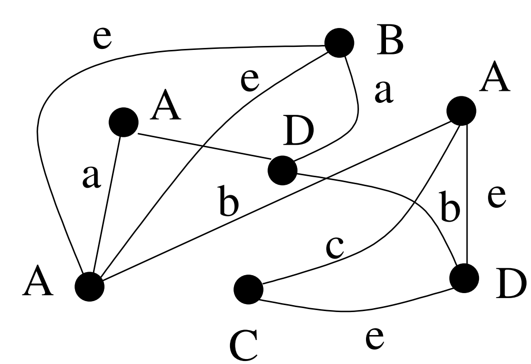






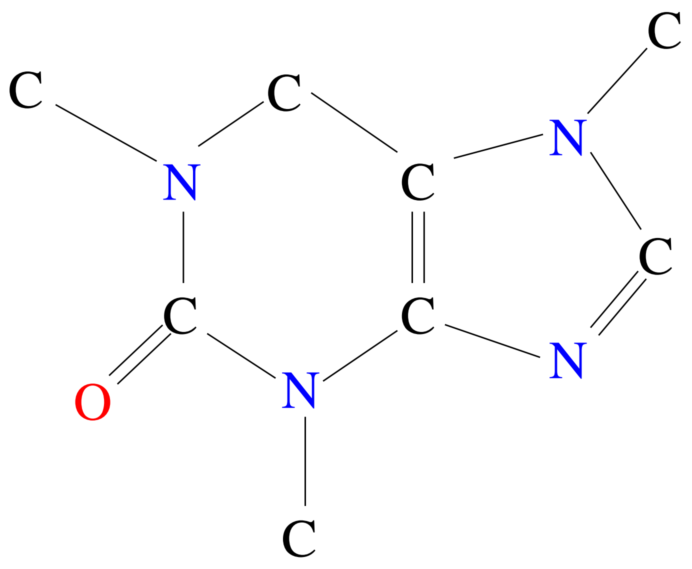






Quick reminders on graphs (2/5)

- Focus: undirected labeled graphs
- A graph is made of
  - labeled vertices or nodes
  - labeled edges (connect nodes)
- For chemical compounds
  - atom/node labels: $\mathcal{A} = \{C, N, O, H, \ldots\}$
  - bond/edge labels: $\mathcal{B} = \{s, d, t, ar, \ldots\}$

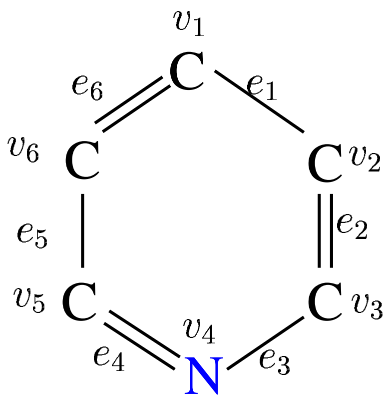


caffeine

Quick reminders on graphs (3/5)

- Notations
  - $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{v_1, \ldots, v_{|\mathcal{V}|}\}$, $\mathcal{E} = \{e_1, \ldots, e_{|\mathcal{E}|}\}$,
    - $n = |\mathcal{V}|$, number of vertices
    - $m = |\mathcal{E}|$, number of edges
    - $label(v_i) \in \mathcal{A} = \{\ell_1^a, \ldots, \ell_{|\mathcal{A}|}^a\}$
    - $label(e_i) \in \mathcal{B} = \{\ell_1^b, \ldots, \ell_{|\mathcal{B}|}^b\}$
  - E: $n \times n$ adjacency matrix
    - $E_{ij} = 1$ if and only if there is an edge between v_i and v_j
  - L: $|\mathcal{A}| \times n$ vertex label matrix
    - $L_{ri} = 1$ if and only if $label(v_i) = \ell_r^b$

Quick reminders on graphs (4/5)



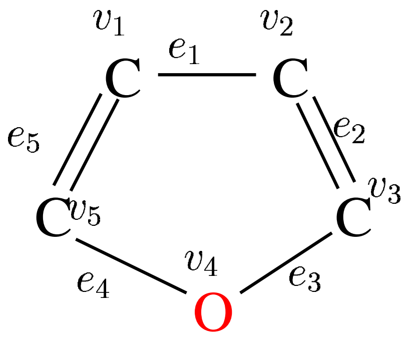
$$\mathcal{A} = \{C, N, O\}, \ |\mathcal{A}| = 3$$
$$\mathcal{B} = \{s, d, t, ar\}, \ |\mathcal{B}| = 4$$

- Pyridine, $n = 6, m = 6$

$$E = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

$$label(v_1) = label(v_2) = label(v_3) = label(v_5) = label(v_6) = C, \quad label(v_4) = N$$
$$label(e_1) = label(e_2) = label(e_3) = label(e_4) = label(e_5) = label(e_6) = ar$$

Quick reminders on graphs (5/5)



$$\mathcal{A} = \{C, N, O\}, \ |\mathcal{A}| = 3$$
$$\mathcal{B} = \{s, d, t, ar\}, \ |\mathcal{B}| = 4$$

- Furane $n = 5, m = 5$

$$E = \begin{pmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}, \qquad L = \begin{pmatrix} 1 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

$$label(v_1) = label(v_2) = label(v_3) = label(v_5) = C, \quad label(v_4) = O$$
$$label(e_1) = label(e_2) = label(e_3) = label(e_4) = label(e_5) = ar$$

Naive/stupid kernels

- $G_1 = (\mathcal{V}_1, \mathcal{E}_1), G_2 = (\mathcal{V}_2, \mathcal{E}_2)$
  - size dependent kernel (this is the stupid kernel !)

$$k(G_1, G_2) = n_1 n_2 + m_1 m_2$$

    feature map: $\phi(G) = [n\ m]^\top$
  - kernel depending on the counts of labels

$$k(G_1, G_2) = \langle \phi_{count}(G_1), \phi_{count}(G_2) \rangle$$

    with

$$\begin{aligned} \phi_{count} : &\mathcal{G} \to \mathbb{R}^{|\mathcal{A}|+|\mathcal{B}|} \\ &G \mapsto [\#C\ \#N\ \#O\ \cdots\ \#s\ \#d\ \#t\ \#ar]^\top \end{aligned}$$

Convolution kernels

- The kernels defined previously don't take the structure into account !!
- Solution: convolution kernels

$$k(G_1, G_2) = \sum_{s_1 \in \mathcal{S}(G_1), s_2 \in \mathcal{S}(G_2)} k_s(s_1, s_2)$$

  where $\mathcal{S}(G)$ is a set of subgraphs of G and k_s a kernel defined on these subgraphs
- Idea similar to the spectrum kernels
- The quality of the kernel depends on $\mathcal{S}(G)$
  - $\mathcal{S}(G)$ must retain as much information as possible on G
  - the enumeration of the elements of $\mathcal{S}(G)$ must be doable in a reasonable time

Walk/path based graph kernels

- A walk on a graph is a sequence of nodes and edges traversed on this graph
- Different kernel approaches
  - deterministic walks: kernels based on powers of some adjacency matrix
  - random walks: marginalized kernels
  - depth-first search walks: fast paths generation and Venn/Tanimoto kernels

Kernels based on powers of adjacency matrices

- $G_1 = (\mathcal{V}_1, \mathcal{E}_1), G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ two graphs
- Let $\langle A, B \rangle = \sum_{ij} A_{ij} B_{ij}$ for two matrices
- General formula for a kernel based on labeled pairs:

$$k(G_1, G_2) = \left\langle L_1 \left(\sum_{i=0}^{\infty} \lambda_i E_1^i \right) L_1^\top, L_2 \left(\sum_{i=0}^{\infty} \lambda_i E_2^i \right) L_2^\top \right\rangle$$

- Takes into account the number of paths of the same length having the same pair of first and last nodes

Kernels based on powers of adjacency matrices

- $G_1 = (\mathcal{V}_1, \mathcal{E}_1), G_2 = (\mathcal{V}_2, \mathcal{E}_2)$ two graphs
- The direct graph product $G_\times$ of G_1 and G_2 is defined as

$$\mathcal{V}_\times = \{(v_1, v_2) \in \mathcal{V}_\times(G_\times) : (label(v_1) = label(v_2))\}$$

$$\begin{aligned}\mathcal{E}_\times = \{((u_1, u_2), (v_1, v_2)) \in \mathcal{E}_\times(G_\times) : (u_1, v_1) \in \mathcal{E}_1 \\ \wedge (u_2, v_2) \in \mathcal{E}_2 \wedge (label(u_1, v_1) = label(u_2, v_2))\}\end{aligned}$$

- For G_1 and G_2, we have $k_\times$:

$$k_\times(G_1, G_2) = \sum_{i,j=1}^{|\mathcal{V}_\times|} \left[\sum_{n=0}^{\infty} \lambda_n E_\times^n \right]_{ij} \tag{1}$$

- Counts the number of common sequences of labels in G_1 and G_2

Conclusion

- Importance
  - convolution kernel
  - strategy to extract the subgraphs
  - efficiency
  - direct graph product
- To be continued