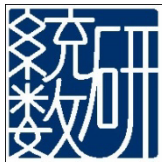
５．構造化データに対するカーネル

正定値カーネルによるデータ解析
― カーネル法の基礎と展開 ―

福水健次
統計数理研究所／総合研究大学院大学

統計数理研究所　公開講座
2011年1月13,14日

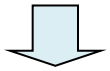


概要

- ストリングデータに対するカーネル
- グラフデータに対するカーネル

構造化データの処理

正定値カーネルは，任意の集合上に定義可能
→ ベクトルデータなくてもよい



– 構造化 (non-vectorial) データ
  - ストリング(文字列): アルファベットの任意長の系列.
  - ツリー(木).
  - グラフデータ，ネットワークデータ

– 正定値カーネルさえ定義されれば，さまざまなカーネル法が非ベクトルデータに適用可能（カーネルPCA, SVM, カーネルリッジ回帰,etc）

– カーネル法のアルゴリズムで計算すべきはグラム行列 $k(x_i, x_j)$

– 本講座ではカーネルの定義法のみを述べる．具体的な応用は，Schölkopf et al (2004) (bioinformatics), Joachims (2002) (text) などを参照.

概要

- ストリングデータに対するカーネル
- グラフデータに対するカーネル

ストリング

- ストリング
  - アルファベット Σ : 有限集合
  - ストリング : Σ の要素の有限長の列
    - 例） $\Sigma = \{ a, b, c, d, \ldots, z \}$
      ストリング cat, head, computer, xyydyaa,…
  - Σ^p : 長さ p のストリング全体
  - Σ^* : 任意の長さのストリング全体 $\Sigma^* = \bigcup_{p=0}^{\infty} \Sigma^p$

    注） $\Sigma^0 = \{\varepsilon\}$: 空ストリング
  - 記号法

    s : $s_1 s_2 \ldots s_n$ ストリングに対し

    $|s|$ ・・・ ストリング s の長さ = n

    $s[i:j]$ ・・・ $s_i \ldots s_j$ という s の部分列

    s, t に対し結合 $st = s_1 s_2 \ldots s_n t_1 t_2 \ldots t_m$

ストリングカーネル

- ストリングカーネル
  - Σ^* 上定義された正定値カーネル・・・ 2つのストリング s, t の類似度
  - 一致する部分列を数え上げるタイプが多い
  - 効率的な計算の工夫が重要 ・・・ 再帰式（漸化式)など
    Dynamical Programming（DP)

- 典型的な応用先
  - 自然言語処理
    - 文字列: $\Sigma = \{a, b, c, \ldots, z\}$
    - 単語列: $\Sigma = \{$単語全体$\}$
  - ゲノム解析
    - ゲノム: $\Sigma = \{A, T, G, C\}$
    - タンパク質: $\Sigma = \{$アミノ酸$\}$（20種類）

ストリングカーネルの応用

- タンパク質の構造予測
  - アミノ酸配列： Σ = 20種のアミノ酸

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

配列 → 立体構造のクラスや進化的関係を予測

  - 立体構造は，進化的，機能的に重要な情報を持つ．
  - データベース
    - SCOP（Structural Classification of Proteins）
      http://scop.mrc-lmb.cam.ac.uk/scop/

p-スペクトラムカーネル

– 長さ p の部分列の出現回数を特徴ベクトルとする

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

$$\phi_u^p(s) = \left|\{(w_1, w_2) \in \Sigma^* \times \Sigma^* \mid s = w_1 u w_2\}\right| \quad \cdots \quad s \text{ の中の } u \text{ の出現回数}$$

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

特徴空間: 長さ p の列全体 $\cdots$ m^p 次元

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

s = “statistics”　　t = “pastapistan”

3-スペクトラム

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

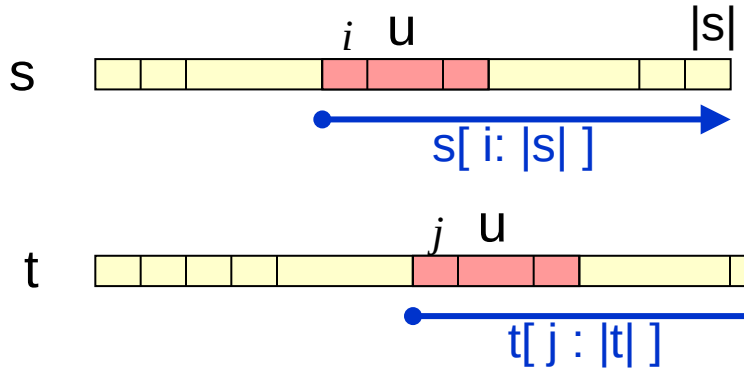
t: pas, ast, sta, tap, api, pis, ist, sta, tan

	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

$K_3(s, t) = 1 \cdot 2 + 1 \cdot 1 = 3$

- p-スペクトラムカーネルの計算法

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



– 直接的な計算

部分列 u を，途中から始まる
部分列 suffix（接尾辞）の
先頭（prefix)と思う

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

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

計算量 ＝ O(p| s || t |)

- p-スペクトラムカーネルの計算法(II)

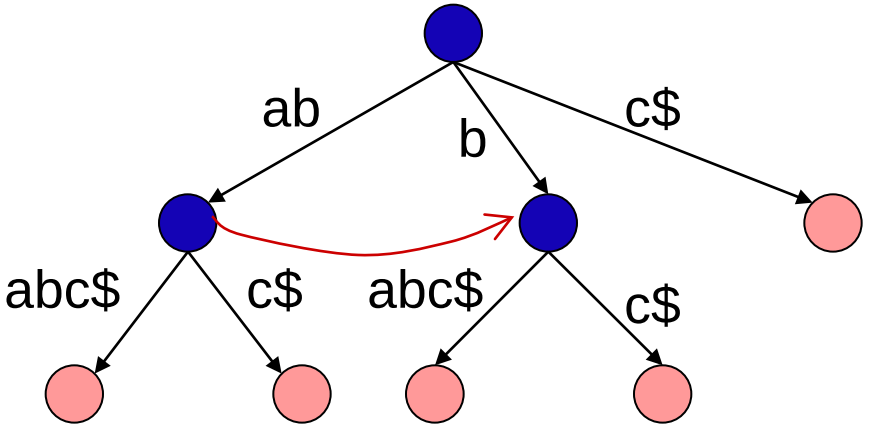
  実は，|s|+|t| に対して線形時間 O(p(| s | + | t |)) で計算する方法がある

  – Suffix Tree

    ストリングのすべての suffix を木構造で効率的に表すアルゴリズム

    例）ababc

ababc
babc
abc
bc
c



  – Gram行列 $(k(X_i, X_j))$ の計算は $O\left(p\sum_{i=1}^{N}|X_i|\right)$

  – 詳しくは Vishwanathan & Smola 03, Gusfield 97.

All-subsequences Kernel

– すべての長さの部分列（gapも許す）の出現回数を特徴ベクトルとする

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

$$\phi_u(s) = \left|\{\vec{i} \mid s[\vec{i}] = u\}\right|$$

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

$$s[\vec{i}] = s_{i_1} s_{i_2} s_{i_3} \cdots s_{i_\ell}$$

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

特徴空間 ＝任意長のストリングを添字に持つベクトル空間（無限次元）

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

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

- gapを許す比較

ATGACTAC
CATGCGATT

ATGACTAC
CATG CGATT

u = ATGCA

- 例

s: ATG
t : AGC

$K(s,t) = 4$

ε: 空ストリング

	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

- All-subsequences kernel の計算

再帰的な式： 過去に計算した結果を利用

初期条件

$k(s, \varepsilon) = k(t, \varepsilon) = 1$ （任意の s, t ）　　ε ： 空ストリング

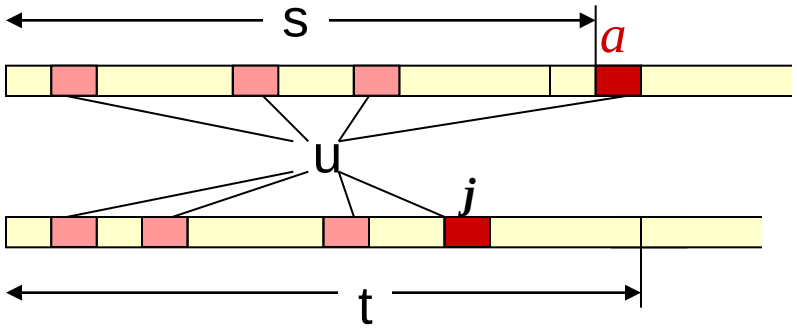
$k(s, t)$ まで求まっているとして $k(sa, t) = ?$

$k(sa, t)$

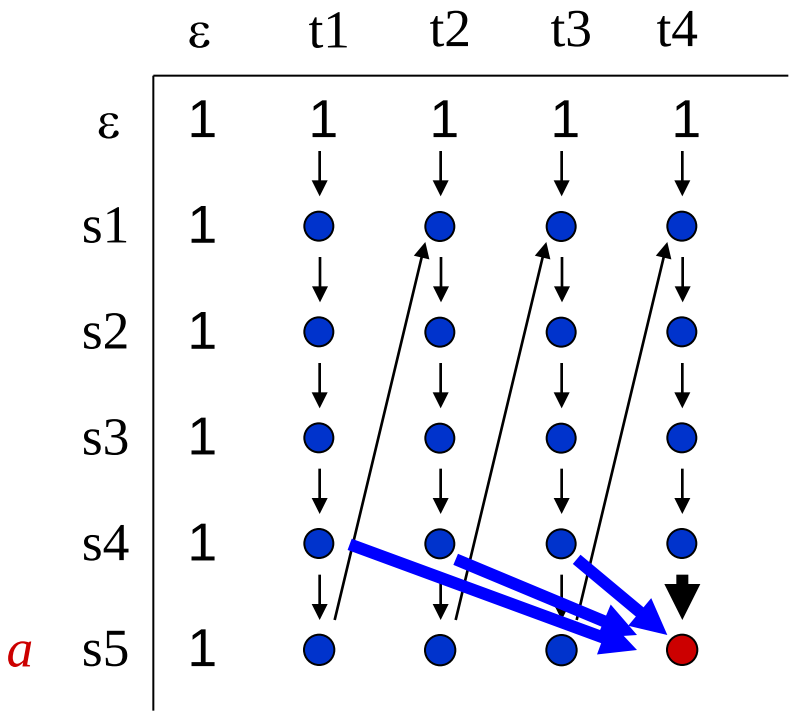
＝（ s 内 と t 内 での一致によるもの）

＋ （a を含む一致）

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



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



計算量 = O(| s | | t |^2)

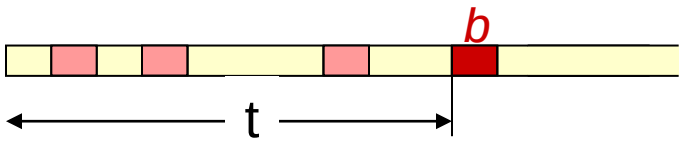
- All-subsequences kernel の計算(II)
  - 計算の効率化

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

t 回の計算をやりたくない

$$\| \quad \tilde{k}(sa,t)$$

とおくと



t に関する再帰式

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

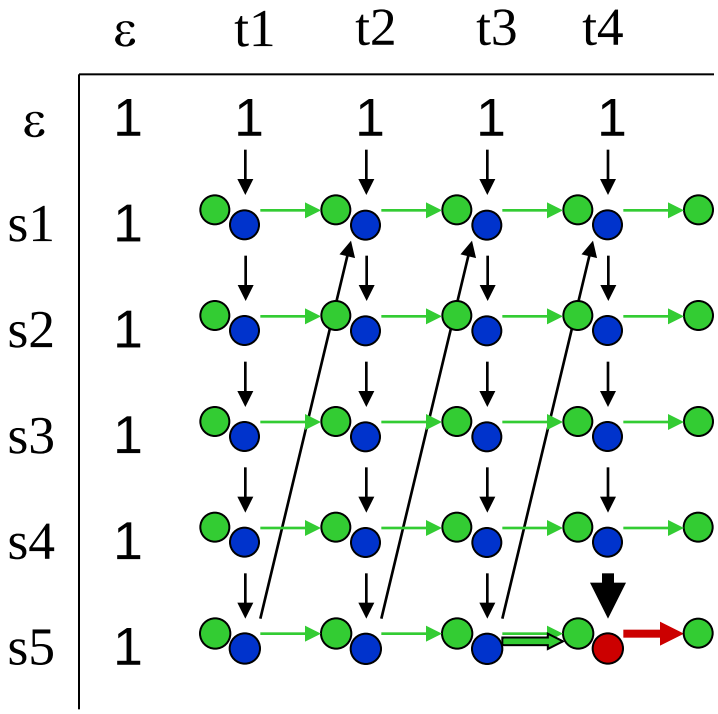
1≦*j* ≦ |*t*| の場合　　*j* = |*tb*| の場合

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

$$\begin{cases} k(sa,t) = k(s,t) + \tilde{k}(sa,t) & (s \text{ についての再帰式}) \\ \tilde{k}(sa,tb) = \tilde{k}(sa,t) + \delta_{ab} k(s,t) & (t \text{ についての再帰式}) \end{cases}$$

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

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



計算量 = O(| s | | t |)

Gap-weighted subsequence kernel

– Gap に対してペナルティをつけた特徴ベクトル

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

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

ただし $\vec{i} = [i_1, \ldots, i_r]$ に対し

$\ell(\vec{i}) = | s[i_1 : i_r] |$

C T G A C T G
u = CAT
➡ $\vec{i} = [1,4,6]$ ➡ $\ell(\vec{i}) = 6$

gap が多い一致は割り引く

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

特徴空間: 長さ p の列全体 ・・・ m^p 次元

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

– Gap-weighted subsequences kernel の例

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

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

– 効率的なアルゴリズムとして再帰式によるものがある
計算量 ＝ O(p | s | | t |)

– 正規化が行われることが多い $\tilde{k}_p(s,t) = \frac{k_p(s,t)}{\sqrt{k_p(s,s)k_p(t,t)}}$

– 詳しくは Lohdi et al. (JMLR, 2002)
Rousu & Shawe-Taylor (2004)

その他のストリングカーネル

- Fisherカーネル (Jaakkola & Haussler,1999)

  Fisher スコア関数に基づく

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

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

  ストリングデータの場合はHMMでモデル化し，推定しておく.

- Mismatch kernel (Leslie et al. 2003)
  - 長さ*p*の部分列を, *r* 個のミスマッチを許してマッチングする.
  - ミスマッチツリーによる効率的計算が可能

    K(s,t) の計算量 = $O(p^{r+1}m^r(|s|+|t|))$ $(m = |\Sigma|)$

周辺化カーネル

- 確率モデルにもとづくカーネル設計

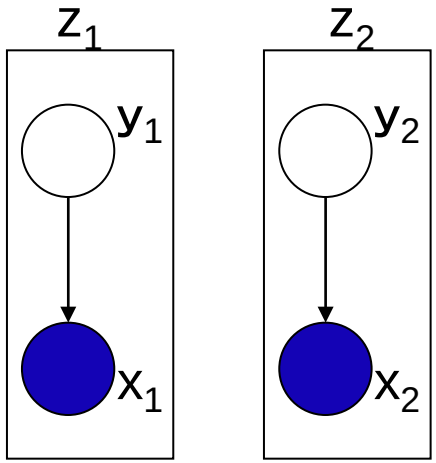
$z = (x, y)$

x：観測される変数

y：観測されない隠れ変数
（データを生成する構造）

$p(x,y)$： (x, y) に対する確率モデル

$k_z(z_1, z_2)$ ： $z = (x, y)$ に対する正定値カーネル



仮定：(x, y)のモデル化とカーネル定義のほうが容易

y_1, y_2 の状態全体

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

- 例

unknown

y　1 1 1 . . . 1 1 1 2 2 2 . . . 2 2 2 1 1 1　exon / intron

x　**A A G . . . G T G G T A . . . C A G A C A**　DNA

known

– $p(x, y)$ は隠れマルコフモデル（HMM）によって記述済み

– $k_z(z_1, z_2) = \frac{1}{|z_1||z_2|} \sum_{i=1,2} \sum_{a \in \{A,T,G,C\}} C_{ai}(z_1) C_{ai}(z_2)$

$C_{ai}(z)$: (a, i) のカウント

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

– 周辺化カーネル（Marginalized kernel）

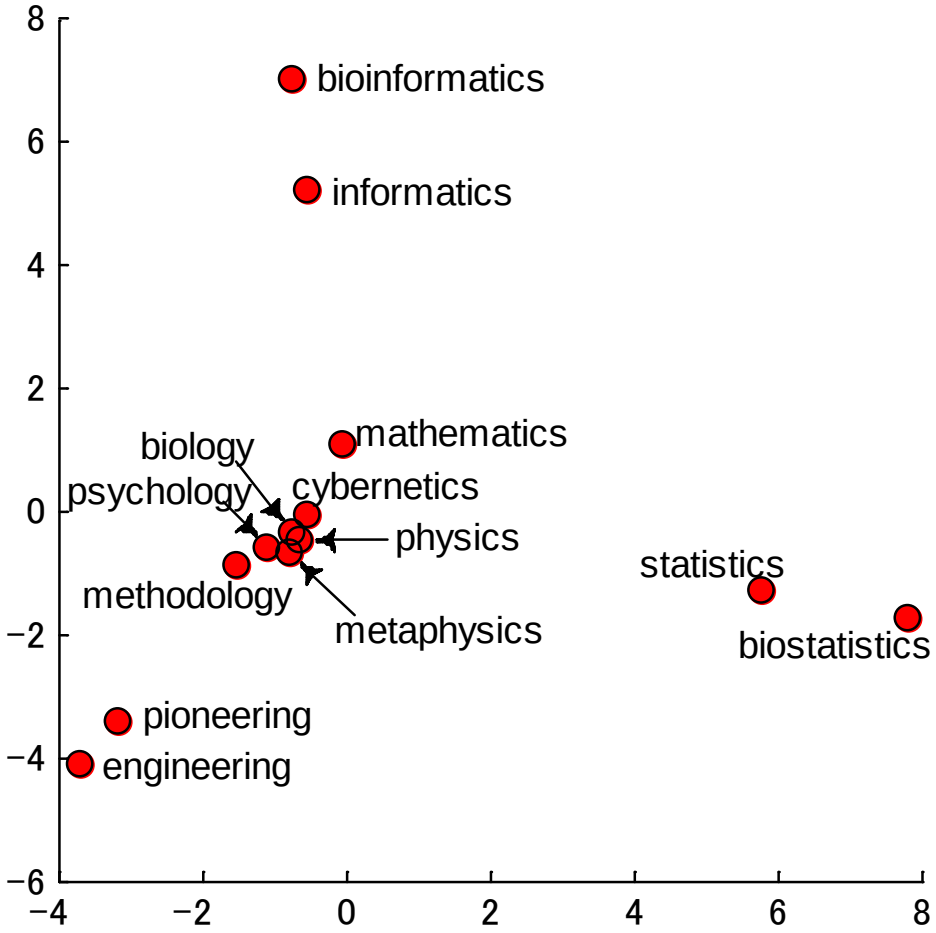
$$k(x_1, x_2) = \sum_{y_1} \sum_{y_2} p(y_1 \mid x_1) p(y_2 \mid x_2) k_z(z_1, z_2)$$

HMMから計算

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

Example: Kernel PCA for “words”

- 13 words
- Gap-weighted subsequence kernel of length 4.

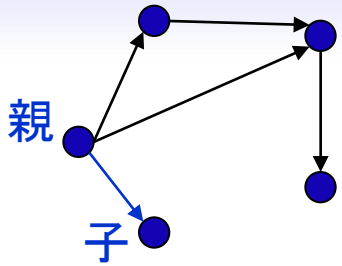


概要

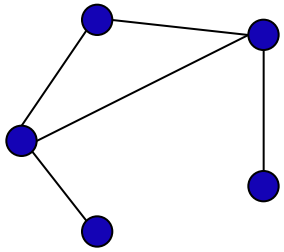
- ストリングデータに対するカーネル
- グラフデータに対するカーネル

グラフとツリー

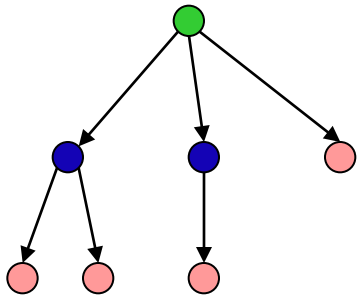
- グラフ
  - V： ノード（node, vertex）・・・ 有限集合
  - E： エッジ（edge）・・・ V x V の部分集合
  - 有向グラフ： E の向きを考えたもの
    (a, b) ∈ E のとき，aからbへ矢印を描く
    - ノード a の親： (b,a) ∈E なる b
    - ノード a の子： (a,b) ∈E なる b
  - 無向グラフ： E の向きを忘れたもの



有向グラフ


無向グラフ

- ツリー（directed rooted tree）
  - 連結した有向グラフで，親の無いルートノードが存在し，他の各ノードは親を1個だけ持つもの
  - リーフ：ツリーの中で子の無いノード


ツリー

ツリーカーネル

– ツリー全体の集合上に定義された正定値カーネル

$$\Phi: \quad \text{ツリー } T \quad \mapsto \quad \Phi(T) \quad \in \quad H \text{ 特徴空間（ベクトル空間）}$$

– 代表的な例

サブツリーの一致によりカーネルを定義する

- All-subtrees kernel

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

$$\phi_S(T) = \begin{cases} 1 & T \text{ が } S \text{ をサブツリーとして含む} \\ 0 & T \text{ が } S \text{ をサブツリーとして含まない} \end{cases}$$

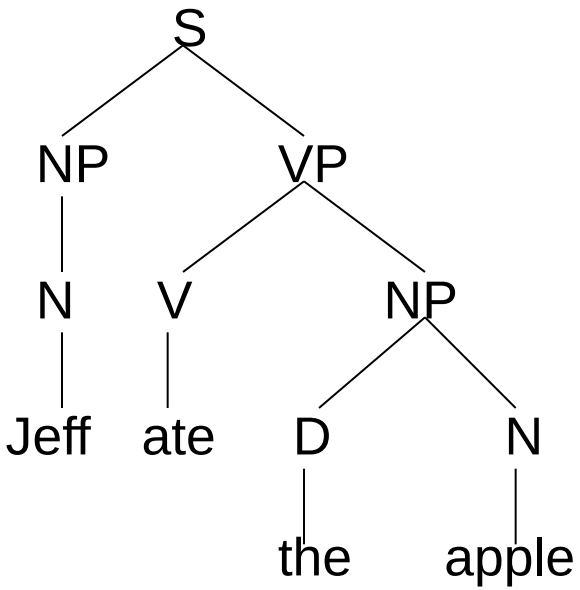
- 再帰式で計算可能.　計算量 ＝ $O(|T_1|\,|T_2|)$

– 詳細は Collins & Duffy (2002, NIPS) などを参照

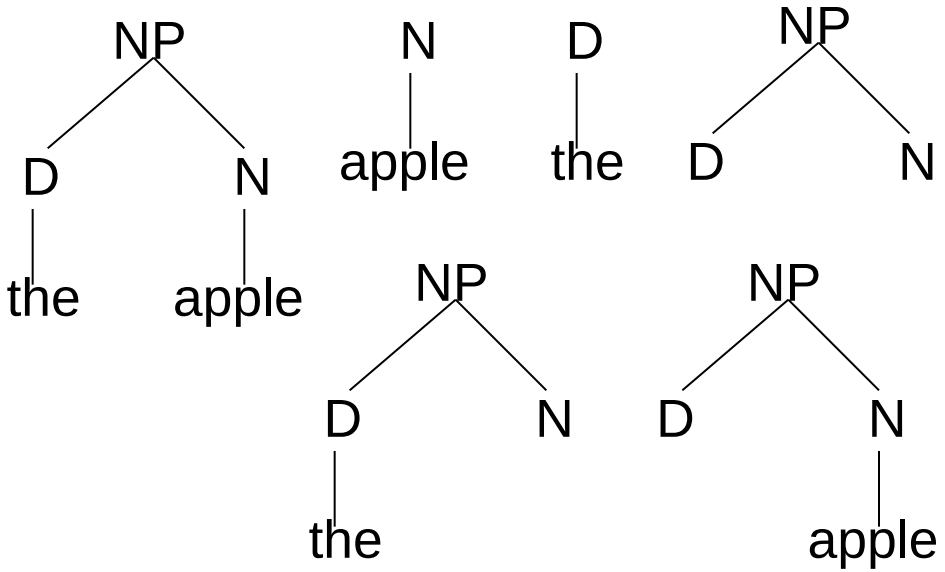
– 自然言語処理への応用

構文解析

Jeff ate the apple.



サブツリーの例



– *e.g.* SVMによって文から構文解析木への写像を学習(構造化出力) Collins & Duffy 2002, NIPS

グラフカーネル

- グラフ上に定義された正定値カーネル
  - グラフとグラフの類似度を測る.
  - ラベル付グラフ

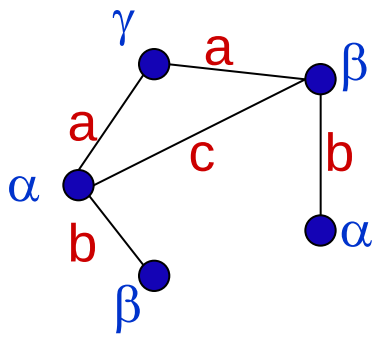
    ノードとエッジにラベルがついている.

    L: ラベルの集合（有限集合）

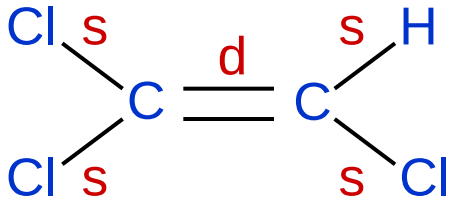
    ラベル付グラフ　G = (V, E, h)

    V: ノード, E : エッジ,

    h : V ∪ E → L　ラベル付けの写像



  - 応用
    - 化合物の毒性予測
    - 自然言語処理

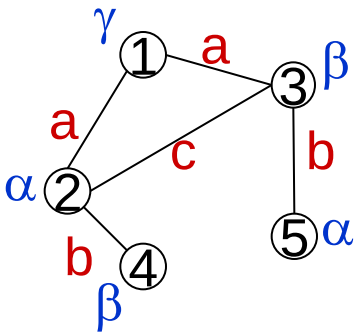


- Marginalized graph kernel
  - 系列のラベル

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

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

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



  - 系列の確率 – ランダムウォーク
    - ノード間の遷移確率

$$p(v_j \mid v_i) = \begin{cases} 1/(i\text{の隣接ノードの数}) & (i,j) \in E \\ 0 & (i,j) \notin E \end{cases}$$

    - 系列の確率

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

グラフ上のランダムウォークにより生じる系列の確率

– ラベル系列に対するカーネル

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

– 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, H_2 : それぞれ h_1, h_2 から決まるラベル関数
V_1^*, V_2^* : それぞれ V_1, V_2 をアルファベットとする系列全体

- ランダムウォークにおいて，同じパスが生じる確率
- Marginalized kernel のひとつとみなせる

– 詳しくは，Kashima et al. (2003), Mahé, et al. (2004)

構造化データ上のカーネルの計算量の問題

- 計算量
  - SCOPデータベース： 配列の長さ～数百， 配列データの数～数千
  - p-spectrum, mismatch kernel
    - k(x,y): O(C(|s|+|t|))
    - 全データ長に対して線形の計算量
  - All subsequences, gap-weighted kernel
    - k(x,y) の計算にかかる時間
      O(|s| |t|). サイズが大きくなると困難
    - Gram行列の計算は （データ数）^{2} のオーダー

セクション5のまとめ

- 構造化データのカーネル
  - 非ベクトルデータのベクトル化
  - 明示的なベクトル表示でカーネルを定義することが多い.
  - 効率的な計算法が重要
  - ストリング
    - p-spectral kernel, all-subsequences kernel, gap-weighted kernel, mismatch kernel…
  - ツリー，グラフ

参考文献

Collins, M. and N. Duffy. (2002) Convolution Kernels for Natural Language. *Advances in Neural Information Processing Systems 14*.

Gusfield, D. *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.

Joachims, T. *Learning to Classify Text using Support Vector Machines*, Kluwer/Springer, 2002.

Kashima, H., K. Tsuda and A. Inokuchi. (2003) Marginalized Kernels Between Labeled Graphs. *Proc. 20th Intern. Conf. Machine Learning (ICML2003).*

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.

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.

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.

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*.

Schölkopf, B., K. Tsuda, J-P. Vert (Editor) *Kernel Methods in Computational Biology*. Bradford Books. 2004.

Tsuda, K., T. Kin, and K. Asai. (2002) Marginalized kernels for biological sequences. *Bioinformatics, 18.* S268-S275.

Vishwanathan, S.V.N. and A.J. Smola (2003) Fast Kernels for String and Tree Matching. *Advances in Neural Information Processing Systems 15*, 569-576., MIT Press.