# QDTIS 補足資料

2024 年 8 月 18 日

QD 部　朝倉響

## 概要

本資料は，QDTIS の課題に関連する機械学習手法についての補足資料である．基本はサンプルコードに記載の手法について，その概略を述べる．各手法について比較的一般的なことを述べているので，実際のデータに対してどうアプローチするかは示していない．「ここまで抑えておけば，サンプルコード以上のことをしなくても，ハイパーパラメータや各モデルの比較だけである程度の発表ができるんじゃないかな」という内容を目指す．

### 表記

- $N$：データ数
- $\boldsymbol{x}_n \in \mathbb{R}^M$：$n$ 番目の入力データ（特徴量）
  生入力データをもとにいじった特徴量を含む．例えば，定値 1 を要素として追加することも可能だし，特徴量の非線形変換を行うことも可能．時系列データでは，過去 $p$ ステップのデータを含むこともある．
- $y_n$：$n$ 番目の出力データ（目標値）
- $\boldsymbol{w} \in \mathbb{R}^M$：重みベクトル

## 1　課題 1：株価予測

### 1.0　線形回帰

入力データに線形な関数での予測 $\boldsymbol{w}^\top \boldsymbol{x}_n$ と正解 $y_n$ の誤差を最小化するような重みベクトル $\boldsymbol{w}$ を求める．

$$\underset{\boldsymbol{w}}{\text{minimize}} \sum_{n=1}^{N} (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2.$$

- 正則化：過学習を防止したい → Ridge / Lasso
  （補足）モデルの次数 $M$ に対し，データ数 $N$ が少ない場合，過学習を引き起こす．つまり，自由度の高いモデルが “今回のデータのランダムノイズ” をも表現しようとしてしまい，汎化性能が低下する．過学習を起こしている場合，ノイズによるズレを無理やり表現するために，入力に対して急峻な変化を要求され，$w_j$ の値が大きくなりがちになる．そこで，$w_j$ の大きさを抑えるように，正則化項を導入すればよい．

### 1.1　SGDRegressor

確率的勾配降下法（Stochastic Gradient Descent）は最適化手法であって，回帰モデルの名前ではない．

- 最急降下法：全データ点を使った目的関数の勾配方向に進むことで，最適解を探す．線形回帰なら，上述の

目的関数の勾配

$$-2\sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)\boldsymbol{x}_n$$

- 確率的勾配降下法：ランダムに（いくつか）選んだデータ点に対して，目的関数の勾配方向に進むことで，最適解を探す．線形回帰なら

$$-2\sum_{n:\ \mathrm{rondom}}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)\boldsymbol{x}_n$$

  - 最適化の際に全データの計算の必要がないため，計算が速くなり，ランダム性をいれることにより局所解に収束するリスクを減らすことができる[*1]．

**ハイパーパラメータ**

- `loss`: 最適化したい損失関数
  - `squared_error`: 二乗和誤差
  - `huber`: 誤差が小さいところでは二乗，大きいところでは絶対値誤差．滑らかな L1 誤差.
  - `epsilon_insensitive`: 誤差が $\varepsilon$ 以下のところは無視．$\max\{0, |y_n - \boldsymbol{w}^\top \boldsymbol{x}_n| - \varepsilon\}$
  - `squared_epsilon_insensitive`: $\max\{0, (y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 - \varepsilon\}$
- `penalty='l1'|'l2'|'elasticnet'|None`: 正則化関数
- `alpha`: 正則化項の重み．正則化項の強さを決める．大きくするほど，訓練データへの適合は弱くなる．
- `l1_ratio`: `penalty='elasticnet'` での L1 正則化の割合

`loss='squared_error', penalty='l2'` ならば，Ridge 回帰と同じ（最適化手法が異なる）.

## 1.2 Ridge 回帰

2 次の正則化項

$$\underset{\boldsymbol{w}}{\mathrm{minimize}} \sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \alpha\sum_{j=1}^{M}|w_j|^2$$

**ハイパーパラメータ**

- `alpha`

## 1.3 Lasso 回帰

1 次の正則化項[*2]

$$\underset{\boldsymbol{w}}{\mathrm{minimize}} \sum_{n=1}^{N}(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n)^2 + \alpha\sum_{j=1}^{M}|w_j|$$

[*1] なんか収束が早いって言ってるサイト存在するけど，ウソな気がする.. そもそも収束するかどうかどうやって証明るんやっけ．learning rate の条件なかったっけ

[*2] 公式ドキュメントによると，Ridge はサンプル数で割ってないのに，Lasso では割っていたりする．したがって，必ずしも $\alpha$ が似たような大きさになるとは限らない.

### Ridge と Lasso の違い（大体まあまあの理解）

違いは正則化項の形のみ．誤差項（前半）と正則化項（後半）の等高線を考えたとき，それぞれの等高線の接点が最適点．図 1 から，Lasso（1 次の正則化）のほうが $w_j^* = 0$ になりやすいことがわかる[*3]．重み $w_j$ が 0 になるということは，入力データ（特徴量）のうち，$j$ 番目の要素は予測では使わないということ．Lasso では特にどの変数が重要かを知ることができる．

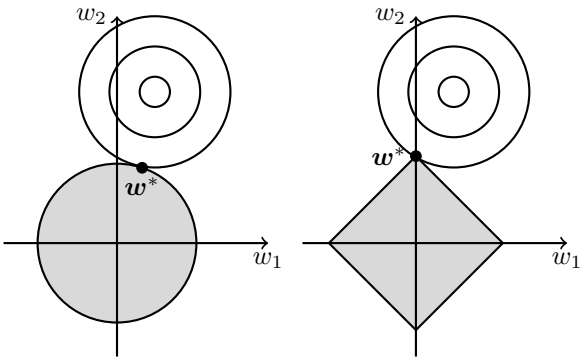


図 1　左：Ridge，右：Lasso．Lasso の場合 $w_1^* = 0$ になっており，1 つ目の特徴量は予測に使われていない：重要でないと考えられる．

## 1.4 Random Forest Regressor

### 1.4.1 決定木

「一つの特徴量に対して，ある値を境にして，2 つのクラスに分ける」を繰り返す木構造をつくる．ある入力 $\boldsymbol{x}_n$ の予測値は，分類されたクラスの訓練データの平均値．
最適化は，各葉 $k$ を訪れ，$n \in A_k = \{$葉 $k$ に分類されるデータ点$\}$ に対し，

$$\underset{j,t}{\text{minimize}} \sum_{n \in A_k, x_{nj} \leq t} (y_n - \bar{y}_L)^2 + \sum_{n \in A_k, x_{nj} > t} (y_n - \bar{y}_R)^2$$

$$\bar{y}_L = \text{average of } \{y_n \mid n \in A_k, x_{nj} \leq t\}, \quad \bar{y}_R = \text{average of } \{y_n \mid n \in A_k, x_{nj} > t\}$$

$$\underset{j,t}{\text{minimize}} \left(\boldsymbol{x}_n \text{ の } j \text{ 番目の要素が } t \text{ 以下のデータ点の分散}\right) + \left(t \text{ より大きいデータ点の分散}\right)$$

を計算し，これが元々の誤差 $\sum_{n \in A_k} (y_n - \bar{y})^2$ より小さくなっていれば，求めた要素 $j^*$ を $t^*$ で分割．

**ハイパーパラメータ**

- `max_depth`: 決定木の深さ
- `min_samples_split`: 分割するための最小データ数
- `min_samples_leaf`: 葉を作るための最小データ数

[*3] l1（もしくは L1）最適化の解が疎（スパース）になる証明はそれなりに面倒だった記憶がある

**特徴**

- `max_depth=None` など可能な限り細かく分割してしまうと，過学習しやすい → ランダムフォレスト
- 全体では階段状の回帰になっており，非線形なモデルである．ただし，分類されたクラス内で定値をとるので，各クラス内では線形モデルで，シンプルな構成である．

### 1.4.2 ランダムフォレスト

元データからランダムにサンプリングし，そのサンプリングデータを用いて決定木を作る．複数の決定木を用い，それぞれの予測値の平均を取る．データのランダムサンプリングは被りを許容する（ブートストラップサンプリング）．

**ハイパーパラメータ**

- `n_estimators`: 決定木の数

## 1.5 Support Vector Regression

予測値と正解の誤差を $\varepsilon$ までは許容する．

$$\underset{\boldsymbol{w}}{\text{minimize}}\ C\sum_{n=1}^{N} E_\varepsilon(y_n - \boldsymbol{w}^\top \boldsymbol{x}_n) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

$$E_\varepsilon(u) = \begin{cases} 0, & |u| \le \varepsilon; \\ |u| - \varepsilon, & |u| > \varepsilon \end{cases}$$

**特徴**

- 下で述べるカーネル法の考え方により，特徴量 $\boldsymbol{x}_n$ の次元を可算無限まで拡張して考えられる：モデルの自由度・表現力が高い[*4]
- それらの特徴量の形を明示的に意識する必要はなく，新しい入力データ $\boldsymbol{x}$ に対する予測値を計算できる
- サポートベクトル：学習結果から得られたモデルで予測する際に使われるデータ点のこと．最適解の解析から，予測の際に使われるデータ点は誤差が $\varepsilon$ より大きい（$\varepsilon$ チューブの外側の）点のみであることがわかる．結果，予測の計算は簡略化される．

**（補足）カーネル法**

自力で特徴量を作ることを前提としていたが，生の入力データ $\boldsymbol{r}_n \in \mathbb{R}^L$ に対して，何らかの変換 $\boldsymbol{\phi} : \mathbb{R}^L \to \mathbb{R}^M$ をして，特徴量入力データ $\boldsymbol{x}_n$ を作っていたとする（$\boldsymbol{x}_n = \boldsymbol{\phi}(\boldsymbol{r}_n)$）．最適化問題は以下で書き換えられる．

$$\underset{\boldsymbol{w}}{\text{minimize}}\ C\sum_{n=1}^{N} E_\varepsilon(y_n - \boldsymbol{w}^\top \boldsymbol{\phi}(\boldsymbol{r}_n)) + \frac{1}{2}\|\boldsymbol{w}\|^2$$

[*4] SVM の話をするのにカーネルの話は不可避なので書いてるんですけど，線形回帰に同じような話はできないんだろうか．

この最適化問題をいじくり回す（スラック変数を用いて目的関数を簡単化（不等式制約の追加）し，KKT 条件から双対問題を導出する）と，

$$\begin{aligned}
\underset{\boldsymbol{a},\widehat{\boldsymbol{a}}}{\text{maximize}} \quad & -\frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N}(a_n-\widehat{a}_n)(a_m-\widehat{a}_m)\boldsymbol{\phi}(\boldsymbol{r}_n)^\top\boldsymbol{\phi}(\boldsymbol{r}_m)-\sum_{n=1}^{N}(a_n+\widehat{a}_n)+\sum_{n=1}^{N}(a_n-\widehat{a}_n)y_n \\
\text{subject to} \quad & 0\le a_n\le C \\
& 0\le \widehat{a}_n\le C \\
& \sum_{n=1}^{N}(a_n-\widehat{a}_n)=0
\end{aligned}$$

最適化問題の決定変数が $\boldsymbol{w}\in\mathbb{R}^M$ から $\boldsymbol{a},\widehat{\boldsymbol{a}}\in\mathbb{R}^N$ に変更されている．また，この双対問題の目的関数において，重要なのは $k(\boldsymbol{r},\boldsymbol{r}')\coloneqq\boldsymbol{\phi}(\boldsymbol{r})^\top\boldsymbol{\phi}(\boldsymbol{r}')$ という関数 $k$ さえ決めてしまえば，$\boldsymbol{\phi}$ の形は気にしなくてよく，$\boldsymbol{\phi}(\boldsymbol{r})$ が無限次元でも問題ない．つまり，特徴量を無限個用意できる[*5]！ 実際には自力で作った特徴量をもとに，カーネル法を用いることで，それを非線形変換したものを扱えることになる．

**ハイパーパラメータ**

- カーネル関数
  - linear: $k(\boldsymbol{x},\boldsymbol{x}')=\boldsymbol{x}^\top\boldsymbol{x}'$
  - poly : $k(\boldsymbol{x},\boldsymbol{x}')=\gamma(\boldsymbol{x}^\top\boldsymbol{x}'+r)^d$
  - rbf (Radial Basis Function): $k(\boldsymbol{x},\boldsymbol{x}')=\exp(-\gamma\|\boldsymbol{x}-\boldsymbol{x}'\|^2)$
  - sigmoid: $k(\boldsymbol{x},\boldsymbol{x}')=\tanh(\gamma\boldsymbol{x}^\top\boldsymbol{x}'+r)$

## 1.6 ニューラルネットワーク

- 入力層: $x_i,\ i=1,2,\dots,M$
- 隠れ層 1: $h_j^{(1)}=\left(\boldsymbol{w}_j^{(1)}\right)^\top\boldsymbol{x},\ j=1,2,\dots,H^{(1)}$
- 隠れ層 $\ell=2,3,\dots,L$: $h_j^{(\ell)}=\left(\boldsymbol{w}_j^{(\ell)}\right)^\top\boldsymbol{\phi}\left(\boldsymbol{h}^{(\ell-1)}\right),\ j=1,2,\dots,H^{(\ell)}$
- 出力層：$y=\left(\boldsymbol{w}^{(L)}\right)^\top\boldsymbol{\phi}\left(\boldsymbol{h}^{(L)}\right)$

- 出力層の $y$ と正解の誤差を最小にするように重み $\boldsymbol{w}$ を最適化する（誤差逆伝播法）．
- $\boldsymbol{\phi}:\mathbb{R}^{H^{(\ell)}}\to\mathbb{R}^{H^{(\ell)}}$ は同じ非線形関数（活性化関数）$\varphi:\mathbb{R}\to\mathbb{R}$ が $H^{(\ell)}$ 個並んだもの．

**特徴**

- 非線形なモデルで，複雑な関数を表現できる[*6]．
- 中身がブラックボックス化しやすく，どの特徴量が重要かを知ることは難しい．
- 過去 $p$ ステップの特徴量をすべて入力に入れるることで，時系列データをそれっぽく扱うこともできる

[*5] たかだか可算無限なので，関数空間が非可算無限次元なことを考えると，たかが知れてる，のか？

[*6] たしか，1 隠れ層の 3 層ニューラルネットワークも，隠れ層のノード数がある程度以上あれば，任意の連続関数の近似ができるハズ（普遍性原理）

**ハイパーパラメータ**

- `hidden_layer_sizes`: 隠れ層のユニット数（$(H^{(1)}, H^{(2)}, \ldots, H^{(L)},)$ という配列を指定）
- `activation`: 活性化関数[*7]
  - `identity`: $\varphi(x) = x$
  - `logistic`: $\varphi(x) = 1/(1 + \exp(-x))$
  - `tanh`
  - `relu`（デフォルト）: $\varphi(x) = \max(0, x)$
- `solver`: 最適化手法
  - `lbfgs`: ニュートン法（最急降下法が勾配情報のみ使うのに対し，曲率（2 次微分）の情報も使って収束性を向上．ただし計算量は増える（はず）．確定的な方法なので，局所解に陥りやすい（はず））
  - `sgd`: 確率的勾配降下法（上述）
  - `adam`: Adam（Adaptive Moment Estimation）法（SGD に更新が振動的にならない工夫（モーメンタム[*8]+RMSprop[*9]が追加されたもの）

（ここからは個人的にできそうだなと思うモデルたち：いわゆる時系列解析っぽいやつら．そもそも時系列データを扱っているのだから，こっちのほうが妥当な気もする．Python では `statsmodels` とかで実装できるハズ）

## 1.7 多次元自己回帰モデル（AR モデル）

自己回帰モデルは「時点 $n$ におけるモデル出力が時点 $n$ 以前のモデル出力に依存する確率過程」（Wikipedia）なので，入出力が同じ次元じゃなくてはならない（はず．多分．ウソかも）．

$$\begin{bmatrix} \boldsymbol{x}_n \\ y_n \end{bmatrix} = \sum_{k=1}^{K} \mathrm{A}_k \begin{bmatrix} \boldsymbol{x}_{n-k} \\ y_{n-k} \end{bmatrix} + \boldsymbol{b} + \boldsymbol{\varepsilon}_n, \quad \boldsymbol{\varepsilon}_n \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{\Sigma})$$

というモデルに対し，最適な $\mathrm{A}_k, \boldsymbol{b}$ を求める．ただし，このままだとモデルに現時点の情報を組み込めないため，ベクトルは $[\boldsymbol{x}_{n+1}^\top, y_n]^\top$ とかでもよいかもしれない．また，AR モデルはトレンドを持つようなデータには適用できないため，差分などを生成してから適用することもある．

参考：`statsmodels.tsa.vector_ar.var_model.VAR`

### 1.7.1 ARMA モデル

自己回帰モデルに移動平均モデルを組み合わせたもの．

$$\begin{bmatrix} \boldsymbol{x}_n \\ y_n \end{bmatrix} = \sum_{k=1}^{K_p} \mathrm{A}_k \begin{bmatrix} \boldsymbol{x}_{n-k} \\ y_{n-k} \end{bmatrix} + \sum_{k=1}^{K_q} \mathrm{C}_k \boldsymbol{\varepsilon}_{n-k} + \boldsymbol{b} + \boldsymbol{\varepsilon}_n$$

参考：statsmodels.tsa.statespace.varmax

---

[*7] 何を選べばいいか明確な指針があるわけではないが，ReLU がデフォルトである理由は，勾配消失問題を回避しやすいためだと思われる．

[*8] SGD は少しのサンプルをランダムに選んで勾配を計算するので，ばらつきやすく，安定しにくい．今回の勾配（更新量）に前回の更新量を足すことで，ばらつきを抑え，アルゴリズムを安定化させられる．

[*9] 勾配の 2 乗の移動平均の逆数を利用することで，急激に勾配が大きくなったら更新量を抑え，勾配が小さくなっても更新量を保つ．

### 1.8 状態空間モデル

有効なパッケージが見つかってないので，紹介だけ

$$\boldsymbol{z}_n = \mathrm{A}\boldsymbol{z}_{n-1} + \mathrm{B}\boldsymbol{x}_{n-1} + \boldsymbol{\varepsilon}_t$$
$$y_n = \mathrm{C}\boldsymbol{z}_n + \mathrm{D}\boldsymbol{x}_n + \eta_n$$

よく知らないけど，時系列分析の文脈ではもう少し簡単な（入力項が存在しなかったり，$\mathrm{A}, \mathrm{C}$ 行列が単位行列だったりする）モデルを指すらしい．そうすると，システムが明示的にかけるから，あとはカルマンフィルタで状態を当てるだけってそういうわけらしい．

## 2 課題 2：行動予測

### 2.1 ロジスティック回帰

交差エントロピー型誤差関数（$y_n = 0$ or $1$）

$$\underset{\boldsymbol{w}}{\text{minimize}} \ -\sum_{n=1}^{N} y_n \log \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n) + (1 - y_n) \log(1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))$$
$$\sigma(u) = \frac{1}{1 + \exp(-u)}$$

**（補足）交差エントロピー誤差関数**
誤差関数の指数を考えると

$$\underset{\boldsymbol{w}}{\text{maximize}} \ \prod_{n=1}^{N} \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n)^{y_n} (1 - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^{1-y_n}$$

シグモイド関数 $\sigma : (-\infty, \infty) \to (0, 1)$ は線形回帰で得られる予測値 $\boldsymbol{w}^\top \boldsymbol{x}$ を確率として解釈し直すことに対応している．交差エントロピー誤差関数は，この確率が正解ラベル $y_n$ にどれだけ適合しているかを表す．なお，これはベイズ的には尤度の最大化に相当する．
二乗和誤差関数 $\sum_{n=1}^{N} (y_n - \sigma(\boldsymbol{w}^\top \boldsymbol{x}_n))^2$ を使ってもよいが，シグモイド関数の形から正解の $0, 1$ 付近で勾配が非常に小さくなり，更新が遅くなるため，交差エントロピー誤差関数がよく用いられる

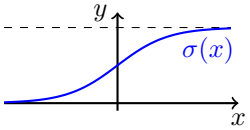


図 2　シグモイド関数

### 2.2 Support Vector Classification

いわゆる Support Vector Machine (SVM) で重要なのは，分類の境界面から最も近い点（サポートベクトル）のその距離（マージン）を最大化する，ということである．$\boldsymbol{x}$ の空間，すなわち $\mathbb{R}^M$ 空間での超平面 $\boldsymbol{w}^\top \boldsymbol{x} + w_0 = 0$

を用いて，2 つのクラスを分離する．いま，一旦この超平面ですべての訓練データ点を正しく分類できたとする．いくつかあり得る超平面の中で最も適切な超平面は，超平面からの距離 $\frac{|\boldsymbol{w}^\top \boldsymbol{x}_n + w_0|}{\|\boldsymbol{w}\|}$ が最も小さい点のそれが最大になるものである（$y_n = -1$ or $1$）．

$$\underset{\boldsymbol{w}, w_0}{\text{maximize}} \ \min_n \frac{y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0)}{\|\boldsymbol{w}\|}$$

ある定数 $\kappa$ を用いて $\kappa\boldsymbol{w}, \kappa w_0$ と定数倍しても，上の式に変化はない．そこで超平面との距離が最も近い点について $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0) = 1$ と固定すると

$$\begin{aligned} &\underset{\boldsymbol{w}, w_0}{\text{maximize}} \ \frac{1}{\|\boldsymbol{w}\|} \quad \Longleftrightarrow \quad \underset{\boldsymbol{w}, w_0}{\text{minimize}} \ \|\boldsymbol{w}\|^2 \\ &\text{subject to} \quad y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0) \geq 1, \quad \forall n \end{aligned}$$

ここまでの分類器は「超平面ですべての訓練データ点を正しく分類できた」としていた．これは「超平面との距離が最も近い点について $y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0) = 1$ と正規化できる」というところ，すなわち「すべての $n$ に対し，$y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0) \geq 1$ を満たす」という条件に表れている．そこで，この条件を少し緩和し「多少はこの不等式を破ってもいいが，その分ペナルティを加える」とする．

$$\begin{aligned} &\underset{\boldsymbol{w}, w_0}{\text{minimize}} \ C\sum_{n=1}^{N} \xi_n + \|\boldsymbol{w}\|^2 \\ &\text{subject to} \quad y_n(\boldsymbol{w}^\top \boldsymbol{x}_n + w_0) \geq 1 - \xi_n, \quad \xi_n \geq 0, \quad \forall n \end{aligned}$$

**（補足）カーネル法**

SVC においても，カーネル法を用いることができる．再び，生の入力データ $\boldsymbol{r}_n \in \mathbb{R}^L$ に対し，何らかの変換 $\boldsymbol{\phi} : \mathbb{R}^L \to \mathbb{R}^M$ をして，特徴量入力データ $\boldsymbol{x}_n$ を作っていたとする（$\boldsymbol{x}_n = \boldsymbol{\phi}(\boldsymbol{r}_n)$）．KKT 条件を考え，双対問題を導出すれば結局，

$$\begin{aligned} &\underset{\boldsymbol{a}}{\text{maximize}} \ \sum_{n=1}^{N} a_n - \frac{1}{2}\sum_{n=1}^{N}\sum_{m=1}^{N} a_n a_m y_n y_m \boldsymbol{\phi}(\boldsymbol{r}_n)^\top \boldsymbol{\phi}(\boldsymbol{r}_m) \\ &\text{subject to} \quad 0 \leq a_n \leq C \\ &\qquad\qquad \sum_{n=1}^{N} a_n y_n = 0 \end{aligned}$$

1.5 節での議論とまったく同様に，カーネル関数を用いることで，特徴量を無限個用意できる．

## 2.3 決定木

分類問題用に書き換える

$$\underset{j,t}{\text{minimize}} \sum_{y=0,1} p_L(t)(1-p_L(t)) + \sum_{y=0,1} p_R(t)(1-p_R(t))$$

$$p_L(y) = \frac{\sum_{n\in A_k, x_{nj}\le t} 1-|y_n-y|}{\sum_{n\in A_k, x_{nj}\le t} 1} = [n \in A_k, x_{nj} \le t \text{ の中で，} y_n = y \text{ のサンプル数確率}]$$

$$p_R(y) = \frac{\sum_{n\in A_k, x_{nj}> t} 1-|y_n-y|}{\sum_{n\in A_k, x_{nj}> t} 1}$$

## 2.4 ランダムフォレスト

略