第6章　カーネル法

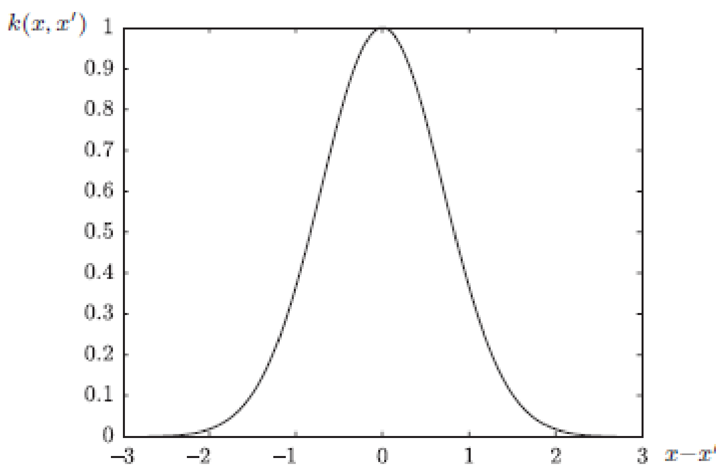
Pattern
Recognition
and
Machine
Learning

修士2年
藤井　敬士

BehaviorInNetworks
都市生活学・ネットワーク行動学研究室

カーネル法とは

- カーネル関数を用いたデータ解析手法
  - カーネル関数とは，二つの入力$\mathbf{x}=(x_1,\ldots,x_d)$, $\mathbf{x}'=(x'_1,\ldots,x'_d)$から計算される関数$k(\mathbf{x},\mathbf{x}')$.
  - 直観的には，$k(\mathbf{x},\mathbf{x}')$は$\mathbf{x}$と$\mathbf{x}'$の近さのようなものである
  - 不変カーネル $k(\mathbf{x},\mathbf{x}')=k(\mathbf{x}-\mathbf{x}')$ と均一カーネル（RBF $k(\mathbf{x},\mathbf{x}')=k(\|\mathbf{x}-\mathbf{x}'\|)$



よく使われるカーネル関数の例

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

$\|\ \|$ の意味は $\boldsymbol{z}=(z_1,\ldots,z_d)$

に対し, $\|\boldsymbol{z}\|^2=\sum_{m=1}^{d} z_m^2$

また，βは適当に決めるパラメータである

カーネル法とは

- xに対して関 $y=\sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}^{(j)},\boldsymbol{x})$ を当てはめ，二乗誤差
  $r_k(y,\boldsymbol{x};\boldsymbol{\alpha})=(y-\sum_{j=1}^{n}\alpha_j k(\boldsymbol{x}^{(j)},\boldsymbol{x}))^2$ を最小化するαを求める.
- 二乗誤差の総和は $R_k(\boldsymbol{\alpha})=\sum_{i=1}^{n}r_k(y^{(i)},\boldsymbol{x}^{(i)};\boldsymbol{\alpha})=(\boldsymbol{y}-K\boldsymbol{\alpha})^{\mathrm{T}}(\boldsymbol{y}-K\boldsymbol{\alpha})$

ここで，

$$K=\begin{pmatrix} k(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(1)}) & k(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(1)}) & \ldots & k(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(1)}) \\ k(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(2)}) & k(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(2)}) & \ldots & k(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(2)}) \\ \vdots & \vdots & \ddots & \vdots \\ k(\boldsymbol{x}^{(1)},\boldsymbol{x}^{(n)}) & k(\boldsymbol{x}^{(2)},\boldsymbol{x}^{(2)}) & \ldots & k(\boldsymbol{x}^{(n)},\boldsymbol{x}^{(n)}) \end{pmatrix}$$

この解は，Kが正則ならば $\boldsymbol{\alpha}=(K^{\mathrm{T}}K)^{-1}K^{\mathrm{T}}\boldsymbol{y}$

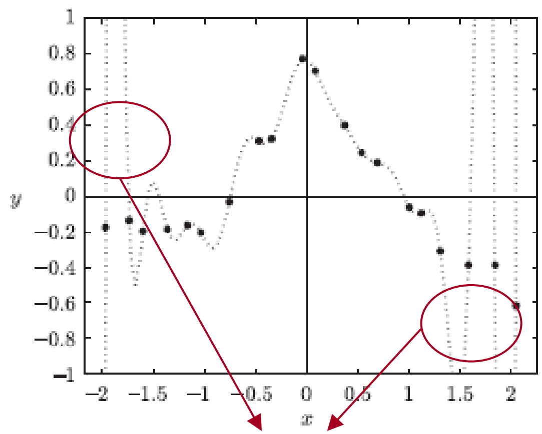
任意の**x**,**x**'について $k(\boldsymbol{x},\boldsymbol{x}')=k(\boldsymbol{x}',\boldsymbol{x})$ が成り立つのでKは対称行列

すなわち，$K^{\mathrm{T}}=K$だから，$(K^{\mathrm{T}}K)^{-1}K^{\mathrm{T}}=(K^2)^{-1}K=K^{-1}$よって，

$$\boldsymbol{\alpha}=K^{-1}\boldsymbol{y}$$

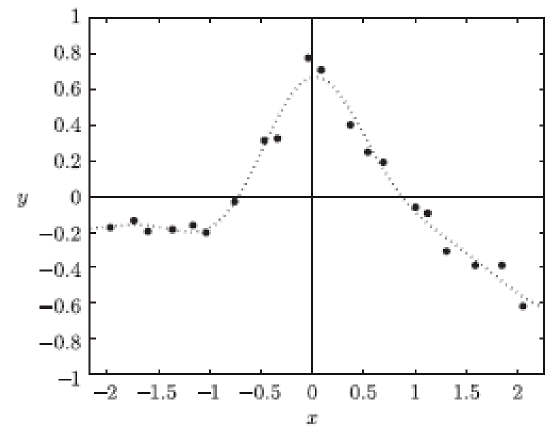
カーネル法とは

- 推定結果



過学習による誤推定

正則化



λ=0.01の場合の関数

$$R_{k,\lambda}(\boldsymbol{\alpha}) = (\boldsymbol{y}-K\boldsymbol{\alpha})^{\mathrm{T}}(\boldsymbol{y}-K\boldsymbol{\alpha})+\underbrace{\lambda\boldsymbol{\alpha}^{\mathrm{T}}K\boldsymbol{\alpha},}_{\text{正則化項}} \quad \lambda > 0$$ を最小化する

$-K(\boldsymbol{y}-K\boldsymbol{\alpha})+\lambda K\boldsymbol{\alpha} = 0$ より，$\boldsymbol{\alpha} = (K+\lambda I_n)^{-1}\boldsymbol{y}$

※正則化パラメータλの取り方には任意性が残る

λが小さいと不安定な解，λが大きいとα＝0に近づく

カーネル法を用いることの利点

1. サンプルの増加と共にどんどん複雑に出来る

   (正則化パラメータを適当に取ると)複雑な関数を表現することが出来る.

2. 線形性と非線形性を両方持つ

   $y = \sum_{j=1}^{n} \alpha_j k(\boldsymbol{x}^{(j)}, \boldsymbol{x})$ より，決めるべきパラメータαについては線形性を持つが，入力データについては非線形な関数を表現できる.

3. 高次元・非数値データへの適用

   カーネル関数の中身は1次元の実数に限らず，高次元，文字列，グラフ構造などについても同様に扱うことが出来る.

4. カーネル関数のモジュール化

   最適解αは行列**K**のみに依存し，カーネル関数がどんなものであるかは関係ない．つまり，カーネル関数を計算する部分とそれ以降の処理を分割できる.

双対表現

- パラメータベクトル**w**を直接扱う代わりに，最小二乗法のアルゴリズムをパラメータベクトル**a**で表現しなおすこと．
- この表現によって，カーネル関数が見える形になる．
  - $$J(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\right\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}.$$ のような正則化された二乗和誤差の最小

化を考える(λ≧0とする).

  - J(**w**)(
$$\mathbf{w} = -\frac{1}{\lambda}\sum_{n=1}^{N}\left\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\right\}\phi(\mathbf{x}_n) = \sum_{n=1}^{N} a_n\phi(\mathbf{x}_n) = \mathbf{\Phi}^{\mathrm{T}}\mathbf{a}.$$

すなわち，係数が**w**の関数であるようなφ(x_n)の線形結合となる．ここ

て$$a_n = -\frac{1}{\lambda}\left\{\mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}_n) - t_n\right\}$$ ρ(x_n)$^{\mathrm{T}}$で与えられるような計画行列．また，

として**a**=(a_1,…,a_N)とする．

双対表現

- w=ΦTaをJ(w)に代入すると, $J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\mathbf{\Phi}^{\mathrm{T}}\mathbf{a}$

  ここで, **t**=(t1,…tn)とする.
- 次に, N×Nの対称行列で, その要素が $K_{nm} = \phi(\mathbf{x}_n)^{\mathrm{T}}\phi(\mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m)$

  で表されるグラム行列**K=ΦΦ**ᵀを定義す $k(\mathbf{x}, \mathbf{x}') = \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{x}')$.

  (

$$J(\mathbf{a}) = \frac{1}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{K}\mathbf{a} - \mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{t} + \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \frac{\lambda}{2}\mathbf{a}^{\mathrm{T}}\mathbf{K}\mathbf{a}$$

$$\mathbf{a} = (\mathbf{K} + \lambda\mathbf{I}_N)^{-1}\mathbf{t}.$$

wを消去して**a**について解くと,

- これを線形回帰モデルに代入し直すことによって, 新たなxに対する予測は以下のように与えられる

$$y(\mathbf{x}) = \mathbf{w}^{\mathrm{T}}\phi(\mathbf{x}) = \mathbf{a}^{\mathrm{T}}\mathbf{\Phi}\phi(\mathbf{x}) = \underline{\mathbf{k}(\mathbf{x})^{\mathrm{T}}\left(\mathbf{K} + \lambda\mathbf{I}_N\right)^{-1}\mathbf{t}}.$$

カーネル関数のみで表現可能

双対：φ(**x**)の要素の線形結合によって**a**が表現できることから, パラメータベクトル**w**を用いたもともとの定式化を復元できる.

特徴ベクトルφ(**x**)を明示的に考えなくても, カーネル関数で表現できる

双対表現

- 双対表現　においては，N×N行列の逆行列を求めることでパラメータ**a**が得ら $\mathbf{a} = (\mathbf{K} + \lambda \mathbf{I}_N)^{-1}\mathbf{t}.$
- もともとの表現においては，M×M行列の逆行列を求めればよかった.

- 通常はN>>M．しかし，双対表現を用いることで特徴ベクトルφ(**x**)を明示的に考えずに，高次元や無限次元の特徴空間を間接的に扱うことが出来る.
- 回帰のための確率的な線形モデルとガウス過程の双対性
- サポートベクトルマシンとの関連性(7章)

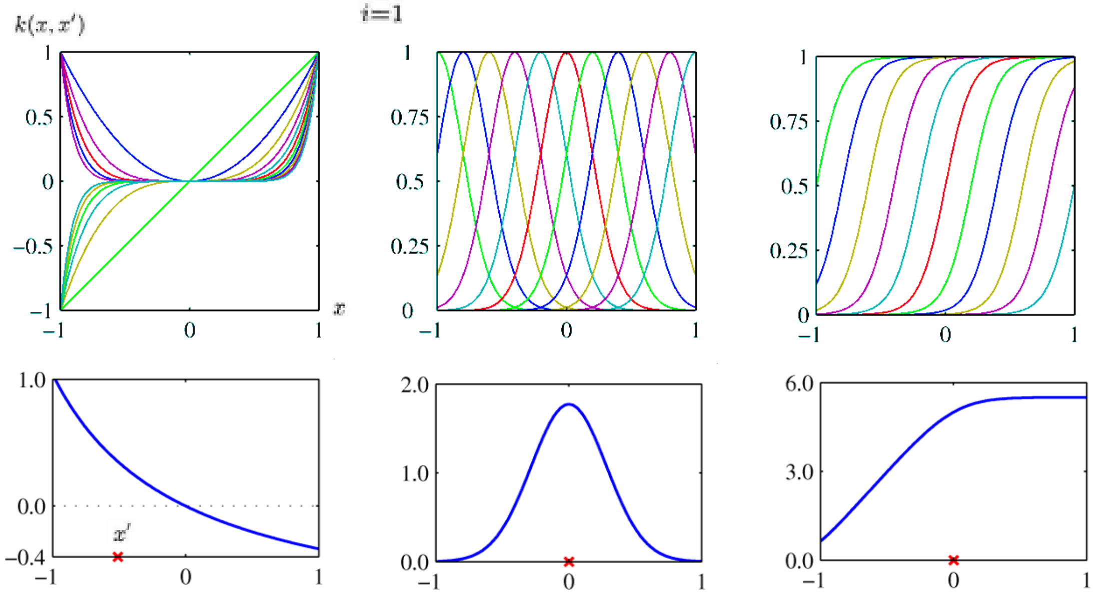
カーネル関数の構成

- カーネル置換を行う＝有効なカーネル関数を構成する必要
  1. 特徴空間への写像φ(x)を考え，対応するカーネルを構成する.
  2. カーネル関数を直接定義する.

1. カーネル関数は以下のように定義されている.

$$k(x, x') = \phi(x)^{\mathrm{T}}\phi(x') = \sum_{i=1}^{M} \phi_i(x)\phi_i(x').$$

φ_i(x)は基底関数



カーネル関数の構成

2. 次の例で考える. $k(\mathbf{x},\mathbf{z}) = \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2$

2次元の入力空間 $\mathbf{x} = (x_1, x_2)$ を考えて，上式を展開

$$
\begin{aligned}
k(\mathbf{x},\mathbf{z}) &= \left(\mathbf{x}^{\mathrm{T}}\mathbf{z}\right)^2 = (x_1 z_1 + x_2 z_2)^2 \\
&= x_1^2 z_1^2 + 2x_1 z_1 x_2 z_2 + x_2^2 z_2^2 \\
&= (x_1^2, \sqrt{2}x_1 x_2, x_2^2)(z_1^2, \sqrt{2}z_1 z_2, z_2^2)^{\mathrm{T}} \\
&= \phi(\mathbf{x})^{\mathrm{T}}\phi(\mathbf{z}).
\end{aligned}
$$

特徴空間への写像は $\phi(\mathbf{x}) = (x_1^2, \sqrt{2}x_1 x_2, x_2^2)^{\mathrm{T}}$ の形を持ち，すべての2次の項を含む.

関数k(**x**,**x**')が有効なカーネル

⇔任意の{$\mathbf{x}_n$}に対して，要素がk($\mathbf{x}_n$,$\mathbf{x}_m$)で与えられるグラム行列**K**が半正定値であること.

新たなカーネル関数の構成法

- $k_1(\mathbf{x},\mathbf{x}')$と $k_2(\mathbf{x},\mathbf{x}')$が有効なカーネルであるとき，下の関数もカーネル関数として有効である．

$$k(\mathbf{x},\mathbf{x}') = ck_1(\mathbf{x},\mathbf{x}')$$
$$k(\mathbf{x},\mathbf{x}') = f(\mathbf{x})k_1(\mathbf{x},\mathbf{x}')f(\mathbf{x}')$$
$$k(\mathbf{x},\mathbf{x}') = q\left(k_1(\mathbf{x},\mathbf{x}')\right)$$
$$k(\mathbf{x},\mathbf{x}') = \exp\left(k_1(\mathbf{x},\mathbf{x}')\right)$$
$$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}') + k_2(\mathbf{x},\mathbf{x}')$$
$$k(\mathbf{x},\mathbf{x}') = k_1(\mathbf{x},\mathbf{x}')k_2(\mathbf{x},\mathbf{x}')$$
$$k(\mathbf{x},\mathbf{x}') = k_3\left(\phi(\mathbf{x}),\phi(\mathbf{x}')\right)$$
$$k(\mathbf{x},\mathbf{x}') = \mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}'$$
$$k(\mathbf{x},\mathbf{x}') = k_a(\mathbf{x}_a,\mathbf{x}'_a) + k_b(\mathbf{x}_b,\mathbf{x}'_b)$$
$$k(\mathbf{x},\mathbf{x}') = k_a(\mathbf{x}_a,\mathbf{x}'_a)k_b(\mathbf{x}_b,\mathbf{x}'_b).$$

•c>0は定数

•f(･)は任意の関数

•q(･)は非負の係数を持つ多項式

•φ(**x**)は**x**から$\mathbf{R}^M$への関数

•k3(･,･)は$\mathbf{R}^M$で定義された有効なカーネル

•Aは対称な半正定値行列

•$\mathbf{x}_a$と$\mathbf{x}_b$は$\mathbf{x}=(\mathbf{x}_a,\mathbf{x}_b)$であるような変数

•k_aとk_bはそれぞれの特徴空間において有効なカーネル関数

カーネル関数の例

- ガウスカーネル

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\|\mathbf{x} - \mathbf{x}'\|^2 / 2\sigma^2\right)$$

- 生成モデルに基づくカーネル

$$k(\mathbf{x}, \mathbf{x}') = p(\mathbf{x})p(\mathbf{x}').$$

- 配列XとX'の類似度を測るカーネル

$$k(\mathbf{X}, \mathbf{X}') = \sum_{\mathbf{Z}} p(\mathbf{X}|\mathbf{Z})p(\mathbf{X}'|\mathbf{Z})p(\mathbf{Z}).$$

- フィッシャーカーネル

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}}\mathbf{F}^{-1}\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}')$$

$$k(\mathbf{x}, \mathbf{x}') = \mathbf{g}(\boldsymbol{\theta}, \mathbf{x})^{\mathrm{T}}\mathbf{g}(\boldsymbol{\theta}, \mathbf{x}').$$

- シグモイドカーネル

$$k(\mathbf{x}, \mathbf{x}') = \tanh\left(a\mathbf{x}^{\mathrm{T}}\mathbf{x}' + b\right)$$

RBFネットワーク

- 線形基底関数モデル(3章)では，基底関数の形を考えていなかった．
→RBF(動径基底関数:radial basis function)が良く使われる．

$$\phi_j(\mathbf{x}) = h(\|\mathbf{x} - \boldsymbol{\mu}_j\|)$$

  - もともとは，関数補間（目的変数の値を正確に表現できる関数を求めること）のために導入された．

$$f(\mathbf{x}) = \sum_{n=1}^{N} w_n h(\|\mathbf{x} - \mathbf{x}_n\|).$$

  - 入力変数にノイズが含まれる場合の補間にも使われる

入力変数xに含まれるノイズが，確率分布v(ξ)に従う確率変数ξによって表されるとき

$$E = \frac{1}{2} \sum_{n=1}^{N} \int \{y(\mathbf{x}_n + \xi) - t_n\}^2 \nu(\xi)\, \mathrm{d}\xi.$$

変分法を用いて以下のように最適化でき，RBFも求められる

$$y(\mathbf{x}) = \sum_{n=1}^{N} t_n h(\mathbf{x} - \mathbf{x}_n), \quad h(\mathbf{x} - \mathbf{x}_n) = \frac{\nu(\mathbf{x} - \mathbf{x}_n)}{\sum_{n=1}^{N} \nu(\mathbf{x} - \mathbf{x}_n)}.$$

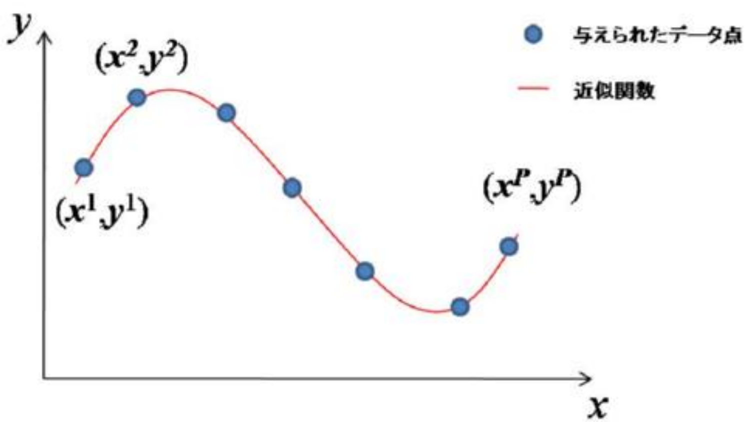
Nadaraya-Watsonモデル

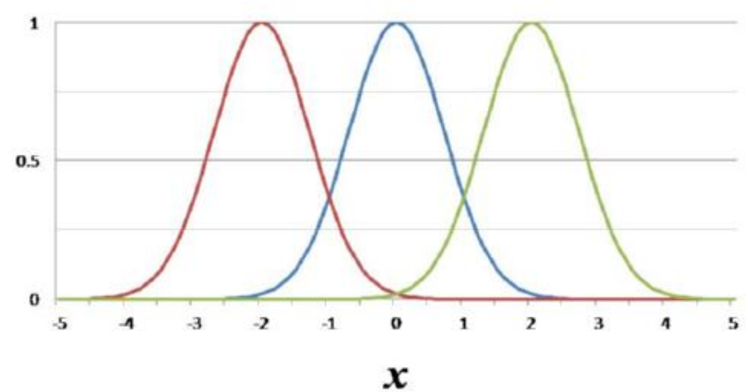
RBFネットワーク

- 3層から構成されるニューラルネットワーク
- 最小二乗法によって関数の最良近似法を導くことができる

＝安定した学習が可能

- ガウス関数を基底関数として用いることが多い

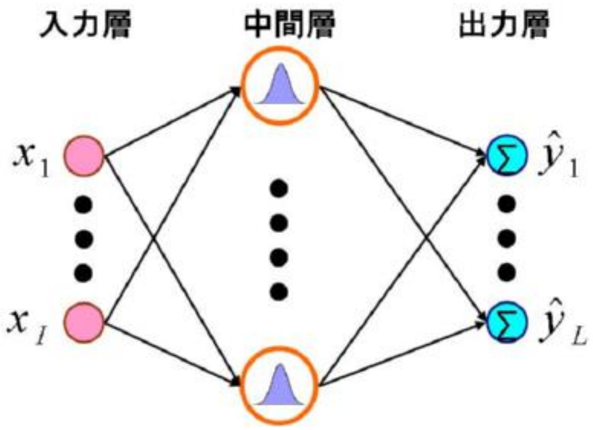


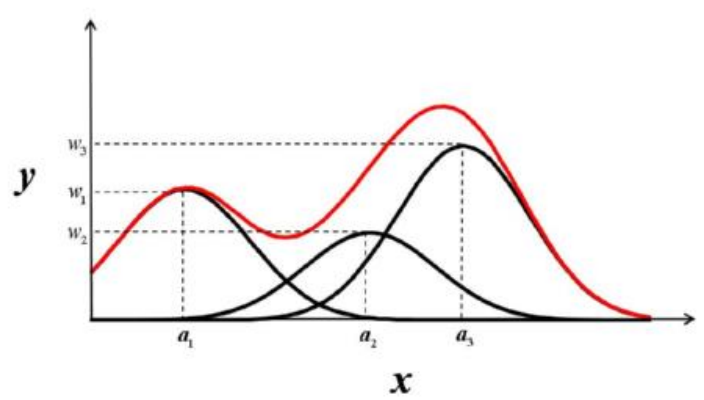


RBFネットワーク

- ネットワーク構造の中間層にさまざまなRBFを使用
- 出力層は中間層出力の多重和

  →複数のRBFに重み付けをして足し合わせることで任意の関数を実現
- RBFの中心値と重みを調整することで学習データと出力の誤差を小さくする.





RBFネットワーク

デモ

Nadaraya-Watsonモデル

- 訓練集合を$\{x_n,t_n\}$として，同時分布p(x,t)を推定するために，Parzen推定法を用いる．f(x,t)は密度関数の要素．

$$p(\mathbf{x},t)=\frac{1}{N}\sum_{n=1}^{N}f(\mathbf{x}-\mathbf{x}_n,t-t_n).$$

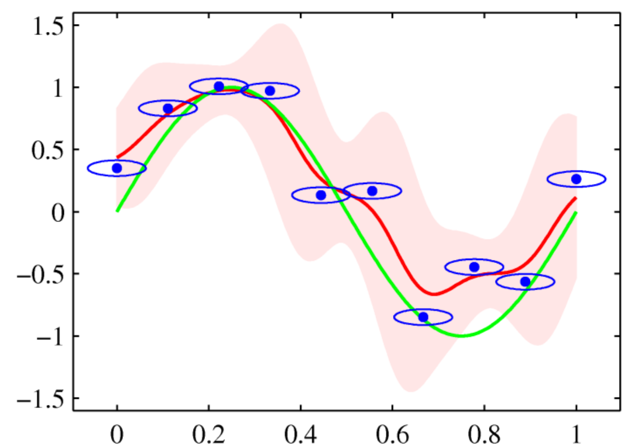
回帰関数y(x)は，目標変数の条件付き期待値

$$\begin{aligned}y(\mathbf{x})=\mathbb{E}[t|\mathbf{x}]&=\int_{-\infty}^{\infty}tp(t|\mathbf{x})\,\mathrm{d}t\\&=\frac{\int tp(\mathbf{x},t)\,\mathrm{d}t}{\int p(\mathbf{x},t)\,\mathrm{d}t}\\&=\frac{\sum_n\int tf(\mathbf{x}-\mathbf{x}_n,t-t_n)\,\mathrm{d}t}{\sum_m\int f(\mathbf{x}-\mathbf{x}_m,t-t_m)\,\mathrm{d}t}.\end{aligned}$$

$$\boxed{\begin{aligned}y(\mathbf{x})&=\frac{\sum_n g(\mathbf{x}-\mathbf{x}_n)t_n}{\sum_m g(\mathbf{x}-\mathbf{x}_m)}\\&=\sum_n k(\mathbf{x},\mathbf{x}_n)t_n.\\k(\mathbf{x},\mathbf{x}_n)&=\frac{g(\mathbf{x}-\mathbf{x}_n)}{\sum_m g(\mathbf{x}-\mathbf{x}_m)}.\\g(\mathbf{x})&=\int_{-\infty}^{\infty}f(\mathbf{x},t)\,\mathrm{d}t\end{aligned}}$$

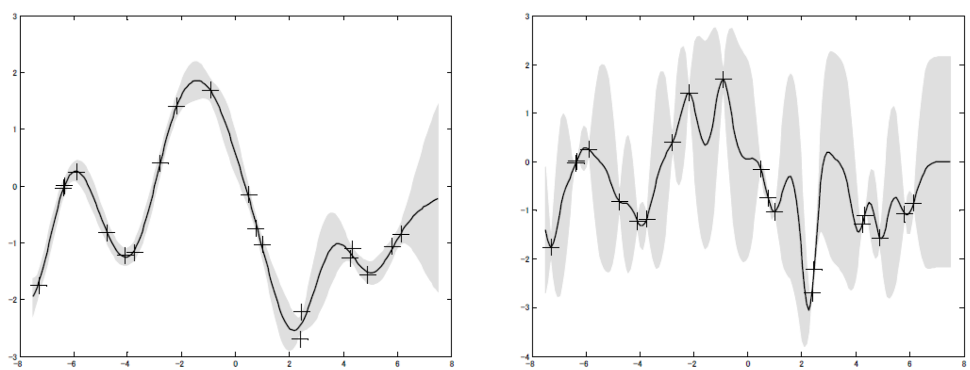
Nadaraya-Watsonモデル(カーネル回帰)

■データ点xに近いデータ点x_nほど大きな重みを与えることが出来る



ガウス過程

- 関数y(**x**)の上の確率分布として定義され，任意の点集合$x_1,\ldots x_N$に対するy(**x**)の値の同時分布がガウス分布に従うもの
- 線形回帰モデルだけでは訓練データを増やせば増やすほど予測がほぼ期待値に一致してしまう．しかし，一般化されたガウス過程のモデルを導入すると，訓練データに近いところでは分散が小さく，離れるごとに分散が大きくなるモデルとなる．



$\sigma = 1$　　　$\sigma = 0.3$

共分散関数 $R(s,t) = \exp\left(-\frac{1}{2\sigma^2}(s-t)^2\right)$