The Elements of Statistical Learning
第 6 章 カーネル平滑法

鬼頭幸助

2020 年 8 月 30 日

局所化 (localization) を考える。(今日は、「局所」という言葉が重要なキーワード。) すなわち、ある点における値を、その点に近い訓練データの値を使って予測する手法を見ていく。基本的なコンセプトとしては、近いもの程大きな重みを与え、遠くでは 0 に収束するような重み関数 (核、カーネル (kernel) と呼ぶ)$K_\lambda(x_0, x_i)$ を考え、損失関数として、重みを掛けた関数を取り、推定値を取り出す。一般的に、カーネルはパラメータ λ を含み、この値によって裾野の広さを調整する。

裾野が狭い = ごく一部の訓練データをより優遇する = 訓練データに強く適合する = 自由度が高い

なので、パラメータ λ で実質的な自由度 (efective digree of freedom) を支配できると思ってよい。

先週も出てきたように、“kernel” という言葉が別の意味で使われることもあるので、注意。(共通点として、「重み付ける関数」という理解は良いかも。)

6.1　1 次元のカーネル平滑

まずは、k 近傍法を発展させることを考える。k 近傍法は、以下の式で推定値を導いた。

$$\hat{f}(x) = \mathrm{Ave}(y_i \mid x_i \in N_k(x))$$

これは、$\mathrm{E}[Y \mid X = x]$ のを推定するために、$X = x$ の時の Y の分布を、訓練データのうち、x に近いもの k 個で代替し、期待値の計算を単純な算術平均で代替した、という解釈をした。

この計算方法の結果は、不必要にギザギザになり、推定値も不連続になる。そこで、「単純な算術平均」の代わりに「重み付き平均」を使っていい感じにしよう、というのが、**Nadaraya-Watson 重み付きカーネル平均** (kernel-weighted average) という方法。適当なカーネル $K_\lambda(x_0, x_i)$ をつかって、以下の式で推定値を計算する。(複雑な式に見えるけど、ただの重み付き平均)

$$\hat{f}(x_0) = \frac{\sum_{i=1}^{N} K_\lambda(x_0, x_i) y_i}{\sum_{i=1}^{N} K_\lambda(x_0, x_i)}$$

カーネルとしてよく使われるものは、以下のようなものがある。

- Epanechnikov の 2 次カーネル

$$K_\lambda(x_0, x) = D\left(\frac{|x - x_0|}{\lambda}\right),$$
$$D(t) = \begin{cases} \frac{3}{4}(1 - t^2) & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

  この 2 式を使った時、$D(t)$ は、重み付けのバランスを定めて、D に食わせる値で、“距離” と λ の変化による近傍の変化の仕方を定める。
- 上記の一個目の式の分母を、λ に関して単調増加な関 数 $h_\lambda(x_0)$ に拡張したもの。
- Tri-cube kernel(tri-cube function, tri-cube weighted function)

$$D(t) = \begin{cases} (1 - |t|^3)^3 & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

  とするもの。
- Gaussian カーネル

$$D(t) = e^{-t^2}$$

  とするもの。ここまでの例と違い、「台がコンパクトでない」(ざっくりいうと、重みが 0 でない点が無限集合)。

ここまでの言葉を使うと、k 近傍法は、カーネルとして以下を取った Nadaraya-Watson 重み付きカーネル平均とみなせる。

$$K_k(x_0, x) = D\left(\frac{|x - x_0|}{|x_0 - x_{[k]}|}\right),$$
$$D(t) = \begin{cases} 1 & \text{if } |t| \leq 1 \\ 0 & \text{otherwise} \end{cases}$$

6.1.1 局所線形回帰

(式の形から、「線形」と名付けられているが、最終的に得られる式は線形関数ではないことに注意。)

ここまで、k 近傍法のカクカクさを避けるために、カーネルを用いて、滑らかな重み付き平均をとる、という方法を考えた。

この方法の問題点は、領域の外縁部でバイアスが大きくなること。理由は、外縁部では、「近傍」の分布が領域の中心方向に偏っていること。

それを防ぐ方法として考えられたのが、**局所線形回帰** (local linear regression)。x_0 における y の推定値を得るために以下の式を最小化する。

$$\sum_{i=1}^{N} K_\lambda(x_0, x_i)(y_i - \alpha(x_0) - \beta(x_0)x_i)^2$$

対比として、k 近傍法、Nadaraya-Watson 法は以下の最小化を考えていると思える。

$$\sum_{i=1}^{N} K_\lambda(x_0, x_i)(y_i - y)^2$$

[メモ]

値を推定するのに一番簡単に思いつく方法は、単純な平均値=大域的な 0 次式。ちょっと複雑化する方法として、「大域的」を「局所的」としたのが k 近傍法で、「0 次式」を「1 次式」としたのが線形回帰。さらに、局所的な 1 次式としたのが、局所線形回帰と思うと、統一的に理解できる？

で、元の式に戻ると、上の式を最小化するような $\hat{\alpha}(x_0)$, $\hat{\beta}(x_0)$ を使って、以下の式より推定値を得る。

$$\hat{f}(x_0) = \hat{\alpha}(x_0) + \hat{\beta}(x_0)x_0$$

ここから先は、前と同様。微分して $= 0$ としてやれば、以下を得る。

$$\begin{aligned} \hat{f}(x_0) &= b(x_0)^T(\mathbf{B}^T\mathbf{W}(x_0)\mathbf{B})^{-1}\mathbf{B}^T\mathbf{W}(x_0)\mathbf{y} &(1)\\ &= \mathbf{L}(x_0)\mathbf{y} &(2)\end{aligned}$$

ただし、$\mathbf{B}$ は訓練データの入力変数 (定数項を表す 1 を含む) がなす行列。また、$\mathbf{W}$ は対角成分が $K_\lambda(x_0, x_i)$ となる、対角行列。$\mathbf{W}$ を単位行列とすれば、通常の線形回帰に他ならない。

最後の式変形は、$\mathbf{L}$ は訓練データの入力変数のみに依存していて、訓練データの出力変数に関して、線形結合として表せることを強調している。この係数行列 $\mathbf{L}$ のことを、**等価カーネル** (equivalent kernel) というらしい (正確な定義を述べた文書を未だ発見できず)。どうやら、合計 1 になり、重み付き平均の係数と思えるよう。

この方法を採用することで、当初問題となっていた、領域外縁部での誤差は、高々 1 次のオーダーとなることが数学的に示せる。

6.1.2 局所多項式回帰 (Local Polynomial Regression)

0 次式を 1 次式にして良くなったなら、次数上げてもっと良くできるんじゃない？という話。以下を最小化する。

$$\sum_{i=1}^{N} K_\lambda(x_0, x_i)\left(y_i - \alpha(x_0) - \sum_{j=1}^{d}\beta_j(x_0)x_i^j\right)^2$$

基本的に、計算は線形の時と同様にできる。局所線形回帰と局所 2 次回帰を比べると、以下の傾向がある。

- 局所線形回帰は、真の関数が曲がっている場所で、より内回りを通る。
  (丘を削り、谷を埋める、trimming the hills and filling the valleys)
- 局所二次回帰はバリアンスが大きい

(まさしく、bias-variance tradeoff に他ならない)

6.2 カーネルの幅の決定

これまでの他のモデルと同じく、カーネルの幅 = モデルの自由度 を決定する必要があるので、その方法について。

交差検証 (cross validation) なども考えられるが、とりあえず素朴な方法として、スプラインの時と同じ方法が考えられる。つまり、$\hat{y} = \mathbf{S}_\lambda \mathbf{y}$ となる行列 $\mathbf{S}_\lambda$、今回の場合は $\mathbf{L}$ のトレースをもって、有効自由度 (effective degree of freedom) とみなし、これを指定することで、λ の値を決定することができる。

この「有効自由度」を用いることで、同じ自由度のスプライン平滑とカーネル平滑の比較、とかができる。

6.3 p 次元での局所回帰

入力変数の次元が上がっても、全く同様に計算できる。ただし、高次元の時は注意点がある。

- 次元の呪い
  次元が高まると、訓練データの密度を保つために、次元に対して指数オーダーのデータ数が必要になる。
- 視覚化しにくい
  3 次元 (出力変数込み) までは、グラフで表現できるが、それでも解釈しにくい。そこで、第 2、第 3 の入力変数については、値の範囲によって、データを分割して、各範囲ごとに別々に局所回帰をした方が良かったりする。
  こういう表示の仕方を、トレリス (trellis、格子) 表示というらしい。

6.4 構造化局所回帰

次元が高く、訓練データが不足の時は、モデルに何かしらの構造を仮定する必要がある。

6.4.1 構造化カーネル

最初の方法では、カーネルに条件を加える。半正定値行列 $\mathbf{A}$ を 2 個目のパラメータとして、カーネルが以下の式で表されると仮定する。

$$K_{\lambda,\mathbf{A}}(x_0, x) = D\left(\frac{(x - x_0)^T \mathbf{A}(x - x_0)}{\lambda}\right)$$

$\mathbf{A}$ に適当な制約を課すことで、特定のパラメータを無視したり、影響を弱めたりできる。

この方法の例として、射影追跡回帰 (projection-pursuit regression) というものがある。詳しくは 11 章にて。

$\mathbf{A}$ にたいして、より一般的な制約を課すこともできるが、面倒なので、それくらいなら回帰関数に制約を加えることの方が多いそう。

6.4.2 構造化回帰関数

今度は、回帰関数に制約を加える、という発想。まず、真の回帰関数 $E[Y \mid X] = f(X_1, X_2, \ldots, X_p)$ を以下のように分解できる。

$$f(X_1, X_2, \ldots, X_p) = \alpha + \sum_j g_j(X_j) + \sum_{k<l} g_{kl}(X_k, X_l) + \cdots$$

ここで、この関数は、一定以上の次数を持つ項を持たない、という仮定をする、という手法。

例えば、加法的なモデルは、相互作用の項を一切含まない、という仮定を置いていると思える。また、2 次に拡張した線形モデルは、3 個以上の相互作用をもたないものである。

こういった、低次元を仮定したモデルの計算方法として、反復的なバックフィッティングアルゴリズム (iterative backfitting algorithm) というものがある。詳しくは 9 章にて。

この、構造化回帰関数のモデルの重要な例として、**係数変化モデル** (varying coefficient model) というものがある。簡単に言えば、入力変数のうちいくつかの線形結合、係数は残りの変数を含む関数とみなす方法。

適当な $q < p$ について、$Z = (X_{q+1}, \ldots, X_p)$ として、

$$f(X) = \alpha(Z) + \beta_1(Z)X_1 + \cdots + \beta_q(Z)X_q$$

という形である、と仮定し最適化をする。係数関数 β 達の値の推定も必要。最適化には、重み付きの最小二乗法を使うことが多い。

6.5 局所尤度やその他の手法

局所回帰や係数変化モデルはより広く応用できる。最小化する式が損失関数の合計の時、損失関数の重み付きの合計を最小化するようにすればよい。

- 尤度を局所化する。つまり、尤度の計算に重みを付ける。対数尤度が以下の式で表されていた。

  $$L(\theta) = \sum_{i=1}^{N} \log P_\theta[Y = y_i \mid X = x_i]$$

  この、回帰関数に含まれるパラメータ θ も x_0 の関数であると思ったうえで、対数尤度を以下のように局所化、つまり重み付けできる。

  $$L(\theta, x_0) = \sum_{i=1}^{N} K_\lambda(x_0, x_i) \log P_\theta[Y = y_i \mid X = x_i]$$

- 次数 k の自己回帰時系列モデル (つまり、時系列データで、次の時点での値の予測に、直近 k 時点分のデータを使うモデル) は、以下の形で定式化される。(古典的に、線形であることは仮定されてきたらしい。最近は非線形のものも考えられているとか)

  $$y_t = \beta_0 + \beta_1 y_{t-1} + \cdots + \beta_k y_{t-k} + \epsilon_t$$

  通常の線形回帰と全く同様に最小二乗法でパラメータを推定できる。また、カーネル $K_\lambda(y_t)$ 局所化することで、モデルをより柔軟にできるとか。(動的線形モデル (dynamic linear model) と比較しているが、良く分からなかった。状態空間モデルというものと関係があるらしいのだけど。)

最尤法の変形が出てきたので、ロジスティック回帰に適用してみる。事後確率が前と同様以下のように書けると思う。

$$P[G = j \mid X = x] = \frac{e^{\beta_{j0} + \beta_j^T x}}{1 + \sum_{k=1}^{J-1} e^{\beta_{k0} + \beta_k^T x}}$$

すると、重み付き対数尤度は以下のように書ける。

$$\sum_{i=1}^{N} K_\lambda(x_0, x_i) \left(\beta_{g_i 0}(x_0) + \beta_{g_i}(x_0)^T (x_i - x_0) - \log \left(1 + \sum_{k=1}^{J-1} \exp(\beta_{k0}(x_0) + \beta_k(x_0)^T (x_i - x_0)) \right) \right)$$

ここで、ちょこっと式をいじって、指数関数の肩の部分を、x_0 について中心化していることに注意。(ただの式変形)

この方法は、次元の呪いを避ける工夫はしていないが、高次元でもよい結果を出した事例もあるとか。カーネル平滑を用いた、一般化された加法モデルと深い関係があるらしい。詳しくは 9 章にて。

6.6 カーネル密度推定と分類

カーネル密度推定は、カーネル回帰より古くから使われてきた教師なし学習の手法。ノンパラメトリックな分類に使えるらしい。

6.6.1 カーネル密度推定

教師なし学習の話。訓練データ $x_1, \ldots, x_N$ が与えられた時、X の確率密度関数を、

$$\hat{f}_X(x_0) = \frac{|\mathcal{N}(x_0)|}{N\lambda}$$

によって推定する方法。ただし、$\mathcal{N}$ は、入力に対して何かしらの近傍を与える関数で、λ によって幅が定まり、上式の全体での積分が 1 になる。

この式は、カクカクになって滑らかじゃなくて気持ち悪いので、これまでのようにカーネルを使って、平滑化する。

$$\hat{f}_X(x_0) = \frac{1}{N\lambda}\sum_{i=1}^{N} K_\lambda(x_0, x_i)$$

これの手法を、**カーネル密度推定** (kernel density estimation)、または Parzen 窓 (Parzen window) というらしい。

カーネル密度推定を考えるときは、カーネル K_λ として、Gaussian カーネル $K_\lambda(x_0, x) = \phi(|x - x_0|/\lambda)$ が使われることが多い。

6.6.2 カーネル密度分類

クラス $j = 1, \ldots, J$ への分類問題を考える。先ほどのカーネル密度推定を使うことで、各クラスにおける密度関数

$$\hat{f}_j(x_0) = \hat{P}[X = x_0 \mid G = j]$$

を推定できる。さらに、訓練データにおける各クラスの割合を事前確率の推定値 $\hat{\pi}_j$ として使い、Bayes の定理を使って事後確率を計算できる。

$$\hat{P}[G = j \mid X = x_0] = \frac{\hat{\pi}_j \hat{f}_j(x_0)}{\sum_k \hat{\pi}_k \hat{f}_k(x_0)}$$

ただし、分類がしたいなら、境界付近を上手に推定することが必須。(それで、結局どういう計算を推奨しているのかを読み取れなかった。)

6.6.3 ナイーブベイズ分類器

昔から使われてきた手法。特に、入力変数の次元が高くてうまく推定できないときに使う。各入力変数が独立であることを仮定する手法。つまり、

$$f_j(X) = \prod_k f_{jk}(X_k)$$

とする。普通、本当に独立なことは少ないが、良い制度を出すこと多いとか。

推定は、各入力変数ごとに 1 次元カーネル密度推定をすればよいので次元の呪いを回避できる。

上の式をもとに算出した事後確率 $P[G = j \mid X]$ をロジット変換すると、以下のようになる。

$$\log \frac{P[G = l \mid X]}{P[G = J \mid X]} = \log \frac{\pi_l f_l(X)}{\pi_J f_J(X)} = \log \frac{\pi_l}{\pi_J} + \sum_k \log \frac{f_{lk}(X_k)}{f_{lJ}(X_J)}$$

これは、9 章で見る一般化された加法的モデルと同じ式の形になっているらしい。

2 つのモデルの関係は、線形判別モデルとロジスティック回帰の関係と似ている。つまり、考える式は同じで、「最適化」の基準が違う。

6.7 放射基底関数とカーネル

基底関数展開の考え方と、カーネル法の考え方を合体させる、というアイデア。基底関数として、ここまで見てきたカーネルを取る。

$$f(x) = \sum_j K_{\lambda_j}(\xi_j, x)\beta_j$$

カーネル関数としては、Gaussian カーネルが使われることが多い。

推定すべきパラメータは β_j に加え、λ_j、ξ_j も。パラメータの推定に使われる手法は何個かあるとか。

最小二乗法を使う方法を 2 つ紹介。

- 単純な最小二乗法
  3 種類のパラメータについて、同時に RSS を最小化する。RBF(Radial Basis Function) ネットワークと呼ばれている手法。
  シグモイドを使ったニューラルネットワークの代わりとして出てくるらしい。ξ と λ が重みのパラメータになる。
  RSS がいくつかの極小値をもち、ニューラルネットワークと似た最適化アルゴリズムが用いられる。
- 先に ξ と λ を推定し、そのあと β を最小二乗法で推定する方法。
  ξ と λ は、訓練データの入力変数の分布から教師なし学習で推定することが多い。例えば、混合ガウスモデルを使ったり、クラスタリングしたり。

パラメータを減らすために、分散 λ_j が一定である、と仮定することも考えられる。この時、どの基底関数もカバーできない領域が生まれうる。これは、再度正規化することで防ぐ。

Nadaraya-Watson カーネル回帰は、再正規化した放射基底関数とみなせる。

$$\hat{f}(x_0) = \frac{\sum_i K_\lambda(x_0, x_i)y_i}{\sum_i K_\lambda(x_0, x_i)} = \sum_i y_i \frac{K_\lambda(x_0, x_i)}{\sum_i K_\lambda(x_0, x_i)}$$

最後の式と、先週の再生核ヒルベルト空間の結果の式が似ている。ここからもうちょっと話を進めると、本章の古典的な「カーネル」と再生核ヒルベルト空間の話で出てくる「カーネル」の繋がりが分かるそう。

6.8 密度推定と分類のための混合モデル

放射基底関数の進化版として、混合モデル (mixture model) が考えられる。つまり、いくつかの確率密度関数 (よくあるのは正規分布) の和の形であると考える。すなわち、

$$f(x) = \sum_{m=1}^{M} \alpha_m \phi(x; \mu_m, \Sigma_m)$$

という形。ただし、$\sum_m \alpha_m = 1$。もちろん、必ずしも正規分布でなくてもよい。

通常、パラメータの推定は「EM アルゴリズム」というものを使うらしい。詳しくは 8 章にて。特別な場合として、以下がある。

- 分散共分散行列がスカラー、すなわち $\Sigma_m = \sigma_m \mathbf{I}$ となるとき、これは放射関数基底による推定に他ならない。
- 上記に加えて、σ_m が定数で、基底関数の数 M が訓練データ数 N に下から近づくとき、推定値は、$\alpha_m = 1/N$、$mu_m = x_m$ に近づく。

ベイズの定理を使うと、事後確率 $P[G \mid X]$ の柔軟なモデルが考えられるらしい (良く分からん)。詳しくは 12 章にて。

6.9 計算方法の工夫

カーネル法は、“memory-based” な方法。意味は分からなかった。対立概念も分からん。

計算量が多くて大変で、統計ソフトとかは “triangular method”(三角法?) という手法で実装して計算量を減らしているとか。