カーネル法

鹿島 久嗣*

1 カーネル法

カーネル法 [11] とは（データ集合を $\mathcal{X}$ とするとき）2 つのデータの間のある種の類似度を表す「カーネル関数」$k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ を通じてデータにアクセスするような学習モデルである。より具体的には、n 個のデータ $x^{(1)}, x^{(2)}, \ldots, x^{(n)} \in \mathcal{X}$ が与えられたときに、モデル $f(\cdot)$ が、

$$f(\cdot) = g\left(\sum_{i=1}^{n} \alpha^{(i)} k(\cdot, x^{(i)})\right) \tag{1}$$

のような形、つまり、カーネル関数の線形結合の関数として表現されるようなモデルを扱うのがカーネル法である。なお、g はある関数、また $\{\alpha^{(i)}\}_{i=1}^{n}$ はモデルパラメータとする。カーネル関数はどのような関数でもよいというわけではなく、内積として書くことができる必要がある。つまり、データ $x^{(i)}$ の d 次元の特徴空間中でのベクトル表現を $\boldsymbol{\phi}(x^{(i)})$ とすると、i 番目と j 番目のデータの間のカーネル関数が

$$k(x^{(i)}, x^{(j)}) = \left\langle \boldsymbol{\phi}(x^{(i)}), \boldsymbol{\phi}(x^{(j)}) \right\rangle$$

のように定義されている必要がある。

式 (1) で注目すべきは、モデルがデータにアクセスする際には常にカーネル関数を経由している点である。つまり、モデルの学習時にも、モデルを用いた予測時にも、d 次元の特徴ベクトル $\boldsymbol{\phi}(x^{(i)})$ が直接的に扱われることはなく、データは必ず 2 つのデータのカーネル関数を通して扱われる。

カーネル法の例としては、何と言ってもサポートベクトルマシン [2] が代表的であるが、多くの “古典的な”機械学習法のカーネル法版が提案されている [11]。まずは、2 値分類のための教師付き学習法であるパーセプトロンのカーネル化 [3] を通して、カーネル法の具体例を紹介する。まず、n 個の訓練データ集合、すなわちデータの特徴ベクトルと対応する出力の組 $(\boldsymbol{\phi}(x^{(1)}), y^{(1)}), (\boldsymbol{\phi}(x^{(2)}), y^{(2)}), \ldots, (\boldsymbol{\phi}(x^{(n)}), y^{(n)})$ が与えられているとする。ここで、$\boldsymbol{\phi}(x^{(i)})$ は、それぞれ d 次元の実数値ベクトル、つまり $\mathcal{X} = \mathbb{R}^d$ であるとする。また、$y^{(i)} \in \{+1, -1\}$、すなわち出力集合が $\mathcal{Y} = \{+1, -1\}$ であるとする。パーセプトロンのモデル f の形は、パラメータベクトル $\boldsymbol{w} \in \mathbb{R}^d$ および閾値 $b \in \mathbb{R}$ として

$$f(x) = \mathrm{sign}\left(\left\langle \boldsymbol{w}, \boldsymbol{\phi}(x) \right\rangle\right) \tag{2}$$

のように線形識別器として表される。なお、sign は引数の符号（$+1$ か -1）を返す関数であるとする。パーセプトロンの学習は、まず初期パラメータとして $\boldsymbol{w} = \boldsymbol{0}$ から学習をスタートし、その後、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習アルゴリズムである。例えば i 番目のデータ $x^{(i)}$ を処理するとき、まずパーセプトロンはそのデータの出力を現在のモデルを用いて $y^{(i)} = f(x^{(i)})$ によって予測する。これが正しい出力 $y^{(i)}$ と異なったときに限り、以下の規則にしたがってパラメータベクトルを更新する。

$$\boldsymbol{w} \leftarrow \boldsymbol{w} + y^{(i)} \boldsymbol{\phi}(x^{(i)}) \tag{3}$$

訓練データを完全に分類できるパラメータ $\boldsymbol{w}^*$ 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。

ここでパーセプトロンをカーネル法の視点から捉えなおしてみる。パラメータの更新式 (3) において $y^{(i)} \in \{+1, -1\}$ であることに注意すると, 更新の度にパラメータベクトルに特徴ベクトル $\boldsymbol{\phi}(x^{(i)})$ が加えられているか、もしくは引かれているかのどちらかであることがわかる。従って、パラメータベクトルは特徴ベクトルの線形和によって

$$\boldsymbol{w} = \sum_{i=1}^{n} \alpha^{(i)} \boldsymbol{\phi}(x^{(i)}) \tag{4}$$

のように表現できるはずであることがわかる。ここで、$\alpha^{(i)}$ は i 番目の訓練例の重みである。これを (2) の式

*東京大学 情報理工学系研究科 数理情報学専攻

に代入してみると、次の式が得られる。

$$f(x) = \mathrm{sign}\left(\sum_{i=1}^{n} \alpha^{(i)} \left\langle \boldsymbol{\phi}(x^{(i)}), \boldsymbol{\phi}(x) \right\rangle\right) \tag{5}$$

また、学習も、すべての i について $\alpha^{(i)} = 0$ から学習を開始して、更新式 (3) は

$$\alpha^{(i)} \leftarrow \alpha^{(i)} + y^{(i)} \tag{6}$$

のように書きかえることができる。

式 (1) と式 (5) を見比べてみることで、$g(\cdot) = \mathrm{sign}(\cdot)$ と $k(x^{(i)}, x) = \left\langle \boldsymbol{\phi}(x^{(i)}), \boldsymbol{\phi}(x) \right\rangle$ とすれば、両者が対応していることがわかる。カーネル関数が内積となっていることもここからわかる。また、学習の更新式 (6) も、$\alpha^{(i)}$ に $+1$ もしくは -1 を加えているだけであり、特徴ベクトル $\boldsymbol{\phi}^{(i)}$ は直接的に扱っていないことがわかる。

2 カーネル関数

カーネル法の重要な特徴として、カーネル法ではカーネル関数の上での線形モデルを考えている点が挙げられる。これは、本来の特徴空間におけるモデルが、特徴空間の上の線形モデルを明示的に考えていたのとは対照的である。つまり、たとえば、データ x がグラフや時系列であったりなどの何らかの理由により、明示的に特徴ベクトル $\boldsymbol{\phi}(x)$ を構成することが自明ではなくとも、カーネル関数 k として適当な類似度関数を構成できれば、特徴空間を経由せずに学習を実現できることになる。ただし、そのためには先に述べたようにカーネル関数が特徴ベクトルの内積として解釈できることが保証される必要がある。そのためには、カーネル関数が半正定、すなわち k が対称 $k(x, x') = k(x', x)$ かつ、

$$\sum_{x \in \mathcal{X}} \sum_{x' \in \mathcal{X}} \beta(x)\beta(x')k(x, x') \geq 0$$

が任意の $\sum_{x \in \mathcal{X}} \beta(x) < \infty$ である w について成立することが示せればよい。この条件は Mercer の定理として知られている。

カーネル法において、よく用いられるカーネル関数としては、m 次の多項式カーネル

$$k(x, x') = \left\langle \boldsymbol{\phi}(x), \boldsymbol{\phi}(x)' \right\rangle$$

や、幅 $\sigma > 0$ をもつガウスカーネル

$$k(x, x') = \exp\left(-\frac{\| \boldsymbol{\phi}(x) - \boldsymbol{\phi}(x)' \|^2}{\sigma}\right)$$

などがある。多項式カーネルの場合には、本来の ($\boldsymbol{\phi}$ の) 特徴の全ての m 個の組み合わせによる特徴空間が、、ガウスカーネルの場合には無限次元の特徴空間が暗黙のうちに構成され、カーネル関数はこれらの空間における内積として解釈できる。

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、カーネル関数を合成して新しいカーネル関数を構成することができる。たとえば、2 つのタンパク質の間に相互作用関係があるかどうかなどの、2 つのデータ $(x^{(i)}, x^{(j)})$ の組み合わせに対して予測を行うような問題はデータのペア同士のカーネル関数 $k^{\mathrm{pair}}((x^{(i)}, x^{(j)}), (x^{(k)}, x^{(\ell)}))$ を用意する必要がある。このとき、2 つのデータ間のカーネル関数 k を元にして、

$$\begin{aligned} &k^{\mathrm{pair}}((x^{(i)}, x^{(j)}), (x^{(k)}, x^{(\ell)})) \\ &= k(x^{(i)}, x^{(k)})k(x^{(j)}, x^{(\ell)}) + k(x^{(i)}, x^{(\ell)})k(x^{(j)}, x^{(j)}) \end{aligned}$$

のように構成することができる。

3 様々な機械学習法のカーネル法化

パーセプトロンのほかにも、主成分分析、判別分析、正準相関分析、ロジスティック回帰など、様々な機械学習法がカーネル法化されている。

例えば、通常の主成分分析では、データの特徴ベクトルを並べた行列 $\Phi = (\boldsymbol{\phi}^{(1)}, \boldsymbol{\phi}^{(2)}, \ldots, \boldsymbol{\phi}^{(n)})$ として

$$\begin{aligned} \Phi\Phi^\top \boldsymbol{w} &= \lambda \boldsymbol{w} \\ \boldsymbol{w}^\top \boldsymbol{w} &= 1 \end{aligned}$$

を満たす $\boldsymbol{w}$ と λ を固有値問題を解くことによって求めるが、カーネル主成分分析では、やはり式 (4) を仮定することによって、

$$\begin{aligned} K\boldsymbol{\alpha} &= \lambda \boldsymbol{\alpha} \\ \boldsymbol{\alpha}^\top K \boldsymbol{\alpha} &= 1 \end{aligned}$$

のように書きなおすことができる [10]。求まった $\boldsymbol{\alpha} = (\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(n)})^\top$ を用いて、データ x に対する主成分は $\sum_{i=1}^{n} \alpha^{(i)} k(x, x^{(i)})$ として得ることができる。

ここで、行列 K は、要素 $[K]_{i,j} = k(x^{(i)}, x^{(j)})$ をもつような行列で、カーネル行列と呼ばれる。

判別分析（フィッシャー判別分析）では目的関数

$$J(\boldsymbol{w}) = \frac{\boldsymbol{w}^\top S_1 \boldsymbol{w}}{\boldsymbol{w}^\top S_2 \boldsymbol{w}}$$

を最小化するような $\boldsymbol{w}$ を求める。ここで S_1 はクラス間共分散行列、S_2 はクラス内共分散行列と呼ばれる行列である。主成分分析と同様に、式 (4) を用いることで

$$J(\boldsymbol{w}) = \frac{\boldsymbol{\alpha}^\top M \boldsymbol{\alpha}}{\boldsymbol{\alpha}^\top N \boldsymbol{\alpha}}$$

のようにカーネル法版の判別分析の目的関数が導かれる [9]。ここで、2 つの行列 M と N はカーネル行列から決まる行列である。この目的関数を最小化するパラメータ $\boldsymbol{\alpha}$ は、以下の一般化固有値問題を解くことによって求める。

$$M\boldsymbol{\alpha} = \lambda N \boldsymbol{\alpha}$$

以上の例で見たように、モデル中の $\langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle$ を、式 (4) によってカーネル関数を用いて書きかえることで、カーネル関数の線形結合を導くのが標準的なカーネル法の構成法である。式 (4) の仮定が成り立つためには、最適なパラメータ $\boldsymbol{w}^*$ が特徴ベクトルの線形和によって書けることが保証される必要がある。一般に、目的関数が

$$J(\boldsymbol{w}) = \sum_{i=1}^{n} l(y^{(i)}, \langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle) + r(\| \boldsymbol{w} \|_2)$$

と表されるとする。なお、l は正しい出力 $y^{(i)}$ とモデルの出力 $\langle \boldsymbol{w}, \boldsymbol{\phi}(x) \rangle$ の食い違いを評価する任意の損失関数、r は単調増加関数とする。このとき、この目的関数を最小化するパラメータは式 (4) の形で表される。この定理は表現定理と呼ばれ、カーネル法を支える根拠となっている。

4 カーネル関数の設計

文書や DNA 配列などのように文字列で表されるようなデータ、XML 文書や HTML 文書、RNA 構造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、これらに対して適切にカーネル関数を設計することで、そのままカーネル法化されたアルゴリズムが適用できるところがカーネル法の利点である。

ここでは、これら構造をもったデータに対象に対して提案されているカーネル関数を紹介する。構造をもったデータに対する特徴定義として自然なものとしては、その部分構造が考えられるであろう。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畳み込みカーネル [5] である。畳み込みカーネルは一般的に

$$k(x, x') = \sum_{s \in \mathcal{S}(x)} \sum_{s' \in \mathcal{S}(x')} k_{\mathcal{S}}(s, s')$$

と定義される。ここで $\mathcal{S}(x)$ は x からとりだされる部分構造の集合を表し、$k_{\mathcal{S}}$ 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。畳み込みカーネルは, x と x' から取り出された部分集合 $\mathcal{S}(x)$ と $\mathcal{S}(x')$ 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。

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 $\mathcal{S}(x)$ 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 [4]。

たとえば、x が配列データの場合、$\mathcal{S}(x)$ は、x 中に含まれるすべての部分配列（x =ABCD における AC のように間があいていて出現していてもよい）とする。また、こうしてそれぞれの配列データから取り出された部分配列間のカーネル関数を $k_{\mathcal{S}}(s, s') = \delta(s = s')\lambda^{\ell(s,s')}$ のように定義する。ここで $\delta(s = s')$ は、s と s' の文字が一致するなら 1 を、そうでないなら 0 を返す関数とする。つまり、2 つの部分配列が一致するときのみ 0 以外の値をとる。また、λ は 0 から 1 の間の定数、$\ell(s, s')$ を、s が x 内で出現した際に占めた長さと、s' が x' 内で出現した際に占めた長さの和とする。たとえば、上の例の場合、部分配列 AC は x =ABCD において長さ 3 の位置を占めている。これは出現領域の長さについて減衰するように重みを与えていることになり、λ はその減衰の程度を指定している。

こうして取り出される部分配列の数は、配列の長さに関して指数的であるため、カーネル関数を単純に計算することは非常に困難である。しかし、部分配列の再帰性を利用した、動的計画法によって、2 つの配列の長さの積に比例する時間で効率的に計算することができる [8]。

x が木構造データの場合には、$\mathcal{S}(x)$ は、x 中に含まれるすべての部分木（部分グラフ）として定義される。

この場合も、木に含まれる部分木の数は膨大であるが、順序木の場合には、部分木の再帰的構造を利用することで、やはり動的計画法によって効率的に計算することが可能になる [1, 6]。

最後に、x がグラフ構造をもつデータの場合を考える。これまでの延長線上で考えれば $\mathcal{S}(x)$ として部分グラフを用いるのが自然であるが、残念ながら、この場合、カーネル関数を効率よく計算することはできない（NP 困難）ため、より限定された部分構造を用いる必要がある。たとえば $\mathcal{S}(x)$ を、x 上のランダムウォークによって生成されるパスの集合とすると、計算が効率よく行えるようになる [7]。

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 $x = (\mathcal{V}(x), \mathcal{E}(x))$ は頂点の集合 $\mathcal{V}(x)$ と辺の集合 $\mathcal{E}(x)$ からなり、各頂点 $v \in \mathcal{V}(x)$ にはラベルが振られているものとする。このグラフの上でランダムウォークを行うことで辿った頂点の列からラベルの列が得られる。すなわち一様に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一様に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。2 つのグラフ x と x' から取り出された長さ ℓ の 2 つのパス s と s' の間のカーネル関数 $k_{\mathcal{S}}(s, s')$ は、2 つのパスのラベルが完全に一致するときに λ^{ℓ}、そうでないとき 0 であるとする。こうして定義されたカーネル関数を実際に計算するにあたって、パスの個数は無限個あるため総当たり的な計算は不可能であるが、やはりパスの持つ再帰的な構造を利用することで効率的に計算することが可能になる。頂点 v を終点とする頂点パスの集合 $\mathcal{S}_v(x)$ を定義すると、

$$k_{\mathcal{V}}(v, v') = \sum_{s \in \mathcal{S}_v(x)} \sum_{s' \in \mathcal{S}_{v'}(x')} k_{\mathcal{S}}(s, s') \qquad (7)$$

とすることでカーネル関数は、

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

と書くことができる。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であるため式 (7) は

$$k_{\mathcal{V}}(v, v') = \lambda \delta(v = v') \left(1 + \sum_{\tilde{v} \in \mathcal{N}(v)} \sum_{\tilde{v}' \in \mathcal{N}(v')} k_{\mathcal{V}}(\tilde{v}, \tilde{v}') \right)$$

のように再帰的に書くことができる。なお、$\delta(v = v)$ は、v と v' のラベルが一致するときに 1、そうでないときに 0 を返すような関数であるとする。また $\mathcal{N}(v)$ は頂点 v に隣接する頂点の集合とする。この式は $k_{\mathcal{V}}$ 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。

References

[1] M. Collins and N. Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*, Cambridge, MA, 2002. MIT Press.

[2] C. Cortes and V. N. Vapnik. Support vector networks. *Bioinformatics*, 20:273–297, 1995.

[3] Y. Freund and R. Shapire. Large margin classification using the perceptron algorithm. *Machine Learning*, 37(3):277–296, 1999.

[4] T. Gärtner. A survey of kernels for structured data. *SIGKDD Explorations*, 5(1):S268–S275, 2003.

[5] D. Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California in Santa Cruz, 1999.

[6] H. Kashima and T. Koyanagi. Kernels for semi-structured date. In *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 291–298, San Francisco, CA, 2002. Morgan Kaufmann.

[7] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, San Francisco, CA, 2003. Morgan Kaufmann.

[8] H. Lodhi, C. Saunders, J. Shawe-Taylor, N. Cristianini, and C. Watkins. Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444, 2002.

[9] S. Mika, G. Ratsch, J. Weston, B. Scholkopf, and K. Muller. Fisher discriminant analysis with

kernels. *Neural networks for signal processing IX*, pages 41–48, 1999.

[10] B. Schölkopf, A. Smola, and K.-R. Müller. Nonlinear component analysis as a kernel eigenvalue problem. *Neural Computation*, 10(5):1299–1319, 1998.

[11] J. Shawe-Taylor and N. Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.