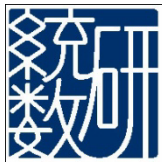
Kernel Method: Data Analysis with Positive Definite Kernels

6. Structured Data

Kenji Fukumizu

The Institute of Statistical Mathematics
Graduate University for Advanced Studies /
Tokyo Institute of Technology

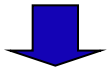
Nov. 17-26, 2010
Intensive Course at Tokyo Institute of Technology



Structured Data

- Kernel method for structured data

A positive definite kernel can be defined on an arbitrary set.
Data may not be vectors.



– Structured (non-vectorial) data
  - String: sequence of alphabets with different length.
  - Tree.
  - Graph data, network data

– Once positive definite kernels are defined, various kernel methods can be applied, such as kernel PCA, SVM, kernel ridge regression, etc.

– What we need to compute for kernel methods
  = Gram matrix $k(x_i, x_j)$

Outline

- Kernels for string data
- Kernels for trees and graphs
- Kernels for probability measures

- Kernels for string data
- Kernels for trees and graphs
- Kernels for probability measures

String

- Alphabet Σ : finite set
- String : finite sequence of elements of Σ
  - *e.g.*, Σ = { a, b, c, d,…, z }
    String: cat, head, computer, xyydyaa,…
- Σ^p : set of strings of length p
- Σ^* : set of strings of arbitrary length

$$\Sigma^* = \bigcup_{p=0}^{\infty} \Sigma^p$$

Note: $\Sigma^0 = \{\varepsilon\}$: empty string

- Notations:

  $s = s_1 s_2 \ldots s_n$ string

  - $|s|$: length of string s.
  - $s[i:j]$: subsequence of s defied by $s_i \ldots s_j$
  - The concatenation of two strings s and t is
    $st = s_1 s_2 \ldots s_n t_1 t_2 \ldots t_m$

String Kernel

- String kernel
  - Positive definite kernel defined on Σ^*
  - Similarity measure for two strings
  - Typically, defined by counting common substrings
  - Efficient computation is very important -- recursion

- Typical applications
  - Natural language processing
    - Alphabet: $\Sigma = \{a, b, c, \ldots, z, A, B, ..., Z\}$
    - Words: $\Sigma =$ all the words
  - Bioinformatics
    - Genome sequence: $\Sigma = \{A, T, G, C\}$
    - Protein sequence $\Sigma =$ {amino acid}（20 types）

Applications

- Structural classification of proteins
  - Protein = sequence of 20 types of amino acids.

```
7LES_DROME    LKLLRFLGSGAFGEVYEGQLKTE....DSEEPQRVAIKSLRK.......
ABL1_CAEEL    IIMHNKLGGGQYGDVYEGYWK........RHDCTIAVKALK........
BFR2_HUMAN    LTLGKPLGEGCFGQVVMAEAVGIDK.DKPKEAVTVAVKMLKDD.....A
TRKA_HUMAN    IVLKWELGEGAFGKVFLAECHNLL...PEQDKMLVAVKALK........
```

String → Types of 3D-structure (fold) / evolutionary relationship.

  - Structural information is important to evolutionary and functional similarity.
  - SCOP（Structural Classification of Proteins）

    http://scop.mrc-lmb.cam.ac.uk/scop/

p-Spectrum Kernel I

– Feature vector = frequency of substrings of length p.

$$|\Sigma| = m, \quad u \in \Sigma^p$$

$$\phi_u^p(s) = \left|\{(w_1, w_2) \in \Sigma^* \times \Sigma^* \mid s = w_1 u w_2\}\right|$$

frequency of *u* in string *s*

$$\Phi : \Sigma^* \to H \cong \mathbf{R}^{m^p}, \qquad \Phi^p(s) = \left(\phi_u^p(s)\right)_{u \in \Sigma^p}$$

Feature space H = the strings of length *p* (m^p dim.)

$$K_p(s,t) = \sum_{u \in \Sigma^p} \phi_u^p(s)\phi_u^p(t) = \left\langle \Phi^p(s), \Phi^p(t) \right\rangle_H$$

p-Spectrum Kernel II

s = “statistics” t = “pastapistan”

3-spectrum

s: sta, tat, ati, tis, ist, sti, tic, ics

t: pas, ast, sta, tap, api, pis, ist, sta, tan

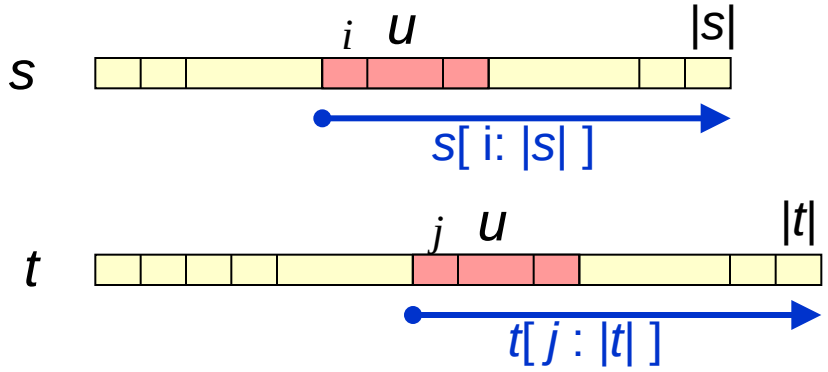
	sta	tat	ati	tis	ist	sti	tic	ics	pas	ast	tap	api	pis	tan
Φ(s)	1	1	1	1	1	1	1	1	0	0	0	0	0	0
Φ(t)	2	0	0	0	1	0	0	0	1	1	1	1	1	1

$K_3(s, t) = 1x2 + 1x1 = 3$

Computation of *p*-Spectrum Kernel I

$$K_p(s,t) = \sum_{u \in \Sigma^p} \phi_u^p(s)\phi_u^p(t) \quad = \left\langle \Phi^p(s), \Phi^p(t) \right\rangle_H$$

– Direct computation



Substring *u* : prefix of a suffix
(substring from a middle to the end)

$$h_p(s,t) = \begin{cases} 1 & p\text{-prefix of } s = p\text{-prefix of } t \\ 0 & p\text{-prefix of } s \neq p\text{-prefix of } t \end{cases}$$

$$K_p(s,t) = \sum_{i=1}^{|s|-p+1} \sum_{j=1}^{|t|-p+1} h_p(s[i:|s|], t[j:|t|])$$

Computational cost = $O(p|s||t|)$

Computation of *p*-Spectrum Kernel II

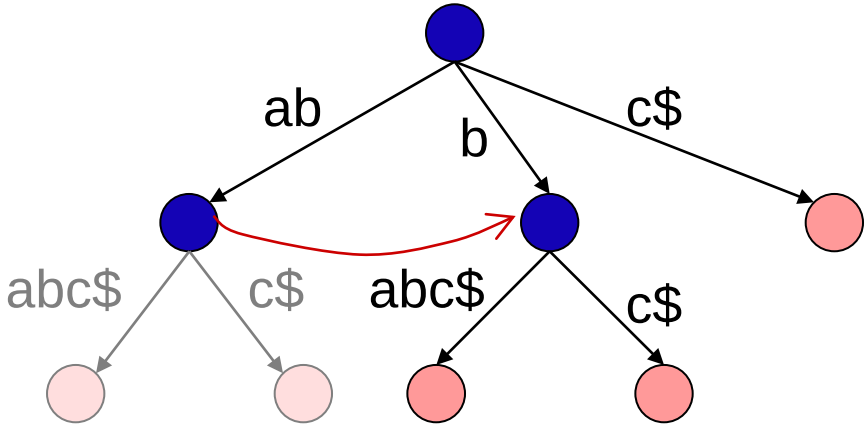
There is a clever way of computing $K(s,t)$ in $O(p(|s|+|t|))$ (linear to the lengths) .

– Suffix Tree

- Algorithm to express all the suffixes in a tree structure.
- The computational cost for constructing a suffix tree is linear to the string length in time and space.

  Example) ababc

ababc
babc
abc
bc
c



- See Gusfield 1997, Vishwanathan & Smola 2003

All-subsequences Kernel I

- Feature vector = frequency of all the substrings (gap allowed).
- Feature space = vector space indexed by arbitrary strings. (infinite dim.)
- Feature map

$$\phi_u(s) = \left|\{\vec{i} \mid s[\vec{i}] = u\}\right| \quad \text{where} \quad s[\vec{i}] = s_{i_1} s_{i_2} s_{i_3} \cdots s_{i_\ell}$$

$$\text{for} \quad \vec{i} = [i_1, i_2, i_3, \ldots, i_\ell] \quad (1 \le i_1 < i_2 < \cdots < i_\ell \le |s|)$$

ex) s: statsitics $\vec{i} = [2,3,9] \Rightarrow s[\vec{i}] =$ tac

$$\Phi : \Sigma^* \to H \cong \mathbf{R}^\infty, \qquad \Phi(s) = \left(\phi_u(s)\right)_{u \in \Sigma^*}$$

$$k(s,t) = \sum_{u \in \Sigma^*} \phi_u(s)\phi_u(t) \quad = \quad \left\langle \Phi(s), \Phi(t) \right\rangle$$

All-subsequences Kernel II

- Matching with gap allowed

ATGACTAC
CATGCGATT

ATGACTAC
CATG CGATT

u = ATGCA

- Example

s: ATG t : AGC $K(s,t) = 4$

	ε	A	T	G	C	AT	AG	AC	TG	GC	ATG	AGC
Φ(s)	1	1	1	1	0	1	1	0	1	0	1	0
Φ(t)	1	1	0	1	1	0	1	1	0	1	0	1

ε: empty string

Computation of All-subsequences Kernel I

Recursive formula: reuse of computations in the past

– Initial:

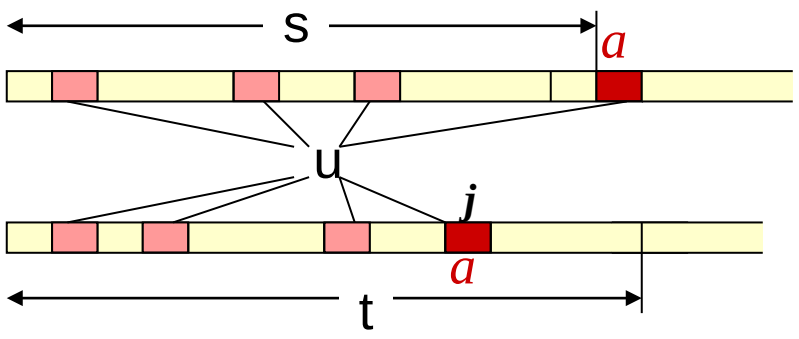
$k(s, \varepsilon) = k(t, \varepsilon) = 1$ （for any s, t ） ε : empty string

– Suppose we have computed up to $k(s, t)$. How can we compute $k(sa, t)$?

$k(sa, t)$

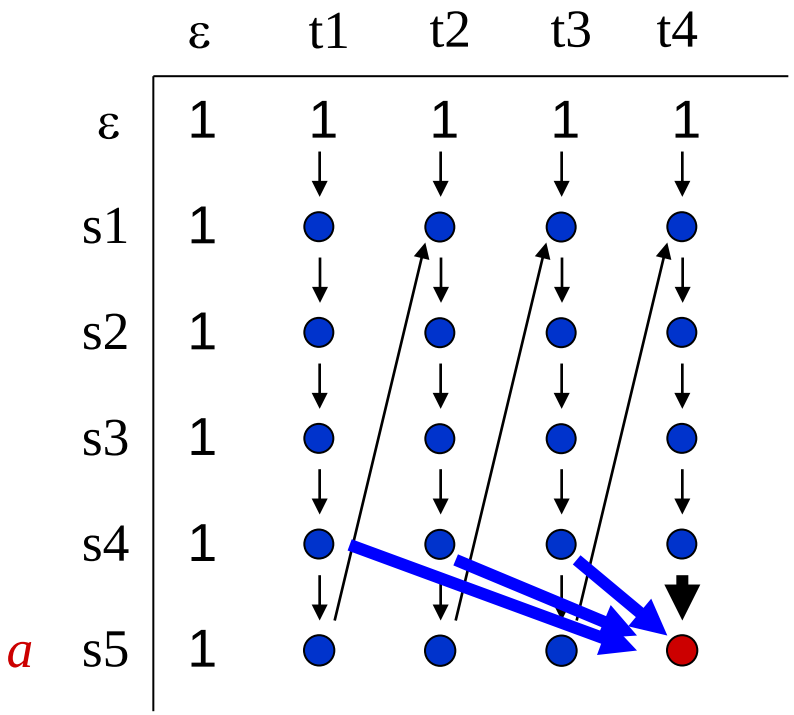
= (Matching within s and t) + (Matching including a)

$$= k(s,t) + \sum_{\substack{1 \leqq j \leqq |t| \\ j:\ t[j] = a}} k(s, t[1:\ j-1])$$



Computation of All-subsequences Kernel II

$$k(sa,t) = k(s,t) + \sum_{\substack{1 \leqq j \leqq |t| \\ j:\ t[j] = a}} k(s, t[1:\ j-1])$$



Cost = O(| s | | t |^2)

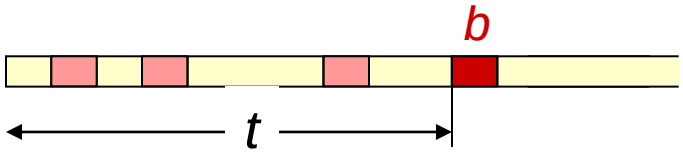
Computation of All-subsequences Kernel III

– More efficient computation

$$k(sa,t) = k(s,t) + \underbrace{\sum_{1 \leqq j \leqq |t|,\ j:\, t[j] = a} k(s, t[1:\, j-1])}_{\tilde{k}(sa,t)}$$

t computation is not desirable

Recursion on *t*



$$\tilde{k}(sa,tb) = \sum k(s, t[1:\, j-1]) + \delta_{ab} k(s,t)$$

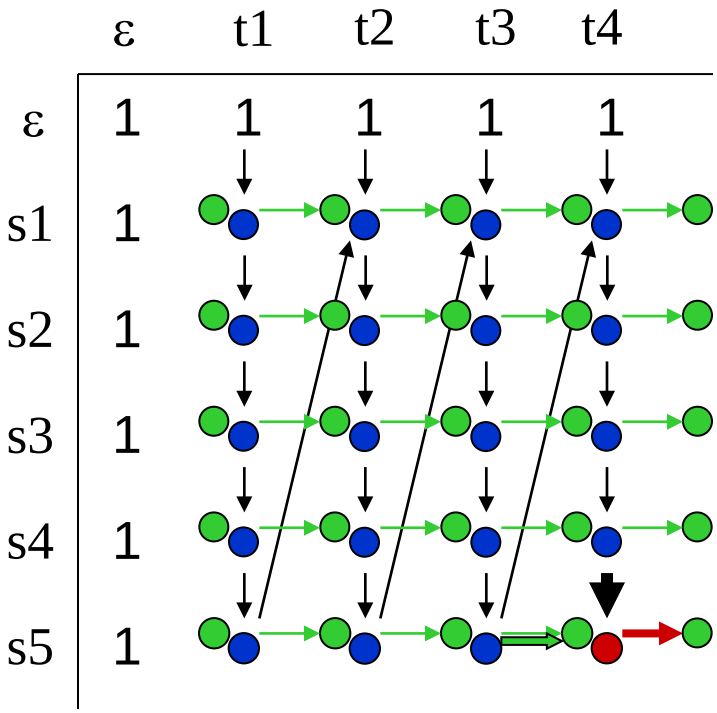
Case: $1 \leqq j \leqq |t|$ Case: $j = |tb|$

$$= \tilde{k}(sa,t) + \delta_{ab} k(s,t)$$

$$\begin{cases} k(sa,t) = k(s,t) + \tilde{k}(sa,t) & \text{(recursion on s)} \\ \tilde{k}(sa,tb) = \tilde{k}(sa,t) + \delta_{ab} k(s,t) & \text{(recursion on t)} \end{cases}$$

Computation of All-subsequences Kernel IV

- $k(sa,t) = k(s,t) + \tilde{k}(sa,t)$
- $\tilde{k}(sa,tb) = \tilde{k}(sa,t) + \delta_{ab} k(s,t)$



Cost = O(| s | | t |)

(but, product)

Gap-weighted Subsequence Kernel I

- Feature space = indexed by the strings of lengths p (m^p dim).
- Feature vector = frequency of gap-weighted substrings

$$|\Sigma| = m, \quad u \in \Sigma^p, \quad 0 < \lambda \leqq 1$$

$$\phi_u^p(s) = \sum_{\vec{i} : u = s[\vec{i}]} \lambda^{\ell(\vec{i})} \qquad \text{where} \quad \ell(\vec{i}) = |s[i_1 : i_r]| \quad \text{for} \quad \vec{i} = [i_1, \ldots, i_r]$$

C T G A C T G
u = CAT

$\Rightarrow \vec{i} = [1,4,6] \Rightarrow \ell(\vec{i}) = 6$

Discounts matching with gaps

- Feature map $\Phi : \Sigma^* \to H \cong \mathbf{R}^{m^p}, \qquad \Phi^p(s) = \left(\phi_u^p(s)\right)_{u \in \Sigma^p}$

$$K_p(s,t) = \sum_{u \in \Sigma^p} \phi_u^p(s)\phi_u^p(t) = \left\langle \Phi^p(s), \Phi^p(t) \right\rangle_H$$

Gap-weighted Subsequence Kernel II

– Example

s: ATGC
t : AGCT
p = 2

	AT	AG	AC	TG	TC	GC	GT	CT
Φ(s)	λ^2	λ^3	λ^4	λ^2	λ^3	λ^2	0	0
Φ(t)	λ^4	λ^2	λ^3	λ^4	0	λ^2	λ^3	λ^2

$$k(s,t) = \lambda^4 + \lambda^5 + 2\lambda^6 + \lambda^7$$

– Recursion is possible. Computational cost = O(p | s | | t |)

– Normalization is preferred. $$\tilde{k}_p(s,t) = \frac{k_p(s,t)}{\sqrt{k_p(s,s)k_p(t,t)}}$$

– For details, Lohdi et al. (JMLR, 2002), Rousu & Shawe-Taylor (2004).

Other Kernels for Strings

- Fisher kernel (Jaakkola & Haussler,1999)

  Based on the Fisher score function

$$k(x,y) = \frac{\partial \log p(x \mid \theta_*)}{\partial \theta} I(\theta_*)^{-1} \frac{\partial \log p(y \mid \theta_*)}{\partial \theta}$$

  $p(x \mid \theta_*)$: p.d.f. generating x and y

  HMM is used for string data.

- Mismatch kernel (Leslie et al. 2003)
  - Matching substrings of length *p*, but allows at most *r* mismatches.
  - Efficient computation by mismatch tree

    Computational cost for $K(s,t)$ = $O(p^{r+1}m^r(|s|+|t|))$

    $(m = |\Sigma|)$

Marginalized Kernel I

– Probabilistic model with hidden variables

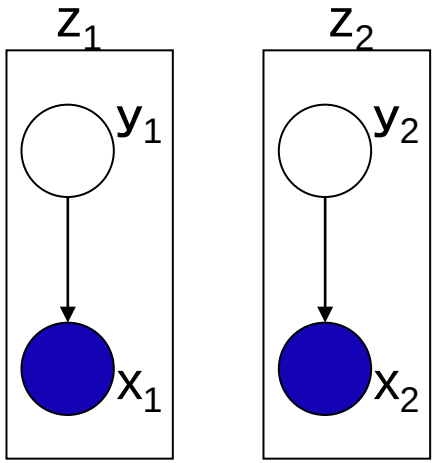
$z = (x, y)$

x : observable variable

y : unobservable (hidden variable)

$p(x,y)$: probability density function of (x, y)

$k_z(z_1, z_2)$: positive definite kernel for the joint variable



Assumption: kernel for z is easy to model.

$$k(x_1, x_2) = \sum_{y_1} \sum_{y_2} p(y_1 \mid x_1)\, p(y_2 \mid x_2)\, k_z((x_1, y_1), (x_2, y_2))$$

All hidden states

Marginalized Kernel II

- Example

	unknown									unknown								
y_1	1	2	2	1	2	2	1	2	2	y_2	1	2	2	1	2	2	1	exon / intron
x_1	A	C	G	G	T	T	C	A	A	x_2	A	C	C	G	T	A	C	DNA
	known										known							

  - Assume we have estimated $p(x, y)$ with hidden Markov model (HMM)
  - $$k_z(z_1, z_2) = \frac{1}{|z_1||z_2|} \sum_{i=1,2} \sum_{a \in \{A,T,G,C\}} C_{ai}(z_1) C_{ai}(z_2)$$

$C_{ai}(z)$: count of (a, i)

1 A	1 C	1 G	1 T	2 A	2 C	2 G	2 T
1	1	1	0	2	1	1	2

1 A	1 C	1 G	1 T	2 A	2 C	2 G	2 T
1	1	1	0	1	2	0	1

  - Marginalized kernel

$$k(x_1, x_2) = \sum_{y_1} \sum_{y_2} p(y_1 \mid x_1)\, p(y_2 \mid x_2)\, k_z(z_1, z_2)$$

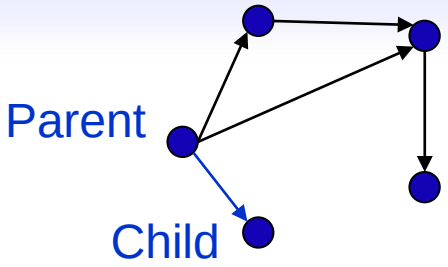
computable by HMM

- Kernels for string data
- Kernels for trees and graphs
- Kernels for probability measures

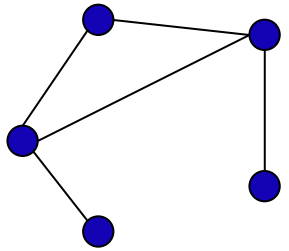
Graph and Tree

- Graph
  - V: finite set of nodes（vertex）.
  - E: set of edges, subset of V x V.
  - Directed graph

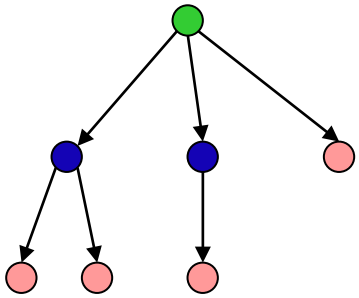
    Orient a from b if $(a,b) \in E$
    - Parents of $a = \{ b \in V \mid (b,a) \in E \}$
    - Children of $a = \{ b \in V \mid (a,b) \in E \}$
  - Undirected graph: forget the directions



Directed graph


Undirected graph

- Tree（directed rooted tree）
  - Directed connected graph such that there is a root with no parents and each of other nodes has only one parent.
  - Leaf: nods with no children.


Tree

Tree Kernel

– Tree kernel: kernel defined on the set of all trees

$$\Phi: \quad \text{tree } T \quad \mapsto \quad \Phi(T) \quad \in \quad H$$

– Examples

Typically, defined by matching of subtrees.

- All-subtrees kernel

$$k(T_1, T_2) = \sum_{S\text{: tree}} \phi_S(T_1)\phi_S(T_2)$$

$$\phi_S(T) = \begin{cases} 1 & S \text{ is a subtree of } T \\ 0 & \text{otherewise} \end{cases}$$

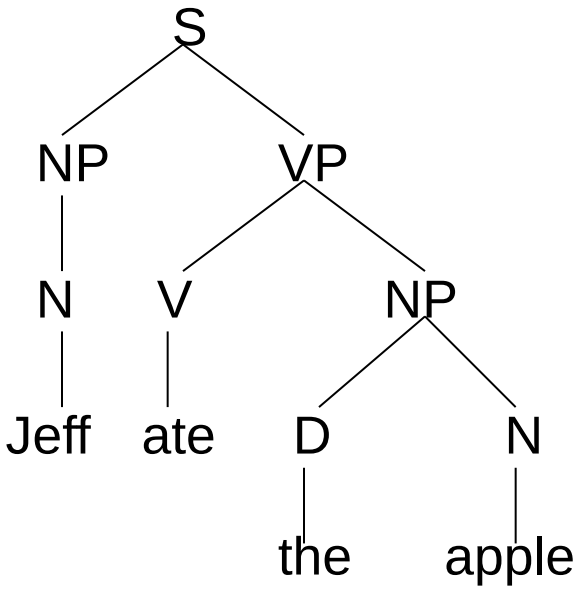
- Recursion is possible. Computational cost = $O(|T_1|\,|T_2|)$

– See Collins & Duffy (2002, NIPS) for details.

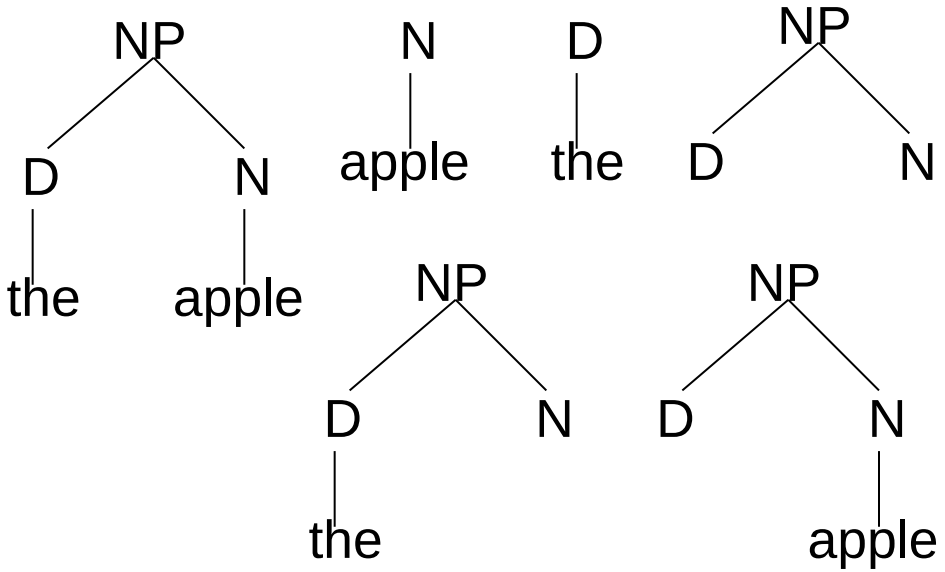
- Application to natural language processing

  Parsing of sentences

  Jeff ate the apple.



Examples of subtree



  – *e.g.* SVM to learn the map: sentence → parsing tree
    (Collins & Duffy 2002, NIPS)

Graph Kernel

- $K(G_1, G_2)$: similarity measure for two graphs
- Labeled graph

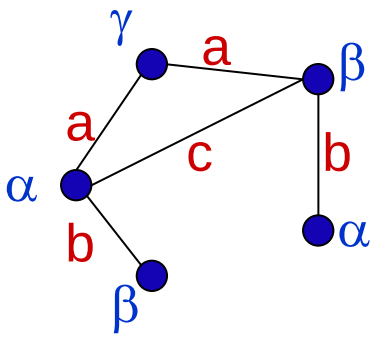
  Each node and edge has a label.

  L_V, L_E: label sets

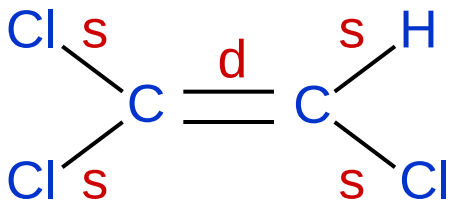
  Labeled graph $G = (V, E, h_V, h_E)$

  V: nodes, E : edges,

  $h_V : V \rightarrow L_V$, $h_E : V \rightarrow L_E$. Label mapping



- Applications
  - Chemical compounds. Toxicology. Medicine.
  - Natural language processing



Marginalized Graph Kernel I

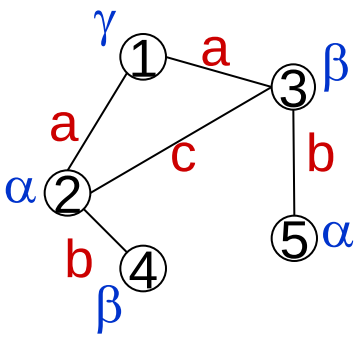
(Kashima et al., 2003; Mahé, et al., 2004)

– Labeled sequence

s: $v_1\ v_2\ v_3\ v_5\ v_3 \cdots$

→ $H(s) = h(v_1)h(e_{12})h(v_2)h(e_{23})h(v_3)h(v_{35})h(v_5)\cdots$

$= \gamma\ a\ \alpha\ c\ \beta\ b\ \alpha\ b\ \beta \cdots$



– Random walk model

- Transition

$$p(v_j \mid v_i) = \begin{cases} 1/|\text{neighbor of } i| & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

- Probability of sequence

$$p(s) = p(v_1)\,p(v_2 \mid v_1)\,p(v_3 \mid v_2)\,p(v_5 \mid v_3)\,p(v_5 \mid v_3)\cdots$$

Occurrence probability given by the random walk.

Marginalized Graph Kernel II

– Positive definite kernel for the sequences

$$K_L : L^* \times L^*, \qquad K_L(H_1, H_2) = \begin{cases} 1 & (H_1 = H_2) \\ 0 & (H_1 \neq H_2) \end{cases}$$

– Marginalized graph kernel

$G_1 = (V_1, E_1, h_1), \quad G_2 = (V_2, E_2, h_2)$

$$K(G_1, G_2) = \sum_{\substack{s \in V_1^* \\ t \in V_2^*}} p_1(s)\, p_2(t) K_L(H_1(s), H_2(t))$$

$H_1(s)$, $H_2(t)$: labels given by h_1, h_2
V_1*, V_2* : the sequences generated by V_1, V_2

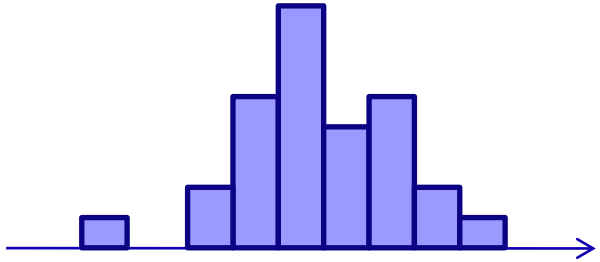
- $K(G_1,G_2)$ is the probability of generating the same label sequence.
- An example of marginalized kernel.

- Kernels for string data
- Kernels for trees and graphs
- Kernels for probability measures

Kernels for Histograms and Probabilities

- Histograms / positive measures

$H(i)$ = # data within i-th bin



- Probabilities

$$p(i) = \frac{\#\text{ data within } i\text{-th bin}}{\text{total number of data}}$$

$$p(x): \quad \int p(x)dx = 1, \quad p(x) \geq 0.$$

- Kernels for positive measures / probabilities: ‘Vector’ is sufficient?

Vector (function) interpretation and application of ordinary kernels, e.g., Gaussian, is possible, but not reasonable.

- Value is not shift invariant.
- Subtraction may not be reasonable.

Kernel by Jensen-Shannon Divergence

(Hein and Bousquet. 2005)

– Entropy function: $h(f) = -\int f(x) \log f(x) dx$

Fact: entropy function is negative definite on the set

$$\{f : \Omega \to \mathbf{R} \mid f(x) \geq 0, h(f) < \infty\}.$$

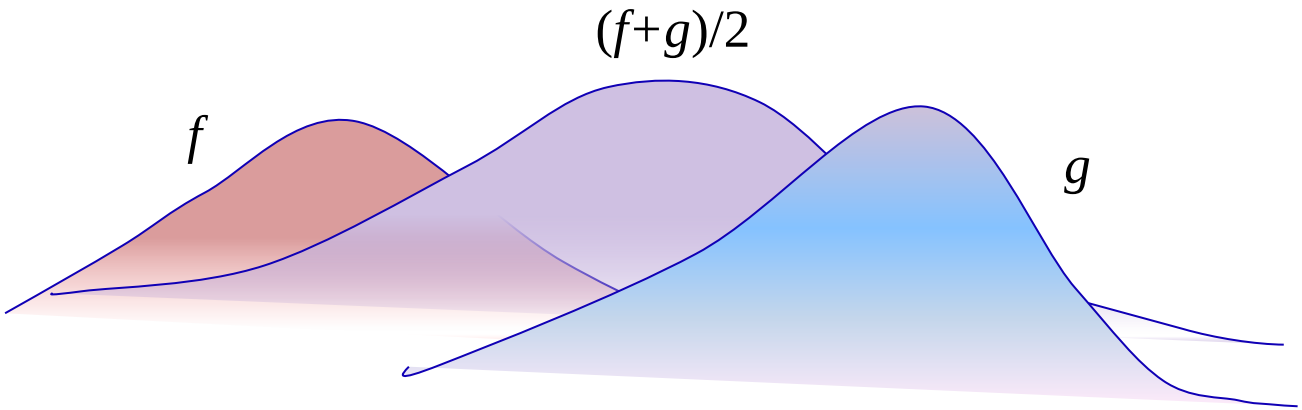
∵) Use the fact that $\psi(a,b) = -(a+b)\log(a+b)$ is negative definite on $\mathbf{R}_{>0}$ [Exercise].

– Jensen-Shannon divergence:

$$D(f,g) = h\left(\frac{f+g}{2}\right) - \frac{h(f)+h(g)}{2}$$: negative definite kernel

– Jensen-Shannon divergence kernel: (Recall Schoenberg's theorem)

$$k(f,g) = \exp\left\{-h\left(\frac{f+g}{2}\right) + \frac{h(f)+h(g)}{2}\right\}.$$



$$D(f,g) = h\left(\frac{f+g}{2}\right) - \frac{h(f)+h(g)}{2}$$

Inverse Generalized Variance Kernel

(Cuturi, Fukumizu, Vert. 2005)

– $P_2(\mathbf{R}^m)$: the probabilities on $\mathbf{R}^m$ with finite covariance.

– Define

$$\Sigma(P) = E_P[xx^T] - E_P[x]E_P[x]^T \qquad \text{(covariance matrix)}$$

– Inverse generalized variance kernel on $P_2(\mathbf{R}^m)$:

$$k(P,Q) = \frac{1}{\det\left[\Sigma\left(\frac{P+Q}{2}\right) + \varepsilon I_m\right]}$$

- Fact: IGV kernel is positive definite.
- For probabilities on general measurable spaces, the covariance on RKHS can be used for defining IGV kernel.

Proof of positive definiteness.

For any $y \in \mathbf{R}^m$,, $P_1,..., P_n \in P_2(\mathbf{R}^m)$, $c_1,...,c_n \in \mathbf{R}$ such that $\sum_{i=1}^n c_i = 0$,

$$\sum_{i,j=1}^n c_i c_j y^T \Sigma\left(\frac{P_i + P_j}{2}\right) y = \sum_{i,j=1}^n c_i c_j y^T \int xx^T d\left(\frac{P_i + P_j}{2}\right)(x) y$$

$$- \sum_{i,j=1}^n c_i c_j y^T \int x d\left(\frac{P_i + P_j}{2}\right)(x) \int x^T d\left(\frac{P_i + P_j}{2}\right)(x) y$$

$$= \frac{1}{2}\sum_{i,j=1}^n c_i c_j E_{P_i}\left[(y^T x)^2\right] + \frac{1}{2}\sum_{i,j=1}^n c_i c_j E_{P_j}\left[(y^T x)^2\right]$$

$$- \frac{1}{2}\sum_{i=1}^n c_i c_j \left(E_{P_i}[y^T x]^2 + 2E_{P_i}[y^T x]E_{P_j}[y^T x] + E_{P_j}[y^T x]^2\right)^2$$

$$= -\left(\sum_{i=1}^n c_i E_{P_i}[y^T x]\right)^2 \le 0$$

Thus, $(P,Q) \mapsto y^T \Sigma\left(\frac{P+Q}{2}\right) y$ is negative definite for any y. From Shoenberg's theorem

$$k(P,Q) = \int \exp\left(-\frac{1}{2} y^T \Sigma\left(\frac{P+Q}{2}\right) y\right) dy$$

is positive definite.

Summary of Section 6

- Kernels for structured data
  - Various kernels have been proposed based on the structure.
  - Once a kernel is defined, all the kernel methods can be applied.
  - String, tree, graph → many kernels are based on counting common substructures.
    - Problem: computational cost
      - O(|s| |t|) for computing K(s,t) is not feasible for long s and t.
      - Note that Gram matrix has N(N-1)/2 elements.
      - *e.g.* SCOP database: length ~ 10^2, N ~ 10^3
  - Other examples.
    - Fisher kernel
    - Kernel for histograms.

References

- Lodhi, H., C. Saunders, J. Shawe-Taylor, N. Cristianini, C. Watkins. (2002) Text Classification using String Kernels. *J. Machine Learning Research*, 2 (Feb): 419-444.
- Leslie, C., E. Eskin, A. Cohen, J. Weston and W. S. Noble. (2003) Mismatch string kernels for SVM protein classification. *Advances in Neural Information Processing Systems 15*, pp. 1441-1448.
- Rousu, J., and J. Shawe-Taylor. (2004) Efficient computation of gap-weighted string kernels on large alphabets. *Proc. PASCAL Workshop Learning Methods for Text Understanding and Mining*.
- Dan Gusfield. *Algorithms on Strings, Trees, and Sequences.* Cambridge Univ. Press. 1997.
- Jaakkola, T.S. and D. Haussler. (1999) Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems 11*. pp.487-493.
- Collins, M. & N. Duffy. (2002) Convolution Kernels for Natural Language. *Advances in Neural Information Processing Systems 14.*
- Tsuda, K., T. Kin, and K. Asai. (2002) Marginalized kernels for biological sequences. *Bioinformatics, 18.* S268-S275.

- Kashima, H., K. Tsuda and A. Inokuchi. (2003) Marginalized Kernels Between Labeled Graphs. *Proc. 20th Intern. Conf. Machine Learning* (*ICML2003*).
- Mahé, P., N. Ueda, T. Akutsu, J.-L. Perret and J.-P. Vert. (2004) Extensions of marginalized graph kernels. *Proc. 21th Intern. Conf. Machine Learning* (*ICML 2004*), p.552-559.
- Schölkopf, B., K. Tsuda, J-P. Vert (Editor) Kernel Methods in Computational Biology. Bradford Books. 2004.
- Hein, M. and O. Bousquet. Hilbertian metrics and positive definite kernels on probability measures. *Proc. AISTATS 2005*, 136-143 (2005).
- Cuturi, M., K. Fukumizu, and J-P. Vert. Semigroup Kernels on Measures, *Journal of Machine Learning Research*, vol.6, pp.1169-1198 (2005).