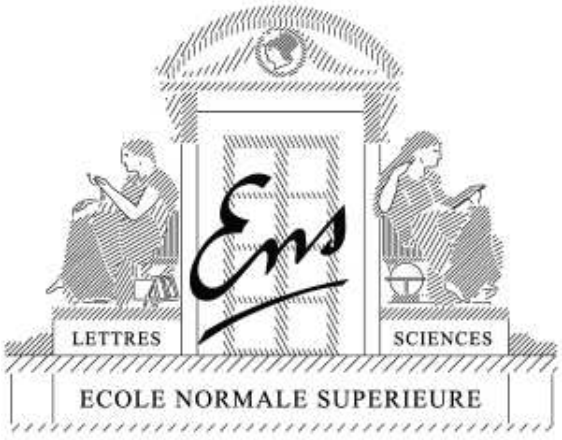
Graph Kernels between Point Clouds

Francis Bach

Willow project, INRIA - Ecole Normale Supérieure, Paris

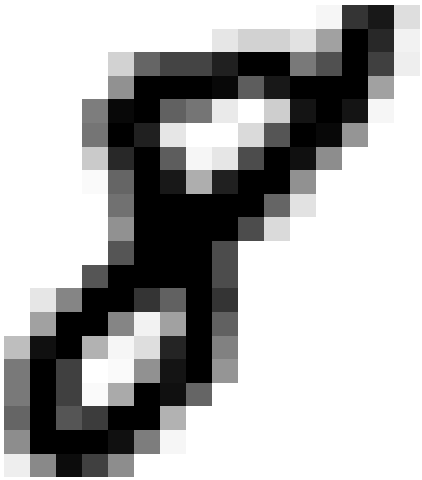


July 2008

(Attributed) point clouds

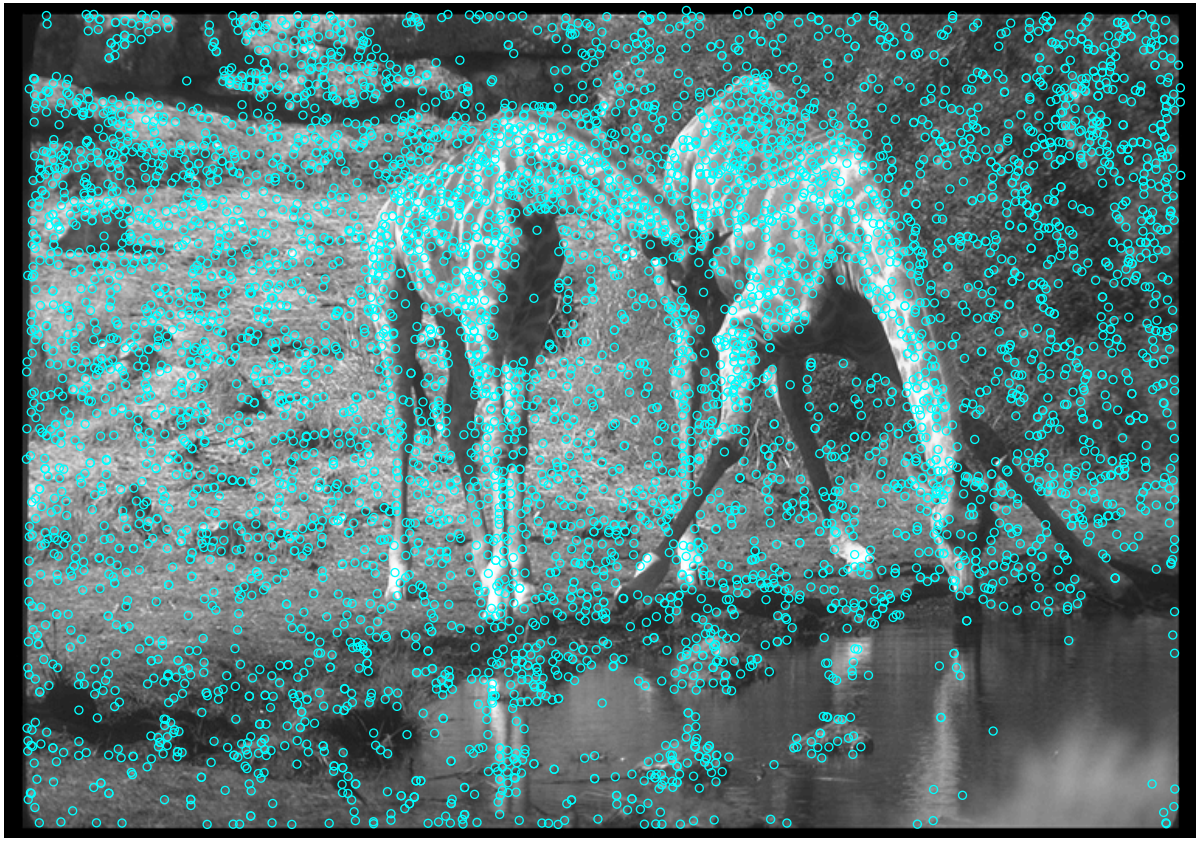
- Set of labelled points in $\mathbb{R}^2$ or $\mathbb{R}^3$ (and more generally $\mathbb{R}^d$)
  - **no *a priori* ordering of the points**
- Many applications:
  - Point clouds as samples from a distribution
    $\Rightarrow$ comparing distributions through samples
  - Computer vision
  - Bioinformatics

Point clouds for computer vision
Line drawings

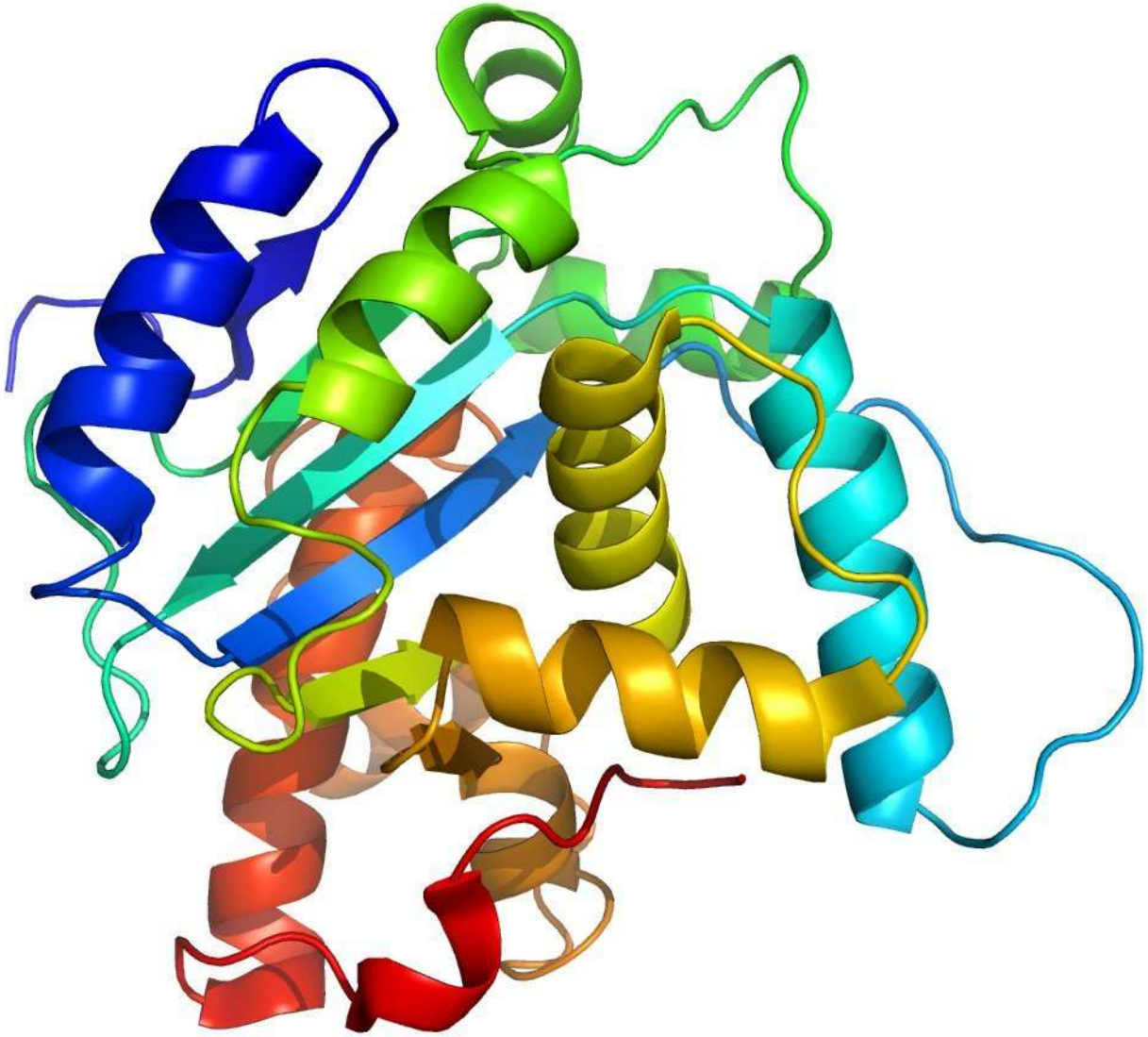


Point clouds for computer vision

Point clouds for computer vision
Corner detectors and SIFT (Lowe, 2004)



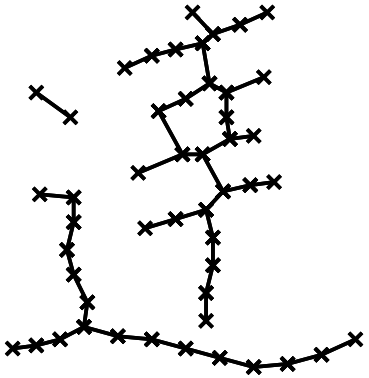
Protein 3D structures (Borgwardt et al., 2005)

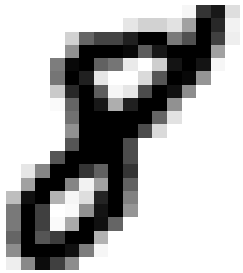


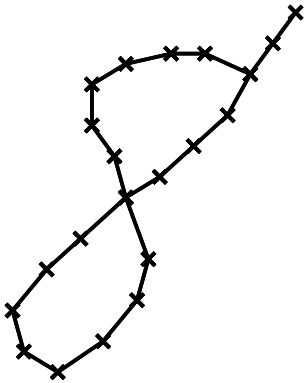
Kernels for point clouds

- Simplest approach: implicit independent and identically distributed assumption
  - define kernel on point clouds by using kernels on **moments** (i.e., only based on **averages of well chosen functions**)
  - Kondor & Jebara (2003), Cuturi & Vert (2005), Grauman & Darrell (2005), Hein & Bousquet (2005), etc...
  - Spatial geometry/invariances not taken into account
- Our main assumption: see point clouds as graphs
  - Needs to define neighborhoods (e.g., k-nearest neighbors)
  - Adapt **graph kernels**

Point clouds for computer vision
Line drawings and graphs







Kernels between point clouds (e.g., labelled undirected graphs)

- Two graphs $G = (V, E, a, x)$ and $H = (W, F, b, y)$, where :
  - V, W are **vertex sets**
  - E, F are **edge sets**
  - a, b, x, y are **vertex labelling functions**

- Two types of labels are considered:
  - attributes $a(v), b(w) \in \mathcal{A}$
  - positions $x(v), y(w) \in \mathcal{X}$

- Line drawings: $\mathcal{X} = \mathcal{A} = \mathbb{R}^2$

Graph kernels

- Two main classes of kernels
  1. Kernels based on existing similarity measures from the graph matching literature (Neuhaus & Bunke, 2006, Fröhlich et al., 2005)
     – Non positive definite, require *ad hoc* post-processing

Graph kernels

- Two main classes of kernels

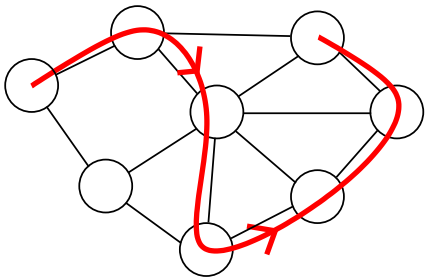
1. Kernels based on existing similarity measures from the graph matching literature (Neuhaus & Bunke, 2006, Fröhlich et al., 2005)
   - Non positive definite, require *ad hoc* post-processing

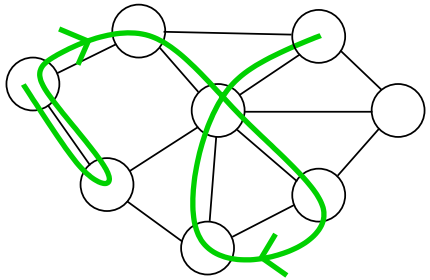
2. Compare substructures of the graphs
   - Random walk kernels sums over all possible walks (with **all lengths**) and can be computed by solving a sparse linear system (Kashima et al., 2004, Borgwardt et al., 2005, Vishwanathan et al., 2007)
   - **Fixed length** walks and tree-walks: dynamic programming recursions (Harchaoui & Bach, 2007)

Paths - walks

- Path: sequence of distinct neighboring vertices
- Walk: sequence of neighboring vertices
- β-walk: walk where any $\beta + 1$ consecutive vertices are distinct

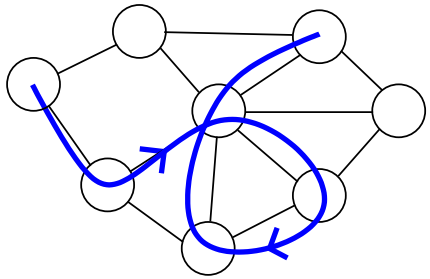
path

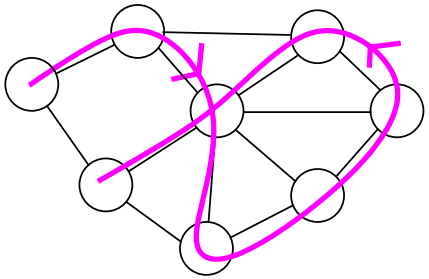




1-walk which is not a 2-walk

2-walk which is not a 3-walk





4-walk

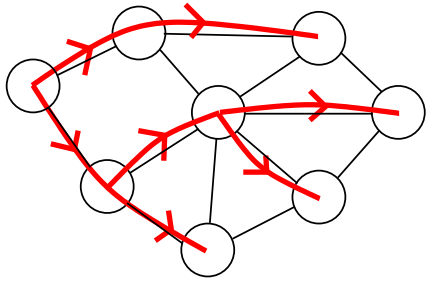
Tree-walks (Ramon and Gärtner, 2003)

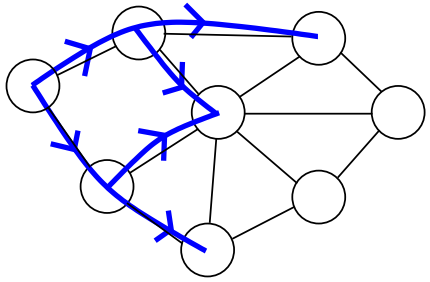
- Extension of subtrees and walks

- α-ary β-tree-walk of depth γ of G, defined by:

  1. **tree-structure**: rooted labelled α-ary tree of depth γ
  2. **consistent labelling**: nodes labelled by vertices in G, such that the labels of neighbors in the tree-walk must be neighbors in G
  3. **distinct labels/vertices**: for nodes up to β generations apart in T

binary 2-tree-walk





binary 1-tree-walk which is not a 2-tree-walk

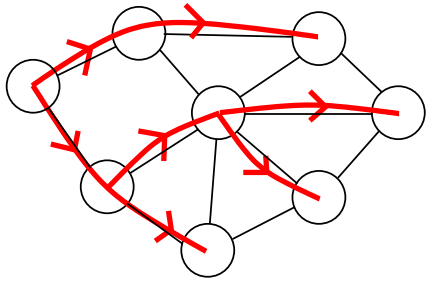
Tree-walks (Ramon and Gärtner, 2003)

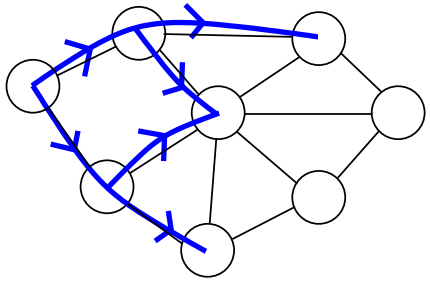
- Extension of subtrees and walks

- α-ary β-tree-walk of depth γ of G, defined by:

  1. **tree-structure**: rooted labelled α-ary tree of depth γ
  2. **consistent labelling**: nodes labelled by vertices in G, such that the labels of neighbors in the tree-walk must be neighbors in G
  3. **distinct labels/vertices**: for nodes up to β generations apart in T

binary 2-tree-walk





binary 1-tree-walk which is not a 2-tree-walk

- Notations: given tree structure $T \in \mathcal{T}_{\alpha,\gamma}$, $\mathcal{J}_\beta(T, G) =$ set of consistent labellings in $V^{|T|}$

Graph kernels

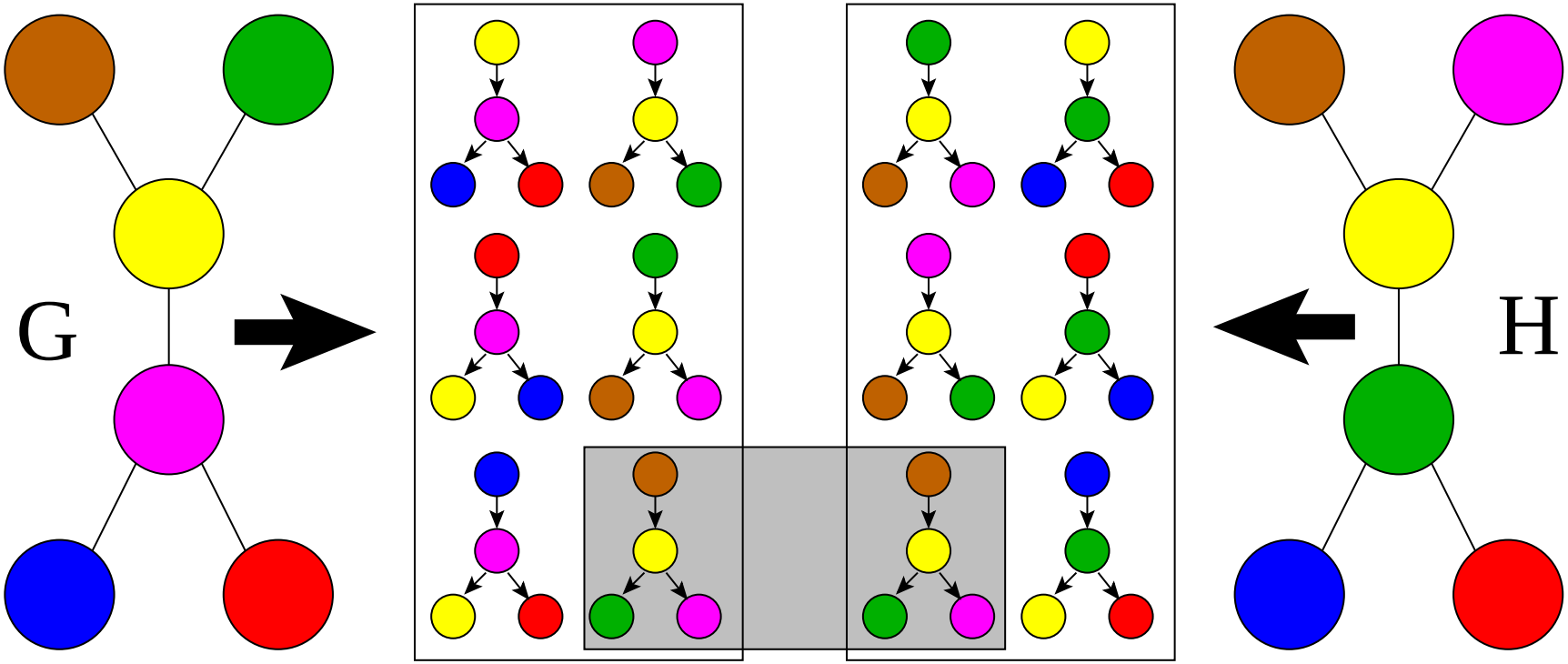
- Main idea: sum a **local kernel** $q_{T,I,J}(G,H)$ between matching walks or tree-walks (Ramon and Gärtner, 2003)

$$k^{\mathcal{T}}_{\alpha,\beta,\gamma}(G,H) = \sum_{T \in \mathcal{T}_{\alpha,\gamma}} f_{\lambda,\nu}(T) \sum_{I \in \mathcal{J}_\beta(T,G)} \sum_{J \in \mathcal{J}_\beta(T,H)} q_{T,I,J}(G,H)$$

- Tree-dependent penalization $f_{\lambda,\nu}(T) = \lambda^{|T|}\nu^{\ell(T)}$
  - downweight bushy graphs (Mahé & Vert, 2006)
- If the local kernel is positive definite, then the graph kernel is positive definite
- Main issue: choice of the local kernel

Graph kernels

- Example with Dirac local kernel



Local kernel on attributes and positions

- **Product of kernel between attributes and kernel between positions**

- For attributes, usual factorized form (independent of tree structure T)

$$q_{\mathcal{A}}(a(I), b(J)) = \prod_{p=1}^{|I|} k_{\mathcal{A}}(a(I_p), b(J_p))$$

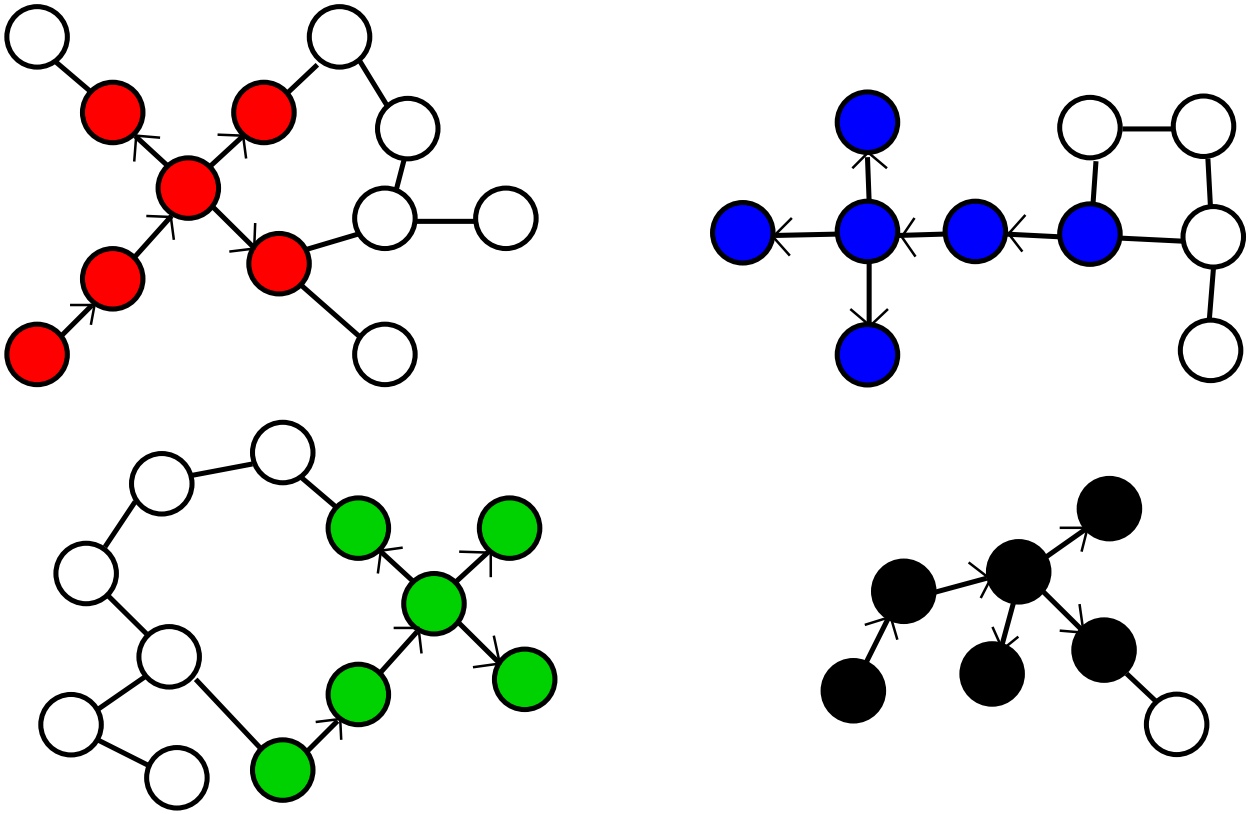
  where $k_{\mathcal{A}}$ is a positive definite kernel on $\mathcal{A} \times \mathcal{A}$

- Fully factorized form allows:
  - dynamic programming recursions (Harchaoui & Bach, 2007)
  - linear system if $\gamma \to \infty$ (Kashima et al., 2004, etc...)

- For positions, cannot be factorized if we want translation invariance

Local kernels and invariances

- Which subgraphs should be similar in terms of positions?



Translation/rotation invariance kernels for positions $x(I) \in \mathcal{X}^{|I|}$ and $y(J) \in \mathcal{X}^{|J|}$

- **Kernel must depend on all $x(I) \in \mathcal{X}^{|I|}$ and $y(J) \in \mathcal{X}^{|J|}$, and not simply as a product of functions on the pairs $(x(I_k), y(J_k))$**
  - **Translation invariance** : only depend on **pairwise differences**
    $$x(i) - x(i') \text{ and } y(j) - y(j')$$
  - **Rotation invariance** : further reduce these to **pairwise distances**
    $$\|x(i) - x(i')\| \text{ and } \|y(j) - y(j')\|$$

Translation/rotation invariance kernels for positions $x(I) \in \mathcal{X}^{|I|}$ and $y(J) \in \mathcal{X}^{|J|}$

- **Kernel must depend on all $x(I) \in \mathcal{X}^{|I|}$ and $y(J) \in \mathcal{X}^{|J|}$, and not simply as a product of functions on the pairs $(x(I_k), y(J_k))$**
  - **Translation invariance** $\Leftrightarrow$ only depend on **pairwise differences** $x(i) - x(i')$ and $y(j) - y(j')$
  - **Rotation invariance** $\Rightarrow$ further reduce these to **pairwise distances** $\|x(i) - x(i')\|$ and $\|y(j) - y(j')\|$
  - Reduce to kernel matrices corresponding to a translation/rotation invariant positive definite kernel $k_{\mathcal{X}}(x_1 - x_2)(= e^{-\upsilon\|x_1 - x_2\|^2})$
- All positions reduced to full kernel matrices $K \in \mathbb{R}^{|V|\times|V|}$ and $L \in \mathbb{R}^{|W|\times|W|}$ for each graph.
  - Local kernel is a kernel between kernel submatrices $K_{I,I} = K_I$ and $L_{J,J} = L_J$

Local kernel on kernel matrices

- Data reduced to positive submatrices $K_I = K_{I,I}$ and $L_J = L_{J,J}$
- Kernels on positive matrices:
  - Bhattacharyya kernel $k_{\mathcal{B}}$ (Kondor & Jebara, 2003), defined as

$$k_{\mathcal{B}}(K, L) = |K|^{1/2} |L|^{1/2} \left| \frac{K + L}{2} \right|^{-1},$$

    where $|K|$ denotes the determinant of K.
  - $k_{\mathcal{B}}(K_I, L_J)$ does not yet depend on the tree structure T
- Recursion may be efficient only if it can be computed recursively
- How to use the graph to compute the local kernel recursively?
  -

Local kernel on kernel matrices

- Data reduced to positive submatrices $K_I = K_{I,I}$ and $L_J = L_{J,J}$
- Kernels on positive matrices:
  - Bhattacharyya kernel $k_{\mathcal{B}}$ (Kondor & Jebara, 2003), defined as

$$k_{\mathcal{B}}(K, L) = |K|^{1/2} |L|^{1/2} \left| \frac{K + L}{2} \right|^{-1},$$

    where $|K|$ denotes the determinant of K.
  - $k_{\mathcal{B}}(K_I, L_J)$ does not yet depend on the tree structure T
- Recursion may be efficient only if it can be computed recursively
- How to use the graph to compute the local kernel recursively?
  - **graph + positive matrices = graphical models**

Positive matrices and graphical models

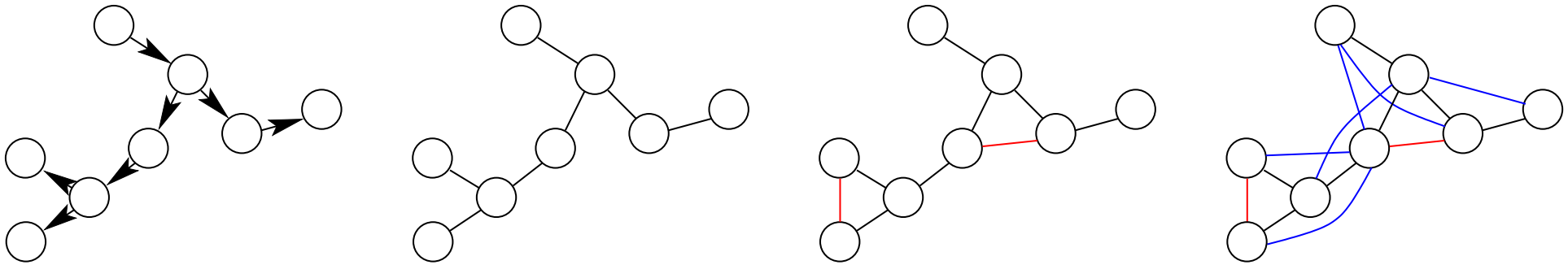
- Symmetric positive definite matrices $=$ covariance matrices
- Probabilistic graphical models for associated Gaussian random vectors
- **Main idea**
  - design factorized approximations of covariance matrices
  - replace determinants by the determinant of the projection of the covariance matrix onto the graphical model
- Decomposable graph Q with cliques $\mathcal{C}(Q)$ and separators $\mathcal{S}(Q)$:

$$\log|\Pi_Q(K)| = \sum_{C\in\mathcal{C}(Q)} \log|K_C| - \sum_{S\in\mathcal{S}(Q)} \log|K_S|$$

  - See paper/poster for more details

Choice of graphical model

- Connect all points to its β-descendents (i.e., all descendants up to the β-th generation)
  - **Densest possible graph**

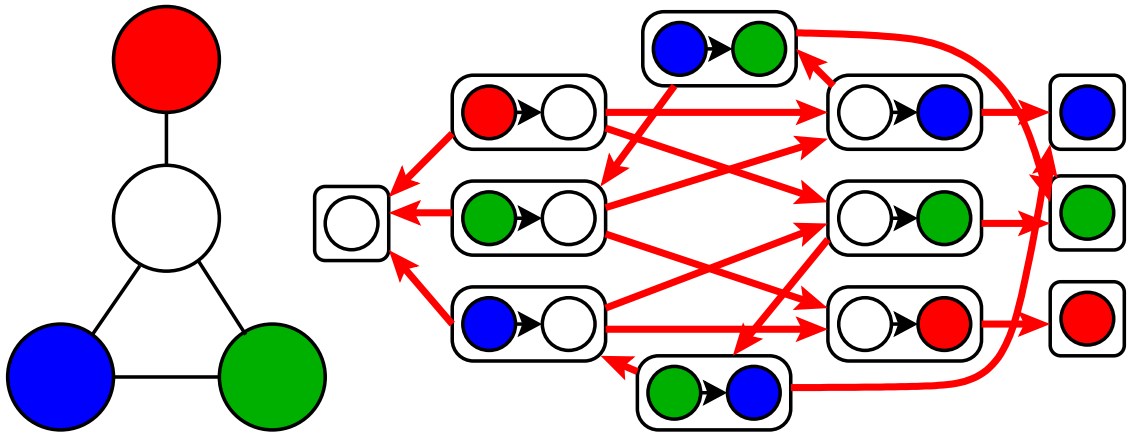


- Approximate Bhattacharyya kernel by the Bhattacharyya kernel between factorized approximation
  - More subtleties (see paper)

Dynamic programming recursions

- Main idea: α-ary β-tree-walks of G can essentially be defined through 1-tree-walks on the augmented graph $G_{\alpha,\beta}$ of all rooted subtrees of G of depth at most β and arity less than α (Mahé & Vert, 2006)

- Example



graph $G_{1,2}$

Recursions

- Recursions for sums restricted to tree-walks with given starting trees:

$$k^{\mathcal{T}}_{\alpha,\beta,\gamma}(G,H,R_0,S_0) = k^{\mathcal{T}}_{\alpha,\beta,\gamma-1}(G,H,R_0,S_0) +$$

$$\sum_{p=1}^{\alpha} \sum_{\substack{R_1,\ldots,R_p \in \mathcal{N}_{G_{\alpha,\beta}}(R_0) \\ R_1,\ldots,R_p \text{ disjoint}}} \sum_{\substack{S_1,\ldots,S_p \in \mathcal{N}_{H_{\alpha,\beta}}(S_0) \\ S_1,\ldots,S_p \text{ disjoint}}}$$

$$\left[\lambda \prod_{i=1}^{p} k_{\mathcal{A}}(a(\mathrm{root}(R_i)), b(\mathrm{root}(S_i))) \times \frac{k_{\mathcal{B}}^{\cup_{i=1}^{p} R_i | R_0, \cup_{i=1}^{p} S_i | S_0}(K,L)}{\prod_{i=1}^{p} k_{\mathcal{B}}^{R_i,S_i}(K,L)} \left(\prod_{i=1}^{p} k^{\mathcal{T}}_{\alpha,\beta,\gamma-1}(G,H,R_i,S_i)\right)\right].$$

Recursions

$$k^{\mathcal{T}}_{\alpha,\beta,\gamma}(G, H, R_0, S_0) = k^{\mathcal{T}}_{\alpha,\beta,\gamma-1}(G, H, R_0, S_0) +$$

$$\sum_{p=1}^{\alpha} \sum_{\substack{R_1,\ldots,R_p \in \mathcal{N}_{G_{\alpha,\beta}}(R_0) \\ R_1,\ldots,R_p \text{ disjoint}}} \sum_{\substack{S_1,\ldots,S_p \in \mathcal{N}_{H_{\alpha,\beta}}(S_0) \\ S_1,\ldots,S_p \text{ disjoint}}}$$

$$\left[\lambda \prod_{i=1}^{p} k_{\mathcal{A}}(a(\mathrm{root}(R_i)), b(\mathrm{root}(S_i))) \times \frac{k_{\mathcal{B}}^{\cup_{i=1}^{p} R_i | R_0, \cup_{i=1}^{p} S_i | S_0}(K, L)}{\prod_{i=1}^{p} k_{\mathcal{B}}^{R_i, S_i}(K, L)} \left(\prod_{i=1}^{p} k^{\mathcal{T}}_{\alpha,\beta,\gamma-1}(G, H, R_i, S_i)\right)\right].$$

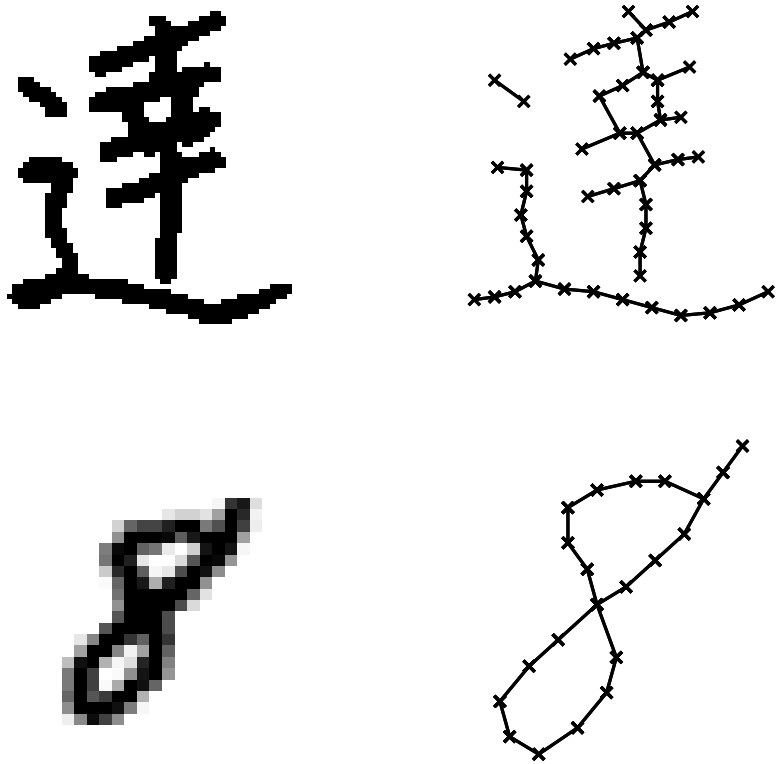
- **Computational complexity**: linear in γ (tree depth), quadratic in the size of $G_{\alpha,\beta}$, exponential in β (order) and α (arity)
  - average cardinality of $V_{\alpha,\beta}$ and time for one kernel evaluation (MNIST, average original graph size $= 18$, $\gamma = 24$):

    $\alpha{=}1,\ \beta{=}2:\ |V_{\alpha,\beta}|{=}36,\ T{=}2$ ms $\alpha{=}2,\ \beta{=}2:\ |V_{\alpha,\beta}|{=}56,\ T{=}25$ ms

    $\alpha{=}1,\ \beta{=}4:\ |V_{\alpha,\beta}|{=}37,\ T{=}3$ ms $\alpha{=}2,\ \beta{=}4:\ |V_{\alpha,\beta}|{=}70,\ T{=}32$ ms

Simulations
Line drawings and graphs

- Handwritten digits: MNIST, 100 examples per class
- Chinese characters: ETL9B, 50 examples per class (5 hardest)



Simulations

- (problem dependent) kernels: $k_{\mathcal{X}}(x,y) = \exp(-\tau\|x-y\|^2)+\kappa\delta(x,y)$
  $k_{\mathcal{A}}(x,y) = \exp(-\upsilon\|x-y\|^2)$

- Free parameters:
  - arity of tree-walks ($\alpha = 1, 2$)
  - order of tree-walks ($\beta = 1, 2, 4, 6$)
  - depth of tree-walks ($\gamma = 1, \ldots, 24$)
  - penalization on number of nodes ($\lambda = 1$)
  - penalization on number of leaf nodes ($\nu = .1, .01$)
  - bandwidth for kernel on positions ($\tau = .05, .01, .1$)
  - ridge parameter ($\kappa = .001$)
  - bandwidth for kernel on attributes ($\upsilon = .05, .01, .1$)

- Selection by 5-fold cross-validation

Results (misclassification rates)

- Competing methods
  - Gaussian-RBF kernel on the vectorized original images
  - random walk kernel which sums over all walk lengths
  - pyramid match kernel (Grauman & Darrell, 2007)

- Two loops of 5-fold cross-validation for parameter selection

	MNIST $\alpha = 1$	MNIST $\alpha = 2$	ETL9B $\alpha = 1$	ETL9B $\alpha = 2$
$\beta = 1$	11.6 ± 4.6	9.2 ± 3.9	36.8 ± 4.6	32 ± 8.4
$\beta = 2$	5.6 ± 3.1	5.6 ± 3.0	29.2 ± 8.8	$\mathbf{25.2 \pm 2.7}$
$\beta = 4$	$\mathbf{5.4 \pm 3.6}$	$\mathbf{5.4 \pm 3.1}$	32.4 ± 3.9	29.6 ± 4.3
$\beta = 6$	5.6 ± 3.3	6 ± 3.5	29.6 ± 4.6	28.4 ± 4.3
RBF	11.6 ± 5.4		50.4 ± 6.2	
Random walk	8.6 ± 1.3		34.8 ± 8.4	
Pyramid match	10.8 ± 3.6		45.2 ± 3.4	

Conclusion

- Kernels between attributed point clouds
  - Point clouds as graphs
  - Compare local structure with factorized local kernel
  - Link between kernels and graphical models
- Current work
  - Application to proteins
  - Approximation properties (see, e.g., Caetano et al., 2006)