カーネル法の応用

松井知子
統計数理研究所

2006年7月6･7日
公開講座「カーネル法の最前線—SVM、非線形データ解、構造化データ— 」

実問題を扱う

- 課題：いかにデータに語らせるか
  - データに潜む構造を扱う。
  - データの本質をつかむ。

- カーネル法によるアプローチ
  - 構造化データを扱う。

    例）ベクトル　⇒　構造化オブジェクト
  - データ x が与えられた時のラベル y の条件付分布をうまく推定する。

    例）Penalized Logistic Regression Machine (PLRM)

構造化データを扱う

- 対象データ内の構造
  - 様々なString/Tree/Graph kernel
    (基本的にConvolution kernelの考えを利用)
  - *P*-kernel, Fisher kernel (確率的な生成モデルを利用)
  - 例)テキスト・音声・画像、構文解析木、
    ゲノム・タンパク質　など
- 対象データ間の構造
  - Diffusion kernel
  - 例)タンパク質の相互作用、WWW、引用ネットワーク　など
- クラスデータの構造
  - Kernel conditional random field
  - 例)形態素解析(単語列⇒品詞列)、
    タンパク質の2次構造予測　など

話の構成

- 実問題を扱う
  - 構造化データを扱う
    - 対象データ内の構造
      - Convolution kernelの考えを利用
      - 確率的な生成モデルを利用
    - クラスデータの構造
  - 条件付分布を推定する
  - ソフトウエアの紹介

Convolution Kernelの考えを利用

- 対象データ x : ストリング、木、グラフなど
  - ストリングや木はグラフの特別な場合と考えられる。
- ストリング/木/グラフ上のカーネル関数を設計

$$k(s, t)=\langle \Phi(s), \Phi(t) \rangle$$

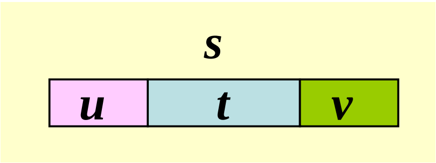
  - $k(s, t)$は半正定値性を満たし、s, tの類似度を表す。
  - s, tは大きさが異なる場合もある。
- 設計上の一般的な考え
  - 部分構造(部分列や部分木など)に分解する。
  - 一致する部分構造を数え上げて$\Phi(s)$を構成する。

基本は convolution kernel [Haussler 99]

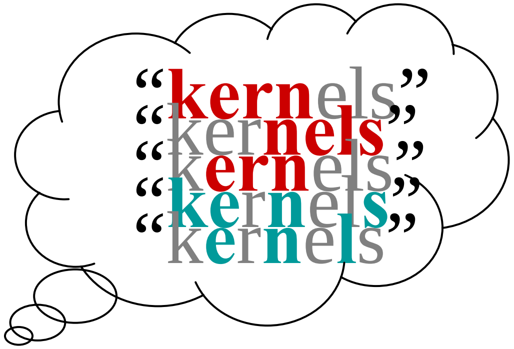
⇒効率的な計算の必要性
  - 動的計画法
  - 接尾辞木　など

ストリングと部分列

- ストリング
  - アルファベット Σ: $|\Sigma|$個のシンボルの有限集合
  - 長さnのストリング Σ^n
  - 全ストリング Σ^*: $\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$
  - ストリングsの部分ストリングt: $s=utv$ (u,v:ストリング)
    - $u=\varepsilon$の時、tは接頭辞
    - $v=\varepsilon$の時、tは接尾辞
    - 長さkの部分ストリング: k-gram
    - $S[i:j]$: $s_i,\ldots,s_j$の部分ストリング



- ストリングsの部分列u
  - インデックス$\mathbf{i}=(i_1,\ldots,i_{|u|})$が存在して、$1 \leqq i_1 < \cdot\cdot < i_{|u|} \leqq |s|$
  - $j=1,\ldots,|u|$ について、$u_j=s_{i_j}$
  - 部分列の長さ$l(\mathbf{i})$は、$l(\mathbf{i})=i_{|u|}-i_1+1$
- 例) s="kernels"
  - 部分ストリング: t="kern"(接頭辞), "nels"(接尾辞), "ern", …
  - 部分列: u="kens", "enl", …



典型的な応用先

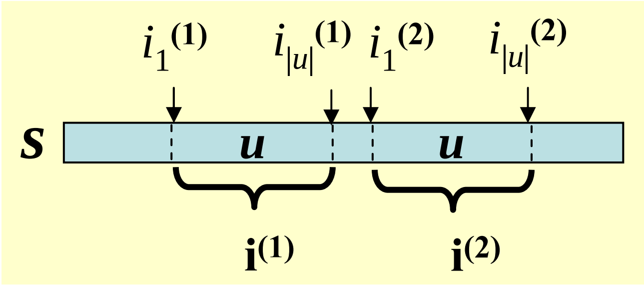
- 自然言語処理
  - 文字列:　$\Sigma=\{a, b, c, \ldots, z\}$
  - 単語列:　Σ={単語全体}
- ゲノム解析
  - ゲノム:　$\Sigma=\{A, T, G, C\}$
  - タンパク質:　Σ={アミノ酸全体}

String Kernels

- ギャップを許した部分列を扱う [Lodhi et al., 2002]
  - Gapped-weighted subsequences kernel、O($n|x||y|$)
- 不一致ペナルティを用いる [Leslie et al., 2002]
  - Mismatch string kernel、O($k^{m+1}l^m$ ($|x|+|y|$))
- 接尾辞木を利用する [Vishwanathan et al., 2002]
  - Suffix-tree-based spectrum kernel、O($|x|+|y|$)
- 局所アライメントをとる [Vert et al., 2004]
  - Local alignment kernel、O($n^3|x||y|$)

Gapped-Weighted Subsequences Kernel
[Lodhi et al, 2002]

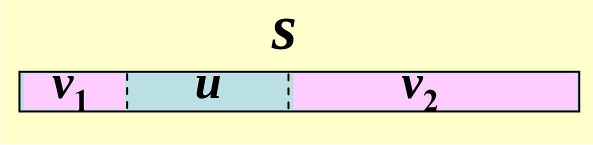
- 関連研究: All-sequences kernel
  - 全ての長さの部分列の出現回数を特徴ベクトルとする。
  - 任意長のストリングを座標とする特徴空間 *Fall*



$$\Phi : \Sigma^* \to F_{all} \cong R^{\infty}$$

$$\Phi(s) = (\phi_u(s))_{u \in \Sigma^*}, \quad \phi_u(s) = |\{\mathbf{i} \mid s[\mathbf{i}] = u\}|$$

- 関連研究:　*p*-Spectrum kernel
  - 長さpの部分ストリングuの出現回数を特徴ベクトルとする。
  - 長さpの部分ストリングuを座標とする特徴空間Fp



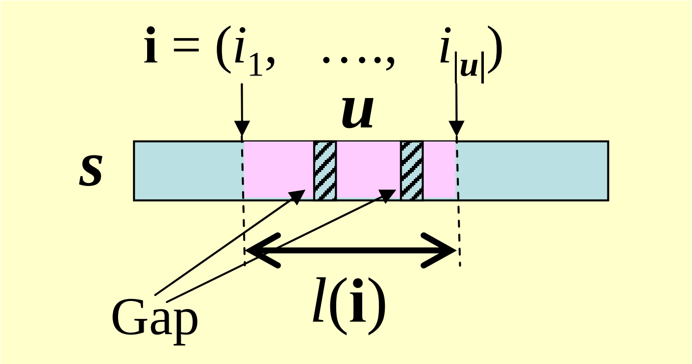
$$\Phi : \Sigma^* \to F_p \cong R^{|\Sigma|^p}$$

$$\Phi(s) = (\phi_u(s))_{u \in \Sigma^p}, \quad \phi_u(s) = |\{(v_1, v_2) : s = v_1 u v_2\}|$$

- Gapped-weighted subsequences kernel
  - ストリング中の長さ$l(\mathbf{i})$に応じて重み付けした、長さnの部分列uの出現回数を特徴ベクトルとする。
    [ギャップが多い場合には割り引く]
  - 長さnの部分列uを座標とする特徴空間F_n

$$\Phi : \Sigma^* \to F_n \cong R^{|\Sigma|^n}$$

$$\Phi(s) = (\phi_u(s))_{u \in \Sigma^n}, \quad \phi_u(s) = \sum_{\mathbf{i}:u=s[\mathbf{i}]} \lambda^{l(\mathbf{i})}, \quad \lambda \le 1$$

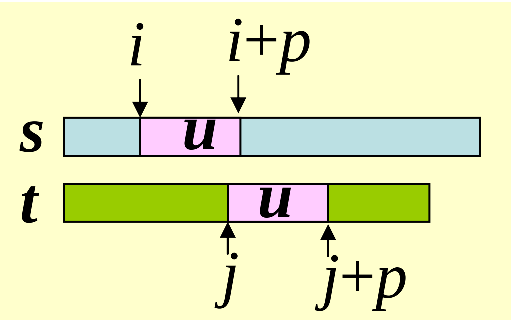


• カーネルの定義

$$K(s,t) = \langle \Phi(s), \Phi(t) \rangle = \sum_{u \in \Sigma^x} \phi_u(s) \cdot \phi_u(t)$$

– All-sequences kernel

$$K_{all}(s,t) = \sum_{u \in \Sigma^*} \sum_{(\mathbf{i},\mathbf{j}):u=s[\mathbf{i}]=t[\mathbf{j}]} 1 = \sum_{(\mathbf{i},\mathbf{j}):s[\mathbf{i}]=t[\mathbf{j}]} 1$$



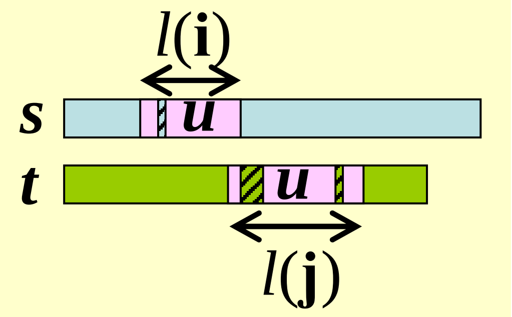
– p-Spectrum kernel

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

$$K_{p-spec}^{suffix}(s,t) = \begin{cases} 1 & \text{if } s = s_1 u,\ t = t_1 u,\ u \in \Sigma^p \\ 0 & \text{otherwise} \end{cases}$$

– Gapped-weighted subsequences kernel

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



uのギャップを除いた長さ: n

- All-sequences kernel: {“cat”, “car”, “bat”, “baa”}の例

ϕ	ε	a	b	c	r	t	ca	ct	at	ba	bt	cr
cat	1	1	0	1	0	1	1	1	1	0	0	0
car	1	1	0	1	1	0	1	0	0	0	0	1
bat	1	1	1	0	0	1	0	0	1	1	1	0
baa	1	2	1	0	0	0	0	0	0	2	0	0

ϕ	ar	aa	cat	car	bat	baa
cat	0	0	1	0	0	0
car	1	0	0	1	0	0
bat	0	0	0	0	1	0
baa	0	1	0	0	0	1

$$\Phi(\text{"cat"}) = (1,1,0,1,0,1,1,1,1,0,0,0,0,0,1,0,0,0)$$

$$\Phi(\text{"car"}) = (1,1,0,1,1,0,1,0,0,0,0,1,1,0,0,1,0,0)$$

$$K_{all}(\text{"cat"},\text{"car"}) = \left\langle \Phi(\text{"cat"}), \Phi(\text{"car"}) \right\rangle = 4$$

- *p*-Spectrum kernel: {“cat”, “car”, “bat”, “baa”}の例 (p=2)

ϕ	ca	at	ar	ba	aa
cat	1	1	0	0	0
car	1	0	1	0	0
bat	0	1	0	1	0
baa	0	0	0	1	1

$$\Phi(\text{"cat"}) = (1,1,0,0,0)$$

$$\Phi(\text{"car"}) = (1,0,1,0,0)$$

$$K_{2-spec}(\text{"cat"},\text{"car"}) = \left\langle \Phi(\text{"cat"}), \Phi(\text{"car"}) \right\rangle = 1$$

- Gap-weighted sequences kernel: {“cat”, “car”, “bat”, “baa”}の例 (n=2)

ϕ	ca	ct	at	ba	bt	cr	ar	aa
cat	λ^2	λ^3	λ^2	0	0	0	0	0
car	λ^2	0	0	0	0	λ^3	λ^2	0
bat	0	0	λ^2	λ^2	λ^3	0	0	0
baa	0	0	0	$\lambda^2 + \lambda^3$	0	0	0	λ^2

$$\Phi(\text{"cat"}) = (\lambda^2, \lambda^3, \lambda^2, 0,0,0,0,0)$$

$$\Phi(\text{"car"}) = (\lambda^2, 0,0,0,0, \lambda^3, \lambda^2, 0)$$

$$K_2(\text{"cat"}, \text{"car"}) = \left\langle \Phi(\text{"cat"}), \Phi(\text{"car"}) \right\rangle = \lambda^4$$

- カーネルの計算

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

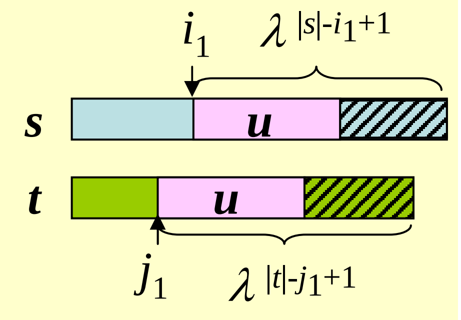
– 直接的に計算する場合： $O(|\Sigma|^n)$を含む計算
– 動的計画法のように再帰的な式を用いる： $O(n|s||t|)$
  - 初期条件：

$$K_n(s,\varepsilon) = K_n(\varepsilon,t) = 1$$

  - 再帰計算（過去に計算した結果を利用）：

$$K_n(sx,t) = \underbrace{\left(s\text{内と}t\text{内での一致によるもの}\right)}_{\text{過去に計算した結果}} + \underbrace{\left(x\text{を含む一致}\right)}_{\text{新たに計算}}$$

$$= \underline{K_n(s,t)} + \underline{\left(x\text{を含む一致}\right)}$$

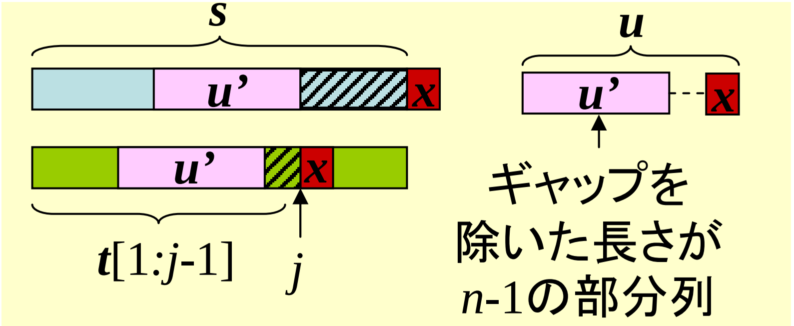


- （xを含む一致）の部分の再帰計算：
  - 補助関数を導入する。

$$K'_i(s,t)=\sum_{u\in\Sigma^i}\sum_{\mathbf{i}:u=s[\mathbf{i}]}\sum_{\mathbf{j}:u=t[\mathbf{j}]}\lambda^{|s|+|t|-i_1-j_1+2},\quad i=1,\dots,n-1$$

  - 補助関数を用いて（xを含む一致）の部分を表し、再帰的に計算する。

$$\left(x\text{を含む一致}\right):\ \sum_{j:t_j=x}K'_{n-1}(s,t[1:j-1])\lambda^2$$



- 初期条件：

$$K'_0(s,t)=1,\quad \forall s,t$$

$$K'_i(s,t)=0,\quad \text{if } \min(|s|,|t|)<i$$

– i=1,…,(n-1)についての再帰計算：

$$K'_i(sx,t) = \lambda \underline{K'_i(s,t)} + \underline{K''_i(sx,t)}$$

過去に計算した結果　新たに計算

$$K''_i(sx,tu) = \begin{cases} \lambda^{|u|} K''_i(sx,t) & \text{if } u \neq x \\ \lambda(K''_i(sx,t) + \lambda K'_{i-1}(s,t)) & \text{otherwise} \end{cases}$$

O(|s||t|)の再帰計算

$$(x\text{を含む一致}):\ \sum_{j:t_j=x} K'_{n-1}(s,t[1:j-1])\lambda^2$$

⇒全体としてはO(n|s||t|)の計算量

- カーネルの正規化（一般に利用される）
  - 長さによる影響を取り除く。

$$\hat{K}(s,t) = \left\langle \frac{\Phi(s)}{\|\Phi(s)\|}, \frac{\Phi(t)}{\|\Phi(t)\|} \right\rangle = \frac{1}{\|\Phi(s)\|\|\Phi(t)\|}\langle \Phi(s), \Phi(t) \rangle$$

$$= \frac{K(s,t)}{\sqrt{K(s,s)K(t,t)}}$$

テキスト分類の実験

- 実験条件
  - データベース： ロイターニュース Reuiters-21578
  - カテゴリー： “所得”、“買収”、“原油”、“穀類”
  - 学習： 390ドキュメント（152 / 114 / 76 / 48）
  - テスト： 90ドキュメント（40 / 25 / 15 / 10）
  - 識別器： SVM（各ドキュメントが正解カテゴリーに分類されるか判定）
  - *N*-grams kernel、Bag-of-words kernelと比較
    - *N*-grams kernel: 特徴ベクトルは*n*-grams（長さ*n*の部分ストリング）を添え字とする。
    - Bag-of-words kernel: 特徴ベクトルは単語を添え字とし、単語の頻度情報を値にとる。
- 結果
  - F尺度（再現率と適合率を考慮した尺度）：
    “所得”：95%、“買収”：88%、“原油”：95%、“穀類”：85%
  - Gap-weighted subsequences kernel ~ *N*-grams kernel
  - Gap-weighted subsequences kernel > Bag-of-words kernel

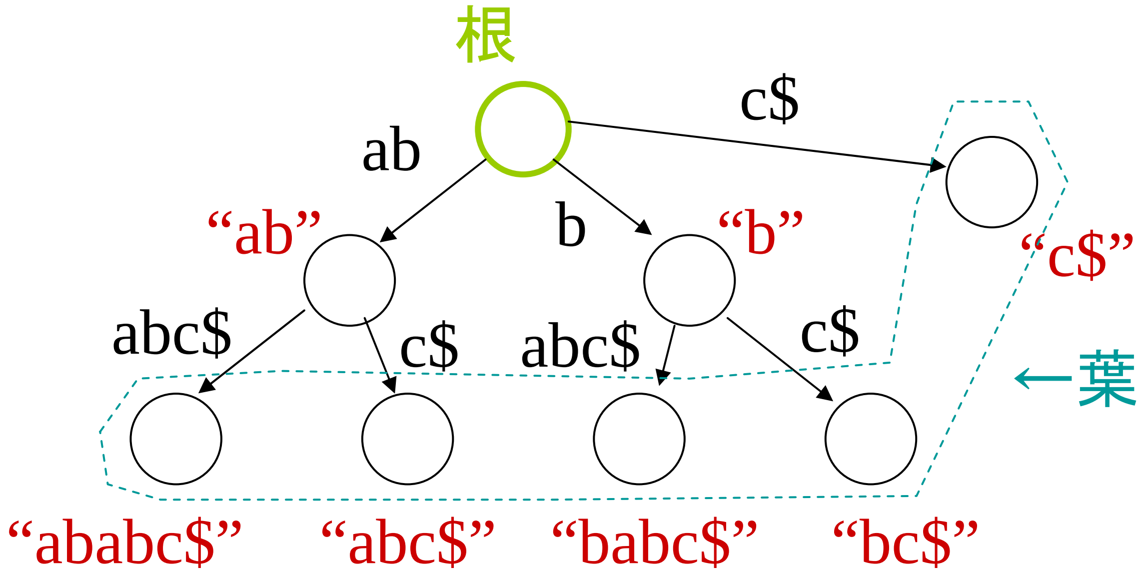
Suffix-Tree-Based Spectrum Kernel
[Vishwanathan et al., 2002]

- 対象とするカーネルの型

$$K(s,t) := \sum_{u \in \Sigma^*} \mathrm{num}_u(s) \cdot \mathrm{num}_u(t) \cdot w_u$$

  - $\mathrm{num}_u(s)$: 部分ストリングuの出現回数
  - 例） p-Spectrum kernel
- 接尾辞木を用いた効率的なカーネルの計算
  - 接尾辞木による文字列照合アルゴリズム [Cormen, et al, 1990]

- s=“ababc”の接尾辞木の例



– 接尾辞木：　ストリングsのすべての接尾辞を表したもの（trie）
  - 辺のラベル: sの部分ストリング
  - 根から葉までのラベルを連結したもの：　sの接尾辞
– 計算時間
  - 接尾辞木の作成：　線形時間
  - あるストリング中の部分ストリングの探索、複数のストリング中に共通に出現する部分ストリングの探索：　線形時間
（根から辿り、頂点に対応する文字ストリングを列挙する。）

$$O(|s|+|t|)$$

グラフ

- ラベル付きグラフ　$G = (V, E)$
  - 頂点の集合 V（$|V|$個）
  - 辺の集合　$E \subset V \times V$
  - ラベルの集合 A
  - ラベル関数 $l: V \cup E \rightarrow A$　($l(x)$: 頂点(or 辺) x上のラベル)
  - 頂点vから出る辺の数　$d(v)$
  - 有限長の頂点列の集合　$V^* = \bigcup_{n=1}^{\infty} V^n$
  - パス　$h \in V^*$　($h = v_1, \ldots, v_n$、$(v_i, v_{i+1}) \in \mathrm{E}$　$i = 1, .., n-1$、長さ$|h|$)
  - グラフGの全パス集合　$H(G) \subset V^*$
  - パス上のラベル　$l(h) = (l(v_1), l(v_1, v_2), \ldots, l(v_{n-1}, v_n), l(v_n)) \in A^{2n-1}$

Marginalized Graph Kernelの応用例 [Mahe et al., 2004]

- カーネルの定義 [Tsuda et al., 2002]

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

$$K(G_1, G_2) = \sum_{(h_1, h_2) \in V_1^* \times V_2^*} p_1(h_1) p_2(h_2) K_L(l(h_1), l(h_2))$$

  - $p1, p2$: V_1^*, V_2^*上の確率分布
  - $K_L: A^* \times A^* \rightarrow R$ ラベル列間のカーネル

$$K_L(l_1, l_2) = \begin{cases} 1 & \text{if } l_1 = l_2 \\ 0 & \text{otherwise} \end{cases}$$

$$p(v_1, \ldots, v_n) = p_s(v_1) \prod_{i=2}^{n} p_t(v_i \mid v_{i-1})$$

- ラベル列間のカーネルの例

$$K_L(l_1, l_2) = \begin{cases} 1 & \text{if } l_1 = l_2 \\ 0 & \text{otherwise} \end{cases}$$

$$p(v_1, \ldots, v_n) = p_s(v_1) \prod_{i=2}^{n} p_t(v_i \mid v_{i-1})$$

の与え方

– $0 < p_q(v) < 1, \quad \sum_{v \in V} p_0(v) = 1$

– $\exists v \in V \quad \sum_{u \in V} p_a(u \mid v) = 1\,, \quad p_a(u \mid v) > 0 \Rightarrow (v, u) \in E$

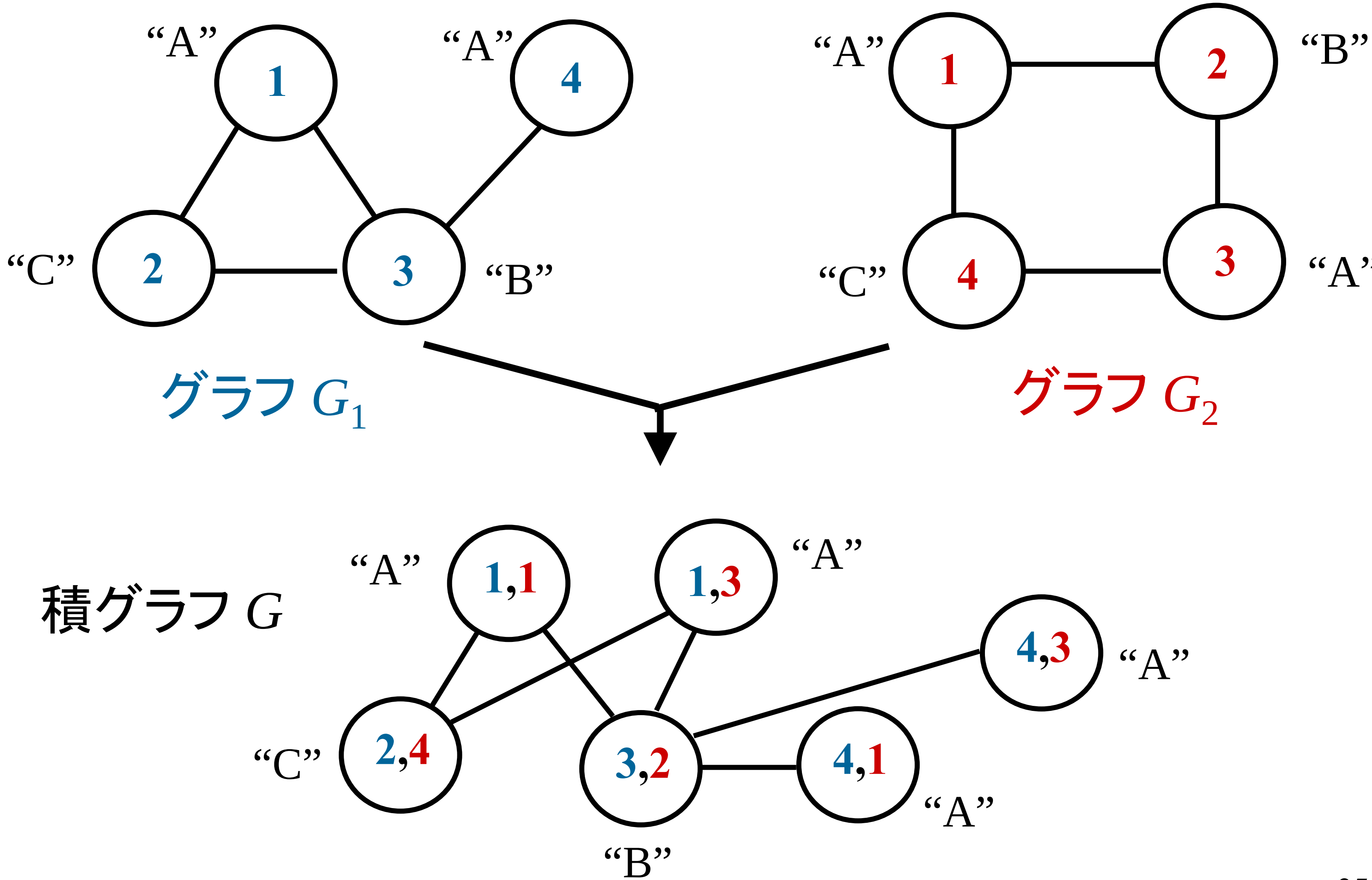
$$p_s(v) = p_0(v) p_q(v)$$

*v*から出て　*u*にいる

$$p_t(u \mid v) = \frac{\boxed{1 - p_q(v)}}{\boxed{p_q(v)}} \boxed{p_a(u \mid v)} \boxed{p_q(u)}$$

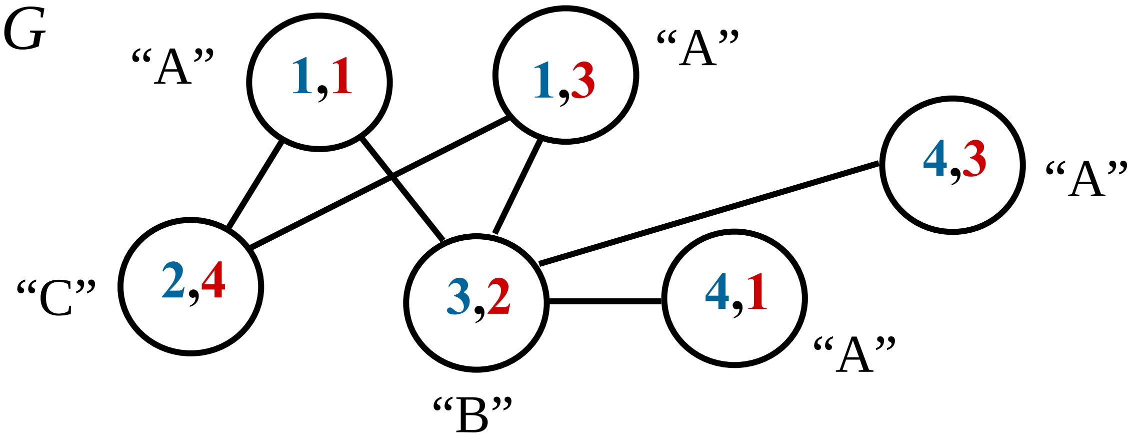
まず*v*にいることが条件　*u*に遷移して

- カーネルの効率的な計算
  - 積グラフの利用



- 積グラフ上の遷移

積グラフ G



遷移行列 Π

	1,1	1,3	2,4	3,2	4,1	4,3
1,1						
1,3						
2,4						
3,2						
4,1						
4,3						

- 積グラフによるカーネルの計算

$$\pi((u_1,v_1)\cdots(u_n,v_n)) = \pi_s(u_1,v_1)\prod_{i=2}^{n}\pi_t((u_i,v_i)\,|\,(u_{i-1},v_{i-1}))$$

$$\begin{cases} \pi_s(u_1,v_1) = p_s^{(1)}(u_1)\,p_s^{(2)}(v_1) \\ \pi_t((u_2,v_2)\,|\,(u_1,v_1)) = p_t^{(1)}(u_2\,|\,u_1)\,p_t^{(2)}(v_2\,|\,v_1) \end{cases}$$

$$K(G_1,G_2) = \sum_{(h_1,h_2)\in V_1^*\times V_2^*} p_1(h_1)p_2(h_2)K_L(l(h_1),l(h_2))$$

$$K(G_1,G_2) = \sum_{h\in H(G)}\pi(\mathrm{h}) = \sum_{n=1}^{\infty}\left(\sum_{h\in H(G),|h|=n}\pi(\mathrm{h})\right) = \pi_s^t(I-\Pi)^{-1}\mathbf{1}$$

計算量 $O(|\Pi|d^N)$

分子構造の分類実験

- データ:　MUTAGデータベース
- 結果
  - Marginalized graph kernel: 90%
  - Neural network: 89%
  - Decision tree: 88%
  - Linear regression: 89%

話の構成

- 実問題を扱う
  - 構造化データを扱う
    - 対象データ内の構造
      - Convolution kernelの考えを利用
      - 確率的な生成モデルを利用
    - クラスデータの構造
  - 条件付分布を推定する
  - ソフトウエアの紹介

確率的な生成モデルを利用

- Hidden Markov model（HMM）などの生成モデルを利用する。
  - 有限状態集合A（初期状態: a_I、最終状態: a_F）、
    状態bからaへの状態遷移確率 $P_M(a|b)$、
    状態aでシンボルσを生成する確率分布 $P(\sigma|a)$ から構成される。
- データに対するモデルの尤度を用いる。
  - P-kernel
  - 拡張例: タンパク質発現プロファイルの解析 [Vert, 2002]
- モデルの尤度だけでなく、幾何情報も利用する。
  - Fisher kernel
  - 拡張例: 音声認識 [Gales et al, 2004]
    （Tangent vector of posterior log odds (TOP) kernel)

P-Kernel

- カーネルはデータ x, z の同時確率で表す。

  $K(x,z) = P(x,z)$

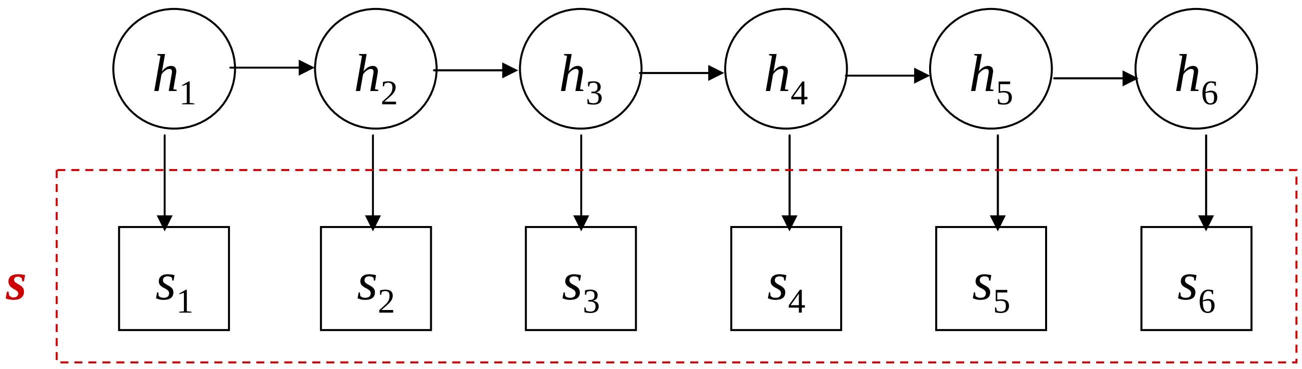
  – ただし、$P(x,z)$は対称、かつ半正定値性を満たすものに限る。
- 条件付き独立を仮定する。[半正定値性の十分条件]

  $P(x,z|m) = P(x|m)P(z|m)$
- Marginalisation kernel
  – モデル集合Mを座標とする特徴空間F

$$\Phi : x \mapsto (P(x \mid m))_{m \in M} \in F$$

$$\begin{aligned} K(x, z) = P_M(x, z) &= \langle \Phi(x), \Phi(z) \rangle \\ &= \sum_{m \in M} P(x \mid m) P(z \mid m) P_M(m) \end{aligned}$$

- HMMから生成される固定長のストリング



HMM: M

– n (= 6) 状態（で表されたストリングh）のHMM Mからストリング s (= $s_1, s_2, \ldots, s_6$)、t (= $t_1, t_2, \ldots, t_6$) が生成される。

$$P(s,t \mid h) = \prod_{i=1}^{n} P(s_i \mid h_i) P(t_i \mid h_i)$$

$$P_M(h) = P_M(h_1) P_M(h_2 \mid h_1) \ldots P_M(h_n \mid h_{n-1})$$

$$K(s,t) = \sum_{h \in A^n} P(s \mid h) P(t \mid h) P_M(h) = \sum_{h \in A^n} \prod_{i=1}^{n} P(s_i \mid h_i) P(t_i \mid h_i) P_M(h_i \mid h_{i-1})$$

タンパク質発現プロファイルの解析

[Vert, 2002]

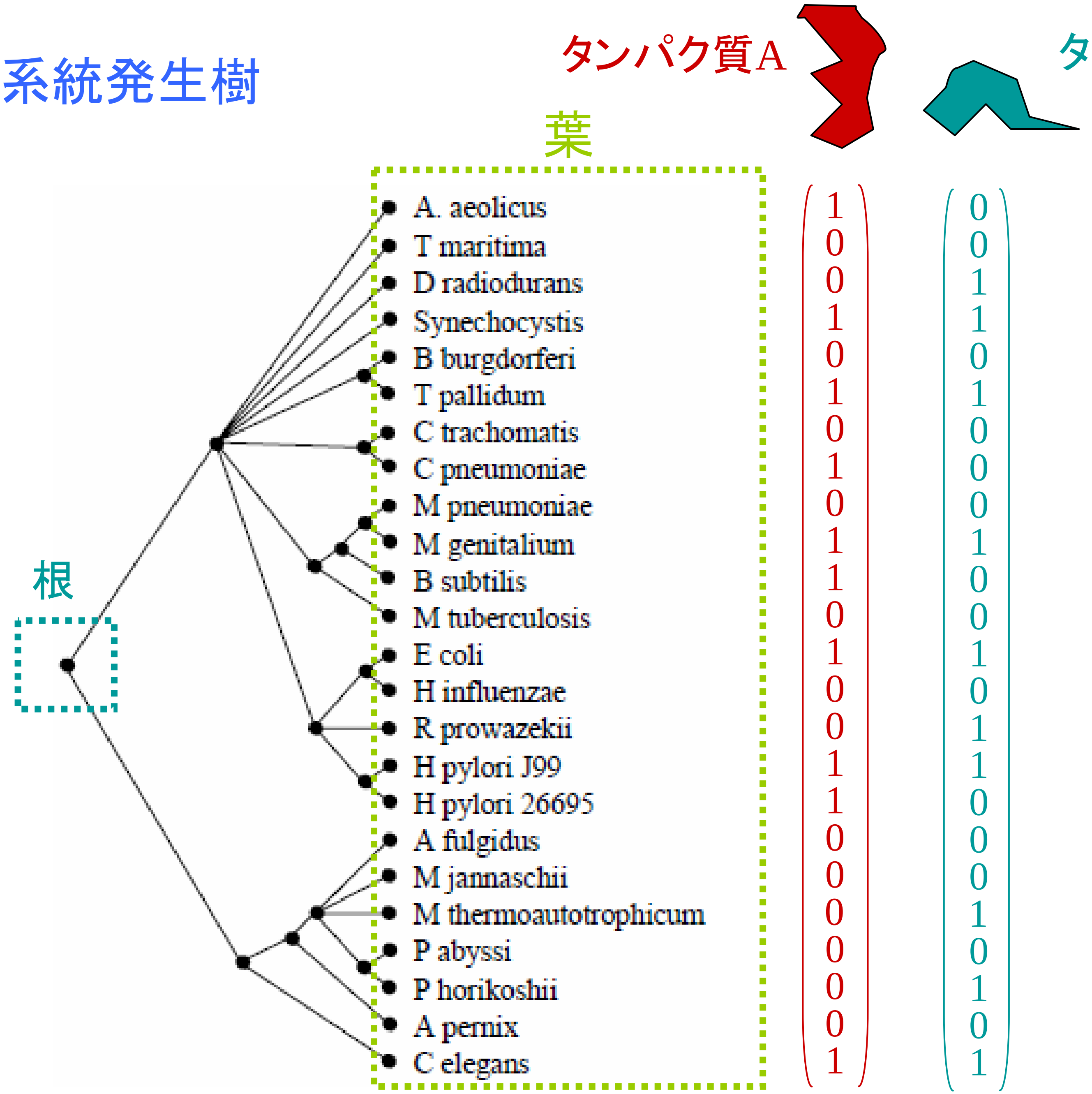
目的:

- タンパク質発現プロファイル間の類似度をはかる。
  - タンパク質発現プロファイル: あるタンパク質が各生体に発現するかどうかの情報

アプローチ:

- 系統発生樹上の進化過程も考慮する。
- Hidden tree modelによるKernel、およびSVMを用いる。

- 系統発生樹



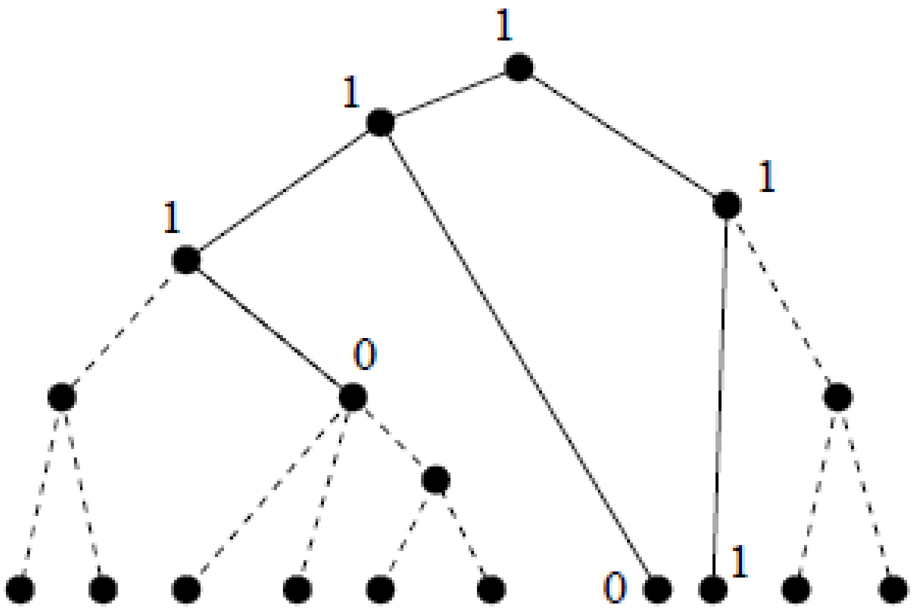
大腸菌やピロリ菌などの生体

- 根付き木
  - 非循環有向グラフ
  - T: 頂点の集合、$T^* = T \setminus \{\lambda\}$、$\lambda$: 根
  - $L \subset T$: 葉の集合
  - $u \in T^*$
    - 親の頂点 $p(u) \in T$
    - 子の頂点の集合 $c(u) \in T$
    - uから辿れる葉の頂点の集合 $l(u) \in T$
  - 頂点に割り当てられる値 $A=\{0,1\}$
  - 同時確率 P が頂点の変数$\{X_u, u \in T\}$に対して定義する。
  - 便宜上、次の確率を定義する。

$$S \subset T, S' \subset T, x_S \in A^S, y_{S'} \in A^{S'}$$

$$p(x_S \mid y_{S'}) \equiv P\left\{\forall u \in S, X_u = x_u \mid \forall u' \in S', X_{u'} = y_{u'}\right\}$$

- 部分木で表される進化パターン (S, z_S): Hidden tree model



- カーネルの定義

$$\Phi : A^L \to R^D$$

$$K(x_L, y_L) \equiv \langle \Phi(x_L), \Phi(y_L) \rangle = \sum_{i=1}^{D} \Phi_i(x_L) \cdot \Phi_i(y_L)$$

$$= \sum_{S \in C(T)} \sum_{z_S \in A^S} p(x_L \mid z_S) p(y_L \mid z_S) p(z_S)$$

効率的な計算アルゴリズムを提案

音声認識
[Gales et al., 2004]

目的:

- 音声データ**O**から、その発声内容（単語w_i（列））を判定する。
  - 従来、HMMが用いられている。

$$\mathbf{O} = o_1, o_2, \ldots, o_T$$

$$\arg\max_i \{P(w_i \mid \mathbf{O})\} = \arg\max_i \left\{ \frac{P(\mathbf{O} \mid w_i) P(w_i)}{P(\mathbf{O})} \right\} \approx \arg\max_i \{P(\mathbf{O} \mid w_i)\}$$

アプローチ:

- Fisher kernel（次式）を拡張する。

$$K(x, z) = g(\hat{\boldsymbol{\theta}}, x)^t \mathbf{I}_M^{-1} g(\hat{\boldsymbol{\theta}}, z)$$

$$\mathbf{I}_M = \mathrm{E}\left[g(\hat{\boldsymbol{\theta}}, x) g(\hat{\boldsymbol{\theta}}, x)^t\right], \quad g(\hat{\boldsymbol{\theta}}, x) = \left(\frac{\partial \log p(x \mid \hat{\boldsymbol{\theta}})}{\partial \theta_i} \right)_{i=1}^{N} = \nabla_{\boldsymbol{\theta}} \log p(x \mid \hat{\boldsymbol{\theta}})$$

- SVMを用いる。

- カーネルの定義
  - 二つのクラスω_1, ω_2のHMM $\theta^{(1)}$, $\theta^{(2)}$による尤度比も用いる。

$$K(\mathbf{x},\mathbf{z}) = \left\langle \Phi(\hat{\boldsymbol{\theta}},\mathbf{x}), \Phi(\hat{\boldsymbol{\theta}},\mathbf{z}) \right\rangle$$

$$\Phi(\hat{\boldsymbol{\theta}},\mathbf{x}) = \frac{1}{T}\begin{bmatrix} \log p(\mathbf{x}\mid\hat{\boldsymbol{\theta}}^{(1)}) - \log p(\mathbf{x}\mid\hat{\boldsymbol{\theta}}^{(2)}) \\ \nabla_{\boldsymbol{\theta}^{(1)}} \log p(\mathbf{x}\mid\hat{\boldsymbol{\theta}}^{(1)}) \\ \nabla_{\boldsymbol{\theta}^{(2)}} \log p(\mathbf{x}\mid\hat{\boldsymbol{\theta}}^{(2)}) \end{bmatrix}, \quad \mathbf{x} = x_1, x_2, \ldots, x_T$$

間違えやすい単語の識別実験

- データ:　大規模音声認識用データベース
  （Fisher LDC、400時間）
- 間違えやすい単語:　“A/The”, “Can/Can’t”, “Know/No”
- 識別率（%）:

	“A/The”	“Can/Can’t”	“Know/No”
従来のHMM	58	82	68
本カーネル	61	85	72
	80	89	86

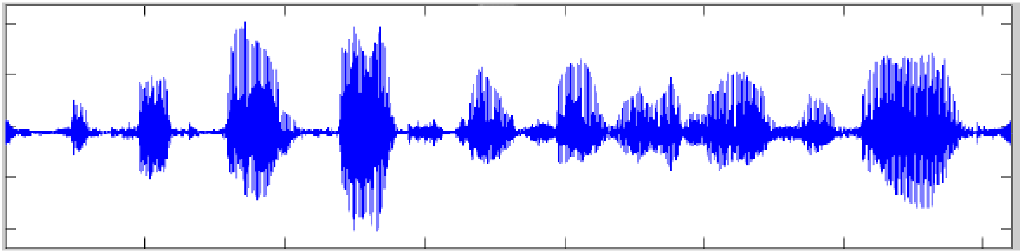
（単語ラティスから計算される事後確率も特徴ベクトルに加えた場合）

話の構成

- 実問題を扱う
  - 構造化データを扱う
    - 対象データ内の構造
      - Convolution kernelの考えを利用
      - 確率的な生成モデルを利用
    - クラスデータの構造
  - 条件付分布を推定する
  - ソフトウエアの紹介

クラスデータが構造を持つ場合

- SVM: クラスデータ $y = \{-1, 1\}$
- 例） 品詞タグ付け
  - x = "The cat ate the mouse"
  - y = "Det N VBD Det N"
- 例） 音声認識



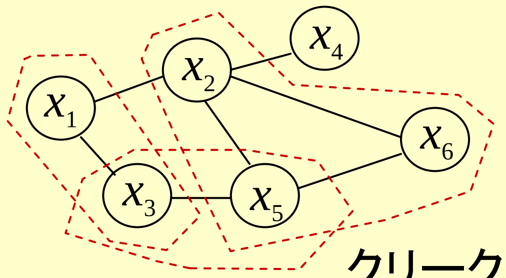
  - $x = x_1, x_2, \ldots, x_T$: 特徴ベクトル列
  - y = "How do we recognize speech"

Kernel Conditional Random Field

[Lafferty et al., 2004]

- x, yを組み合わせて扱う。
  - x: 特徴ベクトル （例）アミノ酸配列、ヒストグラムなど
  - y: ラベル付けしたグラフ

- 対象: グラフデータ
- 目的: グラフの頂点のラベル付け



クリークの例

グラフに関する表記:

- β : グラフの有限集合
- $g \in \beta$: グラフ
- $V(g)$: 頂点の集合
- $|g|=|V(g)|$: グラフの大きさ～頂点数
- $C(g)$: クリーク（頂点が密結合している部分）の集合
- $|c|$: クリーク中の頂点数

グラフに関する表記のつづき:

- Y:　ラベルの有限集合
- $Y(g)$:　グラフg中のラベル集合
- $Y_c(g) = \{(c, y_c) \mid c \in C(g), y_c \in Y^{|c|}\}$:
  グラフg中のラベル付けされたクリークの集合
- X:　入力特徴空間　例)$X = R^n$
- $X(g)$:　グラフgの頂点に割り当てられた特徴ベクトルの集合

- 目的:　関数 $h: X(\beta) \to Y(\beta)$,　$h(g,x) \in Y(g)$を学習したい。
- hを直接推定することは難しい⇒$XY_c(\beta)$上の損失関数を考える。

識別関数

$$f : XY_c(\beta) \to R, \quad c \in C(g), \quad y_c \in Y^{|c|}$$

負の損失関数

$$\phi(y, \{f(g,x,c,y_c)\}) = -\log\left(\frac{\exp\left(\sum_{c \in C(g)} f(g,x,c,y_c) \right)}{\sum_{y' \in Y(g)} \exp\left(\sum_{c \in C(g)} f(g,x,c,y'_c) \right)} \right)$$

- $XY_c(\beta)$上のカーネルを考える。

$$K((g,x,c,y_c),(g',x',c',y'_{c'}))\in R$$

Representer Theoremより

$$\hat{f}(\cdot)=\sum_{i=1}^{N}\sum_{c\in C(g^{(i)})}\sum_{y_c\in Y^{|c|}}\alpha_c^{(i)}(y_c)K((g,x,c,y_c),\cdot)$$ （学習サンプル数: N）

負の損失関数を最大にする識別関数

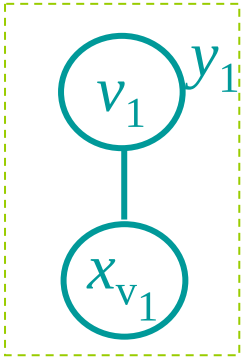
- 具体例:

$$c=(v_1,v_2),\quad y_c=(y_1,y_2)$$

$$K((g,x,(v_1,v_2),(y_1,y_2)),(g',x',(v_1',v_2'),(y_1',y_2'))=\overline{K}(x_{v_1},x'_{v_1'})\delta(y_1,y_2)$$

$$\hat{f}_{(v_1,v_2)}(x,y_1,y_2)=\sum_{i=1}^{N}\sum_{v\in V(g^{(i)})}\alpha_v^{(i)}(y_1)\overline{K}(x_{v_1},x_v^{(i)})$$

エッジ上のカーネル



効率的な f の推定アルゴリズムを提案
（SVMではない）

話の構成

- 実問題を扱う
  - 構造化データを扱う
    - 対象データ内の構造
      - Convolution kernelの考えを利用
      - 確率的な生成モデルを利用
    - クラスデータの構造
  - 条件付分布を推定する
  - ソフトウエアの紹介

条件付分布を推定する

- SVMはサポートベクターだけで識別境界を表す。
- x与えられた時のyの分布をうまく推定することにより、データの本質をつかむ。
  - Penalized logistic regression machine; PLRM [田邉, 2001]
  - Kernel multiple logistic regression; KMLR [Seeger, 2005]
  - Kernel conditional random field; KCRF [Lafferty et al., 2004]

負の損失関数

$$\phi(y, \{f(g,x,c,y_c)\}) \Leftrightarrow p(y \mid g,x) = \frac{\exp\left(\sum_{c \in C(g)} f(g,x,c,y_c)\right)}{\sum_{y' \in Y(g)} \exp\left(\sum_{c \in C(g)} f(g,x,c,y'_c)\right)}$$

ロジスティック変換

Penalized Logistic Regression Machine
[田邉, 2001]

- 多クラスの判別予測を行う。
  - 学習データセット $\{(x_i, y_i)\}_{i=1,\ldots,N}$
    （例） データ $x_i \in R^n$, クラス $y_i \in \{1,\ldots,K\}$）
  - xが与えられた時のyの値を予測する。
- xが与えられた時のyの条件付分布として、$\mathbf{p}(x)$の多項分布を考える。

$$\mathbf{p}(x) = \left(p_1(x), \ldots, p_K(x)\right)^t, \quad p_k(x) = \frac{\exp\left(f_k(x)\right)}{\sum_{j=1}^{K} \exp\left(f_j(x)\right)}$$

$$f_k(x) = \sum_{i=1}^{M} \boxed{w_k^{(i)}} \phi^{(i)}(x) + \boxed{w_k^{(0)}} \omega = \sum_{i=1}^{N} \boxed{v_k^{(i)}} \mathrm{K}\left(x^{(i)}, x\right)$$

$$\overline{W} = V \overline{\Phi}^t$$

Primalパラメータ　　Dualパラメータ

M: 特徴ベクトルの次元 $\gg$ N: サンプル数

PLRMのパラメータ推定

- 負の対数尤度

$$L(\mathbf{V}) \equiv -\sum_{j=1}^{N} \log(p_{c_j}(x_j))$$

$$\mathbf{V} = \begin{pmatrix} v_1^{(1)} & \cdots & v_1^{(N)} \\ \vdots & \ddots & \vdots \\ v_K^{(1)} & \cdots & v_K^{(N)} \end{pmatrix}$$

- 負の罰金付対数尤度

$$PL(\mathbf{V}) \equiv -\sum_{j=1}^{N} \log(p_{c_j}(x_j)) + \boxed{\frac{\delta}{2} \left\| \Gamma^{\frac{1}{2}} \mathbf{V} \mathbf{K}^{\frac{1}{2}} \right\|_F^2}$$

罰金項

$$\mathbf{K} = \left[k(x_i, x_j) \right]_{i,j=1,\ldots,N}$$

KKT conditions

$$\Rightarrow \quad \frac{\partial PL(\mathbf{V})}{\partial \mathbf{V}} = (\mathrm{P}(\mathbf{V}) - \mathbf{Y} + \delta \Gamma \mathbf{V})\mathbf{K} = O_{K,N}$$

Newton+CG法でVを推定

- Newton+CG法によるVの推定

$$\mathbf{V}^{(i+1)} = \mathbf{V}^{(i)} - \alpha^{(i)} \frac{f(\mathbf{V}^{(i)})}{f'(\mathbf{V}^{(i)})} = \mathbf{V}^{(i)} - \alpha^{(i)} \Delta \mathbf{V}^{(i)}$$

Newton法

$$f(\mathbf{V}) = \frac{\partial PL(\mathbf{V})}{\partial \mathbf{V}} = (\mathrm{P}(\mathbf{V}) - \mathbf{Y} + \delta \Gamma \mathbf{V})\mathbf{K} \to O_{K,N}, \quad \mathrm{P}(\mathbf{V}) \equiv [\mathbf{p}(x_1); \cdots; \mathbf{p}(x_N)]$$

$$f'(\mathbf{V}) = \frac{\partial^2 PL(\mathbf{V})}{\partial^2 \mathbf{V}}$$

（ヘシアン行列が厳密な式で与えられている）

KCRF？　KMLR〇

– $\Delta \mathbf{V}^{(i)}$の推定をCG法で行う。

大規模データのための学習アルゴリズム
[Myrvoll et al., 2006]

PLRMによる自動的な特徴抽出
[松井、田邉、2006]

- 目的: 入力音声を発声した話者が誰かを判定する。
  - 従来、HMM/GMMで表された各ユーザモデルと入力音声との尤度を求め、最大尤度を示すユーザモデルの話者をその入力音声を発声した話者とする。
  - 特徴量としてはメルケプストラム係数ベクトルが用いられる。
    - 周波数軸を人間の聴覚の特性を考慮したメルスケールに変換
    - ケプストラム分析（対数パワースペクトルの逆フーリエ変換）
- アプローチ: PLRMにより、音声データから識別的な話者特徴を暗に抽出して用いる。
  - メルスケールなどの事前知識に基づく処理を行うことなく、256次元の対数パワースペクトルを直接用いる。

PLRMによる方法

音声信号
(FFTスペクトル)

特徴抽出

- 対数パワー
(256次元スペクトル)

学習

PLRM

Dualパラメータの推定:
話者間の確率密度の比

識別

確率推定

決定:確率推定量の比較

GMMによる方法

音声信号
(FFTスペクトル)

特徴抽出

- メルスケーリング
- 次元削減
- 対数パワー
- 逆FFT
(26次元メルケプストラム)

学習

GMM

GMMパラメータの推定:
各話者の確率密度

識別

尤度推定

決定:尤度の比較

話者10名の識別実験

テスト データ	方法	識別率 (%)	
		メルケプストラム	対数パワースペクトル
単語音声	PLRM	90.7 (87.3, 94.7)	**92.7 (89.3, 96.0)**
	SVM	91.3 (87.3, 94.7)	88.0 (83.3, 92.0)
	GMM	89.3 (85.3, 93.3)	84.0 (79.3, 88.7)
文音声	PLRM	99.3 (98.7,100)	**100 (99.3, 100)**
	SVM	98.0 (96.0, 99.3)	**100 (99.3, 100)**
	GMM	99.3 (98.7, 100)	99.3 (98.7, 100)

※学習は3時期に発声した3文章（約12秒）

話の構成

- 実問題を扱う
  - 構造化データを扱う
    - 対象データ内の構造
      - Convolution kernelの考えを利用
      - 確率的な生成モデルを利用
    - クラスデータの構造
  - 条件付分布を推定する
  - ソフトウエアの紹介

まとめ

- 構造化データを扱う
  - データ、目的に応じてカーネルを設計
  - カーネルを効率的に計算⇒グラム行列
  - SVM
- 条件付分布をうまく推定する
  - $p(y|x)$をロジスティック変換の形で表す
  - 負の尤度最小化(負の損失関数最大化)に基づくパラメータ推定
    - ニュートン法など

音声認識: 条件付分布の推定＋HMMパラメータの推定
[O. Birkenes, T. Matsui and K. Tanabe. Isolated-Word Recognition with Penalized Logistic Regression Machines. Proc. ICASSP, 2006.]

SVMに関するソフトウエアの紹介

- http://www.support-vector-machines.org/SVM_soft.html
  - SVMlight   http://svmlight.joachims.org/
  - LIBSVM http://www.csie.ntu.edu.tw/~cjlin/libsvm/
  - Spider   http://www.kyb.tuebingen.mpg.de/bs/people/spider/

- LIBSVM＋SpiderのMatlab上のデモ
  - 複数クラスの識別: one-versus-rest（1 vs other)
  - クロスバリデーション

- 目的:　手書き数字(0~9)の識別
  - 10クラス
  - 1000サンプル、各クラス100サンプル

1000サンプル

1000サンプル

グラム行列

Gmatrix

ラベル

Labels

$$\begin{pmatrix} \mathbf{0} \\ \vdots \\ \mathbf{0} \\ \\ \mathbf{9} \\ \vdots \\ \mathbf{9} \end{pmatrix}$$

100サンプル分

100サンプル分

9

- クロスバリデーション
  - 例）3 fold → 666 (学習)　334 (テスト)

⇒平均誤り率、その標準偏差

全サンプル

学習	学習	テスト
学習	テスト	学習
テスト	学習	学習

統数研 ISM

- Matlabスクリプトの主要部分 [© 2004 Marco Cuturi]

```
d1=data(Gmatrice,Labels);     % データの用意

Options=['folds=',num2str(cv_f),';repeats=',num2str(repetitions)];
                                % クロスバリデーションのfold数と繰り返し数

a1=svm;                         % SVMパラメータの設定
a1.alpha_cutoff=0.00001;
a1.C=1000;
a1.child=kernel('custom',Gmatrice);
                                % ここではグラム行列は事前に計算

oK=one_vs_rest(a1);             % 複数クラスの識別法としてone_vs_restを指定
vect_er=eval(['get_mean(loss(train(cv(oK,''',Options,'''),d1)));'])
                                % クロスバリデーションをしながら学習して
                                %                  平均誤り率を計算する

e=vect_er.Y(1,1)                % 平均誤り率
s=vect_er.Y(1,2)                % 標準偏差
```

参考文献

- サーベイ
  - J. Shawe-Taylor and N. Cristianini. Kernel Methods for Pattern Analysis.Cam- bridge University Press, 2004.
  - Scholkopf, B, Tsuda, K., and Vert, J.P., Kernel Methods in Computational Biology, MIT press, 2004.
  - B. Schlkopf and A. J. Smola. Learning with Kernels. MIT Press, Cambridge, MA, 2002.
  - 津田 宏治, カーネル設計の方法, 日本神経回路学会誌, 9, 3, pp.190--195, 2002.
  - 鹿島 久嗣, カーネル法による構造データの解析, 電子情報通信学会技術研究報告 言語理解とｺﾐｭﾆｹｰｼｮﾝ/ﾊﾟﾀｰﾝ認識・ﾒﾃﾞｨｱ理解, 2005.
  - 鹿島 久嗣, カーネル法による構造データマイニング, 情報処理, Vol. 46, No. 1, 2005.
  - 鹿島 久嗣, 坂本 比呂志, 小柳 光生: 木構造データに対するカーネル関数の設計と解析, 人工知能学会論文誌, Vol.21, No.1, 2006.
  - 赤穂昭太郎, カーネルマシン, 信学技法 NC2003-34, 2003.
  - 福水健次, カーネル法, 統数研公開講座「機械学習の最近の話題」資料, 2004.
- 構造化データ
  - D. Haussler. Convolution kernels on discrete structures. In Technical Report UCS-CRL-99-10,1999.
  - Huma Lodhi, John Shawe-Taylor, Nello Cristianini, and Christopher J. C. H. Watkins. Text classification using string kernels. In NIPS, pp. 563--569, 2000.
  - C. Leslie, E. Eskin, J. Weston, and W. Noble. Mismatch string kernels for SVM protein classification. In Advances in Neural Information Processing Systems, 15, 2002.
  - S. V. N. Vishwanathan and A. J. Smola. Fast kernels on strings and trees. In Advances in Neural Information Processing Systems, 15, 2002.
  - Cormen, T.H., Leiserson Introduction to Algorithms. The MIT Electrical Engineering and Computer Science Series. MIT Press/McGraw, C.E., Rivest, R.L.: Hill, 1990.

- J.-P. Vert, H. Saigo, T. Akutsu. Local alignment kernels for biological sequences. In Kernel Methods in Computational Biology, B. Schölkopf, K. Tsuda and J.-P. Vert (Eds.), MIT Press, p.131-154, 2004.
- K. Tsuda, T. Kin, and K. Asai. Marginalized kernels for biological sequences. Bioinformatics, 18(Suppl. 1): S268-S275, 2002.
- Mahe, P., Ueda, N., Akutsu, T., Perret, J.-L., & Vert, J.-P. Extensions of marginalized graph kernels. Proceedings of ICML'04, pp. 552-559, 2004.
- Vert, J., A tree kernel to analyze phylogenetic profiles, Bioinformatics, 18:s276--s284, 2002.
- M.J.F. Gales and M.I. Layton. SVMs, Score-spaces and Maximum Margin Statistical Models. Beyond HMM Workshop, 2004.
- N. Smith and M. Gales. Speech recognition using SVMs. In Advances in Neural Information Processing Systems 14. 2002, MIT Press.

- 条件付分布

- K. Tanabe. Penalized Logistic Regression Machines: New methods for statistical prediction 1. ISM Cooperative Research Report 143, pp. 163-194, 2001.
- K. Tanabe. Penalized Logistic Regression Machines: New methods for statistical prediction 2. Proc. IBIS, Tokyo, pp. 71-76, 2001.
- K. Tanabe. Penalized Logistic Regression Machines and Related Linear Numerical Algebra. KOKYUROKU 1320, Institute for Mathematical Sciences, Kyoto University, pp. 239-249, 2003.
- M. Seeger, Cross-Validation Optimization for Structured Hessian Kernel Methods, Technical report, 2005.
- Lafferty, J., Zhu, X., and Liu, Y.. Kernel conditional random fields: Representation and clique selection. Proc.International Conf. on Machine Learning, 2004.
- T. A. Myrvoll and T. Matsui. On a greedy learning algorithm for dPLRM with applications to phonetic feature detection. Proc. Interspeech, 2006.
- T. Matsui and K. Tanabe. Comparative Study of Speaker Identification Methods: dPLRM, SVM and GMM. IEICE, Vol.E89-D, 3, pp. 1066-1073, 2006.
- O. Birkenes, T. Matsui and K. Tanabe. Isolated-Word Recognition with Penalized Logistic Regression Machines. Proc. ICASSP, 2006.

- ソフトウエア
  - http://www.support-vector-machines.org/SVM_soft.html

総合研究大学院大学
「統計科学専攻」