カーネル法による構造データマイニング

鹿 島 久 嗣 †

Kernel Methods for Mining Structured Data

Hisashi Kashima †

1. 構造データマイニングとカーネル法

WWW, Web サービス, blog, ···, 日々進化し続けるインターネット技術は, 今や HTML や XML などの, 構造をもったデータなしには語れなくなってきている. これらは相互に参照する機能をもち, データをまたいだ, さらなる構造が形成されている.

一方, ヒトを含むさまざまな生物の全ゲノム配列が解読され, また, タンパク質はその 3 次元立体構造などが解明されつつある. さらには, 遺伝子やタンパク質がお互いに作用し, 制御し合うという相互関係が徐々に明らかになってきている. バイオインフォマティクスと呼ばれる分野では, これらのデータを解析することで, 新たな生物学的, 医学的知見が得られることが期待されている.

こういったデータはすべて, なんらかの形で内的ないし外的な「構造」をもっているといえる. そして, 長年研究されてきたデータマイニングや機械学習の手法を用いてこれらのデータを解析し, なんらかの知見を見出すことは当然の期待といえるだろう. しかしながら, 従来のデータマイニングや機械学習で用いられる多くの手法では, 解析の対象となるデータはベクトル形式で表現されているものと仮定されている.

近年, このような「データに潜む構造」を扱えるようにデータマイニングや機械学習の手法を拡張しようという試みが活発になされている. 例えば, 本特集での鷲尾や、工藤らの解説もこういった取り組みのひとつであるといえる. そして本解説では, サポートベクトルマシンに代表される, 近年の機械学習の分野で特に活発に研究が進んでいるカーネル法[1] とよばれる手法に基づく, 構造をもったデータを扱うためのいくつかのアプローチを紹介する.

さて, まずはカーネル法とはそもそも一体何かというところから出発しよう. 例として, 健康診断の各種の測定結果から, ある人が健康かどうかを診断するような問題を考えてみる. 人類すべての集合 X の中の, ある人 $x \in X$ に対して, 全部で D 種類ある測定を行うとする. それらの測定の結果を D 次元の実数ベクトル形式で $\Phi(x) = (\phi_1(x), \phi_2(x), \ldots, \phi_D(x))$ と表すとする. これを特徴ベクトルと呼ぶことにする. また, y を健康状態を表す変数とし, $y = +1$ は健康である状態を, $y = -1$ はそうでない状態を表すとする. すると, この問題は典型的な 2 値分類学習の問題として定式化される.

2 値分類学習の問題とはデータの D 次元の特徴ベクトルの空間から 2 つのクラス $Y = \{+1, -1\}$ へのマッピング $h : \Re^D \to Y$ を求める問題である. 学習アルゴリズムは h を求めるために過去の事例として, N 人分の測定結果と, 実際に健康だったかどうかを示すラベルの組, すなわちデータの特徴ベクトルとクラスの組 $(\Phi(x^{(1)}), y^{(1)}), (\Phi(x^{(2)}), y^{(2)}), \ldots, (\Phi(x^{(N)}), y^{(N)})$(ただし $x^{(i)} \in X$, $y^{(i)} \in Y$) を訓練例として用いることができる. h の形としては, 最も簡単な識別器として, 重みベクトル $\mathbf{w} \in \Re^D$ および閾値 $b \in \Re$ を使って, 線形識別器

$$h(x) = \mathrm{sign}\left(\left\langle \mathbf{w}, \Phi(x) \right\rangle + b\right) \qquad (1)$$

を考えよう. ここで, $\langle \cdot, \cdot \rangle$ は内積, sign は引数の符号を返す関数であるとする.

与えられた事例から h を (すなわち $\mathbf{w}$ と b を) 学習するためのアルゴリズムとしては, パーセプトロンがよく知られている. パーセプトロンは, 初期値として $\mathbf{w} = \mathbf{0}$ および $b = 0$ から学習をスタートし, 訓練例をひとつづつ処理しながら学習を進めていく逐次型の学

† 日本アイ・ビー・エム株式会社 東京基礎研究所
IBM Research, Tokyo Research Laboratory

習アルゴリズムである.

例えば i 番目のデータ $x^{(i)}$ に対し, パーセプトロンはそのデータのクラス予測 $\hat{y}^{(i)} = h(x^{(i)})$ を行い, これが正しいクラス $y^{(i)}$ と異なったときだけ, 以下の規則にしたがって重みベクトルと閾値を更新する.

$$\begin{aligned} \mathbf{w} &\leftarrow \mathbf{w} + y^{(i)}\Phi(x^{(i)}) \\ b &\leftarrow b + y^{(i)}R^2 \end{aligned} \qquad (2)$$

ここで R は原点を中心として, データをすべて包含するような最小の球の半径である. 訓練データを完全に分類できる h が存在するとき, 必ずこの学習は収束する, すなわち誤りが 0 になることが知られている.

さて, ここでパーセプトロンを別の視点から眺めてみよう. 重みの更新式 (2) で $y^{(i)} \in \{+1, -1\}$ であることに注意すると, 更新の度に重みベクトルに特徴ベクトル $\Phi(x^{(i)})$ が加えられているか, 引かれているかのどちらかであることがわかる. 従って, 重みベクトルは特徴ベクトルの線形和によって

$$\mathbf{w} = \sum_{i=1}^{N} \alpha_i \Phi(x^{(i)})$$

のように表現できそうである☆. ここで, α_i は i 番目の訓練例の重みである. これを (1) の式に代入してみると, 次の式が得られる.

$$h(x) = \mathrm{sign}\left(\sum_{i=1}^{N} \alpha_i \left\langle \Phi(x^{(i)}), \Phi(x) \right\rangle + b\right) \qquad (3)$$

また更新式 (2) も, すべての i に対し $\alpha_i = 0$ から学習をスタートし,

$$\alpha_i \leftarrow \alpha_i + y^{(i)}$$

のように書きかえることができる.

さて, このように書き変えてみると, 予測や学習において, データの特徴ベクトルが単独で現れることはなく, 常に 2 つの特徴ベクトルの内積の形で現れているところに気づく. この内積を $K(x, x') = \left\langle \Phi(x), \Phi(x') \right\rangle$ と置き換え, **カーネル関数**と呼ぶことにすると, データアクセスの部分がすべてカーネル関数によって表現されることになる. このような学習アルゴリズムを総称して**カーネル法**と呼ぶ.

さて, このようにわざわざ内積をカーネル関数に置き換えて表現してみることで, 2 つの利点があることに気づく. 1 つ目は, 特徴ベクトルの次元が非常に高く (時には無限次元), 明示的に特徴ベクトルを構成して内積をとることが不可能であったとしても, 何らかの呪術的仕組みによって内積**だけ**を高速に計算できるブラックボックス K が存在すれば, 高速な学習や予測が可能である点である.

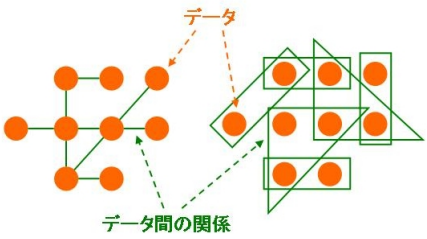

図 1 データの外的構造

また, もう 1 つは, 適当に定義された, 類似度を表す関数 K をカーネル関数として用いることが可能になる点である. この場合, K がカーネル関数になっている, すなわち特徴ベクトル内積であることを保証する必要があるが, そのためには K が半正定, すなわち K が対称 $K(x, x') = K(x', x)$ かつ,

$$\sum_{x \in X} \sum_{x' \in X} w(x) w(x') K(x, x') \geq 0$$

が任意の $\sum_{x \in X} w(x) < \infty$ である w について成立することが示せればよい.

また, いくつかのカーネル関数を足し合わせたり, 掛け合わせたりしたものもやはりカーネル関数になるため, 複数の情報源の情報を自然に統合して用いることができるという便利な特徴もある.

さて, ここで x として, DNA 配列などの文字列で表されるようなデータ, XML 文書や HTML 文書などの木構造をもったデータ, あるいは原子が共有結合によって結び付けられた化合物のようなグラフ構造をもったデータを扱いたい場合, これらをどのように表現したらよいだろうか? あるいは血縁関係や友人関係などの人と人との関係や, タンパク質とタンパク質の間の相互作用など, データそのものではなく, データ間の関係がわかっている場合, これらの情報をどのように学習に役立てたらよいだろうか? このようにデータを構成する要素間, あるいはデータ間になんらかの構造があるとき, 先ほどの例で出てきたようなベクトルによる表現 $\Phi(x)$ は自明ではない.

次節以降では, これら構造をもつデータを扱う問題に対して, カーネル法に基づくいくつかのアプローチを紹介する. 本解説では, 構造データに対するカーネル法の適用を 2 つのカテゴリーに分類して紹介する. ひとつは**外的構造** (図 1), すなわちデータの関係やデータの分布などの, データをとりまく世界の構造を用いてカーネル関数を設計する場合であり, もうひとつは**内的構造** (図 2), すなわちデータに含まれる部分構造を用いてカーネル関数を設計する場合である.

☆ 一般的には最適な $\mathbf{w}$ が特徴ベクトルの線形和であることは, Representer 定理と呼ばれる定理によって保証される.

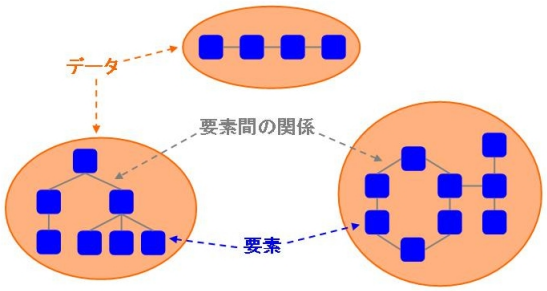

図 2 データの内的構造

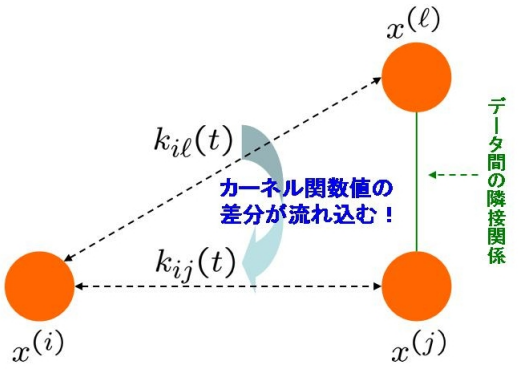

図 3 拡散カーネルの解釈

2. 外的構造を用いたカーネル関数

外的構造の与えられ方のひとつとしては, 直接関係のあるデータとデータの「隣接関係」が考えられる. 一般に, すべてのデータ同士の類似度を定義することが困難な場合でも, データ同士の隣接関係は比較的容易に定義できることが多い. 例えば, リンク構造でむすばれたハイパーテキスト, 人間間の関係を表現した社会ネットワーク, タンパク質の相互作用などは, こういった隣接関係の表現といえる. 隣接関係で表現されたデータ空間の構造は, データを頂点, 隣接関係を辺とした無向グラフ (図 1 左) によって自然に表現することができる☆.

これから紹介する**拡散カーネル**[2] は, 無向グラフによって表現された局所的な関係から, 全データ間のカーネル関数を定義することができるカーネル関数の設計法であり, 実際にテキスト分類や代謝パスウェイの解析などに応用され, 効果をあげている.

まず, 各要素 l_{ij} が i 番目のデータ $x^{(i)}$ と j 番目のデータ $x^{(j)}$ の局所的な関係をあらわす実対称行列 L が与えられるとする. L としては例えば, グラフスペクトルの解析において一般的に使われる量であるグラフのラプラシアンの符号を逆転したもの

$$l_{ij} = \begin{cases} 1 & (i \sim j \text{ のとき}) \\ -d_i & (i = j \text{ のとき}) \\ 0 & (\text{それ以外}) \end{cases}$$

が使われる[2]. ここで d_i は頂点 i の次数とし, $i \sim j$ は $x^{(i)}$ と $x^{(j)}$ との間に隣接関係があることを表すとする.

では, L で与えられた局所的な関係をもちいて, いかにして遠くはなれたデータ間の関係を定義できるだろうか？ 拡散カーネルは, 与えられた L を使って

$$K(t) = \exp(tL) = \lim_{n\to\infty} \left(1 + \frac{tL}{n}\right)^n$$

のように定義される. ここで, $t > 0$ は拡散の時間を表す定数パラメータである. $K(t)$ の (i, j) 要素 $k_{ij}(t)$ は $x^{(i)}$ と $x^{(j)}$ との間のカーネル関数の値である. このように定義された $K(t)$ は対称行列であり, 半正定であることが保証される.

さて, $K(t)$ は一体どのような意味をもっているのだろうか？ $K(t)$ も L も対称であることから, $K(t)$ は, 微分方程式

$$\frac{\partial}{\partial t}K(t) = \frac{1}{2}(LK(t) + K(t)L)$$

を $K(0) = I$ のもとで解いたときの解になっている. これを L が先ほどのグラフラプラシアンの符号を逆転したものであるときを例にとって解釈してみよう. 微分方程式の (i, j) 要素

$$\frac{\partial}{\partial t}k_{ij}(t) = \frac{1}{2}\left(\sum_{\ell:i\sim\ell}(k_{\ell j}(t) - k_{ij}(t)) + \sum_{\ell:j\sim\ell}(k_{i\ell}(t) - k_{ij}(t))\right)$$

に注目してみると, これは, $k_{ij}(t)$ が近傍の k(すなわち $k_{\ell j}(t)$ や $k_{i\ell}(t)$) との差に応じて, その差を少なくする方向に時間変化することを表している. 右辺第 1 項の心は「$x^{(i)}$ と $x^{(\ell)}$ は似ている (いない) とすると, $x^{(\ell)}$ と $x^{(j)}$ には隣接関係があるのだから, $x^{(i)}$ と $x^{(j)}$ だって似ている (いない) はずだ. 」という感じで, まさに $x^{(i)}$ と $x^{(\ell)}$ の類似度が $x^{(i)}$ と $x^{(j)}$ の類似度に「流れ込んでくる」と解釈できる. (図 3)

従って, 初期値 $K(0) = \exp(0 \cdot L) = I$, すなわちデータは自分自身とだけ似ているという状態からスタートして, L で表されたデータ間の局所的な関係を使ってデータ同士の類似度, すなわちカーネル関数の値が「伝播」あるいは「拡散」することによって直接の関係のないデータ間の類似度を定義していることがわかる. また,t が大きいほど局所的な類似度が遠くまで強く拡散することになる.

ところで, 拡散カーネルの計算を行うのは大変そう

☆ もっと一般的には, ハイパーグラフ (図 1 右) で表現できる

であるが, L を対角化して, $L = V^{-1}DV$ とすることで,

$$\exp(\beta L) = V^{-1}\exp(\beta D)V$$

と表すことが出来る. D は対角行列であるから $e^{\beta D}$ は対角成分のみが 0 以外の値を持ち, i 番目の対角成分は $e^{d_{ii}}$ によって計算することができる.

3. 内的構造を扱うカーネル法

では, データ自体が内部に構造をもつデータに場合にはどのようにカーネル関数を設計してやればよいだろうか? データの内的構造は一般的に, データに含まれる変数と変数の間の関係をグラフ, あるいはその部分クラスである配列や木で表すことで表現できる (図 2). 例えば, DNA 配列は塩基の隣接関係が配列で表現される. また, 自然言語処理で現れる構文解析木は単語や句をノードとし, その親子関係を辺で結んだ木の形で表される. 化合物は, 原子をノードとし, これらの間の共有結合を辺とすることでグラフで表現することができる. つまり, それぞれのデータは配列や木, あるいはグラフなどの構造データであり, 我々は 2 つの構造データの間のカーネル関数を定義すればよいことになる.

もしも事前知識として, データの確率的な生成モデル, 例えば配列データの場合にはマルコフモデルあるいは隠れマルコフモデル, 構文解析木データの場合には確率的文脈自由文法などが与えられているとすると, それらのモデルを利用して **Fisher カーネル**[1] と呼ばれるカーネル関数を構成することができる. 生成モデルが, パラメータ θ をもつ確率分布 $P(x|\theta)$ であるとし, 事前知識としてあるパラメータ $\theta = \theta^*$ で指定されるモデル $P(x|\theta^*)$ が与えられたとすると, Fisher カーネルはデータ x の対数尤度が最大の勾配を持つ方向ベクトル

$$\Phi(x) = \nabla_\theta \log P(x|\theta)|_{\theta=\theta^*}$$

を使って, $K(x,x') = \Phi(x)F^{-1}\Phi(x')^\top$ と定義される. ここで $F := E_\theta[\Phi(x)^\top\Phi(x)]$ は Fisher 情報行列と呼ばれる☆. 直感的には $\Phi(x)$ の各次元は与えられたモデルの各パラメータが x を生成するのに, どれだけ貢献しているかの度合いを表す.

では, このようにデータを生成する確率モデルが与えられない場合はどうしたらよいだろうか? 配列データの場合で考えてみよう. 配列データの確率モデルとしては n グラムモデル (マルコフモデル) がよく使われる. n グラムモデルは連続する $n-1$ 文字によって, 後に続く 1 文字の確率分布が決まるとするモデルだが, これはすなわち連続する部分文字列が配列の特徴を表現できると仮定しているといえる. この考え方を一般化すると, 構造をもったデータは, その部分構造をもって特徴を表現できそうである. この心をカーネル設計の一般的な枠組みとしてあらわしたのが Haussler によって提唱された**畳み込みカーネル**である. 畳み込みカーネルは一般的に

$$K(x,x') = \sum_{s\in S(x)}\sum_{s'\in S(x')} f(s|x)f(s'|x')K_S(s,s')$$

と定義される☆☆. ここで $S(x)$ は x からとりだされる部分構造の集合を表し, K_S は 2 つの部分構造の間に定義されるカーネル関数であるとする. また, $f(s|x)$ は x から取り出された部分構造 s の重みを表す関数である. 畳み込みカーネルは, x と x' から取り出された部分集合 $S(x)$ と $S(x')$ を取り出し, それらの間のカーネル関数値をすべて足し合わせることで定義される. 2 つのデータの類似度が, データの部分構造の類似度の和によって定義されるのである. (図 4)

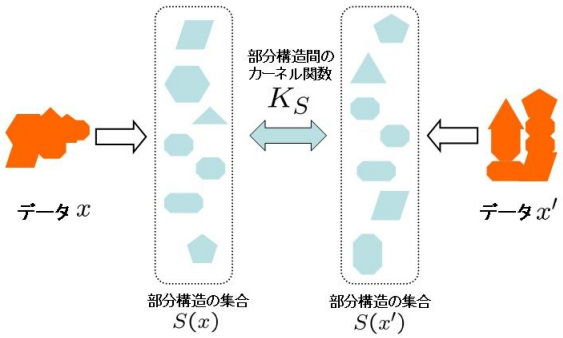

図 4 畳み込みカーネルの概念図

畳み込みカーネル自体はあくまで枠組みであり, 扱いたいデータに応じて部分構造 $S(x)$ とカーネル関数を効率よく計算するためのアルゴリズムの設計が必要である. これまでに配列や木, グラフなど様々な構造を持つデータに対するカーネル関数とアルゴリズムが提案されている[1),5)].

3.1 配列同士の畳み込みカーネル[1),5)]

では, 具体例として配列に対する畳み込みカーネルを考えてみよう. それぞれ長さ ℓ と ℓ' の 2 つの配列を $x = (x_1, x_2, \ldots, x_\ell)$ および $x' = (x'_1, x'_2, \ldots, x'_{\ell'})$ とする. また Σ をラベルの集合 (アルファベット) とし, $\sigma(x_i) \in \Sigma$ は配列の各要素のラベルを表すとする.

部分構造の集合 $S(x)$ としては, x の長さ n のすべての部分配列の集合 $S^{(n)}(x)$ をとる. すなわち, 配列

☆ 実際には $F^{-1} = I$ と置いてしまうことが多い.

☆☆ オリジナルの畳み込みカーネル[3)] では部分構造の重み f は用いられてないが, ここでは部分構造の重みを使った**周辺化カーネル**[1),4)] の形で書いた.

を長さ n の部分配列の集合として捉えようというのである. ただし, 長さ n の部分配列は単に連続した部分配列 (substring) だけでなく, 飛び飛びになってもよい部分配列 (subsequence) も許すことにしよう. 例えば, 4 つの配列"able to", "can", "cannot", "can't" があり, これらを「できる」「できない」という意味に分類したいとする. だが, これらを上手くひとつで分類できる連続した部分文字列はなさそうである. しかし, 間に何文字か入ることを許せば"cant"という文字列は, "cannot"の中に"cannot"あるいは"cannot"という形で, "can't"の中には, "can't"という形でそれぞれ出現しているということができる. 一方, "able to"や"can"の中にはこれは出現しない. 従って, このような飛び飛びを許した柔軟なマッチングを行うことで「できる」と「できない」を分類する特徴として"cant"を見つけることができるというわけである.

しかしながら, ほぼ連続して現れる部分配列と, 非常に広い範囲にまたがって現れる部分配列をまったく同じに扱うのは少々気が引ける. そこで部分配列 s の重み $f(s|x)$ を s が飛び飛びになっているほど小さくなるように定義する. ここでは, 小さい定数 $\lambda > 0$ を使って, s が長さ $g(s)$ の領域 ($g(s) \geq n$) にわたるとき, その重み $f(s|x) = \lambda^{g(s)}$ と定義する. 先ほどの例では, $x =$"can't"からは, $s = (x_1, x_2, x_3, x_5)$ を取り出すことで, "cant"というラベル列が見つかるが, s の領域は長さ $g(s) = 5$ にわたるから, $f(s|x) = \lambda^5$ となる.

従って, 配列の畳み込みカーネルは

$$K(x, x') = \sum_{s \in S^{(n)}(x)} \sum_{s' \in S^{(n)}(x')} \lambda^{g(s)} \lambda^{g(s')} K_S(s, s') \quad (4)$$

のように書くことが出来る. ここで, 2 つの部分配列 $s = (s_1, s_2, \ldots, s_n)$ と $s' = (s'_1, s'_2, \ldots, s'_n)$ の間のカーネル関数は

$$K_S(s, s') = \prod_{i=1}^{n} K_\Sigma(\sigma(s_i), \sigma(s'_i)) \quad (5)$$

のようにラベルごとのカーネル関数 K_Σ の積で定義されるとする. 例えば $\sigma, \sigma' \in \Sigma$ に対し, ラベルごとのカーネル関数を

$$K_\Sigma(\sigma, \sigma') = \begin{cases} 1 & (\sigma = \sigma' \text{のとき}) \\ 0 & (\sigma \neq \sigma' \text{のとき}) \end{cases} \quad (6)$$

と定義したとすると, $K_S(s, s')$ は 2 つの部分列 s と s' に対応するラベル列が同一の時に 1, そうでないとき 0 となる☆.

さて, 配列の畳み込みカーネルを計算するには, まず x と x' から長さ n の部分配列を全て取り出してきて計算すればよいかもしれないが, その数は ${}_\ell C_n$ と ${}_{\ell'} C_n$ と膨大な数になってしまうという計算量的な問題がある. そこで, 動的計画法を使って効率的に計算する方法を示そう.

まず $S_t^{(n)}(x)$ を, x の長さ n の部分配列で, 右端が x_t である (つまり右端が x の t 番目の位置に出現する) ものの集合と定義すると, (4) は,

$$K(x, x') = \sum_{t=1}^{\ell} \sum_{t'=1}^{\ell'} k^{(n)}(t, t')$$

$$k^{(n)}(t, t') := \sum_{s \in S_t^{(n)}(x)} \sum_{s' \in S_{t'}^{(n)}(x')} \lambda^{g(s)} \lambda^{g(s')} K_S(s, s') \quad (7)$$

と書き直せる. (7) は部分配列の右端を x_t および x'_t に固定したときのカーネル関数である.

さて, x_t を右端にもつ長さ n の部分配列は, x_t より左方に右端にもつ長さ $n-1$ の部分配列に x_t を追加することで構成できる点と (5) に注意すると, (7) は

$$k^{(n)}(t, t') = \sum_{\tau=1}^{t-1} \sum_{\tau'=1}^{t'-1} \lambda^{t-\tau} \lambda^{t'-\tau'} k^{(n-1)}(\tau, \tau') K_\Sigma(\sigma(x_t), \sigma(x'_{t'}))$$

のように n,t,t' について再帰的に書ける. この時点で $O(n\ell^2\ell'^2)$ の計算量になるが, さらにもう少しだけ工夫して

$$J^{(n)}(t, t') := \sum_{\tau=1}^{t-1} \sum_{\tau'=1}^{t'-1} \lambda^{t-\tau} \lambda^{t'-\tau'} k^{(n-1)}(\tau, \tau')$$

と定義すれば, 再帰式

$$J^{(n)}(t, t') = \lambda J^{(n)}(t-1, t') + \lambda J^{(n)}(t, t'-1) - \lambda^2 J^{(n)}(t-1, t'-1) + k^{(n-1)}(t-1, t'-1)$$

が成立するので, これと,

$$k^{(n)}(t, t') = J^{(n)}(t, t') K_\Sigma(\sigma(x_t), \sigma(x'_{t'}))$$

を使うことで, $O(n\ell\ell')$ で効率的に計算できる.

3.2 グラフ同士の畳み込みカーネル[4),5)]

今度は 2 つのグラフ同士の畳み込みカーネルを考えよう. ここでは各データはノードにラベルのついた無向グラフであるとする. すなわちグラフ $x = (V, E)$ は頂点の集合 V と辺の集合 E からなり, 各頂点 $v \in V$ にはラベル $\sigma(v) \in \Sigma$ が振られているとする. また, 簡

☆ 因みに, 部分配列が飛び飛びになることを許さず, かつ (6) を仮定する場合には, 接尾辞木を使うことで配列の畳み込みカーネルを線形時間で計算する方法が存在する[1)].

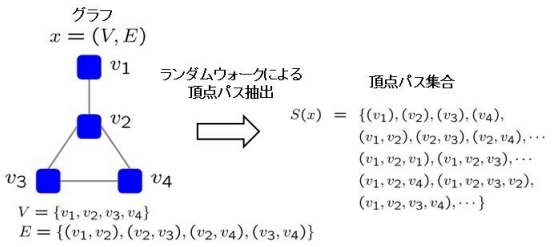

図 5 グラフ同士の畳み込みカーネル

単のため 2 つの頂点を直接結ぶ辺の数は高々1 本であると仮定する. 例えば化合物は原子を頂点, 原子の種類をそのラベル, 原子間の共有結合を辺とすると, ノードにラベルのついた無向グラフとして表現できる☆.

部分構造の集合 $S(x)$ としては, 配列カーネルの設計で使った部分配列の考え方を拡張すると, グラフに含まれる頂点パスの集合を $S(x)$ として使えそうである. そこで, ここでは $S(x)$ は x の上でランダムウォークをすることによって得られうる頂点の列の集合であると定義する (図 5). ランダムウォークは, ランダムに選んだ頂点からスタートし, 現在の頂点と隣接する頂点の中からランダムにひとつを選び, その頂点に移動するものとする. このようにして辿った頂点の列, すなわち頂点パス $s = (s_1, s_2, \ldots, s_n)$ $(s_i \in V)$ が得られる.

また, 長さ n の頂点パス s に与えられる重み $f(s|x)$ は, $0 < \lambda < 1$ である十分小さい定数を使って, 頂点パスの長さに応じて $f(s|x) = \lambda^n$ と定義する.

2 つの頂点パス $s = (s_1, s_2, \ldots, s_n)$ と $s' = (s'_1, s'_2, \ldots, s'_n)$ の間のカーネル $K_S(s, s')$ は, $n = n'$ のときは (5) と同様にラベル毎のカーネルの積になるとし, $n \neq n'$ のときは $K_S(s, s') = 0$ となるとする.

さて, こうして定義した 2 つのグラフのカーネルを計算したいが, ランダムウォークによって生成される頂点パスの集合は無限にあるので, パスを明示的に列挙して計算したのでは計算が終わらない. そこで, 配列同士のカーネルを計算したときのように再帰的な表現を試みることにしよう.

配列のカーネルのときと同じように頂点 v を終点とする頂点パスの集合 $S_v(x)$ を定義すると,

$$k(v, v') = \sum_{s \in S_v(x)} \sum_{s' \in S_{v'}(x')} f(s|x) f(s'|x') K_S(s, s') \quad (8)$$

とすることでカーネル関数は,

$$K(x, x') = \sum_{v \in V} \sum_{v' \in V'} k(v, v')$$

とかける. v を終点とする頂点パスの集合は, ランダムウォークが v でスタートした途端に終了する場合 (v のみを含む頂点パスを生成する) と, 隣接した頂点で終了する頂点パスに v を足して作られる頂点パス集合の和集合なので, やはり (8) は

$$k(v, v') = \lambda^2 K_\Sigma(\sigma(v), \sigma(v')) \left(1 + \sum_{\tilde{v} \in N(v)} \sum_{\tilde{v}' \in N(v')} k(\tilde{v}, \tilde{v}')\right)$$

のように再帰的に書くことができる. なお $N(v)$ は頂点 v に隣接する頂点の集合とする. これは k について連立方程式の形をしているので, 頂点数に関して多項式時間で解くことが可能になる.

4. 構造カーネル法の今後

本解説では, 従来のデータマイニングや機械学習の手法ではうまく扱うことができなかった, 内的・外的な様々な構造を持つデータをカーネル法を用いてうまく扱うための基本的な手法をいくつか紹介した.

紙面の関係上, すべての手法を網羅的に紹介することはできなかったが構造データを扱うカーネル法のサーベイ及び解説としては,[1),5),6)] あたりが詳しいので興味を持たれた方はそちらを参照していただきたい.

今回, 教師ありの分類問題を念頭において話を進めたが, 構造カーネル法は教師ありの分類問題に限らず回帰問題や順序付けの問題, あるいは教師無し学習や強化学習の問題など, 実に多くの問題に適用可能であり, 毎年多くの研究成果が発表されている.

特に近年, X だけでなく Y のほうにも構造があるような学習問題が注目を集めている[7),8)]. X と Y が同一の構造をもつ場合などのように Y の構造が予め与えられている場合には, Y に含まれる各変数に対してラベルを割り当てていくような問題となる. このような問題は構造ラベリングの問題と呼ばれ, 自然言語処理においての単語品詞付け問題や固有表現抽出問題, またバイオインフォマティクスにおいてのタンパク質のアミノ酸配列の 2 次構造予測問題などがこれに当たる. また, より一般的には Y の持つ構造も予測する問題が考えられる. 例えば X が配列構造を持ち, Y が木構造持つようなケースとして, 構文解析の問題などがある. これらの問題は, 一般的には構造間のマッピングを学習するような問題として捉えることができる. そして現在, このような問題に対しても, 構造カーネル法を適用できるように拡張していこうという試みが始まっている[9)].

カーネル法は, ともすれば明示的には扱えないほど高次元の表現になってしまう複雑な構造を, カーネル関数の形で隠蔽し, 美しく, かつ効率的に扱うことが出来るという非常に魅力的な手法である. この解説に

☆ 共有結合にも種類があるので, 本当は辺にもラベルがあるのだが, ここでは簡単のため無視する.

よって読者が少しでもカーネル法に興味を持っていただければ幸いである．

参 考 文 献

1) Shawe-Taylor, J. and Cristianini, N: *Kernel Methods for Pattern Analysis*, Cambridge University Press (2004).
2) Kondor, R. I. and Lafferty, J.: Diffusion Kernels on Graphs and Other Discrete Input Spaces, *Proceedings of the 19th International Conference on Machine Learning* , pp. 315–322 (2002).
3) Haussler, D.: Convolution Kernels on Discrete Structures, Technical Report UCSC-CRL-99-10, University of Calfornia in Santa Cruz (1999).
4) Kashima, H., Tsuda, K., and Inokuchi, A.: Marginalized Kernels for Labeled Graphs, *Proceedings of the 20th International Conference on Machine Learning*, pp. 321–328 (2003).
5) Schölkopf, B., Tsuda, K., and Vert, J.-P.: *Kernel Methods in Computational Biology*, MIT Press (2004).
6) Gärtner, T.:A Survey of Kernels for Structured Data, *SIGKDD Explorations*, Vol. 5, No. 1, pp. S268–S275 (2003)
7) Weston, J., O. Chapelle, A. Elisseeff, Schölkopf, B., and Vapnik, V.: Kernel Dependency Estimation, *Advances in Neural Information Processing Systems 15*, pp. 873–880 (2003).
8) Tsochantaridis, I., Hofmann, T., Joachims, T., and Altun, Y.: Support Vector Machine Learning for Interdependent and Structured Output Spaces, *Proceedings of the 21th International Conference on Machine Learning*, pp. 823–830 (2004).
9) Kashima, H., and Tsuboi, Y.: Kernel-based Discriminative Learning Algorithms for Labeling Sequences, Trees and Graphs, *Proceedings of the 21th International Conference on Machine Learning*, pp. 457–464 (2004).