Graph Kernels

S.V.N. Vishwanathan

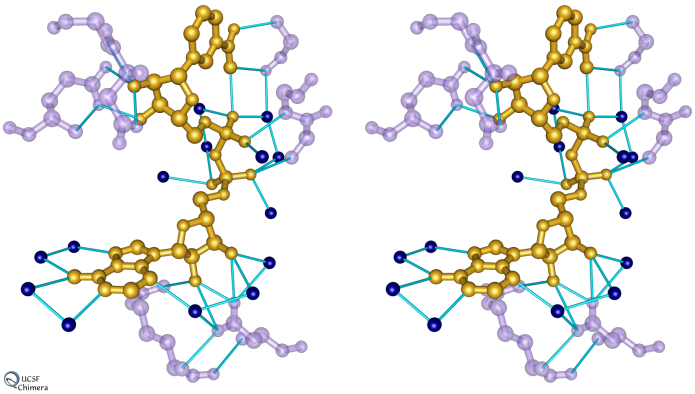
vishy@stat.purdue.edu

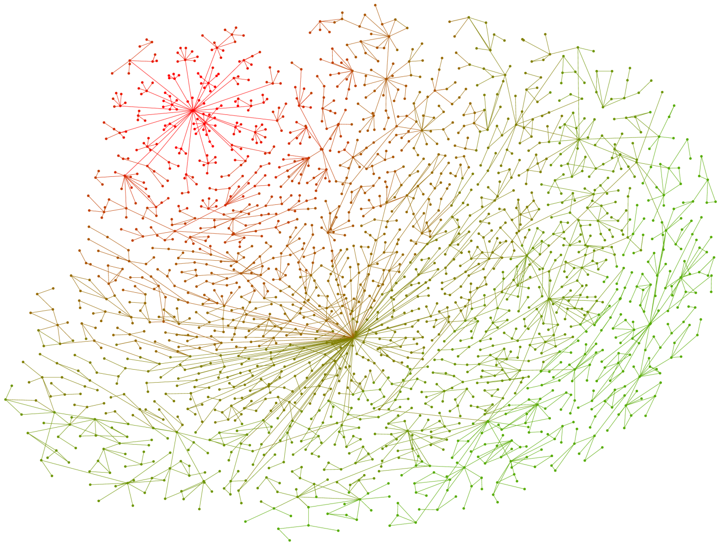
http://www.stat.purdue.edu/~vishy

Purdue University

Joint work with Karsten Borgwardt, Nic Schraudolph, and Risi Kondor

Graphs are Everywhere





Two protein molecules

The Internet

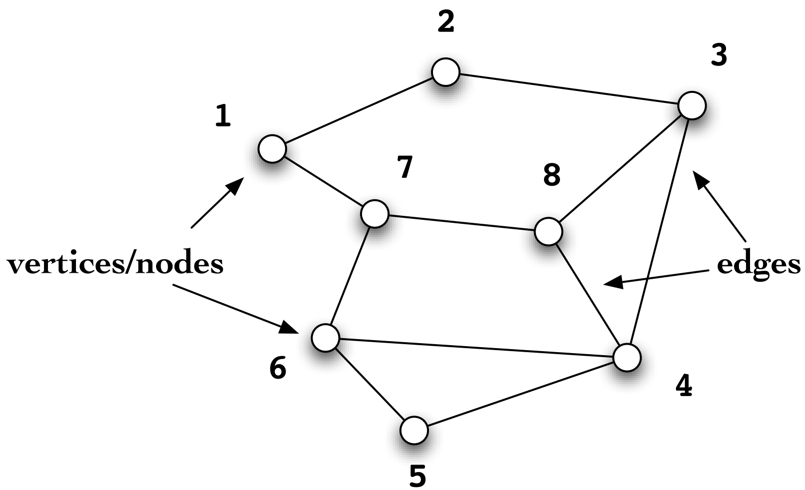
Comparing Graphs:

- How similar are two graphs?

Comparing Nodes:

- How similar are two nodes of a graph?

Adjacency Matrix

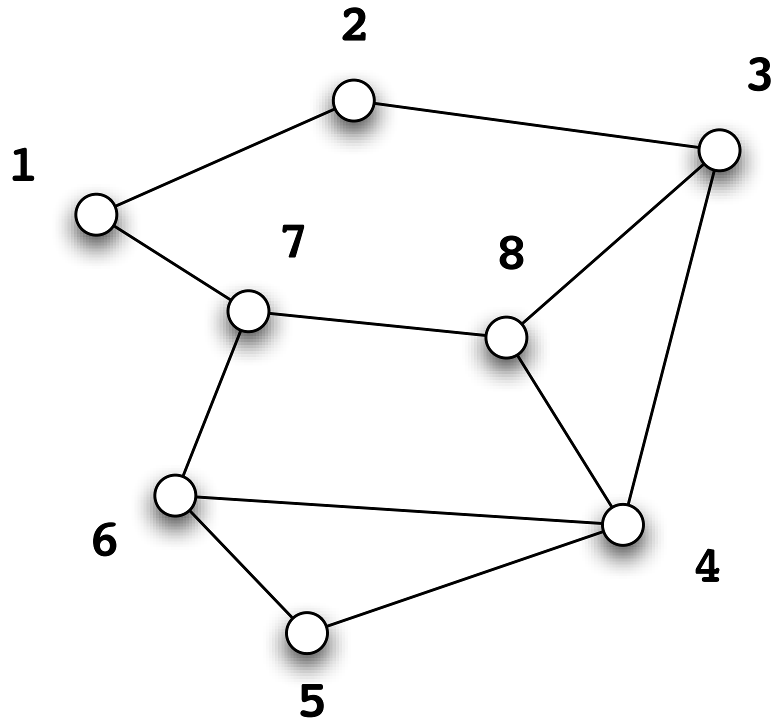


Undirected Graph G(V, E)

$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

sub-matrix of A = a subgraph of G

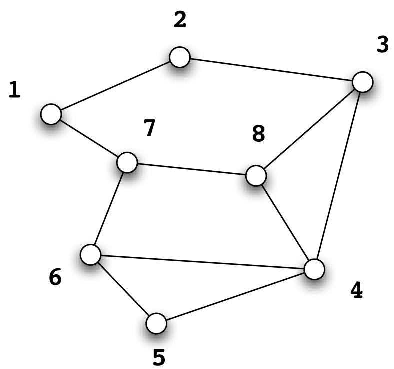
Degree Matrix



$$D = \begin{bmatrix} 2 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 3 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 4 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 3 \end{bmatrix}$$

Normalized Adjacency matrix $\tilde{A} = D^{-1}A$ is a stochastic matrix (each row sums to one)

Graph Laplacian



$$L = \begin{bmatrix} 2 & -1 & 0 & 0 & 0 & 0 & -1 & 0 \\ -1 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 3 & -1 & 0 & 0 & 0 & -1 \\ 0 & 0 & -1 & 4 & -1 & -1 & 0 & -1 \\ 0 & 0 & 0 & -1 & 2 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & -1 & 3 & -1 & 0 \\ -1 & 0 & 0 & 0 & 0 & -1 & 3 & -1 \\ 0 & 0 & -1 & -1 & 0 & 0 & -1 & 3 \end{bmatrix}$$

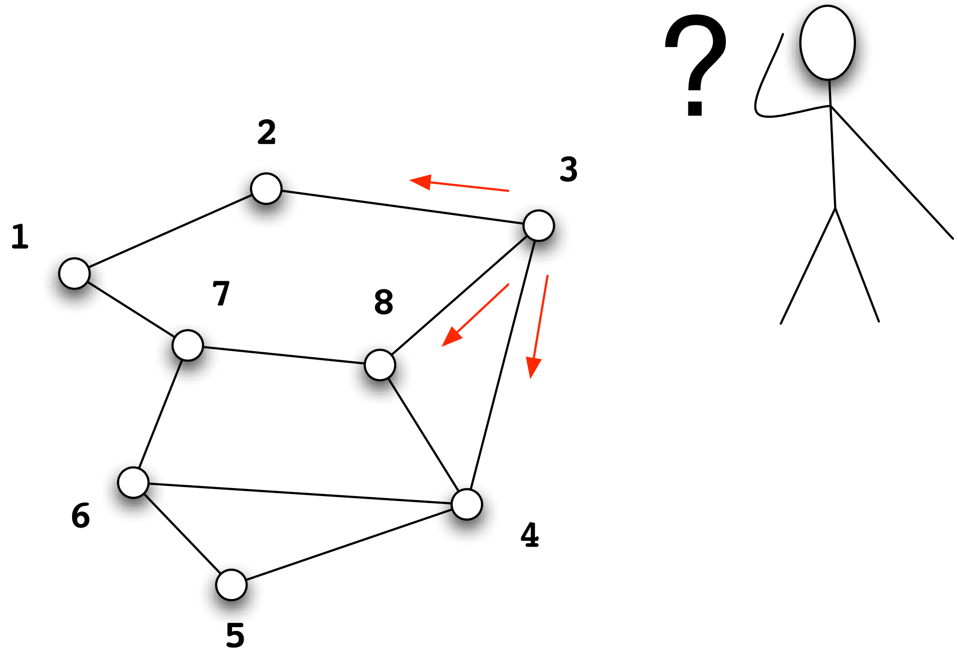
$$L = D - A$$

Normalized version

$$\tilde{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$$

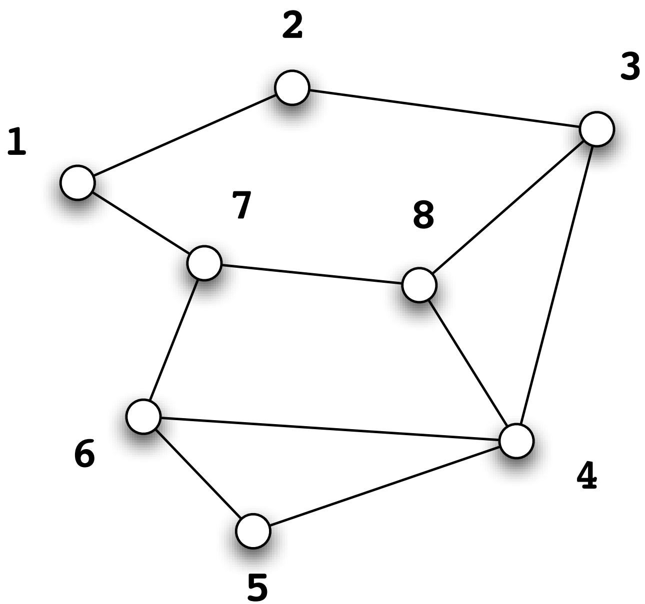
Spectrum bounded between 0 and 2

Random Walk



- From a vertex i randomly jump to any adjacent vertex j
- Probability of jumping to j proportional to $\tilde{A}_{ij}$

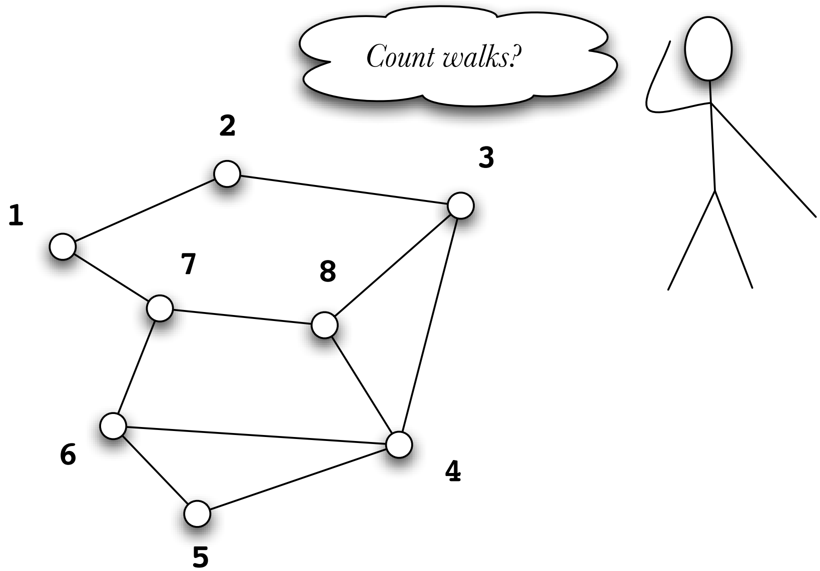
Walks of Length 2



$$A^2 = \begin{bmatrix} 2 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 2 & 0 & 1 & 0 & 0 & 1 & 1 \\ 1 & 0 & 3 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 4 & 1 & 1 & 2 & 1 \\ 0 & 0 & 1 & 1 & 2 & 1 & 1 & 1 \\ 1 & 0 & 1 & 1 & 1 & 3 & 0 & 2 \\ 0 & 1 & 1 & 2 & 1 & 0 & 3 & 0 \\ 1 & 1 & 1 & 1 & 1 & 2 & 0 & 3 \end{bmatrix}$$

- Entries of A^2 = number of length 2 walks
- Entries of $\tilde{A}^2$ = probability of length 2 walks

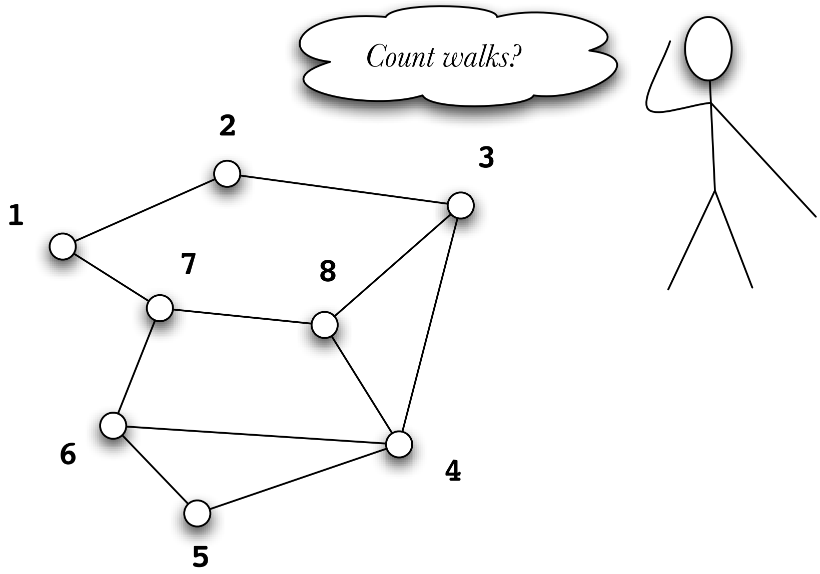
Idea!



- Count number of walks between two nodes
- Two nodes are similar if they are connected by many walks

- Does not work :(
- If graph has cycles then number of walks goes to ∞

A Better Idea!



- Discount contribution of longer walks
- Count number of walks between two nodes
- Two nodes are similar if they are connected by many walks

- Works if discounting factor chosen appropriately!

Diffusion Kernels

Discounting Factor:

- Discount a k length walk by $\lambda^k/k!$ for $0 \leq \lambda \leq 1$

Similarity:

- Similarity defined as

$$k(i,j) = \left[\sum_k \frac{\lambda^k}{k!} A^k\right]_{ij} = [\exp(\lambda A)]_{ij}$$

Kondor and Lafferty:

- Work with diffusion and hence the graph Laplacian

$$k(i,j) = \left[\sum_k \frac{\lambda^k}{k!} L^k\right]_{ij} = [\exp(\lambda L)]_{ij}$$

- They show that this is a valid p.s.d kernel

Extensions

Laplacian as a regularizer:

- For any real-valued function f on the vertices of a graph

$$\langle f, Lf \rangle = f^\top L f = -\frac{1}{2} \sum_{i \sim j} (f_i - f_j)^2$$

- Can regularize differently if we replace L by

$$r(L) := \sum_i r(\rho_i) l_i l_i^\top$$

- Any monotonically increasing function of ρ admissible

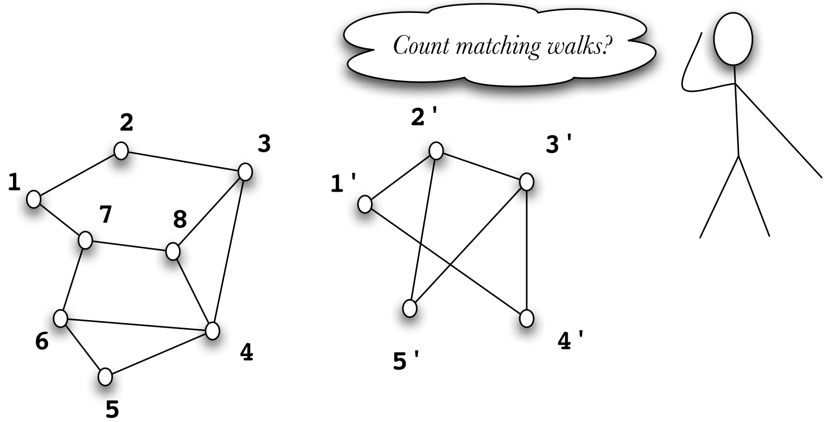
Smola and Kondor

Other Kernels:

$$r(\rho) = 1 + \sigma^2 \rho, \qquad K = (I + \sigma^2 L)^{-1}$$ regularized Laplacian

$$r(\rho) = (1 - \lambda\rho)^{-p}, \quad K = (I - \lambda L)^p$$ p-step random walk

Comparing Graphs

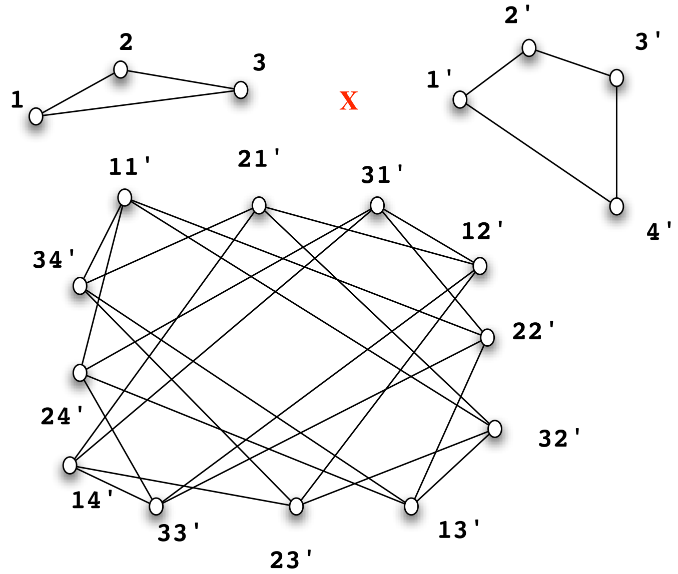


- Count number of matching walks in two graphs
- Discount contribution of longer walks
- Two graphs are similar if many walks are matching

Three Questions:

- How to formalize this intuition?
- How to compute this efficiently?
- How is this related to diffusion kernels?

Direct Product Graph



- Formal Definition

$$V_{\times}(G \times G') = \{(v, v') : v \in V, v' \in V'\}$$
$$E_{\times}(G \times G') = \{((v, v'), (w, w')) : (v, w) \in E, (v', w') \in E'\}$$

Key Insight

Random Walk on Product Graph:

- Equivalent to simultaneous random walk on input graphs

Kernel Definition:

$$k(G, G') = \frac{1}{|G|\,|G'|} \sum_k \frac{\lambda^k}{k!} \, \mathbf{e}^\top A_\times^k \, \mathbf{e} = \frac{1}{|G|\,|G'|} \mathbf{e}^\top \exp(\lambda A_\times) \, \mathbf{e}$$

Extensions

Different Decay Factor (Gärtner et al.):

- Using a λ^k decay

$$k(G,G') = \frac{1}{|G||G'|}\sum_k \lambda^k \, \mathbf{e}^\top A_\times^k \, \mathbf{e}$$
$$= \frac{1}{|G||G'|}\mathbf{e}^\top (\mathbf{I} - \lambda A_\times)^{-1} \, \mathbf{e}$$

Taking expectations:

- Instead of summing, take expectations

$$k(G,G') = \sum_k \lambda^k \, q_\times^\top A_\times^k p_\times = q_\times^\top (\mathbf{I} - \lambda A_\times)^{-1} p_\times$$

- $p_\times$ and $q_\times$ are initial and stopping probabilities resp.

Efficient Computation

Product Graph is Huge:

- If G and G' have n vertices then product graph has n^2 vertices
- Adjacency matrix $A_\times$ is of size $n^2 \times n^2$

Houston we have a problem:

- Kernel computation involves

$$k(G, G') = q_\times^\top \underbrace{\exp(\lambda A_\times)}_{O(n^6)\,!} p_\times$$

or

$$k(G, G') = q_\times^\top \underbrace{(\mathbf{I} - \lambda A_\times)^{-1}}_{O(n^6)\,!} p_\times$$

Kronecker Products

Definition (by example):

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \text{ and } B = \begin{bmatrix} 2 & 5 & 2 \\ 5 & 2 & 5 \\ 1 & 5 & 2 \end{bmatrix}$$

then

$$A \otimes B = \begin{bmatrix} 2 & 5 & 2 & 0 & 0 & 0 \\ 5 & 2 & 5 & 0 & 0 & 0 \\ 1 & 5 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 2 & 5 & 2 \\ 0 & 0 & 0 & 5 & 2 & 5 \\ 0 & 0 & 0 & 1 & 5 & 2 \end{bmatrix}$$

Key Insight

- The adjacency matrix of the product graph

$$A_\times = A \otimes A'$$

- Can compute $\exp(A_\times)$ as

$$\exp(A_\times) = \underbrace{\exp(A)}_{O(n^3)} \otimes \underbrace{\exp(A')}_{O(n^3)}$$

- Computing $(\mathbf{I} - \lambda A)^{-1}$ involves a bit more work ...

Sylvester Equations

Claim:

- Computing the Gärtner et. al kernel is no harder than solving

$$X = A' X A^\top + P$$

Sylvester Equations:

- The above equation is called a Sylvester equation
- Well studied in control theory
- Efficiently solvable in $O(n^3)$ time
- `dylap` method in Matlab

Before the proof . . .

vec operator:

$$B = \begin{bmatrix} 2 & 5 & 2 \\ 5 & 2 & 5 \\ 1 & 5 & 2 \end{bmatrix} \text{ and } \mathrm{vec}(B) = \begin{bmatrix} 2 \\ 5 \\ 1 \\ 5 \\ 2 \\ 5 \\ 2 \\ 5 \\ 2 \end{bmatrix}$$

Key Equation:

$$\mathrm{vec}(ABC) = (C^\top \otimes A)\,\mathrm{vec}(B)$$

The Proof

Rewrite the Sylvester equation as

$$\operatorname{vec}(X) = \operatorname{vec}(A' X A^\top) + \operatorname{vec}(P)$$

Apply key equation

$$\operatorname{vec}(X) = (A \otimes A') \operatorname{vec}(X) + \operatorname{vec}(P)$$

Rearrange

$$(\mathbf{I} - A \otimes A') \operatorname{vec}(X) = \operatorname{vec}(P)$$

or equivalently

$$\operatorname{vec}(X) = (\mathbf{I} - A \otimes A')^{-1} \operatorname{vec}(P)$$

Let $p_\times = \operatorname{vec}(P)$ and multiply both sides by $q_\times$

$$q_\times^\top \operatorname{vec}(X) = q_\times^\top (\mathbf{I} - A \otimes A')^{-1} p_\times = K(G, G')$$

Other Schemes

Basic Idea:

$$\underbrace{\mathrm{vec}(A' X A^\top)}_{O(n^3)} = \underbrace{(A \otimes A')\,\mathrm{vec}(X)}_{O(n^4)}$$

- Can exploit sparsity of A and A' to speed up things

Fixed Point Iteration:

- Solve for a fixed point (Kashima et. al):

$$(\mathbf{I} - A \otimes A')\,\mathrm{vec}(X_\infty) = \mathrm{vec}(X_\infty)$$

Conjugate Gradient:

- Fast matrix-vector multiplication to speed up CG solver
- Convergence depends on spectrum of A and A'

Relation to Diffusion Kernels

Laplacian of the Direct Product Graph:

- In general $L_\times \neq L_1 \otimes L_2$:(
- But there is a fix ...

Cartesian Product of Graphs:

$$
\begin{aligned}
V_\square =& \{(v, v') : v \in V, v' \in V'\} \\
E_\square =& \{((v, v'), (w, w')) : (v, w) \in E, (v', w') \in E'\}
\end{aligned}
$$

- For Cartesian products

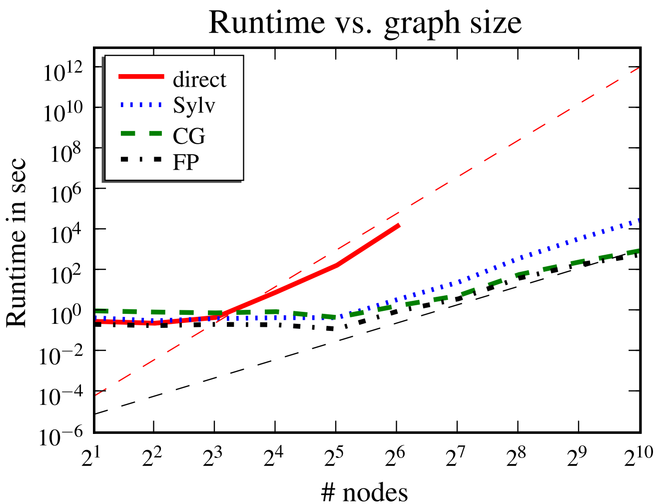
$$
\begin{aligned}
A_\square &= A_1 \oplus A_2 := A_1 \otimes I + I \otimes A_2 \\
L_\square &= L_1 \oplus L_2
\end{aligned}
$$

- All our efficient computation tricks apply!
- Is the kernel PSD?

Experiments

Scaling Behavior - I:

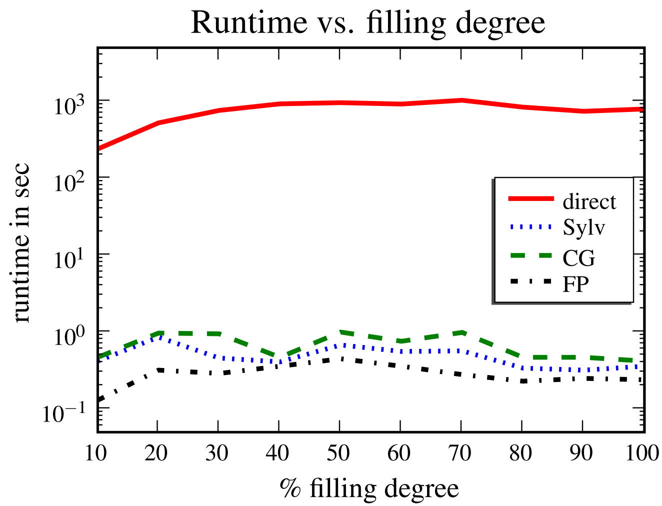
- Begin with empty graphs of size 2^k where $k = 1, \ldots 10$
- Randomly insert edges until
  - avg. degree at least 2 or
  - graph is full
- Generate 10 random graphs and compute kernel matrix



Experiments

Scaling Behavior - II:

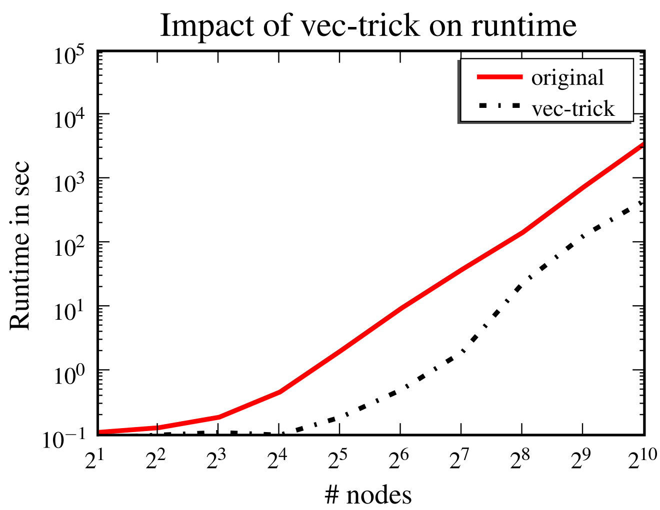
- Begin with empty graphs of size 32
- Randomly insert edges until
  - avg. fill-in of adjacency matrix is $10\% \ldots 100\%$ and
  - Graph is connected
- Generate 10 random graphs and compute kernel matrix



Experiments

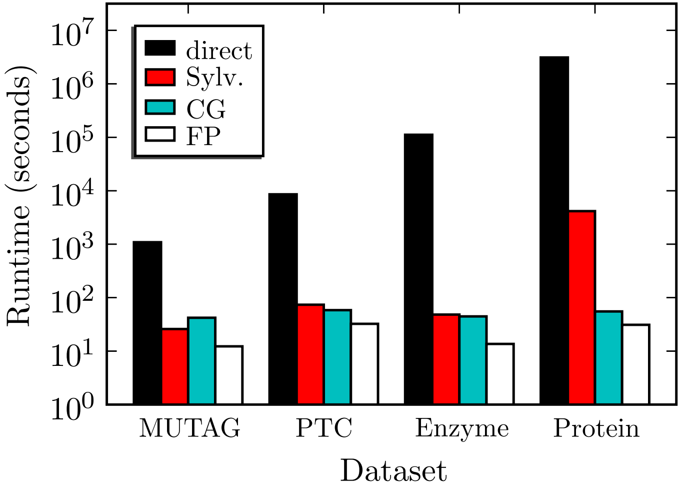
Impact of the vec-trick:

- Same graphs as the runtime vs nodes experiment
- Use the vec trick in the fixed point iteration
- Compare to original fixed point iteration



Experiments

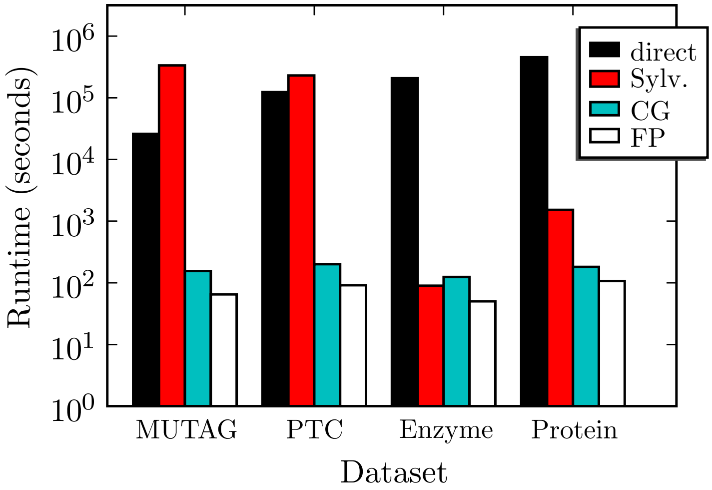
Unlabeled Graphs: We computed graph kernels on four datasets for molecular function prediction: MUTAG and PTC (chemical compounds), Enzyme and Protein (protein structures). We report runtimes for computing a 100×100 kernel matrix.



dataset	MUTAG	PTC	Enzyme	Protein
nodes/graph	17.7	26.7	32.6	38.6
edges/node	2.2	1.9	3.8	3.7
Direct	18'09"	142'53"	31h*	36d*
Sylvester	25.9"	73.8"	48.3"	69'15"
Conjugate	42.1"	58.4"	44.6"	55.3"
Fixed-Point	12.3"	32.4"	13.6"	31.1"

Experiments

Labeled Graphs: We repeated the above graph kernel computation, now using either a linear or delta kernel between node labels as well.



kernel	delta		linear	
dataset	MUTAG	PTC	Enzyme	Protein
Direct	7.2h	1.4d*	2.4d*	5.3d*
Sylvester	3.9d*	2.7d*	89.8"	25'24"
Conjugate	2'35"	3'20"	124.4"	3'01"
Fixed Point	1'05"	1'31"	50.1"	1'47"

What I did not talk about

- Random walks on other semirings e.g. $(\min, +)$
- Why $(\min, +)$ does not yield p.s.d kernels
- Differences between A, $\tilde{A}$, L, and $\tilde{L}$
- Kernels on vertices (yields marginal graph kernels of Kashima et. al)
- Extensions to trajectories of ARMA models (joint work with René Vidal and Alex Smola)
- General theory using Binet-Cauchy theorem (joint work with Alex Smola)
- Connections to Rational kernels of Cortes et. al
- Connections to R-Convolution kernels of Haussler

Overview of my Research

Structured Input:

- Strings
- Graphs
- ARMA models

Structured Output:

- Exponential families in feature space

Optimization for Machine Learning:

- Bundle methods
- subBFGS

Theory

- Fundamental limitations of kernels
- Rates of convergence of boosting algorithms

Conclusion

- First unifying view of
  - Diffusion kernels
  - Regularization on graphs
  - Geometric and random walk kernels
  - Marginal graph kernels
- Efficient computation by exploiting Kronecker products
- Papers at `http://www.stat.purdue.edu/~vishy`

Big Open Question

- Comparing paths in two different graphs is polynomial
- Subgraph isomorphism is known to be NP-hard
- Computing the so-called universal graph kernel which counts all common subgraphs of two graphs is harder than subgraph isomorphism
- When we compare any other subgraphs e.g.
  - simple paths (where vertices do not repeat)
  - cycles
  - trees

  we seem to lose polynomial run-time
- Are there other subgraphs for which efficient computation is possible?

References

Journal Papers

[1] S. V. N. Vishwanathan, Karsten Borgwardt, Nicol N. Schraudolph, and Imre Risi Kondor. On graph kernels. *J. Mach. Learn. Res.*, 2008. submitted.

[2] S. V. N. Vishwanathan, A. J. Smola, and R. Vidal. Binet-Cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. *International Journal of Computer Vision*, 73(1):95–119, 2007.

Conference Papers

[1] S. V. N. Vishwanathan, Karsten Borgwardt, and Nicol N. Schraudolph. Fast computation of graph kernels. Technical report, NICTA, 2006.

[2] S. V. N. Vishwanathan and A. J. Smola. Binet-Cauchy kernels. In L. K. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems 17*, pages 1441–1448, Cambridge, MA, 2005. MIT Press.

Applications to Bioinformatics

[1] Karsten M. Borgwardt, H.-P. Kriegel, S. V. N. Vishwanathan, and N. Schraudolph. Graph kernels for disease outcome prediction from protein-protein interaction networks. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany Murray, and Teri E Klein, editors, *Proceedings of the Pacific Symposium of Biocomputing 2007*, Maui Hawaii, January 2007. World Scientific.

[2] Karsten M. Borgwardt, S. V. N. Vishwanathan, and H.-P. Kriegel. Class prediction from time series gene expression profiles using dynamical systems kernels. In Russ B. Altman, A. Keith Dunker, Lawrence Hunter, Tiffany Murray, and Teri E Klein, editors, *Proceedings of the Pacific Symposium of Biocomputing 2006*, pages 547–558, Maui Hawaii, January 2006. World Scientific.

[3] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.P. Kriegel. Protein function prediction via graph kernels. In *Proceedings of Intelligent Systems in Molecular Biology (ISMB)*, Detroit, USA, 2005.