知的情報処理
10.サポートベクターマシン

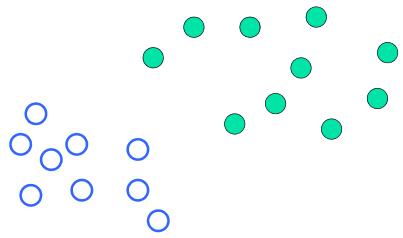
理工学部管理工学科
櫻井彰人

SVM の歴史

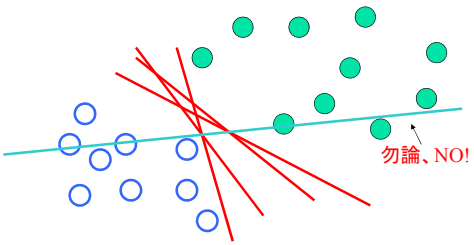
- SVM はVapnik の統計的学習理論から生まれた [3]
- SVM は1992年のCOLTで発表された(Boser, Guyon and Vapnik)[1]
- SVM は素早く普及した。手書き数字認識の精度が高かった
  - SVMは1.1% のテスト誤り. これは、念入りに作ったニューラルネットワーク(LeNet 4)と同じくらいであった.
    - 参考: [2]のSection 5.11. [3]のdiscussion
- SVM は今では、カーネル法の代表的な例と見なされている
  - 注: カーネル(核関数, kernel)の意味はいくつもあるので、注意のこと

[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.

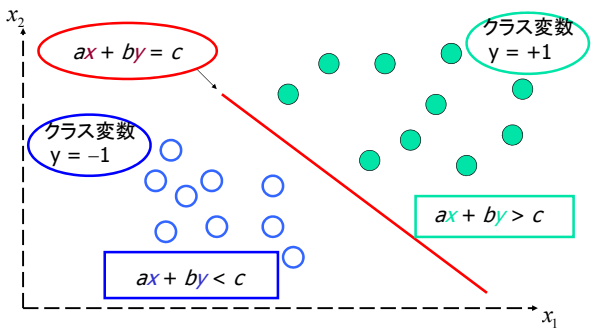
暫定的な仮定: 線形分離可能



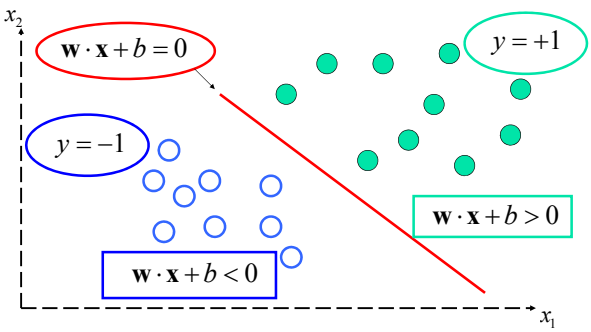
とはいえ、線形境界候補はたくさん



式で表すと

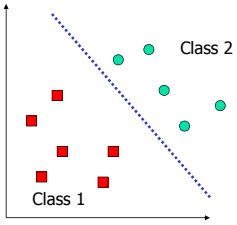


一般に



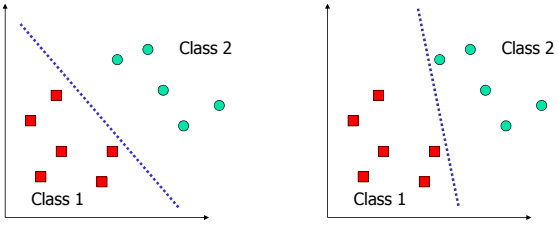
再: 決定境界候補は複数ある

- (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、あるーつを見つける(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?



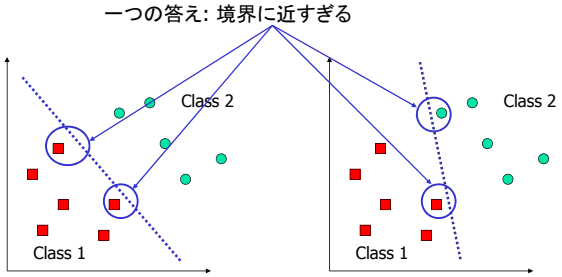
あまり芳しくない境界の例

- 「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

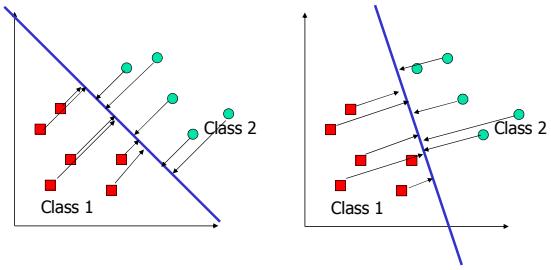
一つの答え: 境界に近すぎる



では、境界に近すぎると、なぜ、芳しくないのか?

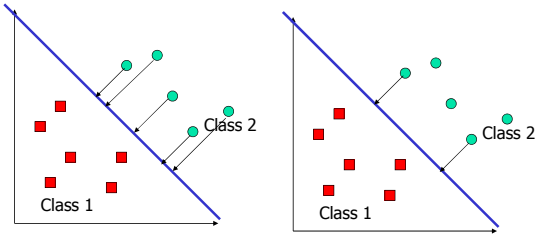
境界を離す

- 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



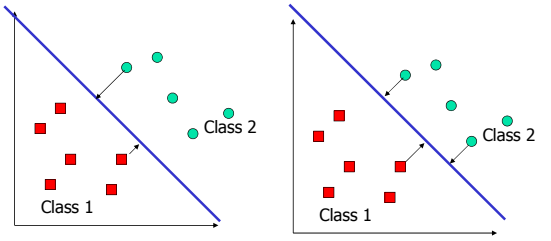
境界を離す: どの点について?

- 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?



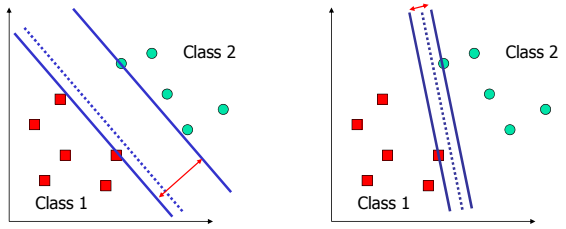
境界を離す: 最近接点について

- 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。

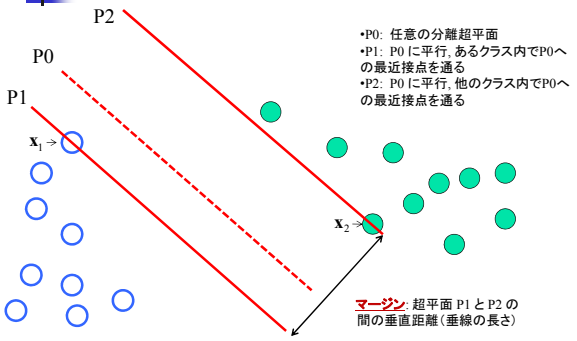


「マージン」の導入

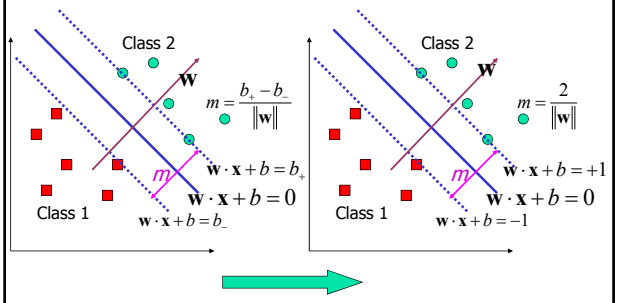
- マージン（大辞林）
  - (1)売買の差額金。利鞘(りざや)。
    (2)販売手数料。
    (3)本などの印刷部分を除く周辺の余白。
- 今回は、「余白」が近いでしょう



「マージン」の導入　その2



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　⇔　$\forall i \;\; y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1$
- 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

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

- この解き方は、既に知っていますよね？

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:

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

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

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

制約付き最適化問題の解法

- 制約が不等式 $g_i(\mathbf{x}) \leq 0$ で表されるときも同様。ただし、ラグランジュ乗数 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

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

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

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

- 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

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

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

- ラグランジュ関数は

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

- 微分を 0 とおくと

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

$$\Downarrow \qquad\qquad \Downarrow$$

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

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

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

  - ただし、$\sum_i \alpha_i y_i=0$ を用いた

- これは α_i だけの関数である

これは学習データである

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:

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

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

ラグランジュ乗数を導入したときの α_i の性質

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

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

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

- 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**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

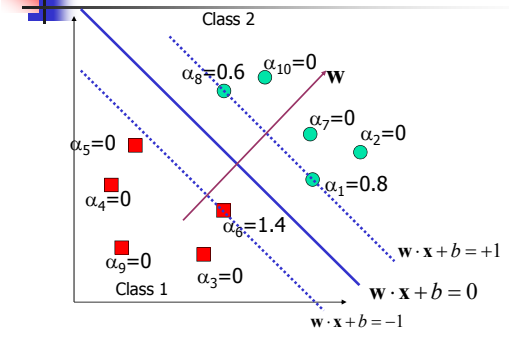
$$\mathbf{w}=\sum_{j=1}^{s}\alpha_{t_j} y_{t_j}\mathbf{x}_{t_j}$$

- 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

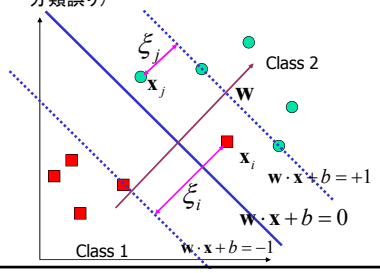
- 多くの手法が提案されている
- その多くは内点法に属する
  - 制約を満たしていないかもしれない、初期点から開始する
  - 解（候補）を、目的関数を最適化方向に進めたり、満たしていない制約の個数を減らしたりしながら、改善していく
- SVM については SMO (sequential minimal optimization) が良く知られている（他にもいろいろ）
  - 2変数の QP はトリビアル
  - SMO の繰り返しでは一対の (α_i,α_j) を取り、当該 QP をこの二変数について解く. この過程を収束するまで繰り返す
- 実際には, QP 解法プログラムをブラックボ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

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

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
- 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

実際に計算するときには適宜設定する必要がある。

- 最適化問題

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

主問題のラグランジアンは

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

従って、

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

となるゆえ、停留点は、

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

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、

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

最適化問題

- この制約付き最適化問題の双対問題は

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

- $\mathbf{w}$ 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

線形 SVM: まとめ

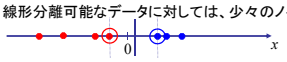
- 分類器は、分離超平面 *separating hyperplane.*
- 最も重要な訓練データ点がサポートベクターとなる; それが当該超平面を決める.
- 2次計画問題を解けば、どの点 $\mathbf{x}_i$ がサポートベクターで非零のラグランジュ乗数 α_i に対応するかが分かる.
- 当該問題の双対問題においても解法においても、訓練データ点は、内積の中にしか現れない:

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

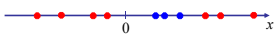
$$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$$

非線形 SVM

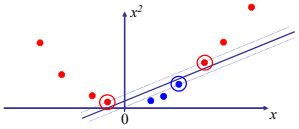
- 線形分離可能なデータに対しては、少々のノイズがあっても、うまくいく:



- しかし、データ集合が線形分離可能でなかったらどうしよう?

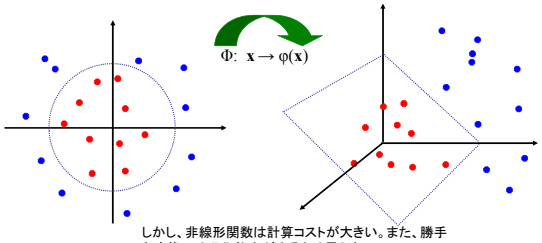


- 例えば… データをより高次元の空間に写像したらどうだろうか:



非線形 SVM: 特徴空間

- 一般的なアイデア: もともとの特徴空間は、いつでも、ある高次元特徴空間に写像すれば、線形分離可能となる:



しかし、非線形関数は計算コストが大きい。また、勝手な変換では汎化能力が出るとは思えない

カーネルトリック “Kernel Trick”

- 線形分類器が依拠していたのは、ベクター間の内積 $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^T\mathbf{x}_j$
- もし各点を高次元空間に、変換 Φ: $\mathbf{x} \rightarrow \varphi(\mathbf{x})$ を用いて写像すると, その高次元空間内での内積は:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$$

- **カーネル関数**は、変換後の内積の値が、変換前の内積の関数となるようなもの.
  - そうなると、計算が楽(計算量が少ない)
- 例:

2-次元ベクトル $\mathbf{x}=[x_1\ x_2]$ に対し $K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2$ とおく
このとき、次の式が成立する $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$:

$K(\mathbf{x}_i,\mathbf{x}_j)=(1+\mathbf{x}_i^T\mathbf{x}_j)^2 = 1+ x_{i1}^2x_{j1}^2 + 2\,x_{i1}x_{j1}x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2+ 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$

$= [1\ \ x_{i1}^2\ \sqrt{2}\,x_{i1}x_{i2}\ \ x_{i2}^2\ \sqrt{2}x_{i1}\ \sqrt{2}x_{i2}]^T\,[1\ \ x_{j1}^2\ \sqrt{2}\,x_{j1}x_{j2}\ \ x_{j2}^2\ \sqrt{2}x_{j1}\ \sqrt{2}x_{j2}]$

$= \varphi(\mathbf{x}_i)^T\varphi(\mathbf{x}_j)$ ただし $\varphi(\mathbf{x}) = [1\ \ x_1^2\ \sqrt{2}\,x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \sqrt{2}x_2]$

カーネル関数

- なぜカーネルを用いるか?
  - 分離可能でないものを分離可能にする.
  - データをより適切な表現空間に写像する
- よく使われるカーネル
  - 線形
  - 多項式 **$K(x,z) = (1+x^Tz)^d$**
  - RBF Radial basis function

$$K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\left\|\mathbf{x}_i-\mathbf{x}_j\right\|^2/2\sigma^2}$$

RBFがカーネルであることを示すのは少し面倒である。

$f(\mathbf{x}): X \rightarrow \mathbf{R} \Rightarrow f(\mathbf{x})f(\mathbf{y})$ はカーネル, カーネルの積はカーネル、 $\forall \mathbf{x},\mathbf{y} \in X,\ \lim_{n\to\infty} K_n(\mathbf{x},\mathbf{y}) = K(\mathbf{x},\mathbf{y})$

まとめ

- サポートベクターマシン (SVM) は
  - サポートベクターに基づいて超平面を決める
    - Support vector = 判定境界付近のクリティカルな点
  - 線形 SVM は線形分類器.
  - カーネル: 高次元へ写像するが、その内積は低次元の内積で簡単に計算できる
  - リスクの上界 (リスク = テストデータでの期待誤り)
  - (邪魔な属性が多いときの)分類器としてベスト?
    - 数1000も属性があるときは、安定的に強い
  - ポピュラー: SVMlight がきっかけ?
    - 速くて無料 (研究目的には)
    - 他にもいくつか: TinySVM, libsvm, ….

参考

- A Tutorial on Support Vector Machines for Pattern Recognition (1998) Christopher J. C. Burges
- S. T. Dumais, Using SVMs for text categorization, IEEE Intelligent Systems, 13(4):21-23, Jul/Aug 1998
- S. T. Dumais, J. Platt, D. Heckerman and M. Sahami. 1998. Inductive learning algorithms and representations for text categorization. *Proceedings of CIKM '98*, pp. 148-155.
- A re-examination of text categorization methods (1999) Yiming Yang, Xin Liu 22nd Annual International SIGIR
- Tong Zhang, Frank J. Oles: Text Categorization Based on Regularized Linear Classification Methods. Information Retrieval 4(1): 5-31 (2001)
- Trevor Hastie, Robert Tibshirani and Jerome Friedman, "Elements of Statistical Learning: Data Mining, Inference and Prediction" Springer-Verlag, New York.
- 'Classic' Reuters data set: http://www.daviddlewis.com /resources /testcollections/reuters21578/
- T. Joachims, *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.

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
> #
> banknote.svm <- svm( Y ~ ., banknote )
>
> # confusion matrix: 見てびっくりしないで下さい。
> table( banknote$Y, predict(banknote.svm) )
   略
```

```
> # 何が起こったかというと:
> # 被説明変数 Y の値は、0と1が 09banknote.csv に書かれています。
> # svm はこれは数値だと思い、svm 回帰という、svm に基づく回帰方法を行ったのです。
> # そのため出力値が実数になっているのです。
> # もし、被説明変数の値が、factor として legitimate なら、svm は分類だと考えます。
> #
> # 分類をさせるには、出力値が範疇値であることを指示するか、結果を丸めればよい
> # 分類をさせるには、type='C' とすればよい(?svm でヘルプを見てください)
>
> # 回帰をして結果を丸めてみる。
> banknote.svm <- svm( Y ~ ., banknote )
> ( cm <- table( banknote$Y, round(predict(banknote.svm)) ) )

     0   1
  0 99   1
  1   0 100
> ( accuracy <- sum(diag(cm))/sum(cm) )
[1] 0.995
>
> # 分類してみよう
> banknote.svm <- svm( Y ~ ., banknote, type='C' )
> ( cm <- table( banknote$Y, predict(banknote.svm) ) )

     0   1
  0 99   1
  1   0 100
> ( accuracy <- sum(diag(cm))/sum(cm) )
[1] 0.995
>
```

```
> # 閑話休題
> # kernel の default値は 'radial' つまり、radial basis function
> # 線形カーネルを試してみよう
> #
> banknote.svm <- svm( Y ~ ., banknote, type='C', kernel='linear')
> ( cm <- table( banknote$Y, predict(banknote.svm) ) )

     0   1
  0 99   1
  1   0 100
> ( accuracy <- sum(diag(cm))/sum(cm) )
[1] 0.995
>
> # いずれにしても良すぎるね。
> # 過学習かもしれない。では、10-fold cross validation で調べてみよう
>
> library(bootstrap)                # crossval を使用するために
>
> # 関数 crossval に必要な関数を定義する
> # なお、次では、svm に分類させている。
> theta.fit <- function (x,y) {
+   return( svm( x, y, type='C'  ) )
+ }
> theta.predict <- function( fit, x) {predict( fit, x) }
>
> # では crossval を使ってみよう
> # こちらは "formula" を使わない仕様なので
> # 説明変数( xy[,-length(xy)] )と被説明変数(xy[,length(xy)])を指定する
> xy <- banknote
> results <- crossval(xy[,-length(xy)], xy[,length(xy)], theta.fit,
+  theta.predict, ngroup=10)
```

```
>
> # confusion matrix と分類精度
> # svm には分類させたので
> (cm <- table( xy[,length(xy)], results$cv.fit  ))

    1  2
  0 98  2
  1  1 99
> (accuracy10CV <- sum(diag(cm))/sum(cm))
[1] 0.985
> # 回帰をさせたいなら results$cv.fit を round( results$cv.fit ) とすればよい
> #
> theta.fit <- function (x,y) {
+   return( svm( x, y ) )
+ }
> theta.predict <- function( fit, x) {predict( fit, x ) }
> xy <- banknote
> results <- crossval(xy[,-length(xy)], xy[,length(xy)],
+   theta.fit, theta.predict, ngroup=10)
> (cm <- table( xy[,length(xy)], round( results$cv.fit ) ))

     0   1
  0 98   2
  1   0 100
> (accuracy10CV <- sum(diag(cm))/sum(cm))
[1] 0.99
>
```

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)
```

10BreastCancer.csv

第三回 レポート課題

- これまで(皆さんが)学習した機械学習手法のうち、naïve Bayes, 決定木 と SVM(線形カーネルとRBFカーネル) とを iris データと Wisconsin Breast Cancerデータに適用し、その結果に基づき、手法の性質の違いを述べてください。
  - Breast Cancer データの取扱い方法については、次のスライドに記すことを参考にして下さい。
  - なお、svm にも、解の複雑度を制御するパラメータがあります。e1071のsvm では cost という引数です。default値は1です。今回はdefault で結構です。RBFカーネルには、このsvmでは、gammaがあります。defaultは1/ncol(x) で、BreastCancerのときは1/9になっています。その10倍と1/10倍を比較して下さい。
- 締め切りは、1/13(火曜日) 23:59 とします。
- 他の注意はこれまでと同じです。**特に: 必ず、自分でやって下さい。**

```
library(e1071)

#  Breast-cancer data
#     良性 ('benign') と悪性 ('malignant') を区別する。
#     パッケージ mlbench にあるデータ BreastCancer
#     nnet のプログラム例では、数値 0 と1 に直して回帰を行ったが、
#     本プログラム例では、もとのカテゴリデータのまま用いる。
#
setwd("D:/R/Sample")
bcdata <- read.csv('10BreastCancer.csv',header=TRUE)

# 属性名とデータの最初の部分を見ると、不要な(というよりあるべきでない)
# 属性が分る。勿論、慎重に考えるべきであるが
# 属性数、データ数もみてみよう

names(bcdata)
dim(bcdata)
head(bcdata)

# 測定値のない、すなわち、NA と書いてある行がある。
# アルゴリズムによっては、単に無視したり、誤動作したりする。
# 削除しておくと便利である(安易な解決方法ではある)

bcdata <- na.omit(bcdata)
dim(bcdata)
ndata <- dim(bcdata)[1]
```

```
# subset() 関数を用いて、不要な属性の削除ができる
# Classは分離してみよう

bc <- subset(bcdata,select=c(-Id,-Class))
bcclass <- subset(bcdata,select=Class)

# CV ではなく、学習データとテストデータを分けて実験してみよう
# 訓練データ(全データの部分集合)を作る

bctrain <- bc[1:400,]
bctrainclass <- bcclass[1:400,]

# テストデータ(全データの部分集合)を作る

bctest <- bc[401:ndata,]
bctestclass <- bcclass[401:ndata,]
```

svm での学習には、このニつのデータを用いて下さい。

```
# iris のデータから各クラス40個の学習データと
# 10個のテストデータは、例えば、次のようにして、
# 取り出すことができます
iristrain <- iris[c(1:40,51:90,101:140),]
iristest <- iris[c(41:50,91:100,141:150),]
```

テストデータを用いた予測は次のように行う

```
pred <- predict(model, bctest)
(cm <- table(bctestclass, pred ) )
(accuracy <- sum(diag(cm))/sum(cm))
```