Kernel Methods ...

... for mining graph data

Thomas Gärtner

Fraunhofer Institut Autonome Intelligente Systeme
Department of Computer Science III, University of Bonn

thomas.gaertner@ais.fraunhofer.de

Fraunhofer Institut Autonome Intelligente Systeme

Uni Bonn
RHEINISCHE FRIEDRICH-WILHELMS-UNIVERSITÄT

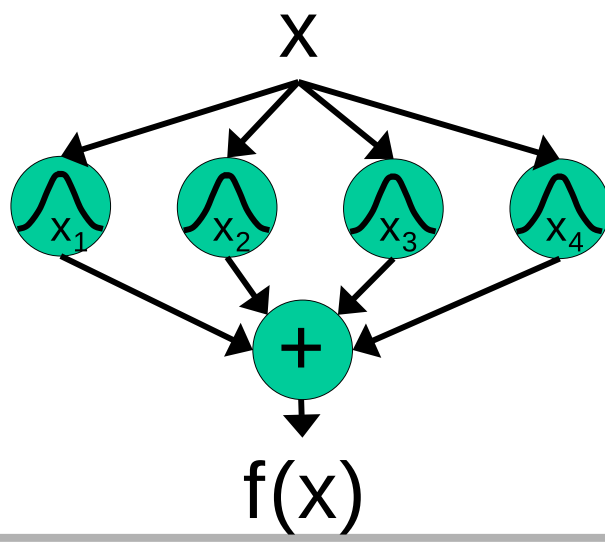
Overview

kernel methods and set kernels

string kernels, tree kernels, and graph kernels

open issues

Basis Function Networks



Thomas Gärtner - MGTS, 2003

Kernel Methods

find linear combination of basis functions

$$f(\cdot) = \sum c_i k(x_i, \cdot)$$

with positive definite k : $\sum c_i c_j k(x_i, x_j) \geq 0$

- convex optimisation problem
- geometric interpretation

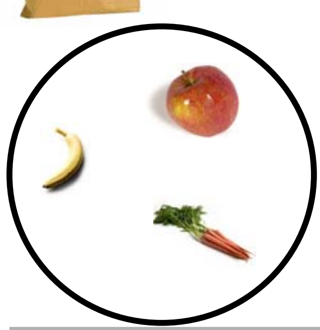
$$k(x_i, x_j) = \left\langle \phi(x_i), \phi(x_j) \right\rangle$$

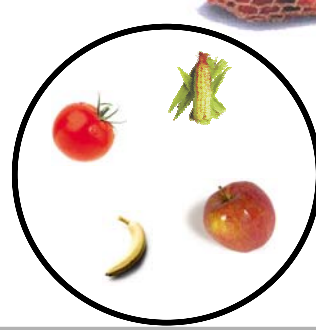
- covariance function

Thomas Gärtner - MGTS, 2003

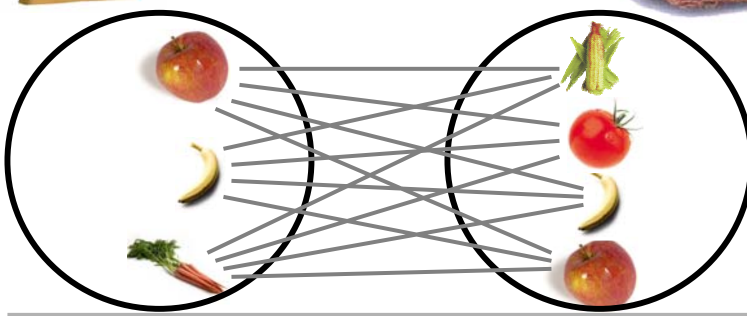
Kernels for Sets

Kernels for Sets



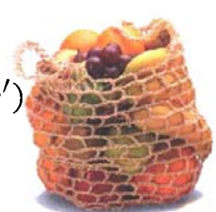


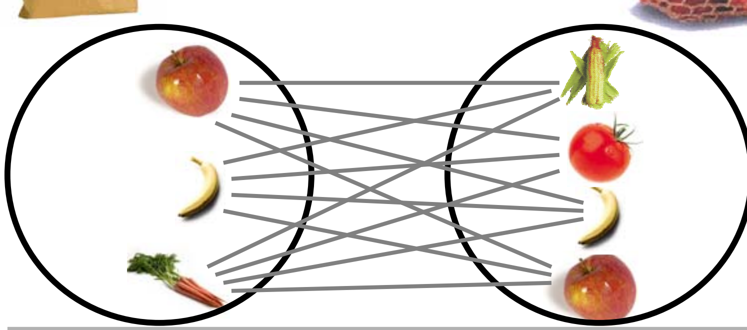
Kernels for Sets



Kernels for Sets

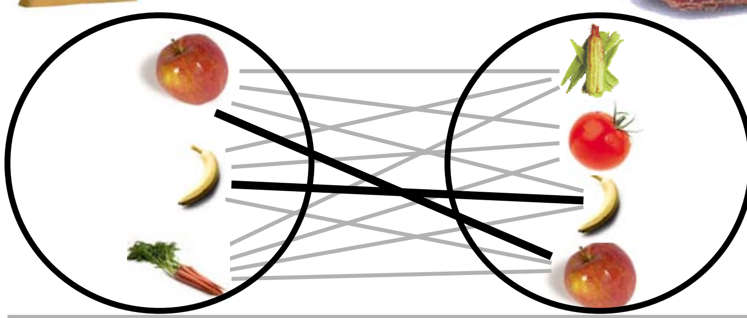
$$K(X, X') = \sum_{(x,x') \in X \times X'} k(x, x')$$





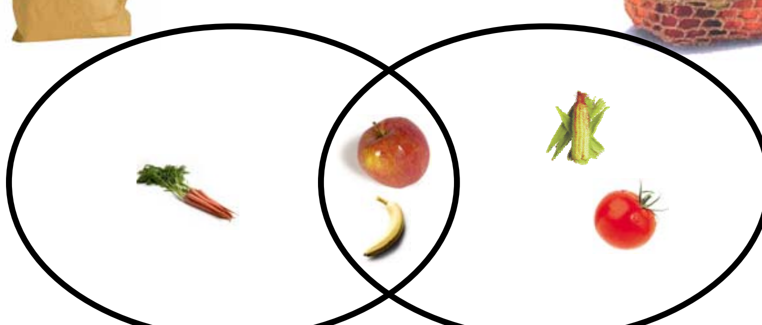
Kernels for Sets

$$K(X, X') = \sum_{(x,x') \in X \times X'} k(x, x')$$



Thomas Gärtner - MGTS, 2003

Kernels for Sets



Thomas Gärtner - MGTS, 2003

Kernels for Sets

1		1
1		1
1		0
0		1
0		1

Kernels for Multisets

2		1
3		2
1		0
0		2
0		5

Abstraction Kernels

$X, X' : \mathcal{X} \to \mathcal{Y}$

$$k_{\mathcal{X}\to\mathcal{Y}}\left(X, X'\right) = \sum_{(x,x')\in\mathcal{X}\times\mathcal{X}} k_{\mathcal{X}}\left(x, x'\right) k_{\mathcal{Y}}\left(X(x), X'(x')\right)$$

abstractions generalise sets, multisets, and measures

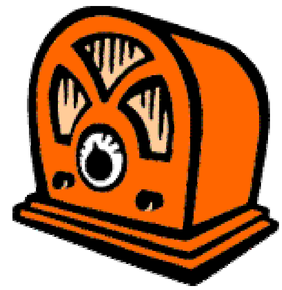
Multiset Example

$$k_{\text{coset}}(X, X') = \sum_{x\in\mathcal{X}, x'\in\mathcal{X}} k_x(x, x') k_y(X(x), X'(x'))$$

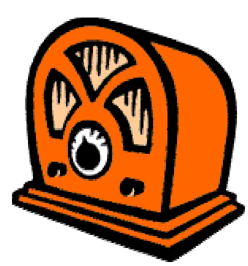
$X : \{a, b, c, d\} \to \mathbb{N}$

if X represents $\langle a, a, b, b, b, c\rangle$
and X' represents $\langle a, a, a, b, c, c, c, c\rangle$
then $k_{\text{coset}}(X, X') = 6 + 3 + 4 = 13$

Kernels for Objects



Kernels for Objects



Thomas Gärtner - MGTS, 2003

Convolution Kernels

$$R^{-1} : \mathcal{X} \rightarrow \{\mathcal{X}_1 \times \cdots \times \mathcal{X}_D\}$$

$$k_{conv}(x,y) = \sum_{\vec{x} \in R^{-1}(x), \vec{y} \in R^{-1}(y)} \prod_{d=1}^{D} k_d(x_d, y_d)$$

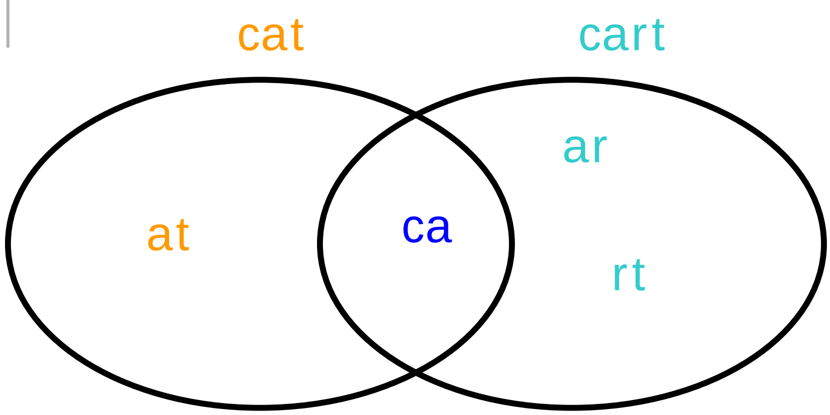
R^{-1} decomposes Instances into their parts
all background knowledge goes in defining R^{-1}
d iterates over the components of the parts (*D*-tuples)

Kernels for

Strings and Sequences

Thomas Gärtner - MGTS, 2003

n-gram Kernels (n = 2)



$$k(\text{cat}, \text{cart}) = 1$$

Thomas Gärtner - MGTS, 2003

n-gram Kernels (n = 2)

cat		cart
1	ca	1
0	cr	0
0	ct	0
0	ar	1
1	at	0
0	rt	1

$$k(\text{cat}, \text{cart}) = 1$$

Thomas Gärtner - MGTS, 2003

Subsequences vs. n-gram

```
subs                        subs
sub  e                       ubse
sub    q                      bseq
...                            sequ
sub            s                eque
...                              quen
 u  s         es                  uenc
...                                ence
            nces                    nces
```


Subsequences vs. n-gram

```
subs               :  4     subs
sub  e             :  5      ubse
sub    q           :  6       bseq
...                            sequ
sub            s :12            eque
...                              quen
 u  s         es :11              uenc
...                                ence
            nces :  4               nces
```

Subsequence Kernel

$$\phi_u(s) = \sum_{i:u=s[i]} \lambda^{l(i)}$$

$$k_n(s,t) = \sum_{u\in\Sigma^n} \phi_u(s)\phi_u(t)$$
$$= \sum_{u\in\Sigma^n} \sum_{i:u=s[i]} \sum_{j:u=t[j]} \lambda^{l(i)+l(j)}$$

Subsequence Kernel

cat		cart
λ^2	c...a	λ^2
	c...r	λ^3
λ^3	c...t	λ^4
	a...r	λ^3
λ^2	a...t	λ^3
	r...t	λ^2

$$k(\text{cat}, \text{cart}) = \lambda^2\lambda^2 + \lambda^3\lambda^4 + \lambda^2\lambda^3$$

String Kernels

n-gram kernels
subsequence kernels
gappy / mismatch / wildcard kernels
rational kernels

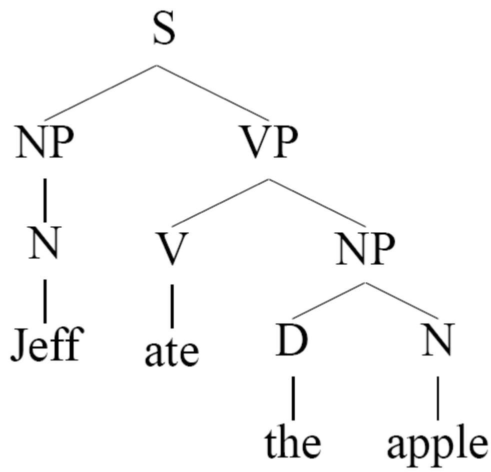
computation tricks
other kernels for sequences
application areas

Kernels for

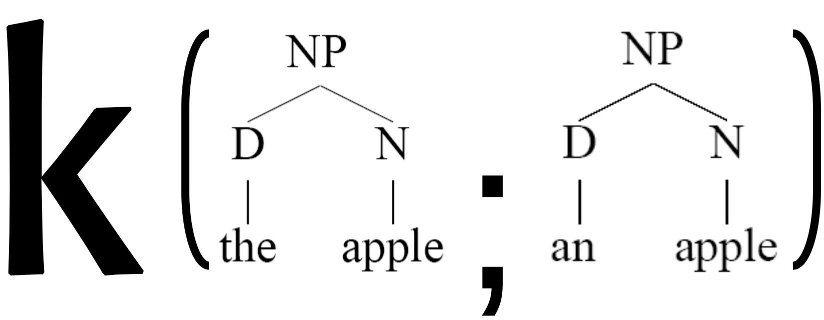
Trees

Thomas Gärtner - MGTS, 2003

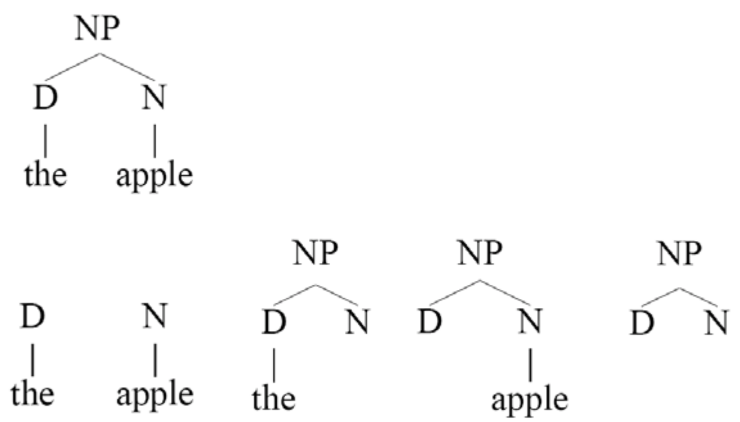
Parse-Tree



Parse-Tree Kernels

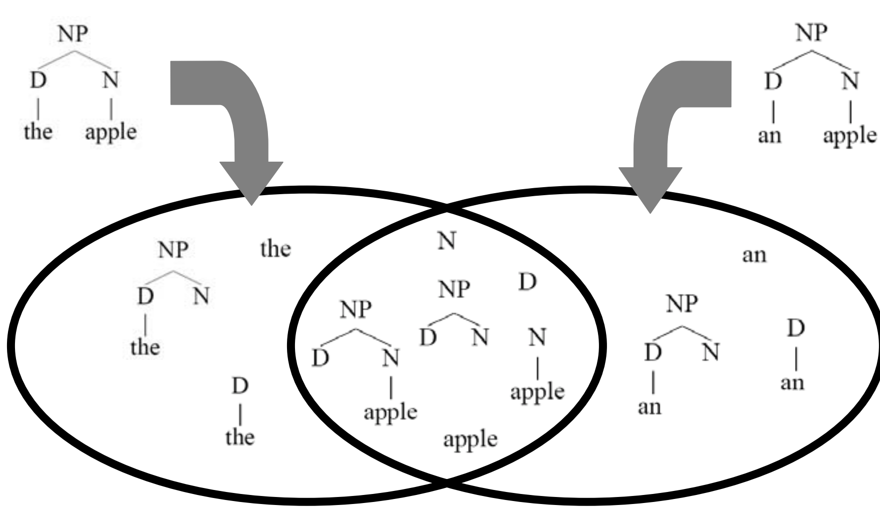


Parse-Tree Subtrees



Thomas Gärtner - MGTS, 2003

Parse-Tree Kernel



Thomas Gärtner - MGTS, 2003

Parse-Tree: Subtree

$$k(T_1, T_2) = \sum_i h_i(T_1) h_i(T_2) = \sum_{v_1 \in \mathcal{V}_1, v_2 \in \mathcal{V}_2} S(v_1, v_2)$$

$S(v_1, v_2) = 0$ if $label(v_1) \neq label(v_2)$

else

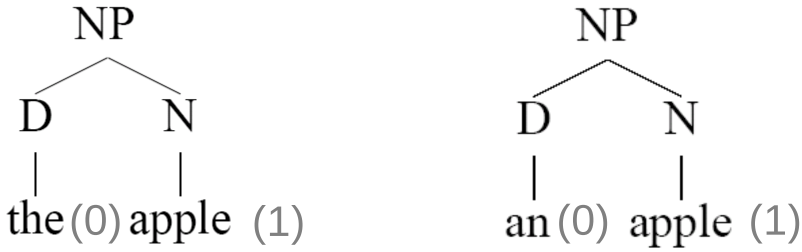
$S(v_1, v_2) = 1$ if $\delta^+(v_1) = 0$

else

$$S(v_1, v_2) = \prod_{j=1}^{|\delta^+(v_1)|} \left[1 + S\left(\delta_j^+(v_1), \delta_j^+(v_2)\right)\right]$$

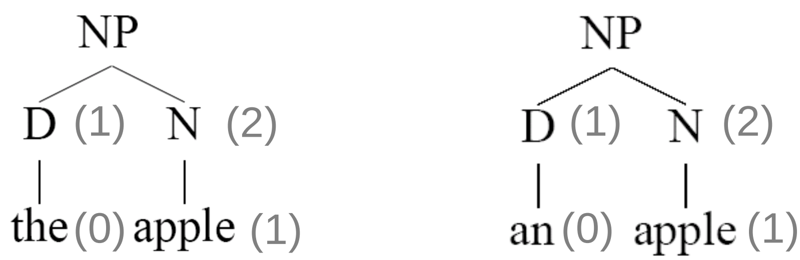
[slight modification of original kernel]

Parse-Tree Kernel



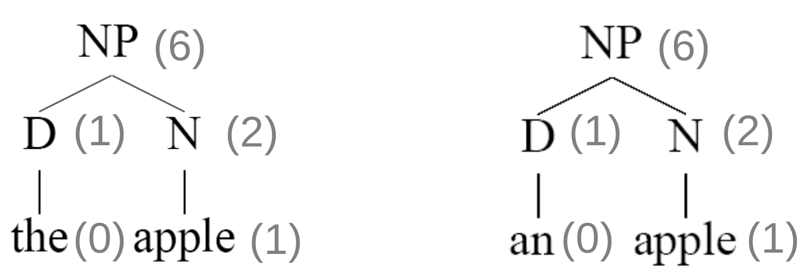
$S(v_1, v_2) = 0$ if $label(v_1) \neq label(v_2)$ else $S(v_1, v_2) = 1$ if $\delta^+(v_1) = 0$

Parse-Tree Kernel



$$S(v_1, v_2) = \prod_{j=1}^{|\delta^+(v_1)|} \left[1 + S\left(\delta_j^+(v_1), \delta_j^+(v_2)\right)\right]$$

Parse-Tree Kernel



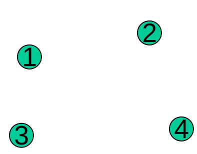
$$S(v_1, v_2) = \prod_{j=1}^{|\delta^+(v_1)|} \left[1 + S\left(\delta_j^+(v_1), \delta_j^+(v_2)\right)\right]$$

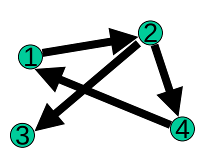
Kernels for

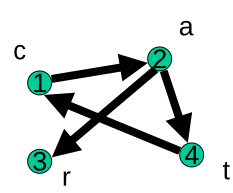
Graphs

Labelled Directed Graphs

vertices	edges	labels







$\mathcal{V} = \{\nu_1, \nu_2, \nu_3, \nu_4\}$ $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ $label : \mathcal{V} \cup \mathcal{E} \to \mathcal{L}$

$\mathcal{E} = \{(\nu_1, \nu_2), (\nu_2, \nu_4), (\nu_4, \nu_1), (\nu_2, \nu_3)\}$

Thomas Gärtner - MGTS, 2003

Sub-Graph Kernels

$$\Phi_H(G) = \lambda_{|\mathcal{E}(H)|} \left| \left\{ G' \text{ is subgraph of } G : G' \simeq H \right\} \right|$$

Bad News ...

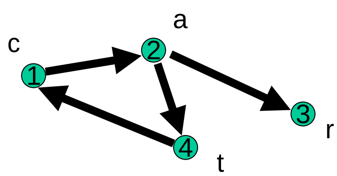
$$\Phi_H(G) = \lambda_{|\mathcal{E}(H)|} \left| \left\{ G' \text{ is subgraph of } G : G' \simeq H \right\} \right|$$

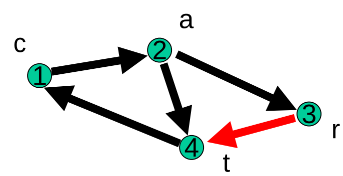
computing the sub-graph kernel is *NP*-hard

proof by mapping to `Hamiltonian Path'

Thomas Gärtner - MGTS, 2003

Path Kernel Example





13 paths
c, a, r, t
at, ca, ar, tc
cat, car, atc, tca
tcar

19 paths
c, a, r, t
at, ca, ar, rt, tc
cat, car, rtc, art, atc, tca
cart, artc, rtca, tcar

Complete Graph Kernels

$$k(G,\cdot) = k(G',\cdot) \Leftrightarrow G \simeq G'$$

Bad News ...

$$k(G, \cdot) = k(G', \cdot) \Leftrightarrow G \simeq G'$$

computing any complete graph kernel is at least as hard as `Graph Isomorphism'

proof by computing distance induced by *k*

$$k(G,G) - 2k(G,G') + k(G',G') = 0 \Leftrightarrow G \simeq G'$$

Interlude

negative results on graph kernels

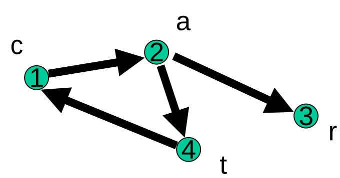
sub-graph kernels and complete graph kernels are computationally hard

positive results on graph kernels

kernels based on walks with common label sequences (with and without gaps) can be computed in polynomial time

Thomas Gärtner - MGTS, 2003

Walks in a Graph



13 paths

c, a, r, t

at, ca, ar, tc

car, cat, atc, tca

tcar

infinitely many walks

Thomas Gärtner - MGTS, 2003

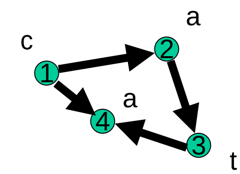
Walk Kernel

$$\phi_s(G) = \sqrt{\lambda_{|s|}} \left| \left\{ w \in \mathcal{W}_{|s|}(G), \forall\, i:\; s_i = l_i(w) \right\} \right|$$

contiguous label sequence

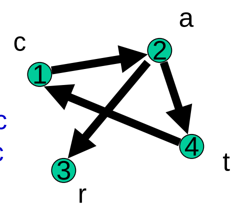
Thomas Gärtner - MGTS, 2003

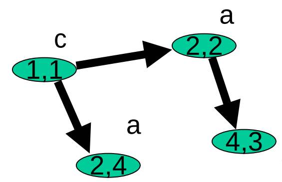
Direct Product Graphs



c, a, a, t
ca, ca, at, ta
cat
cata

c, a, r, t
ca, ar, at, tc
cat, tca, atc
. . .



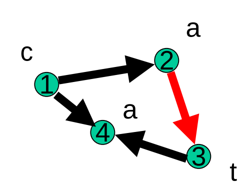


c, a, a, t
ca, ca, at
cat

Thomas Gärtner - MGTS, 2003

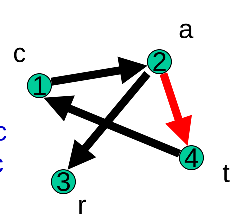
Direct Product Graphs

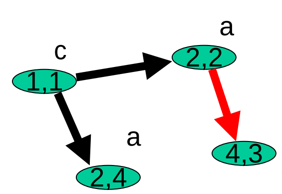
at



c, a, a, t
ca, ca, at, ta
cat
cata

c, a, r, t
ca, ar, at, tc
cat, tca, atc
. . .



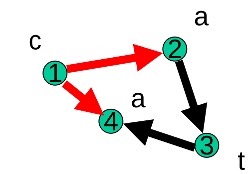


c, a, a, t
ca, ca, at
cat

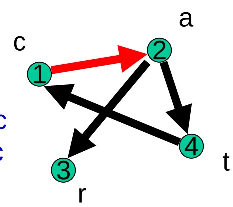
Thomas Gärtner - MGTS, 2003

Direct Product Graphs

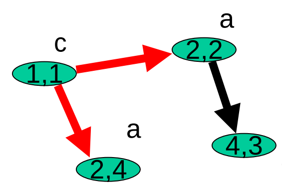
ca



c, a, a, t
ca, ca, at, ta
cat
cata



c, a, r, t
ca, ar, at, tc
cat, tca, atc
. . .

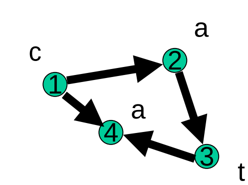


c, a, a, t
ca, ca, at
cat

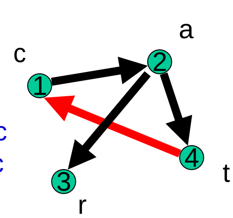
Thomas Gärtner - MGTS, 2003

Direct Product Graphs

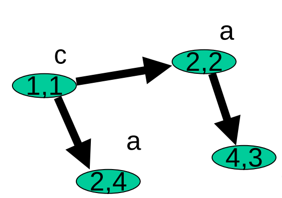
tc



c, a, a, t
ca, ca, at, ta
cat
cata



c, a, r, t
ca, ar, at, tc
cat, tca, atc
. . .

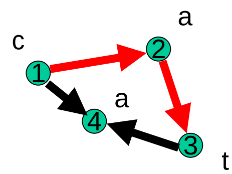


c, a, a, t
ca, ca, at
cat

Thomas Gärtner - MGTS, 2003

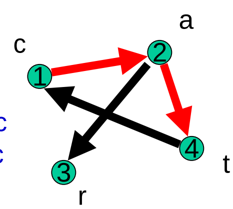
Direct Product Graphs

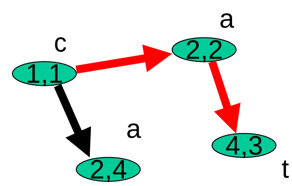
cat



c, a, a, t
ca, ca, at, ta
cat
cata

c, a, r, t
ca, ar, at, tc
cat, tca, atc
. . .





c, a, a, t
ca, ca, at
cat

Computation

$$\phi_s(G) = \sqrt{\lambda_{|s|}} \left| \left\{ w \in \mathcal{W}_{|s|}(G), \forall i : \; s_i = l_i(w) \right\} \right|$$

contiguous label sequences

the inner product can be computed as

$$k_\times(G_1, G_2) = \sum_{ij} \left[\sum_{n=0}^{\infty} \lambda_n E_\times^n \right]_{ij}$$

Walk Kernel (with gaps)

$$\phi_s(G) = \sqrt{\lambda_{|s|}\alpha^m} \left| \left\{ w \in \mathcal{W}_{|s|}(G), \forall i:\ match(s_i, l_i(w)) \right\} \right|$$

label sequences with *m* wildcards

the inner product can be computed as

$$k_*(G_1, G_2) = \sum_{ij} \left[\sum_{n=0}^{\infty} \lambda_n ((1-\alpha)E_\times + \alpha E_o)^n \right]_{ij}$$

Thomas Gärtner - MGTS, 2003

Matrix Power Series

- decompose adjacency matrix

$$E = TDT^{-1}$$

- compute component-wise

$$\sum_n^{\infty} \lambda_n E^n = T \left(\sum_n^{\infty} \lambda_n D^n \right) T^{-1}$$

as $E^n = TD^nT^{-1}$

Thomas Gärtner - MGTS, 2003

Example Power Series

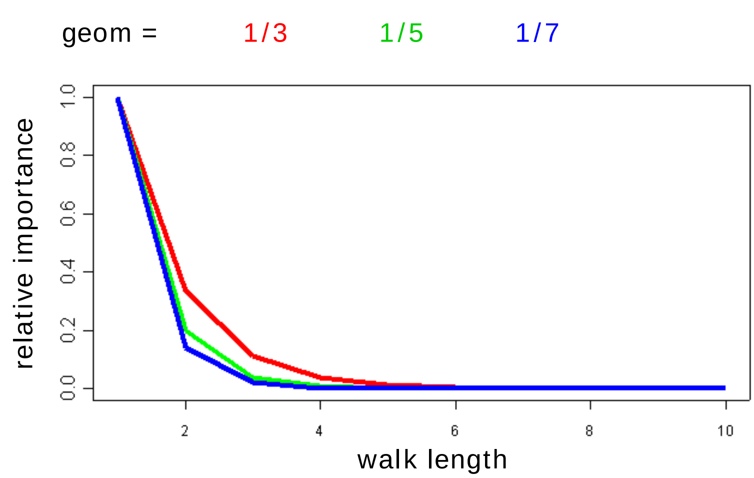
exponential series $\lambda_n = \frac{\beta^n}{n!}$

$$\sum_n^{\infty} \frac{(\beta E)^n}{n!} = e^{\beta E} = T e^{\beta D} T^{-1}$$

geometric series $\lambda_n = \gamma^n$

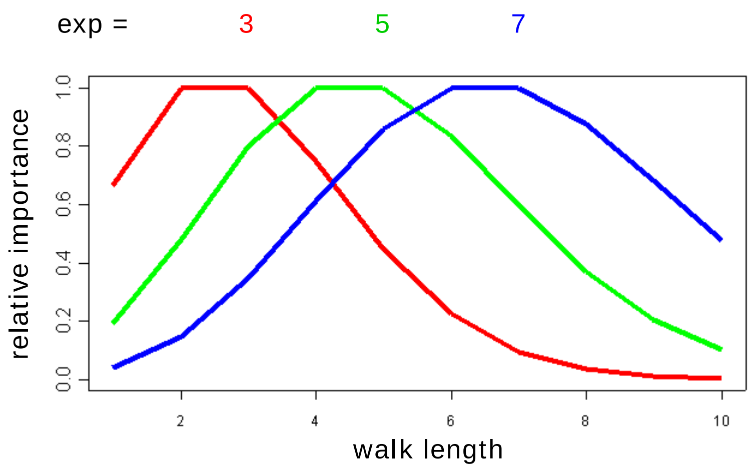
$$\sum_n^{\infty} \gamma^n E^n = (\mathbf{I} - \gamma E)^{-1}$$

GEOM Parameter



Thomas Gärtner - MGTS, 2003

EXP Parameter



Extensions

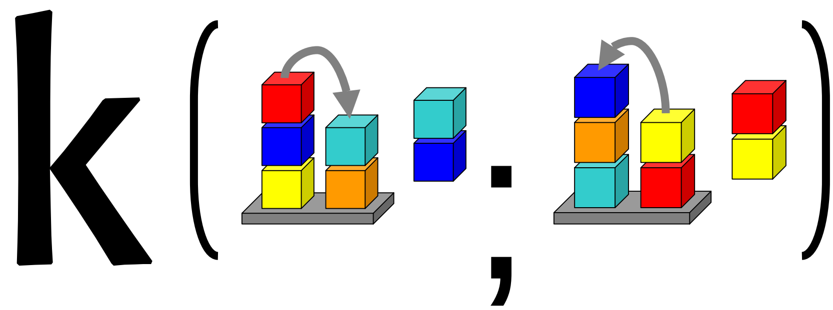
graphs with a transition probability associated with each edge

application of Gaussian processes to relational reinforcement learning [ILP’03]

efficient computation for unlabelled graphs (tensor product of adjacency matrices)

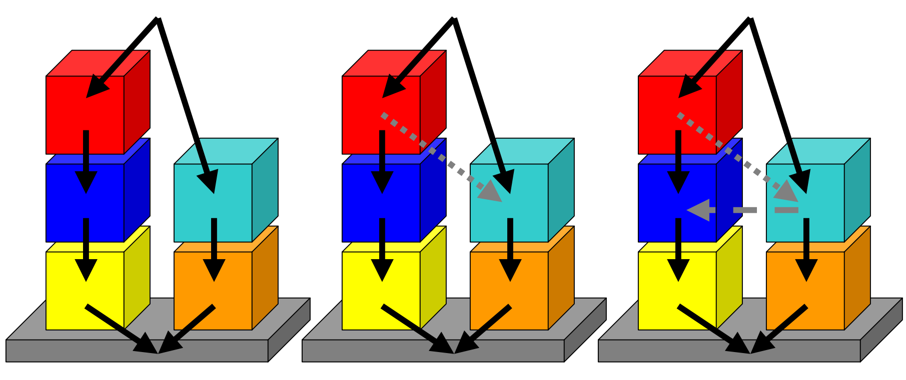
Thomas Gärtner - MGTS, 2003

Blocks World Kernel



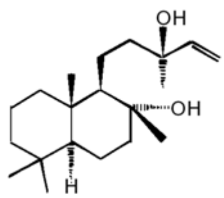
Blocks-World as Graph

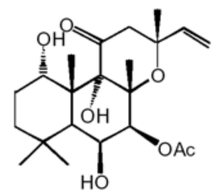
state action goal



Thomas Gärtner - MGTS, 2003

Molecule Kernel

k(

; 

)

Bibliography

A Survey of Kernels for Structured Data

SIGKDD Explorations. Volume 5, Issue 1, July 2003

Recent Papers

Proceedings of the 16th Annual Conference on Learning Theory
and 7th Annual Workshop on Kernel Machines, 2003

Issues

what is a good kernel (given a concept class)
completeness, separation, and convergence

which concept classes are important

applications for graph kernels

incorporate background knowledge
how to choose a kernel / how to choose features