パターン認識
Pattern Recognition

佐藤真一
Shin'ichi Satoh

国立情報学研究所
National Institute of Informatics

June 6, 2023

スケジュール (予定)

5/23	ハイブリッド	線形識別器、パーセプトロン、最小自乗誤差 (MSE) 識別器、Widrow-Hoff 則
5/30	オンライン (zoom) のみ	ニューラルネットワーク、深層学習
6/6	ハイブリッド	サポートベクターマシン
6/13	オンライン (zoom) のみ	直交展開、固有値展開
6/20		休講
6/27	ハイブリッド → オンラインへ変更の可能性あり	クラスタリング、デンドログラム、凝集型 (aggromerative) クラスタリング、k-means
7/4	ハイブリッド	グラフ、ノーマライズドカット、スペクトラルクラスタリング、ラプラシアンアイゲンマップ
7/11		予備日

サポートベクターマシン (Support Vector Machines) とは

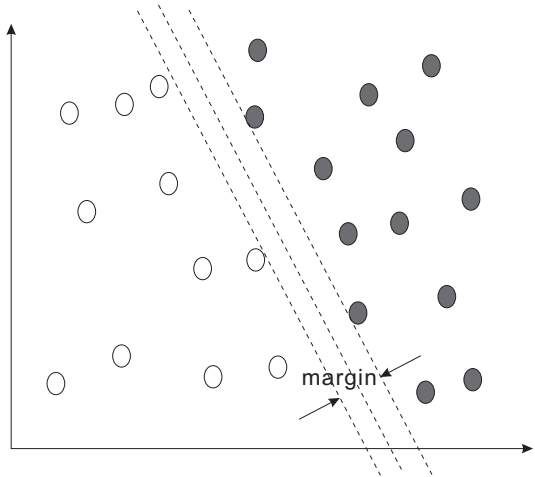
- おおもとのサポートベクターマシン (Support Vector Machines: SVM) アルゴリズム、すなわち線形 SVM は、Vladimir N. Vapnik らにより 1960 年代に考案された
- SVM は構造的損失 (Structural Risk) を最小にする: これは学習誤り (経験的損失: empirical risk) とモデルの複雑さ (VC 次元で測る) との組み合わせであり、それにより過学習 (overfitting) を効果的に避けることができる
- いわゆる「カーネルトリック」を使った SVM の非線形拡張は Bernhard E. Boser, Isabelle M. Guyon and Vladimir N. Vapnik らにより 1992 年に示唆された [1]
- ソフトマージンは Corinna Cortes and Vapnik により 1993 年に実現された [2]

[1] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. Proc. of COLT, 1992.

[2] Cortes, C., Vapnik, V. Support-vector networks. Mach Learn 20, 273–297 (1995).

線形 SVM の直感的説明

線形 SVM (Linear Support Vector Machines) は、マージンを最大化するような識別面を選別する



線形SVMの直感的説明

学習データセット $\mathcal{X} = \{x_i \in \mathbb{R}^d\}$ $(i = 1, \cdots, n)$ とラベル $y_i \in \{-1, 1\}$ を考える
ひとまず学習データは線形分離可能とする
マージンが最大になるような識別 (超) 平面

$$w \cdot x - b = 0$$

を見つけたい
マージンとは、以下の二つの超平面の間のマージンとする

$$w \cdot x - b = 1 \text{ and } w \cdot x - b = -1$$

従ってマージンは

$$\frac{2}{\|w\|}$$

線形SVMの定式化

従って以下の条件を満足しながら $\|w\|$ を最小としたい

$$w \cdot x_i - b \geq 1 \text{ for } x_i \text{ with } y_i = 1$$
$$w \cdot x_i - b \leq -1 \text{ for } x_i \text{ with } y_i = -1$$

この条件は以下と等価

$$y_i(w \cdot x_i - b) \geq 1 \text{ for all } i = 1, \cdots, n$$

よって以下の最適化問題を解けばよい

Minimize $\|w\|$
subject to $y_i(w \cdot x_i - b) \geq 1$ for all $i = 1, \cdots, n$.

主形式 (Primal Form)

これは以下のような二次計画問題 (quadratic programming optimization problem) となる

$$\arg\min_{(w,b)} \frac{1}{2}\|w\|^2$$

subject to (for any $i = 1, \cdots, n$)

$$y_i(w \cdot x_i - b) \geq 1$$

双対形式 (Dual Form)

ラグランジュ未定乗数 (Lagrange multipliers) α_i により、以下に変換できる

$$L = \left\{ \frac{1}{2}\|w\|^2 - \sum_{i=1}^{n} \alpha_i [y_i (w \cdot x_i - b) - 1] \right\}$$

$$\arg\min_{w,b} \max_{\alpha} L$$

以下が付随する Karush-Kuhn-Tucker 条件 (KKT 条件、Karush-Kuhn-Tucker conditions)

$$\frac{\partial L}{\partial w} = 0, \frac{\partial L}{\partial b} = 0$$
$$\alpha_i \geq 0, \alpha_i [y_i (w \cdot x_i - b) - 1] = 0$$

Karush-Kuhn-Tucker 条件

Maximize $f(x)$
subject to $g_i(x) \leq 0,\ h_j(x) = 0$

Karush-Kuhn-Tucker 条件

$$\nabla f(x) - \sum \alpha_i \nabla g_i(x) - \sum \lambda_j \nabla h_j(x) = 0$$

$$g_i(x) \leq 0, h_j(x) = 0$$

$$\alpha_i \geq 0$$

$$\alpha_i g_i(x) = 0$$

参考: ラグランジュ未定乗数決定法 (Lagrange Multipliers Method)

Maximize $f(x)$
subject to $h_j(x) = 0$

ラグランジュ関数 (Lagrangian):

$$L = f(x) - \sum \lambda_j h_j(x)$$

条件:

$$\nabla L = \nabla f(x) - \sum \lambda_j \nabla h_j(x) = 0$$

$$h_j(x) = 0$$

双対形式

$$\frac{\partial L}{\partial w} = 0 \ \rightarrow \ w = \sum \alpha_i y_i x_i$$

$$\frac{\partial L}{\partial b} = 0 \ \rightarrow \ \sum \alpha_i y_i = 0$$

$$\alpha_i [y_i (w \cdot x_i - b) - 1] = 0 \ \rightarrow \ \cdots$$

if $y_i(w \cdot x_i - b) - 1 > 0$ then $\alpha_i = 0$, otherwise $(y_i(w \cdot x_i - b) - 1 = 0)$ $\alpha_i > 0$
$\alpha_i > 0$ に対応する x_i はサポートベクターと呼ばれる

$$b = \frac{1}{|\{\alpha_i > 0\}|} \sum_{\alpha_i > 0} (w \cdot x_i - y_i)$$

双対形式

すべてを元の定式化に入れ込むと
Maximize:

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to:

$$\alpha_i \geq 0 \text{ and } \sum_{i=1}^{n} \alpha_i y_i = 0$$

これを二次計画問題として解く

二次計画問題 (Quadratic Programming Optimization)

Minimize

$$\frac{1}{2}x^T Qx + p^T x$$

subject to

$$Cx \leq b$$

$$C_{eq}x = b_{eq}$$

$$LB \leq x \leq UB$$

われわれの場合は以下のようにすればよい

$$x = \alpha,\ Q_{i,j} = y_i y_j x_i^T x_j,\ p = -[1\,1\,\cdots\,1],$$

$$LB = 0,\ C_{eq} = y,\ b_{eq} = 0$$

線形SVMの実装 (Python)

```
h = x * l
qpP = cvxopt.matrix(h.T.dot(h))
qpq = cvxopt.matrix(-np.ones(n), (n, 1))
qpG = cvxopt.matrix(-np.eye(n))
qph = cvxopt.matrix(np.zeros(n), (n, 1))
qpA = cvxopt.matrix(l.astype(float), (1, n))
qpb = cvxopt.matrix(0.)
cvxopt.solvers.options['abstol'] = 1e-5
cvxopt.solvers.options['reltol'] = 1e-10
cvxopt.solvers.options['show_progress'] = False
res = cvxopt.solvers.qp(qpP, qpq, qpG, qph, qpA, qpb)
alpha = np.reshape(np.array(res['x']), -1)

w = np.sum(x * (np.ones(n) * (l * alpha)), axis=1)
sv = alpha > 1e-5
isv = np.where(sv)[-1]
b = np.sum(w.T.dot(x[:, isv]) - l[isv]) / np.sum(sv)
```

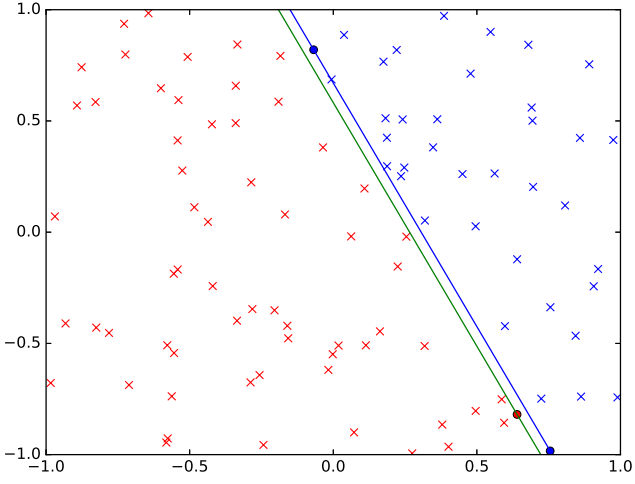
線形 SVM の実装 (Matlab)

```
h=x;
h(:,l<0)=-h(:,l<0);
options=optimset('Algorithm','interior-point-convex');
alpha=quadprog(h'*h,-ones(1,size(x,2)),[],[],l,0,...
    zeros(1,size(x,2)),[],[],options)';
w=sum(x.*(ones(size(x,1),1)*(l.*alpha)),2);
sv=alpha>1e-5;
isv=find(sv);
b=sum(w'*x(:,isv)-l(isv))/sum(sv);
```

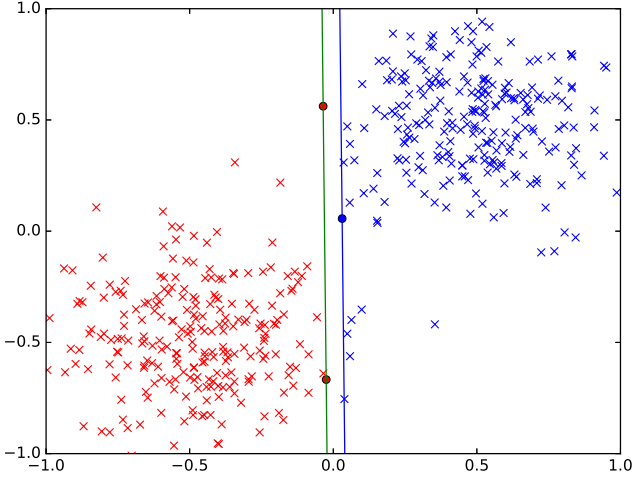
線形 SVM の実装 (Scilab)

```
h=x;
h(:,1<0)=-h(:,1<0);
alpha=quapro(h'*h,-ones(size(x,2),1),1,0,...
  zeros(size(x,2),1),[],1)';
w=sum(x.*(ones(size(x,1),1)*(1.*alpha)),2);
sv=alpha>1e-5;
isv=find(sv);
b=sum(w'*x(:,isv)-1(isv))/sum(sv);
```

実行例 (linear)



実行例 (slinear)



ソフトマージン拡張

学習データが線形分離可能でない場合はどうするか
ソフトマージンにより学習データを「できるだけ」線形分離する拡張法を導入する
非負のスラック変数 (slack variables) ξ_i を導入する

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i$$

われわれの目的関数は以下のようになる

$$\arg\min_{w,\xi,b} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i}^{n} \xi_i \right\}$$

subject to

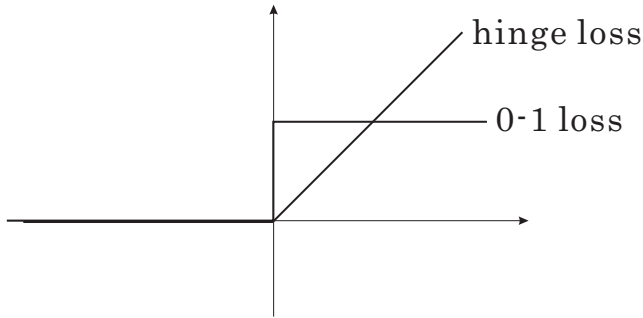
$$y_i(w \cdot x_i - b) \geq 1 - \xi_i, \xi \geq 0$$

ソフトマージン拡張

これは以下に等価

$$\arg\min_{w,b}\left\{\frac{1}{2}\|w\|^2 + C\sum_{i}^{n}\max(1 - y_i(w \cdot x_i + b), 0)\right\}$$

関数 $\max(1 - y_i(w \cdot x_i + b), 0)$ をヒンジ損失 (hinge loss) と呼ぶ



ソフトマージン拡張

これは KKT 条件によるラグランジュ未定乗数決定法で解くことができる
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to

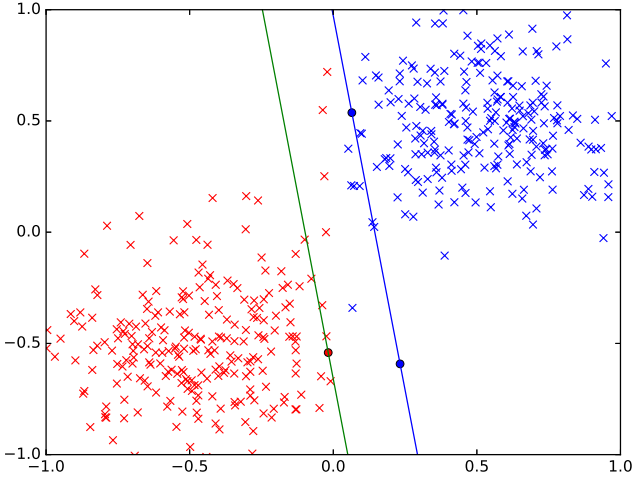
$$0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

サポートベクター:
x_i with $0 < \alpha_i < C$ (x_i with $\alpha_i = C$ は正しく識別されない).

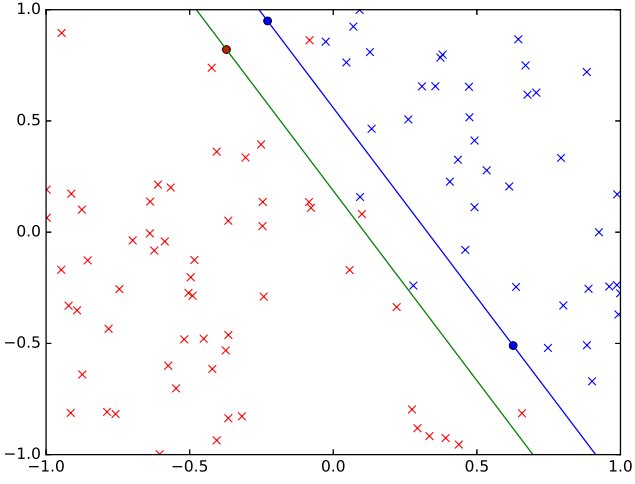
ソフトマージン拡張

実装は線形 SVM に対する非常に単純な変更で対応可能→宿題

実行例 (slinear)



実行例 (qlinear)



カーネル拡張

- 学習データが顕著に線形分離不可能であり、ソフトマージン拡張では対応できない場合は？
- 入力ベクトル x を非線形関数 $\phi(x)$ で変換し、変換後の空間で線形識別を行う
- 例: $x = [x_1\, x_2]^T$, $\phi(x) = [x_1\, x_2\, x_1^2\, x_1 x_2\, x_2^2]^T$.
  $\phi(x)$ に対する線形識別は、元の x に対する二次識別関数の適用と等価になる

カーネル拡張

- もし自明の非線形関数が利用可能ならば、データに対し変換を施し線形 SVM を適用すればよい
- しかし、多くの興味深い非線形関数はカーネル関数としてしか表現されない

$$k(x, y) = \phi(x) \cdot \phi(y)$$

- 例:
  - 多項式カーネル (Polynomial Kernel)

$$k(x, y) = (x \cdot y + 1)^p,\ k(x, y) = (x \cdot y)^p$$

  - ガウスカーネル (Gaussian Kernel, Radial Basis Function (RBF) Kernel)

$$k(x, y) = exp(-\frac{\|x - y\|^2}{2\sigma^2})$$

以下の線形 SVM の定式化を再考しよう
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to

$$0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

すべての x_i は x_j との内積でしか出現しないことに留意

従ってわれわれの問題は
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j \phi(x_i)^T \phi(x_j)$$
$$= \sum_{i=1}^{n} \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

subject to

$$0 \leq \alpha_i \leq C, \sum_{i=1}^{n} \alpha_i y_i = 0$$

α_i は二次計画問題ソルバーにより得られたものとすると、

$$w = \sum \alpha_i y_i \phi(x_i)$$

$\phi(x_i)$ は直接求められないことから、w も直接得ることはできない

$$\begin{aligned} b &= \frac{1}{\#sv} \sum_{i \in sv} (w \cdot \phi(x_i) - y_i) \\ &= \frac{1}{\#sv} \sum_{i \in sv} \left(\sum_j \alpha_j y_j \phi(x_j)^T \phi(x_i) - y_i \right) \\ &= \frac{1}{\#sv} \sum_{i \in sv} \left(\sum_j \alpha_j y_j k(x_j, x_i) - y_i \right) \end{aligned}$$

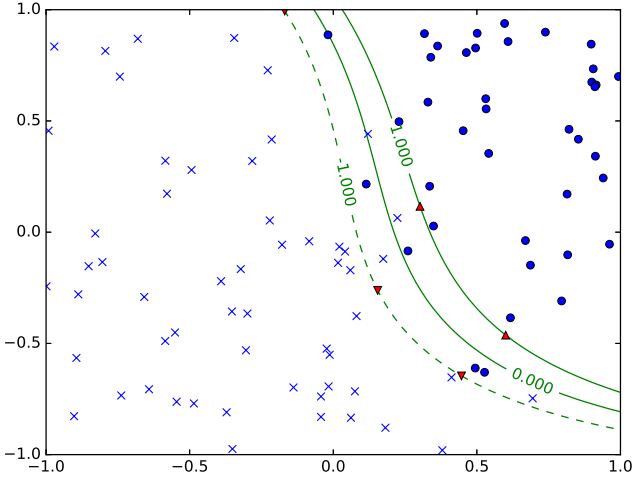
カーネル拡張

未知のデータ x を識別したいとする

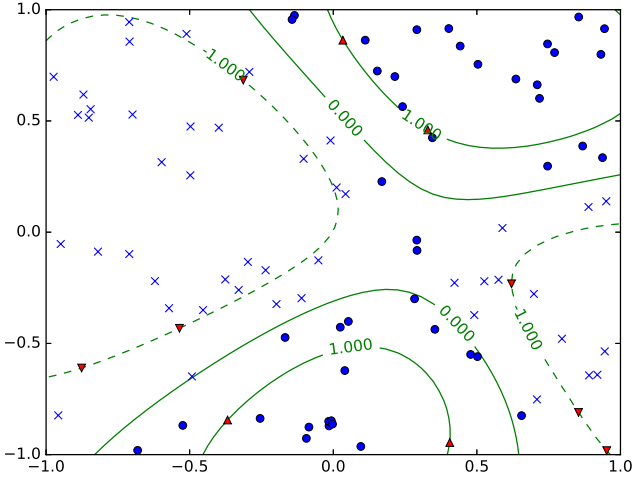
$$
\begin{aligned}
f(x) &= w \cdot \phi(x) - b \\
&= \sum \alpha_i y_i \phi(x_i)^T \phi(x) - b \\
&= \sum \alpha_i y_i k(x_i, x) - b
\end{aligned}
$$

$f(x)$ の正負により x を識別することができる

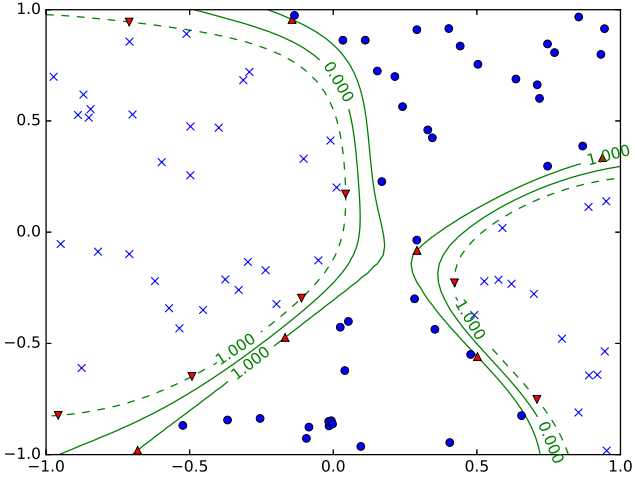
実行例 (qlinear, Polynomial kernel)



実行例 (nonlinear, C=1, RBF kernel)



実行例 (nonlinear, C=1000, RBF kernel)



宿題

- プログラミング課題と非プログラミング課題を課する
- プログラミング課題か非プログラミング課題かのいずれかを解いて提出すること
- プログラミング課題を解くことを推奨する
- もちろん両方解いてもらえると嬉しい
- 締切は 6/27

プログラミング課題

- 線形 SVM を拡張してソフトマージンを扱えるようにし、その振る舞いについてレポートせよ
- さらにカーネル拡張を施し、その振る舞いについてレポートせよ (発展課題)
- 注意: 既存の SVM パッケージを利用しないこと。配布した線形 SVM のコードに基づくなど、自分で実装すること
- 二次計画問題ソルバーは利用してよい
  - Python: cvxopt (“conda install cvxopt” 等)
  - Matlab: quadprog (optimization toolbox が必要)
  - Scilab: quapro (quapro toolbox が必要)
- linear, slinear, qlinear, nonlinear を識別してみて振る舞いを観察せよ

非プログラミング課題1

二つの超平面

$$w \cdot x - b = 1 \text{ と } w \cdot x - b = -1$$

の間のマージンが

$$\frac{2}{\|w\|}$$

であることを示せ

非プログラミング課題2

ソフトマージン SVM の主形式:

$$\arg\min_{w,\xi,b} \left\{ \frac{1}{2}\|w\|^2 + C\sum_{i}^{n} \xi_i \right\}$$

subject to

$$y_i(w \cdot x_i - b) \geq 1 - \xi_i,\ \xi \geq 0$$

から双対形式:
Maximize

$$L(\alpha) = \sum_{i=1}^{n} \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

subject to

$$0 \leq \alpha_i \leq C,\ \sum_{i=1}^{n} \alpha_i y_i = 0$$

を導け