UNIVERSITÉ DE FRIBOURG
UNIVERSITÄT FREIBURG

MASTER IN COMPUTER SCIENCE

Pattern Recognition

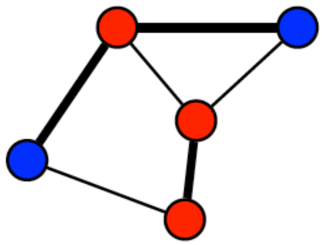
Lecture 11 : Kernel Functions

Dr. Andreas Fischer

andreas.fischer@unifr.ch

Statistical vs Structural Representation

$(x_1, x_2, \ldots, x_n)$



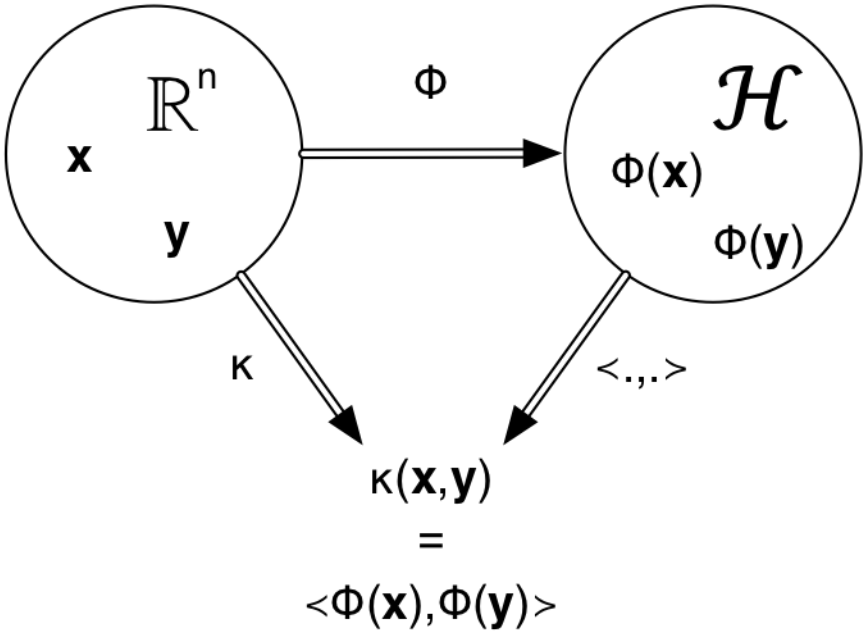
+ rich mathematical structure
+ many efficient algorithms
- low representational power

+ high representational power
- high computational complexity
- lack of efficient algorithms

Kernel Trick (Repetition)

- Avoid an explicit, possibly costly mapping into the new feature space.
- Instead, calculate only the dot product:

$$\kappa(x,y) = \langle \varphi(x), \varphi(y) \rangle$$



Kernelization (Repetition)

- Theorem: For every *valid* kernel function such a feature space exists.
- Replace standard dot product with any valid kernel to solve the problem implicitly in a different feature space.
- Algorithms that can be expressed in terms of dot products only are called *kernelizable*.
- KNN is kernelizable:

$$\|\varphi(x)-\varphi(y)\|^2 = \langle \varphi(x)-\varphi(y), \varphi(x)-\varphi(y) \rangle$$

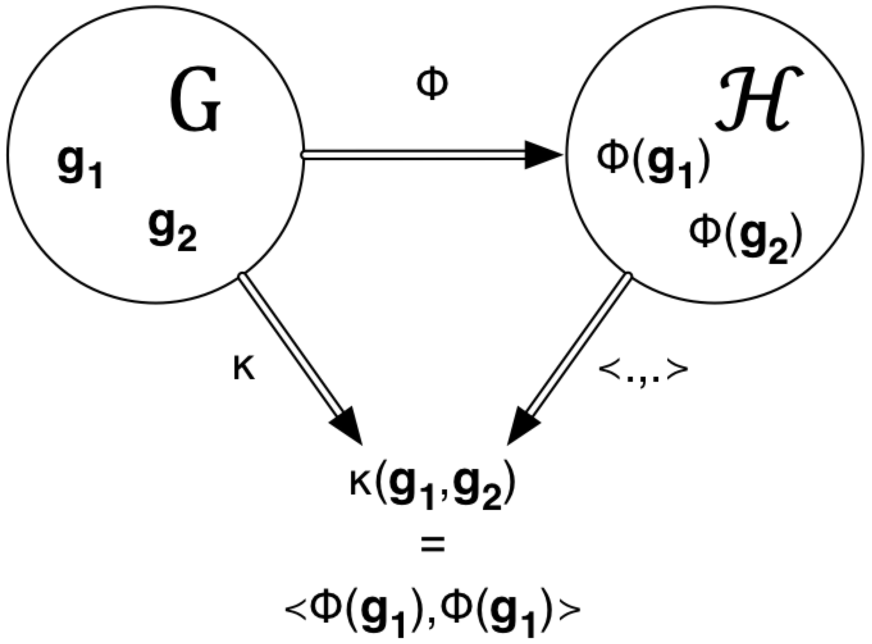
$$= \langle \varphi(x), \varphi(x) \rangle + \langle \varphi(y), \varphi(y) \rangle - 2\langle \varphi(x), \varphi(y) \rangle$$

$$= \kappa(x,x) + \kappa(y,y) - 2\kappa(x,y)$$

Bridging the Gap

- Kernel functions are promising means to bridge the gap between statistical and structural pattern recognition.
- They can be used to make the rich repository of (kernelizable) statistical methods applicable to structural representations.
- The goal is to encode structural matching properties in the kernel. Note that in general, information is lost when doing a vector space embedding ϕ.

$$\kappa(g_1, g_2) = \langle \varphi(g_1), \varphi(g_2) \rangle$$



String Kernels

P-Spectrum Kernel

- Let
  - $x = x_1 \ldots x_N$ and $y = y_1 \ldots y_M$ be strings over alphabet Σ
  - $\{u_1,\ldots,u_n\} = \Sigma^p$ all words of length p
- The *p-spectrum* of string x is the occurrence histogram of substrings with length p:

$$\varphi(x) = (\varphi_{u_1}(x), \ldots, \varphi_{u_n}(x))$$

$$\varphi_{u_i}(x) = \left| \{(v_1, v_2) \mid v_1, v_2 \in \Sigma^*; x = v_1 u_i v_2\} \right|$$

- The *p-spectrum kernel* of x and y is defined as:

$$\kappa(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

- The p-specturm kernel can be computed without explicitly embedding the strings into the vector space N^n.

Example

- 2-spectrum of “bar”, “bat”, “car”, and “cat”

φ	ar	at	ba	ca
bar	1	0	1	0
bat	0	1	1	0
car	1	0	0	1
cat	0	1	0	1

- 2-spectrum kernel

κ	bar	bat	car	cat
bar	2	1	1	0
bat	1	2	0	1
car	1	0	2	1
cat	0	1	1	2

Example

- Consider the words:
  - x = statistics
  - y = computation
- For p = 3 we obtain the substrings:
  - sta, tat, ati, tis, ist, sti, tic, ics, com, omp, mpu, put, uta, tat, ati, tio, ion
- Common substrings with length 3 are “tat” and “ati”. Therefore:

$$\kappa(x, y) = 2$$

All-Subsequence Kernel

- Let
  - $x = x_1 \ldots x_N$ and $y = y_1 \ldots y_M$ be strings over alphabet Σ
  - $\{u_1, \ldots, u_i, \ldots\} = \Sigma^*$
  - $\phi_{u_i}(x)$ the number of occurrences of u_i as a subsequence in x
- Mapping:

$$\varphi(x) = (\varphi_{u_1}(x), \ldots, \varphi_{u_i}(x), \ldots) \in R^\infty$$

- The *all-subsequence kernel* of x and y is:

$$\kappa(x, y) = \langle \varphi(x), \varphi(y) \rangle$$

- For efficient computation, we can focus on subsequences with length min(|x|,|y|) or smaller.
- Normalization that accounts for different lengths of the strings:

$$\hat{\kappa}(x, y) = \frac{\kappa(x, y)}{\sqrt{\kappa(x, x)\kappa(y, y)}}$$

Example

φ	ϵ	a	b	c	r	t	aa	ar	at	ba	br
bar	1	1	1	0	1	0	0	1	0	1	1
baa	1	2	1	0	0	0	1	0	0	2	0
car	1	1	0	1	1	0	0	1	0	0	0
cat	1	1	0	1	0	1	0	0	1	0	0

φ	ca	cr	ct	bar	baa	car	cat
bar	0	0	0	1	0	0	0
baa	0	0	0	0	1	0	0
car	1	1	0	0	0	1	0
cat	1	0	1	0	0	0	1

κ	bar	baa	car	cat
bar	8	6	4	2
baa	6	12	3	3
car	4	3	8	4
cat	2	3	4	8

$\hat{\kappa}$	bar	baa	car	cat
bar	1	$\frac{6}{\sqrt{8*12}}$	$\frac{4}{\sqrt{8*8}}$	$\frac{2}{\sqrt{8*8}}$
baa	$\frac{6}{\sqrt{8*12}}$	1	$\frac{3}{\sqrt{8*12}}$	$\frac{3}{\sqrt{8*12}}$
car	$\frac{4}{\sqrt{8*8}}$	$\frac{3}{\sqrt{8*12}}$	1	$\frac{4}{\sqrt{8*8}}$
cat	$\frac{2}{\sqrt{8*8}}$	$\frac{3}{\sqrt{8*12}}$	$\frac{4}{\sqrt{8*8}}$	1

Graph Kernels

Common Label Kernel

- Let
  - g = (V, E, α, β) and g’ = (V’, E’, α’, β’) two graphs
  - $L_n = \{a_1,\ldots,a_k\}$ the node label alphabet
  - $label(a_i,g)$ the number of occurrences of a_i in g
- Mapping:

$$\varphi(g) = \left(\frac{label(a_1,g)}{|V|},\ldots,\frac{label(a_k,g)}{|V|}\right)$$

- Kernel:

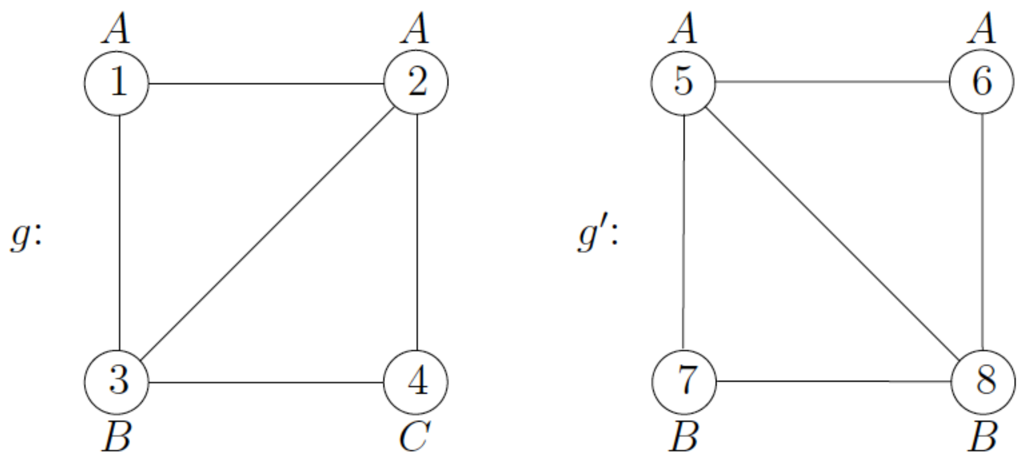
$$\kappa(g,g') = \langle\varphi(g),\varphi(g')\rangle$$

$$= \left(\frac{label(a_1,g)}{|V|},\ldots,\frac{label(a_k,g)}{|V|}\right)\left(\frac{label(a_1,g')}{|V'|},\ldots,\frac{label(a_k,g')}{|V'|}\right)'$$

$$\delta(v,v') = \begin{cases} 1 \text{ if } \alpha(v) = \alpha'(v') \\ 0 \text{ otherwise} \end{cases}$$

$$= \frac{1}{|V||V'|}\sum_{v\in V}\sum_{v'\in V'}\delta(v,v')$$

Example



$$\varphi(g) = \tfrac{1}{4}(2,1,1),\ \varphi(g') = \tfrac{1}{4}(2,2,0),\ \kappa(g,g') = \tfrac{1}{16}(4+2) = \frac{6}{16}$$

Common Subgraph Kernel

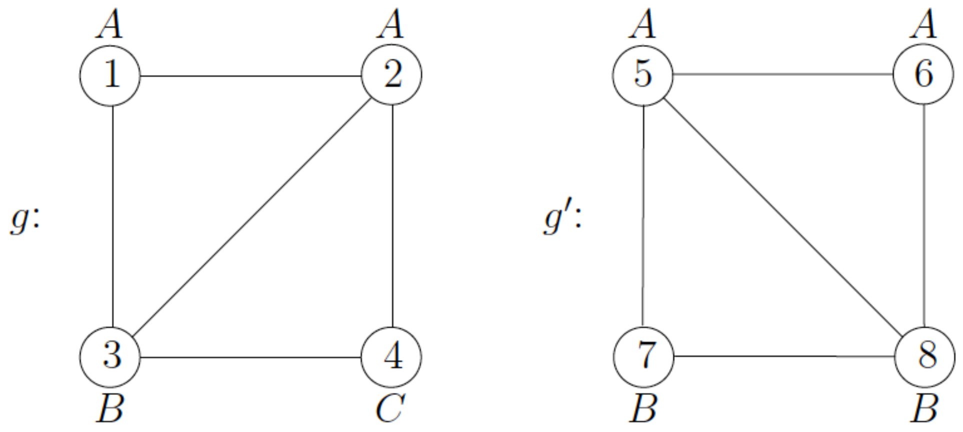
- The common label kernel does not capture the structural properties of the graphs. A more powerful kernel can be derived with respect to a set of reference graphs H = {h_1,…,h_n}.
- Let sg(h_i,g) be the number of occurrences of subgraph h_i in g.
- Mapping:

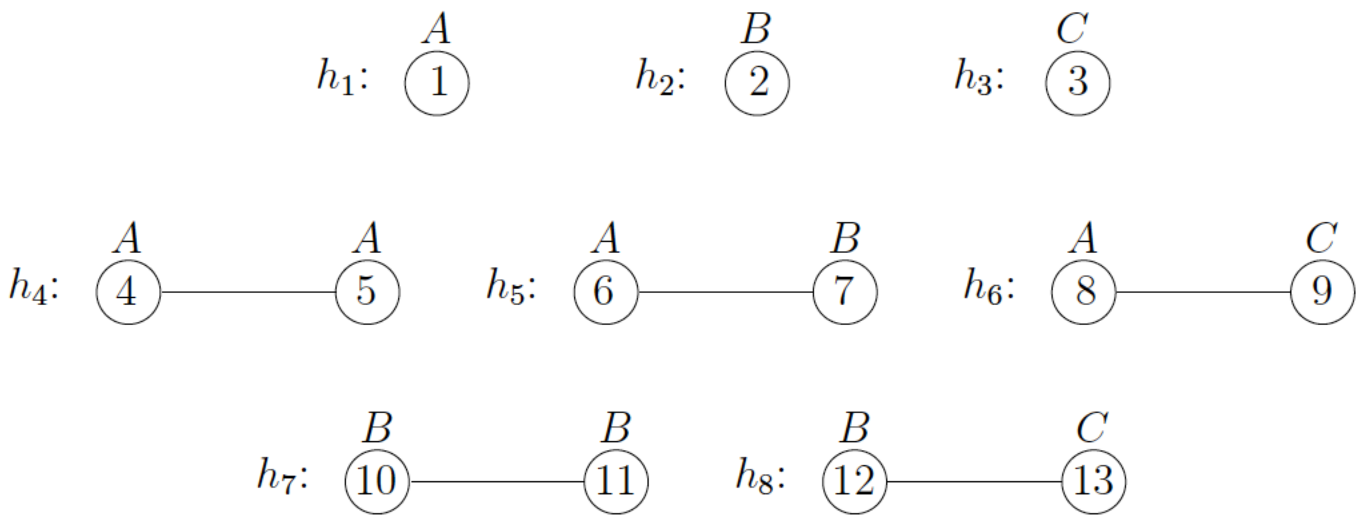
$$\varphi(g) = (sg(h_1, g), \ldots, sg(h_n, g))$$

- Kernel:

$$\kappa(g, g') = \langle \varphi(g), \varphi(g') \rangle$$

Example





$$\begin{aligned}\varphi(g) &= (2,1,1,2,2,1,0,1)\\ \varphi(g') &= (2,2,0,2,3,0,2,0)\\ \kappa(g,g') &= 4+2+0+4+6+0+0+0=16\end{aligned}$$

Common Walk Kernel

- A *walk* in graph g = (V, E, α, β) is a sequence of nodes $(v_1,\dots,v_n)$ with
  - $v_i \in V$
  - $(v_i,v_{i+1}) \in E$
  - length (n-1); like that a single node is a walk of length 0
  - the label sequence $a_1b_1a_2b_2\dots a_{n-1}b_{n-1}a_n$; $\alpha(v_i) = a_i$ and $\beta(v_i,v_{i+1}) = b_i$
  - $\lambda_0,\dots, \lambda_i,\dots \geq 0$ weights assigned to walks of length i
  - $S = \{l_1,\dots,l_i,\dots\}$ the set of all label sequences; infinite for cycles
- Mapping: $$\varphi(g) = (\varphi_{l_1}(g), \varphi_{l_2}(g), \dots, \varphi_{l_i}(g), \dots)$$
$$\varphi_{l_i}(g) = \sqrt{\lambda_m} N(l_i, g)$$
  - $N(l_i,g)$ is the number of walks with length m in graph g that have the label sequence l_i
- Kernel:
$$\kappa(g, g') = \langle \varphi(g), \varphi(g') \rangle$$

Product Graph

- Because of possible cycles, neither the mapping nor the kernel can be computed directly. Instead, walks in the *product graph* can be analyzed.
- The product of two graphs $g_1 = (V_1, E_1, \alpha_1, \beta_1)$ and $g_2 = (V_2, E_2, \alpha_2, \beta_2)$ is a graph $g_x = (V_x, E_x, \alpha_x, \beta_x)$ with:

$$V_x = \{(x_1, x_2) | x_1 \in V_1, x_2 \in V_2, \alpha_1(x_1) = \alpha_2(x_2)\}$$

$$E_x = \{(e_1, e_2) | e_1 \in E_1, e_2 \in E_2, \beta_1(e_1) = \beta_2(e_2)\}$$

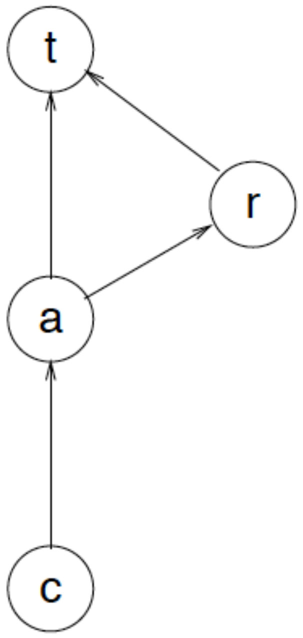
$$\alpha_x(x_1, x_2) = \alpha_1(x_1)$$

$$\beta_x(e_1, e_2) = \beta_1(e_1)$$

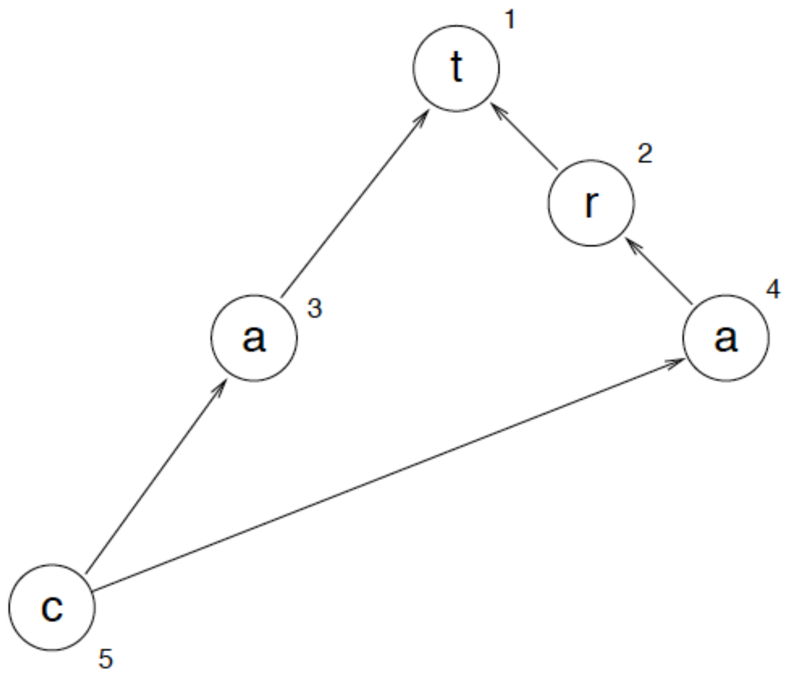
- Product graph g_x contains n nodes with label a ⇔ graph g_1 contains n_1 nodes with label a and graph g_2 contains n_2 nodes; $n = n_1 n_2$
- Product graph g_x contains an edge with label b from (x_1,x_2) to (y_1,y_2) ⇔ graph g_1 contains an edge with label b from x_1 to y_1 and graph g_2 contains an edge with label b from x_2 to y_2; and the node labels match

Example

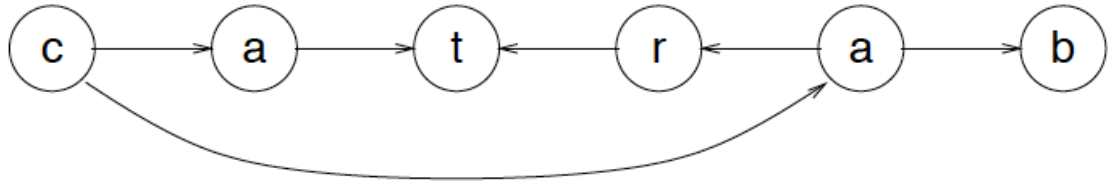
g:



g_x:



g':



Common Walk Kernel

- Idea: A walk with label sequence $a_1b_1\dots a_{n-1}b_{n-1}a_n$ exists in the product graph ⇔ a walk with the same label sequence exists in g and in g'.
- Kernel:

$$\kappa(g_1, g_2) = \sum_{i,j=1}^{|V_x|} \left[\sum_{n=0}^{\infty} \lambda_n \mathcal{E}_x^n \right]_{ij}$$

  - ε_x is the adjacency matrix of the product graph
  - λ_n is the weight for walks with length n
- For any adjacency matrix ε, multiplying it n times with itself ε^n indicates the number of paths between two nodes.
  - If $\varepsilon^n(i,j) = m$ there are m different paths of length n between x_i and x_j

Example

- g and g' have
  - ε_x^0 = I: 5 common walks of length 0 (walks with one node)
  - ε_x^1: 5 common walks of length 1
  - ε_x^2: 3 common walks of length 2
  - ε_x^3: 1 common walk of length 3
  - ε_x^4 = 0: no common walks of length 4 or more

$$\mathcal{E}_x = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix}$$

$$\mathcal{E}_x^3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{E}_x^2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 \end{bmatrix}$$

Convergence

- The common walk kernel includes an infinite sum with γ ≥ 0

$$\lim_{n\to\infty}\sum_{i=0}^{n}\gamma^i\mathcal{E}^i = (\mathbf{I}-\gamma\mathcal{E})^{-1}$$

- Its *limes* exists if γ < 1 / a with
  - a = min{Δ^+(g), Δ^-(g)}
  - Δ^+(g) is the maximum *outdegree* of g (maximum number of outgoing edges)
  - Δ^-(g) is the maximum *indegree* of g (maximum number of ingoing edges)

Graph Edit Distance Kernel

- Let
  - P = {p_1,…,p_m} be a set of prototype graphs
  - d(g,g') the (exact or approximate) graph edit distance
- Mapping, known as *dissimilarity space embedding*:

$$\varphi(g) = (d(g, p_1), \ldots, d(g, p_m))$$

- Kernel:

$$\kappa(g, g') = \langle \varphi(g), \varphi(g') \rangle$$

- Similar to the common subgraph kernel but instead of measuring the number of common subgraphs it measures the dissimilarity to the prototype graphs.

Euclidean Distance

- The Euclidean distance in the vector space after embedding:

$$\begin{aligned}
||\varphi(g) - \varphi(g')||^2 &= \langle \varphi(g), \varphi(g) \rangle + \langle \varphi(g'), \varphi(g') \rangle - 2\langle \varphi(g), \varphi(g') \rangle \\
&= \sum_{i=1}^{m} d(g, p_i)^2 + \sum_{i=1}^{m} d(g', p_i)^2 - 2\sum_{i=1}^{m} d(g, p_i) d(g', p_i) \\
&= \sum_{i=1}^{m} (d(g, p_i) - d(g', p_i))^2
\end{aligned}$$

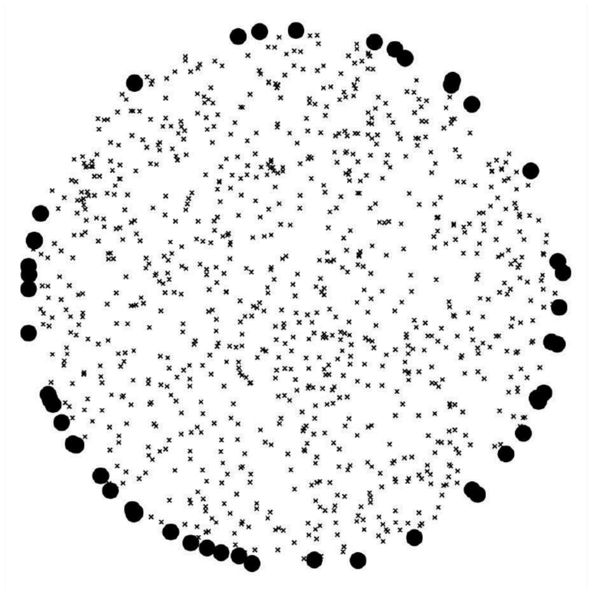
- It corresponds with the quadratic differences of the two edit distances $d(g,p_i)$ and $d(g',p_i)$.
- If g and g' are similar, the differences of the edit distances to the prototpyes are small, and therefore also the Euclidean distance between the embedded graphs ϕ(g) and ϕ(g') in the vector space.
- For this kernel, the mapping is computed explicitly. It has the advantage that not only kernelizable algorithms can be applied but instead the full repository of statistical pattern recognition methods.

Border Prototype Selection

- Prototype selection aims to find diverse reference graphs in the training set T. It can be done randomly or using dedicated selection methods.
- All selection methods can be performed *class-wise*, that is for each class separately, or *class-independently*.
- Border prototype selection iteratively selects graphs that are most dissimilar to all other graphs in T.

$$P_i = \begin{cases} \emptyset, & i = 0 \\ P_{i-1} \cup \{\text{marginal}(T \setminus P_{i-1})\}, & 0 < i \leq n \end{cases}$$

$$\text{marginal}(G) = \underset{g_1 \in G}{\text{argmax}} \sum_{g_2 \in G} d(g_1, g_2)$$

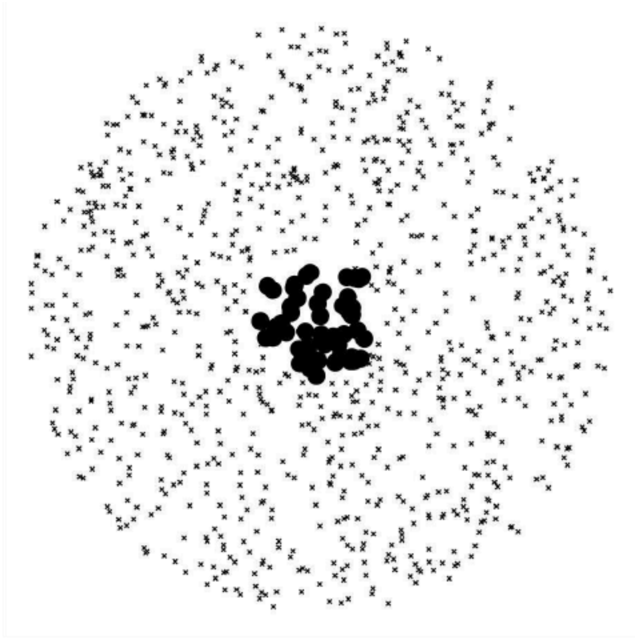


Center Prototype Selection

- Select median graphs that minimize the edit distance to all other graphs of the training set T.

$$P_i = \begin{cases} \emptyset, & i = 0 \\ P_{i-1} \cup \text{median}(T \setminus P_{i-1})\}, & 0 < i \leq n \end{cases}$$

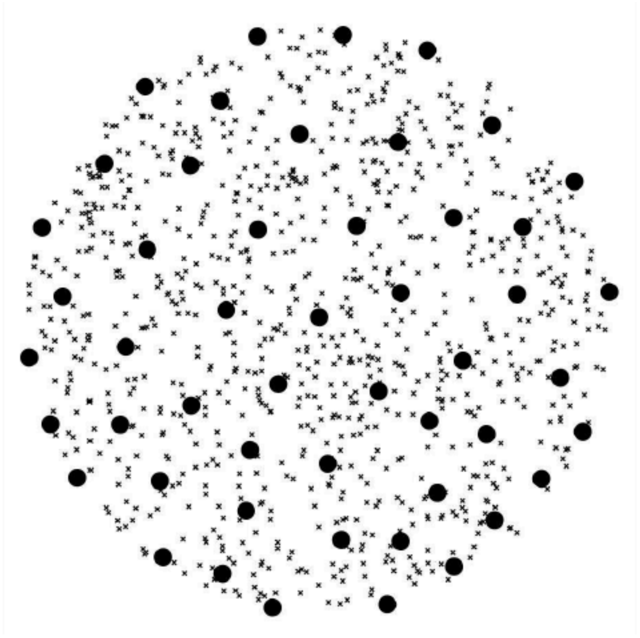
$$\text{median}(G) = \underset{g_1 \in G}{\text{argmin}} \sum_{g_2 \in G} d(g_1, g_2)$$



Spanning Prototype Selection

- Start with the median graph and iteratively add the graphs that are most dissimilar to the already chosen ones.

$$P_i = \begin{cases} \{\text{median}(T)\}, & i = 1 \\ P_{i-1} \cup \{\text{argmax}_{g \in T \setminus P_{i-1}} \min_{p \in P_{i-1}} d(g,p)\}, & 1 < i \leq n \end{cases}$$



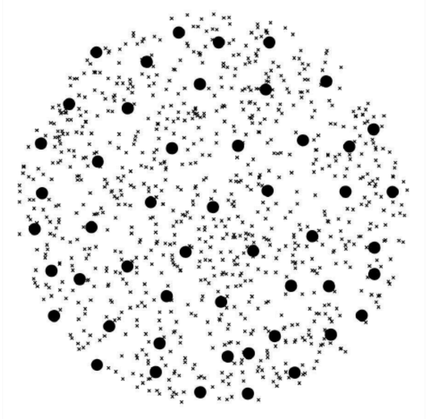
K-Center Prototype Selection

1. Choose an initial set of prototypes $\{p_1, ..., p_n\}$ randomly or by means of another prototype selector.
2. Define n sets $S_i = \{p_i\}$ and add each graph $g \in G \backslash P$ to the set whose prototype p_i has the smallest edit distance to g.
3. Calculate the center graph c_i for each set S_i. The center graph minimizes the maximum edit distance to all other graphs in S_i:

$$c_i = \underset{g_1 \in S_i}{\operatorname{argmin}} \max_{g_2 \in S_i} d(g_1, g_2)$$

4. If the center graph c_i is different from the prototype p_i, let it be the new prototype and return to step 2 until no further changes are made.

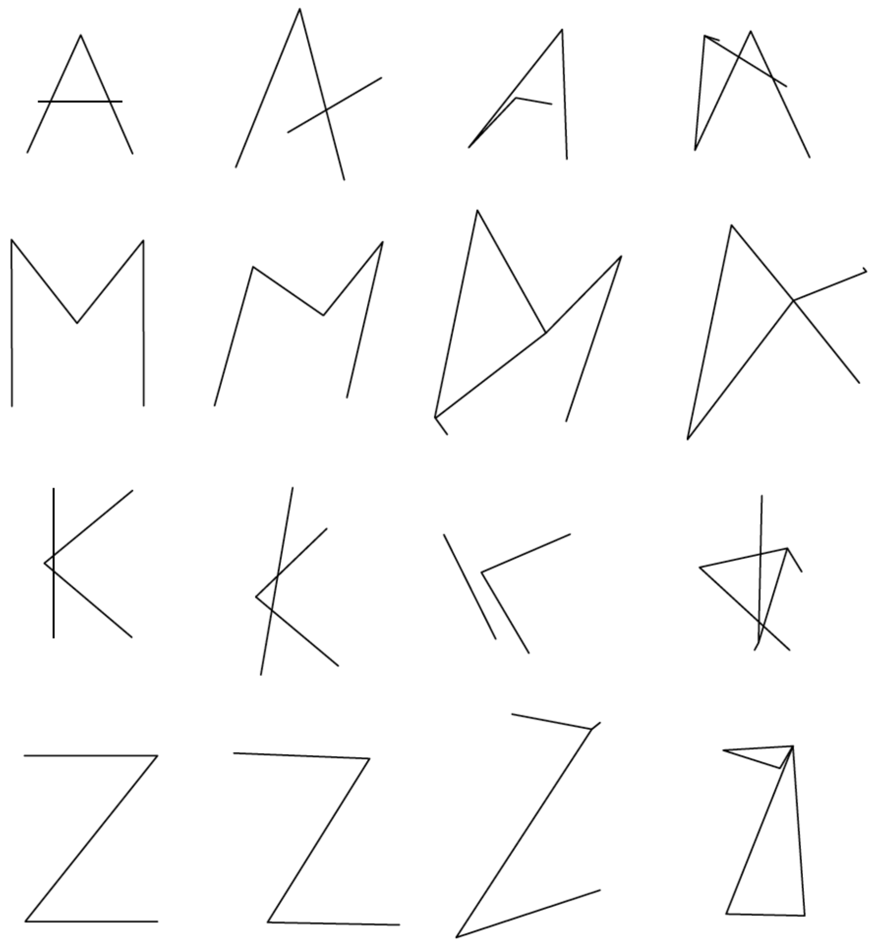
We need to embed graph w.r.t. prototype



Applications

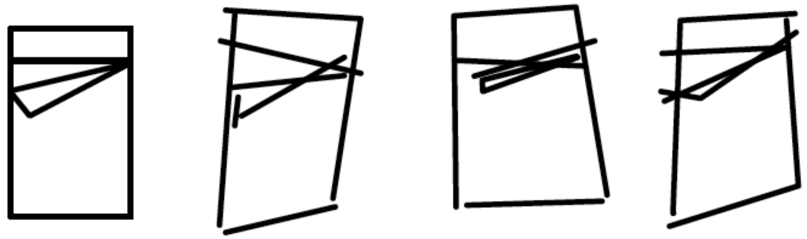
Letter Database

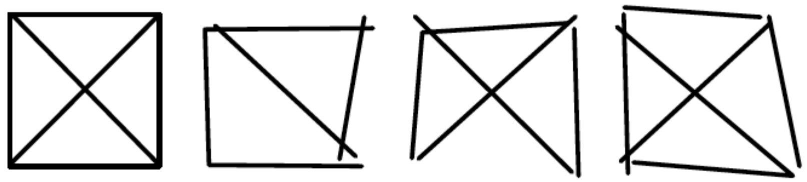
- Graphs representing line drawings of 15 capital letters. Three different levels of artificial distortions.

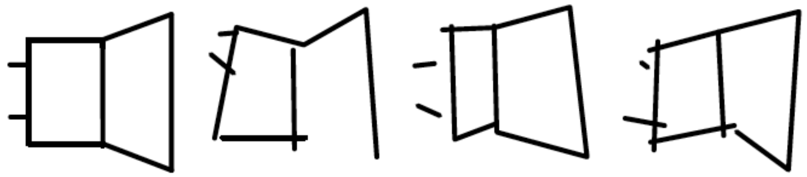


GREC Database

- Graphs representing 32 symbols from technical drawings. Artificial distortions have been applied to the images.

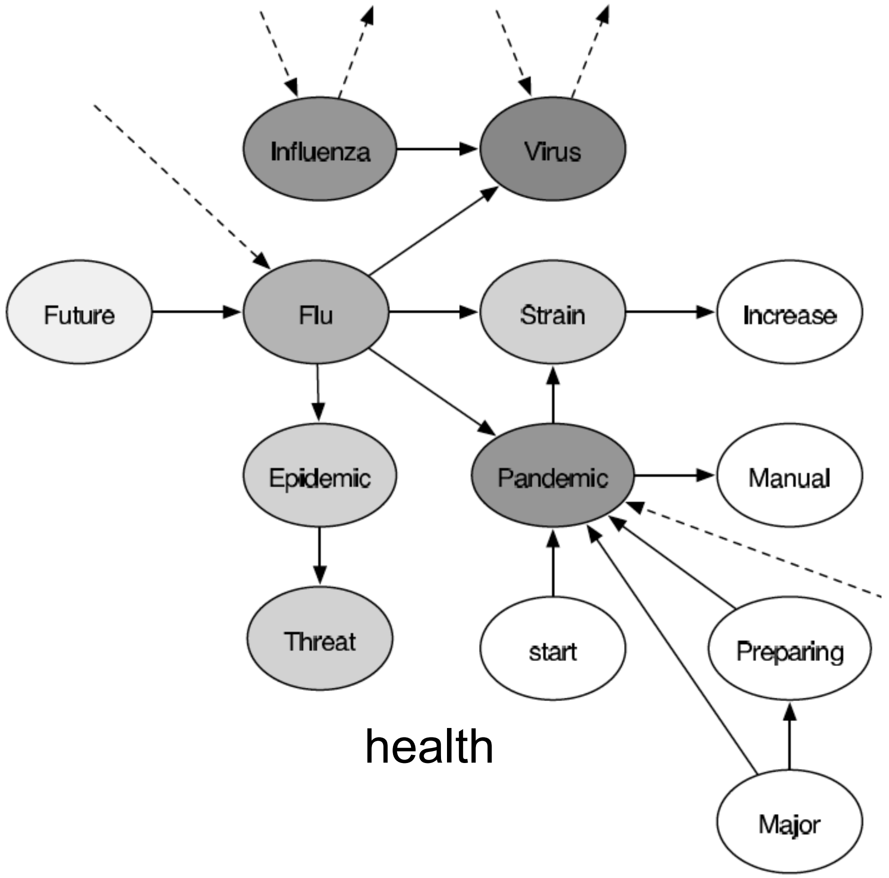

Symbol 1


Symbol 2

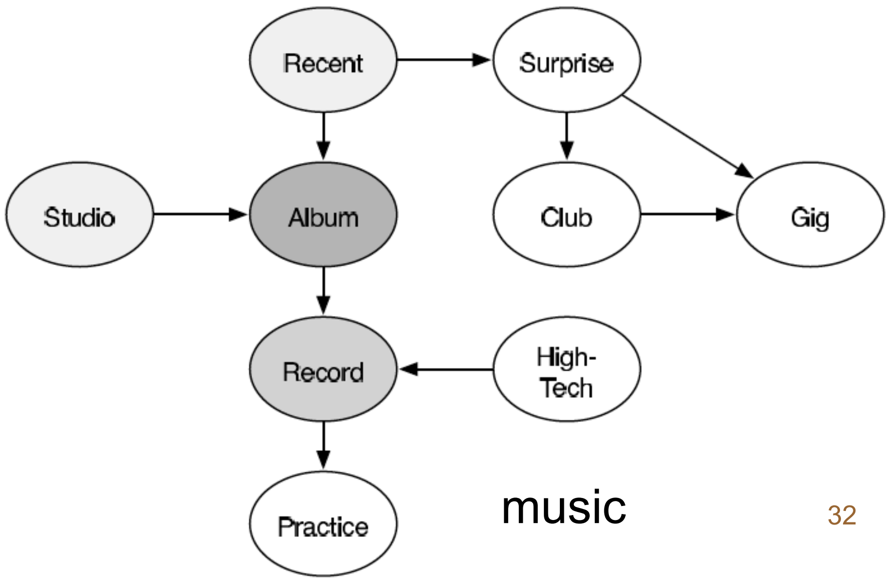

Symbol 3

Webgraph Database

- Word graphs of web documents from 20 classes (business, health, politics, sports, …).



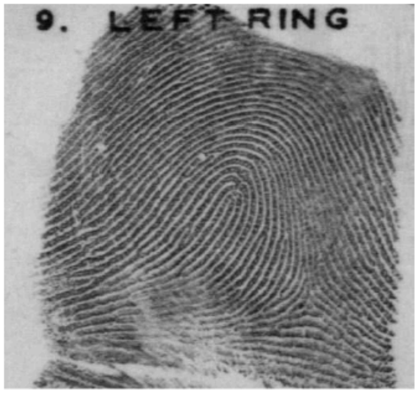
health



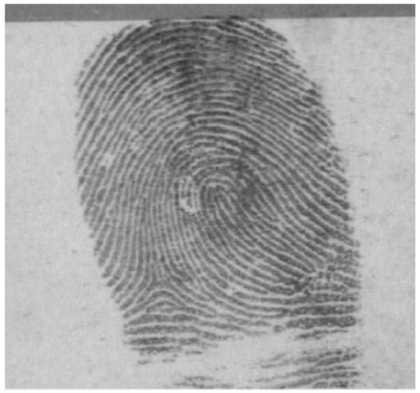
music

Fingerprint Database

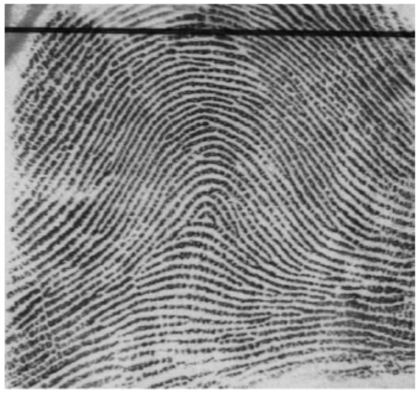
- Graphs representing fingerprint images with 4 general classes.



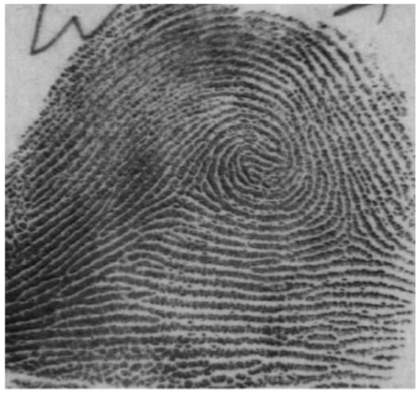
Left



Right



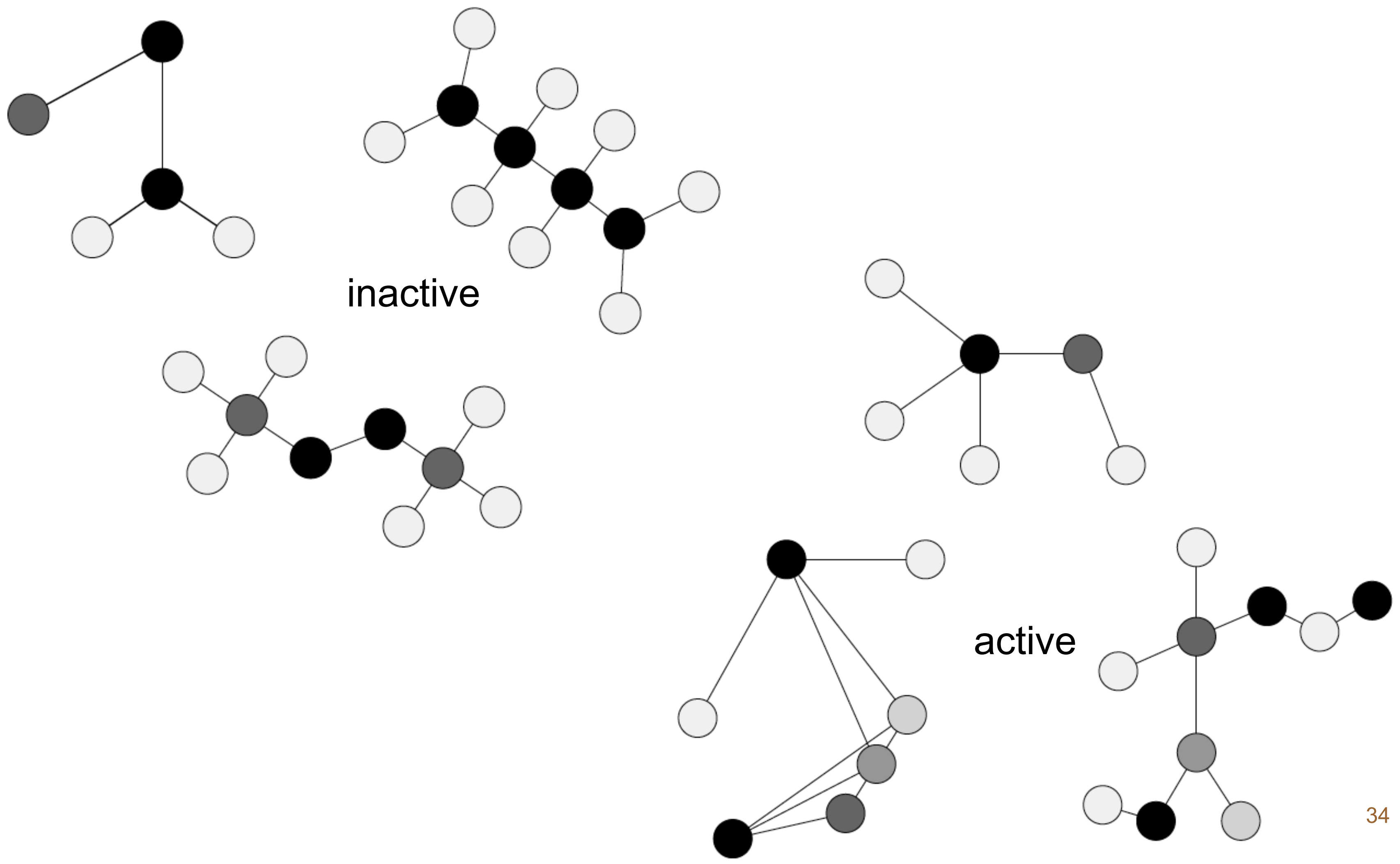
Arch



Whorl

Molecules Database

- Active and inactive molecules from an AIDS antiviral scan database.



Experimental Comparison

- System 1: KNN classification using graph edit distance.
- System 2: SVM classification using common walk kernel.
- System 3: SVM classification using graph edit distance kernel.
  - m is the number of prototypes
  - PS is the prototype selection strategy
- The difference between the systems is analyzed with a statistical significance test (Z-test with α = 0.05).

① / ②	Statistically significantly better than system 1 / 2
❶ / ❷	Statistically significantly worse than system 1 / 2

Results

	Ref. Systems		Embedding classifiers		
Database	k-NN	Walk Kernel	Embedding Kernel	m	PS
Letter Low	91.07	90.40	91.80 ②	225	k-CPS
Letter Med	76.80	75.33	81.80 ①②	450	TPS
Letter High	61.60	60.60	74.07 ①②	270	k-CPS
GREC	86.04	92.71 ①	89.17 ①❷	250	TPS
Image	59.50	76.67 ①	74.67 ①	40	CPS
Webgraphs	75.90	67.69 ❶	82.43 ①②	130	k-CPS
Fingerprints	82.60	83.80	85.00 ①	300	TPS-C
Molecules	97.13	95.67 ❶	98.13 ①②	150	SPS