知的情報処理
10.サポートベクターマシン

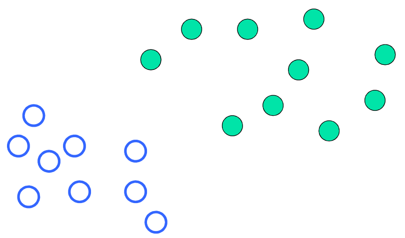
理工学部管理工学科
櫻井彰人

SVM の歴史

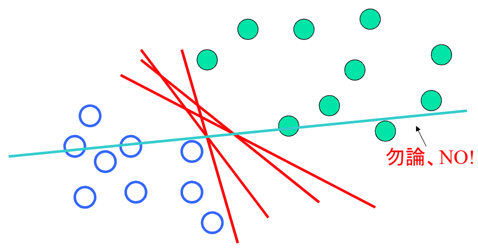
- SVM はVapnik の統計的学習理論から生まれた [3]
- SVM は1992年のCOLTで発表された(Boser, Guyon and Vapnik)[1]
- SVM の普及は速かった。手書き数字認識の精度が高かった
  - SVMは1.1% のテスト誤り. これは、念入りに作ったニューラルネットワーク(LeNet 4)と同じくらいであった.
    - 参考: [2]のSection 5.11. [3]のdiscussion
- SVM は今では、カーネル法の代表的な例と見なされている
  - 注: カーネル(核関数, kernel)の意味はいくつもあるので、注意のこと

[1] B.E. Boser *et al*. A Training Algorithm for Optimal Margin Classifiers. Proceedings of the Fifth Annual Workshop on Computational Learning Theory 5 144-152, Pittsburgh, 1992.
[2] L. Bottou *et al*. Comparison of classifier methods: a case study in handwritten digit recognition. Proceedings of the 12th IAPR International Conference on Pattern Recognition, vol. 2, pp. 77-82.
[3] V. Vapnik. The Nature of Statistical Learning Theory. 2nd edition, Springer, 1999.

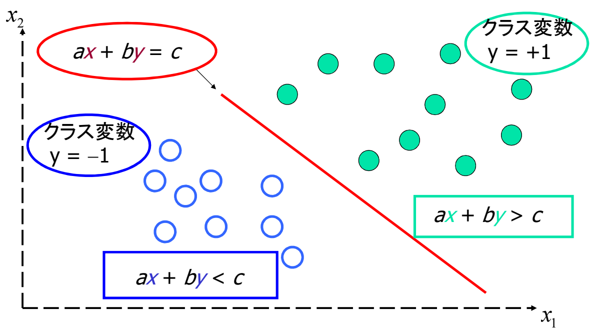
暫定的な仮定: 線形分離可能



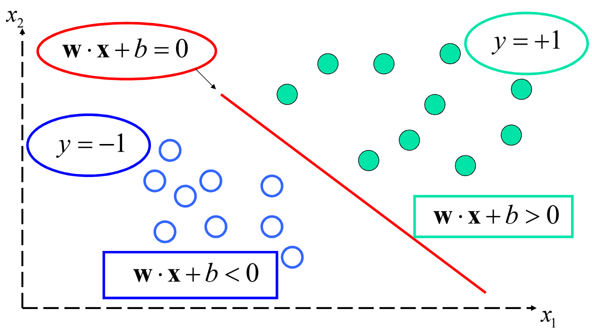
とはいえ、線形境界候補はたくさん



式で表すと

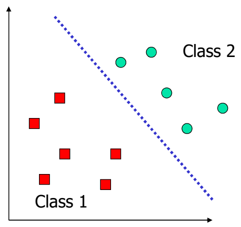


一般に



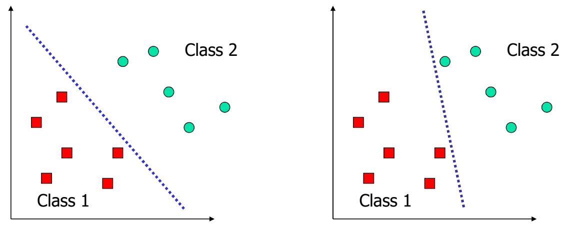
再: 決定境界候補は複数ある

- (簡単のために)分類クラスは2個、線形分離可能と仮定。
- 候補は多数(無限個)!
  - Perceptron学習アルゴリズムは、ある一つを見つける(初期値やデータの提示順序依存)
  - 他にもある
- どの候補も同様に良いのか?



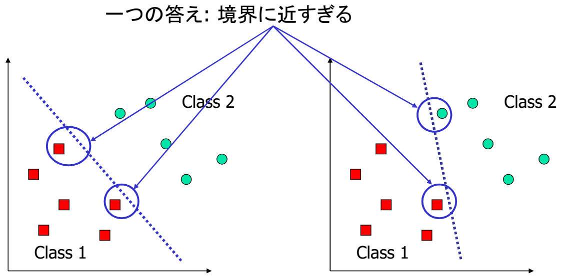
あまり芳しくない境界の例

- 「芳しくない」と考えるのは、何故だと思いますか？



あまり芳しくない境界の例

一つの答え: 境界に近すぎる



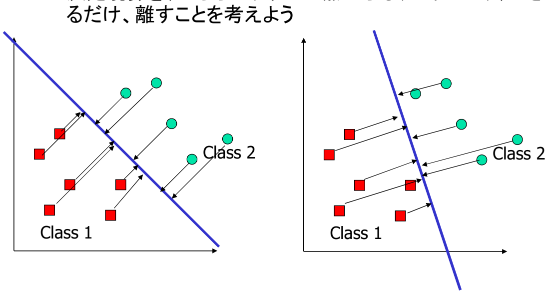
では、境界に近すぎると、なぜ、芳しくないのか？

データ点に近い境界

- 決定境界(判別境界)に近いデータ点があるのはよろしくない。なぜ？
  -
  -

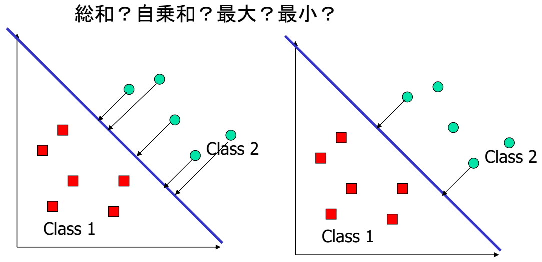
境界を離す

- 決定境界を、どちらのクラスの点からも(公平に！)、できるだけ、離すことを考えよう



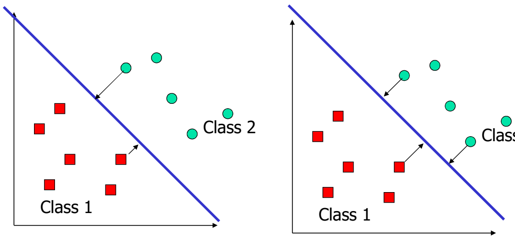
境界を離す: どの点について？

- 同じクラス内なら、どの点を優先するか？公平に？総和？自乗和？最大？最小？



境界を離す: 最近接点について

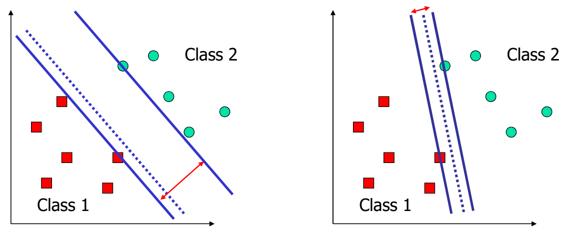
- 同じクラス内の、境界に最も近い点を、できるだけ離そう。境界への最短距離は、どのクラスも同じにしよう。



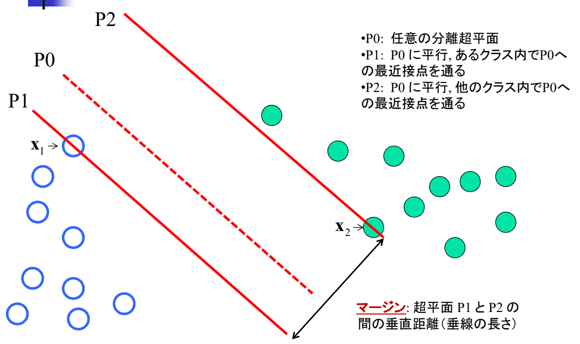
「境界に最も近い点を最大に離す」というのが、それまでになかったアイデア

「マージン」の導入

- マージン（大辞林）
  - (1)売買の差額金。利鞘(りざや)。
    (2)販売手数料。
    (3)本などの印刷部分を除く周辺の余白。
- 今回は、「余白」が近いでしょう



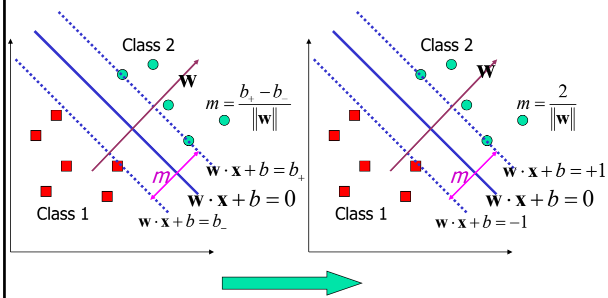
「マージン」の導入　その2



•P0: 任意の分離超平面
•P1: P0 に平行, あるクラス内でP0への最近接点を通る
•P2: P0 に平行, 他のクラス内でP0への最近接点を通る

マージン: 超平面 P1 と P2 の間の垂直距離（垂線の長さ）

式で表すと



改めて記述すると

- 訓練データを $\{x_1, \ldots, x_n\}$, 各x_i のクラスラベルを $y_i \in \{1,-1\}$ とする
- ちょっと工夫すると、決定境界は全ての訓練データを正しく分類する　$\Leftrightarrow$　$\forall i \;\; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$
- そうすると、欲しい決定境界は、次の制約付き最適化問題を解けば求まる

$$\text{Maximize } \frac{2}{\|\mathbf{w}\|} \quad \text{subject to } \forall i \;\; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to } \forall i \;\; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$$

- この解き方は、既に知っていますよね？

このまま解いてもよいのだが

制約付き最適化問題の解法

- 目的: 制約 $g(\mathbf{x}) = 0$　のもと $f(\mathbf{x})$ を最小化する
- $\mathbf{x}_0$ が解である必要条件:

$$\begin{cases} \left.\frac{\partial}{\partial \mathbf{x}}\left(f(\mathbf{x}) + \alpha g(\mathbf{x})\right)\right|_{x=x_0} = 0 \\ g(\mathbf{x}) = 0 \end{cases}$$

- α: ラグランジュ乗数 Lagrange multiplier
- 制約が複数個 $g_i(\mathbf{x}) = 0$, $i=1, \ldots, m$, のとき、ラグランジュ乗数 α_i は各制約ごとに必要

$$\begin{cases} \left.\frac{\partial}{\partial \mathbf{x}}\left(f(\mathbf{x}) + \Sigma_{i=1}^{m}\alpha_i g_i(\mathbf{x})\right)\right|_{x=x_0} = 0 \\ g_i(\mathbf{x}) = 0 \qquad \text{for } i = 1, \ldots, m \end{cases}$$

制約付き最適化問題の解法

- 制約が不等式 $g_i(\mathbf{x}) \le 0$ で表されるときも同様。ただし、ラグランジュ乗数（KKT乗数） α_i は正である必要がある
- もし $\mathbf{x}_0$ が次の制約付き最適化問題の解であるなら

$$\min_{\mathbf{x}} f(\mathbf{x}) \quad \text{subject to } g_i(\mathbf{x}) \le 0 \quad \text{for } i = 1,..,m$$

- $\mathbf{x}_0$ が次の式を満足するような $\alpha_i \ge 0$ (i=1, ..., m) が存在する

$$\begin{cases} \dfrac{\partial}{\partial \mathbf{x}}\left(f(\mathbf{x}) + \Sigma_{i=1}^{m} \alpha_i g_i(\mathbf{x})\right)\Big|_{x=x_0} = 0 \\ g_i(\mathbf{x}) \le 0 \qquad \text{for } i = 1,\dots,m \end{cases}$$

- 関数 $f(\mathbf{x}) + \Sigma_i \alpha_i g_i(\mathbf{x})$ はラグランジュ関数と呼ばれる

元の問題は

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to } \forall i \ \ y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \ge 1$$

$$\Rightarrow \quad \text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 \quad \text{subject to } \forall i \ \ 1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b) \le 0$$

- ラグランジュ関数は

$$L(\mathbf{w}, b, \boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + \sum_i \alpha_i \left(1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)\right)$$

- 微分を 0 とおくと

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \quad \Rightarrow \quad \mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$$

$$\frac{\partial L(\mathbf{w}, b, \boldsymbol{\alpha})}{\partial b} = \sum_i \alpha_i y_i \quad \Rightarrow \quad 0 = \sum_i \alpha_i y_i$$

双対問題

- ラグランジュ関数に $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ を代入すれば

$$\begin{aligned} L(\mathbf{w}, b, \boldsymbol{\alpha}) &= \frac{1}{2}\mathbf{w} \cdot \mathbf{w} + \sum_i \alpha_i \left(1 - y_i(\mathbf{w} \cdot \mathbf{x}_i + b)\right) \\ &= \frac{1}{2}\left(\sum_i \alpha_i y_i \mathbf{x}_i\right) \cdot \left(\sum_j \alpha_j y_j \mathbf{x}_j\right) + \sum_i \alpha_i \left(1 - y_i(\sum_j \alpha_j y_j \mathbf{x}_j \cdot \mathbf{x}_i + b)\right) \\ &= \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j + \sum_i \alpha_i - \sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j - b\sum_i \alpha_i y_i \\ &= \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j \end{aligned}$$

  - ただし、$\sum_i \alpha_i y_i = 0$ を用いた

これは学習データである

- これは α_i だけの関数である

双対問題

- 新しい目的関数は、α_i だけに関するものである
- 双対問題である: **w**が分かれば α_i が分かる; α_i が分かれば**w**が分かる
- 元の問題は主問題と呼ばれる
- 双対問題の目的関数は、最大化する
- 従って、双対問題は:

$$\text{Maximize } W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\text{subject to } \alpha_i \ge 0, \qquad \sum_i \alpha_i y_i = 0$$

ラグランジュ乗数を導入したときの α_i が満たすべき性質

ラグランジュ関数を b に関して微分して（0とおいて）得た条件

双対問題

$$\text{Maximize } W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j \mathbf{x}_i \cdot \mathbf{x}_j$$

$$\text{subject to } \alpha_i \ge 0, \qquad \sum_i \alpha_i y_i = 0$$

- これは2次計画問題(QP; quadratic programming)
  - α_i の大域的最適値を求めることが常に可能
- **w** は $\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i$ から求まる

解の性質

- α_i の多くはゼロ
  - **w** は「少数の」データ点の線形結合
  - このようなスパースな表現は、k-nn と同様に、データ圧縮とみなすことが可能である。
- **非零の** α_i **に対応する** $\mathbf{x}_i$ は support vectors (SV) と呼ばれる
  - 決定境界は SV のみによって決まる
  - t_j (j=1, ..., s) を s 個の SV の添え字とする. そうすると

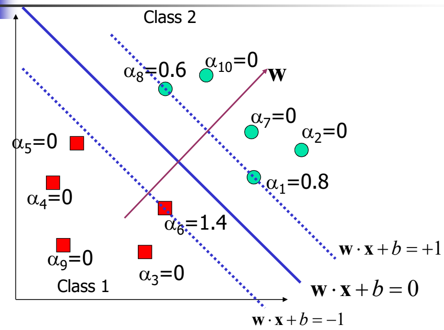
$$\mathbf{w} = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} \mathbf{x}_{t_j}$$

- 未知のデータ **z** に対して
  - $\mathbf{w} \cdot \mathbf{z} + b = \sum_{j=1}^{s} \alpha_{t_j} y_{t_j} (\mathbf{x}_{t_j} \cdot \mathbf{z}) + b$ を計算し、結果が負ならばクラス1とし、そうでなければクラス2とする
  - 注: **w** を明示的に構成する必要はない

2次計画

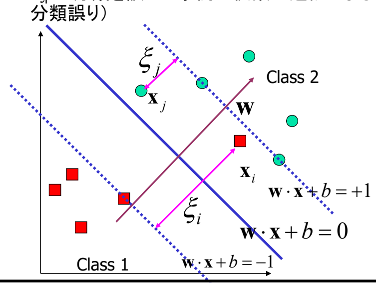
- 多くの手法が提案されている
- その多くは内点法に属する
  - 制約を満たしていないかもしれない、初期点から開始する
  - 解（候補）を、目的関数を最適化方向に進めたり、満たしていない制約の個数を減らしたりしながら、改善していく
- SVM については SMO (sequential minimal optimization) が良く知られている（他にもいろいろ）
  - 2変数の QP はトリビアル
  - SMO の繰り返しでは一対の (α_i,α_j) を取り、当該 QP をこの二変数について解く. この過程を収束するまで繰り返す
- 実際には, QP 解法プログラムをブラックボックスと考え、どう解かれているかは考えないことが多い

幾何的解釈



線形分離可能でないとき

- 分類において「誤り」ξ_i を認めよう; 当該誤り値は決定関数 $\mathbf{w}^T\mathbf{x}$+b の出力値に基づく
- ξ_i は分類を誤った事例の個数の近似となる（1より大の時、分類誤り）



ソフトマージン

- $\sum_i\xi_i$ が最小化されれば, ξ_i は:

$$\begin{cases} \mathbf{w}\cdot\mathbf{x}_i + b \ge 1-\xi_i & \text{if } y_i=1 \\ \mathbf{w}\cdot\mathbf{x}_i + b \le -1+\xi_i & \text{if } y_i=-1 \\ \xi_i \ge 0 & \forall i \end{cases}$$

ここはスラック変数を導入していても、不等式です。サポートベクトル以外のデータ点に対しては、不等式制約だからです

  - ξ_i はスラック変数である
  - 正しく分類されていれば ξ_i=0 である
  - ξ_i は分類誤りの上界である

実際に計算するときには適宜設定する必要がある。

- 最小化するのは $\frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$
  - C: 誤りとマージンのトレードオフを表すパラメータ
- 最適化問題

$$\text{Minimize } \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$$
$$\text{subject to } \forall i \;\; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \ge 1-\xi_i, \quad \xi_i \ge 0$$

主問題のラグランジアンは

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = \frac{1}{2}\mathbf{w}\cdot\mathbf{w} + C\sum_i \xi_i + \sum_i \alpha_i\big((1-\xi_i) - y_i(\mathbf{w}\cdot\mathbf{x}_i + b)\big) - \sum_i v_i\xi_i$$

従って、

$$\frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\partial \mathbf{w}} = \mathbf{w} - \sum_i \alpha_i y_i \mathbf{x}_i \qquad \frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\partial b} = \sum_i \alpha_i y_i \qquad \frac{\partial L(\mathbf{w},b,\boldsymbol{\alpha})}{\partial \xi_i} = C - \alpha_i - v_i$$

となるゆえ、停留点は、

$$\mathbf{w} = \sum_i \alpha_i y_i \mathbf{x}_i \qquad 0 = \sum_i \alpha_i y_i \qquad 0 = C - \alpha_i - v_i$$

これらを主問題に戻せば

$$L(\mathbf{w},b,\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j$$

ただし、条件は、

$$0 = \sum_i \alpha_i y_i \qquad 0 \le \alpha_i \le C \;\; (\text{for all } i)$$

最適化問題

- この制約付き最適化問題の双対問題は

$$\text{Maximize } W(\boldsymbol{\alpha}) = \sum_i \alpha_i - \frac{1}{2}\sum_{i,j}\alpha_i\alpha_j y_i y_j \mathbf{x}_i\cdot\mathbf{x}_j$$
$$\text{subject to } C \ge \alpha_i \ge 0, \qquad \sum_i \alpha_i y_i = 0$$

- **w** は $\mathbf{w} = \sum_{j=1}^{s}\alpha_{t_j} y_{t_j}\mathbf{x}_{t_j}$
- これは、線形分離可能な場合とそっくりである。違いは、各α_i に C という上限があることである
- 同様に、QP solver 用いて解く

多クラスの取り扱い

- SVMは2値分類しかできない。しかし、現実問題には、3クラス以上に分類する多クラス分類問題は多い。どうするか？
- 代表的な方法は、one vs. one と one vs. rest (one vs. all ともいう)である。
- One vs. one: n-クラスに分類する問題を n(n-1)/2 個の2クラス問題に変換する。
- One vs. rest: あるクラスと残りの全てのクラスに分ける2クラス問題(全部で n 個)に変換する

本日使用する e1071 の svm では、下のように書かれている。
For multiclass-classification with k levels, k >2, libsvm uses the 'one-against-one'-approach, in which $k(k$-1)/2 binary classifiers are trained; the appropriate class is found by a voting scheme.

線形 SVM: まとめ

- 分類器は、分離超平面 *separating hyperplane.*
- 最も重要な訓練データ点がサポートベクターとなる; それが当該超平面を決める.
- 2次計画問題を解けば、どの点 $\mathbf{x_i}$ がサポートベクターで非零のラグランジュ乗数 a_i に対応するかが分かる.
- 当該問題の双対問題においても解法においても、訓練データ点は、内積の中にしか現れない:

次のような $\alpha_1 \ldots \alpha_N$ を見出せ:
最大化: $\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i$ - $½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \boxed{\mathbf{x_i}^\mathrm{T}\mathbf{x_j}}$, 但し
(1) $\Sigma\alpha_i y_i = 0$
(2) すべての α_i につき: $0 \leq \alpha_i \leq C$

$$f(\mathbf{x}) = \Sigma\alpha_i y_i \boxed{\mathbf{x_i}^\mathrm{T}\mathbf{x}} + b$$

R における SVM

- svm を含むパッケージには e1071, kernlab, klaR, svmpath などがある。比較は、http://www.jstatsoft.org/v15/i09/paper にあり。
- 今回は、e1071 中の svm を用いてみる。

```
> library(e1071)
 要求されたパッケージ class をロード中です
> setwd("D:/R/Sample")
> # banknote は前回のものを用いる
> banknote <- read.csv ("09banknote.csv", header=TRUE)
> head(banknote)
  Length  Left Right Bottom  Top Diagonal Y
1  214.8 131.0 131.1    9.0  9.7    141.0 0
  略
> # svm の最も簡単な使い方。"formula" で試そう
> # svm( banknote[,-length(banknote)], banknote[length(banknote)] ) でもよい
> # 他のパラメータも指定する必要あり。
> # 分類するには(回帰もできる。それは次回) type="C" と指定する
> # kernel="linear" とする。kernel については次回。
> banknote.svm <- svm( Y ~ ., banknote, kernel="linear", type='C' )
> ( cm <- table( banknote$Y, predict(banknote.svm) ) )
   略
```

```
> ( accuracy <- sum(diag(cm))/sum(cm) )
[1] 0.995
>
> # 良すぎるね。
> # 過学習かもしれない。10-fold cross validation で調べてみたいが、それは次回としよう
>
> # COM実験でもちいた文字認識用データで試してみよう
>
> xy<-read.csv("optdigits.tra.csv", header=F)
> tt<-read.csv("optdigits.tes.csv", header=F)
>
> # formulaではなく、ファイルにて教師信号を指定しよう。
> # cost パラメータは、"C: 誤りとマージンのトレードオフを表すパラメータ" である
> ocr.svm <- svm(xy[,-65], xy[,65], kernel="linear", type="C", cost=1)
 警告メッセージ:
In svm.default(xy[, -65], xy[, 65], kernel = "linear", type = "C",  :
  Variable(s) ‘V1’ and ‘V40’ constant. Cannot scale data.
>
> ( cm <- table( tt[,65], predict(ocr.svm,tt[,-65]) ) )
```

警告メッセージは、今回の場合、V1とV40が全て同じ値である(実は0になっている)ために出された。学習時の警告ではなく、"scale"と書いてあることからわかるように、データの正規化時の警告である。この二つの属性は、学習には(テストにも)使用されない。

本日の課題

- iris データを SVM で分析(Speciesを予測するように学習)してみよう

```
library(e1071)
# iris を使えるようにする
#
data(iris)
# 属性名を調べる( head(iris) でデータを見る方がよい)
names(iris)
```

- "RにおけるSVM" で用いた文字認識データに対して、cost を変えて実験し、精度を比較して下さい。10倍、0.1倍として下さい。例えば、次のように変えて下さい。100000,1000,…,1,0.1,0.01,…,0.000001
- そして、"ソフトマージン" のスライド中の C を含む式をみて、考察して下さい。