「ソフトコンピューティング」(後半)

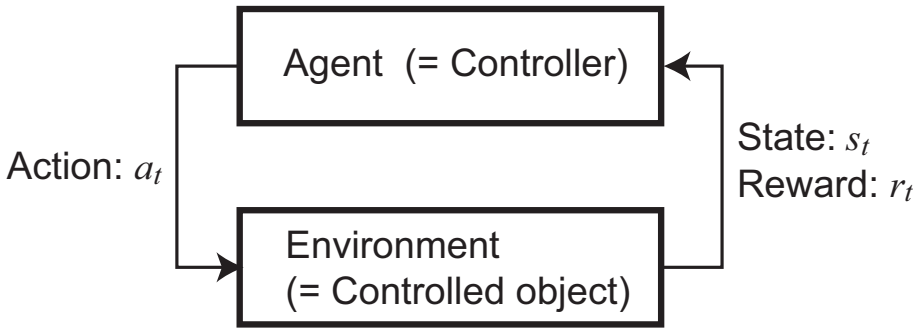
北海道大学 大学院情報科学研究科　　山下　裕

2009 年後期

強化学習

強化学習とは

強化学習 (Reinforcement Learning) とは: ある**エージェント**が試行錯誤を通じて未知の**環境**に適応する**学習制御**の枠組。**機械学習**の一種。一般的な教師付き学習とは異なり、明示的な教師が存在せず、かわりに**報酬**というスカラーの情報を手がかりに学習する。つまり、報酬が最も多く得られるような**方策 (policy)** を学習する。



環境

ここでは、環境 (Environment) は、有限状態数の**マルコフ決定過程 (Markov decision process:MDP)** であるとする。
つまり、離散的な時刻 t における環境の**状態**を s_t とし、環境への行動を a_t とすると、次の時刻 $t+1$ における状態 s_{t+1} の確率密度が、

$$\mathcal{P}^a_{ss'} = Pr(s_{t+1} = s' \,|\, s_t = s, a_t = a)$$

のように、s_t と a_t によって決まるとする。
$\mathcal{P}^a_{ss'}$ は遷移確率 (Transition probabilities) と呼ばれる。

報酬

即時報酬 (Reward): また、環境からの (即時) 報酬の確率密度も、

$$Pr(r_{t+1} \,|\, s_{t+1} = s', s_t = s, a_t = a)$$

のように与えられ、その期待値は、

$$\mathcal{R}^a_{ss'} = E(r_{t+1} \,|\, s_{t+1} = s', s_t = s, a_t = a)$$

時間 t 以降の累積報酬は、

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^T r_{t+T+1} = \sum_{k=0}^{T} \gamma^k r_{t+k+1}$$

で与えられる。ここで、$0 < \gamma < 1$ は割引率で遠い未来に得られる報酬を割り引いて評価するためのものである。T は無限大のこともある。

これら、$\mathcal{P}^a_{ss'}$, $\mathcal{R}^a_{ss'}$ が未知のときに、より多くの累積報酬を得るように学習する方法が強化学習である。

エージェントの方策

エージェントの方策 (Policy) とは、状態 $s_t = s$ のとき、行動 $a_t = a$ を取る確率密度 $\pi(s, a)$ を考える。

- ✔ R_t の何らかの予測ができれば、それを最大化する行動 a を取るのが最適であり、それをグリーディな方策という。
- ✔ 一方確率 ϵ でランダムな行動を取り、それ以外はグリーディな方策を取る場合は、ϵ-グリーディ方策という。

方策 π を固定して考える。$s_t = s$ のときの R_t の期待値を値関数 (Value function) という。

$$V^{\pi}(s) = E\{R_t \,|\, s_t = s\}$$

動的計画法=Bellman 方程式

$\mathcal{P}^a_{ss'}$, $\mathcal{R}^a_{ss'}$ が既知で、$T = \infty$ の場合、値関数は次の方程式を満たす。

$$\begin{aligned} V^\pi(s) &= E_\pi\{R_t \mid s_t = s\} = E_\pi\{r_{t+1} + \gamma R_{t+1} \mid s_t = s\} \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma E_\pi\{R_{t+1} \mid s_{t+1} = s'\}] \\ &= \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')] \end{aligned}$$

Bellman 方程式:

$$V^\pi(s) = \sum_a \pi(s,a) \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^\pi(s')]$$

最適な方策の下での Bellman 方程式:

$$V^*(s) = \max_a \sum_{s'} \mathcal{P}^a_{ss'}[\mathcal{R}^a_{ss'} + \gamma V^*(s')]$$

行動価値関数

行動価値関数:

$$Q^\pi(s,a) = E_\pi\{R_t \mid s_t = s, a_t = a\}$$

$T = \infty$ のとき、

$$Q^\pi(s,a) = E_\pi\{r_{t+1} + \gamma V^\pi(s_{t+1}) \mid s_t = s, a_t = a\}$$

行動価値関数を用いた Bellman 方程式:

$$Q^*(s,a) = \sum_{s'} \mathcal{P}^a_{ss'} \left(\mathcal{R}^a_{ss'} + \gamma \max_{a'} Q^*(s',a') \right)$$

グリーディな行動: $\operatorname{argmax}_a Q^*(s_t,a)$

値関数との関係: $V^\pi(s) = \sum_a \pi(s,a) Q^\pi(s,a), \quad V^*(s) = \max_a Q^*(s,a)$

予測問題と制御問題

ここでは、2 つの問題を考える。

- ✔ **予測問題**: 方策は固定する。値関数 $V(s)$ を学習し、今後の累積報酬の見込みを予測する。
- ✔ **制御問題**: 行動価値関数 $Q(s,a)$ を学習し、今後の累積報酬を最大化するような行動に漸近する。

	予測問題	制御問題
適格度トレースなし	TD(0)	Q 学習, Sarsa, Actor-Critic
適格度トレースあり	TD(λ)	Q(λ), Sarsa(λ), Actor-Critic(λ)

モンテカルロ法

まず、予測問題を考える。つまり、π は固定。
$V(s)$ を推定する問題であるので、その推定途中の $V(s)$ を $V^{(i)}(s)$ と書く。i は学習回数。
$\gamma < 1$ であるから、時刻 t から始めて十分長い試行後に、R_t が観測できる。

$$V^{\pi}(s) = E_{\pi}\{R_t \,|\, s_t = s\}$$

であるから、$V^i(s)$ を R_t に近づければよい。

モンテカルロ法 (Monte-Carlo method; MC 法):

$$V^{(i+1)}(s_t) = V^{(i)}(s_t) + \alpha[R_t - V^{(i)}(s_t)]$$

ここで、$0 < \alpha < 1$。

時刻 t における状態 s_t に関する学習は、即時にできず、十分長い試行後に可能。(R_t が即時にわからないため。)

TD(0) 学習 (1)

モンテカルロ法の欠点 (=学習が即時にできない) を改良する。
$s_t = s$ と $s_{t+1} = s'$ と $r_{t+1} = r$ がわかっているものとする。

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \cdots = r_{t+1} + \gamma R_{t+1}$$

であるので、

$$\begin{aligned} E_\pi\{R_t \,|\, s_t = s, s_{t+1} = s', r_{t+1} = r\} &= r + \gamma E_\pi\{R_{t+1} \,|\, s_{t+1} = s'\} \\ &= r + \gamma V^\pi(s') \end{aligned}$$

そこで、モンテカルロ法の R_t を $r_{t+1} + \gamma V^{(i)}(s_{t+1})$ に置き換える。

TD(0) 学習 (Temporal-Difference Learning):

$$V^{(i+1)}(s_t) = V^{(i)}(s_t) + \alpha[r_{t+1} + \gamma V^{(i)}(s_{t+1}) - V^{(i)}(s_t)]$$

$r_{t+1} + \gamma V^{(i)}(s_{t+1}) - V^{(i)}(s_t)$ は TD 誤差とよばれる。

TD(0) 学習 (2)

TD(0) 学習のアルゴリズム:

$V(s)$ のテーブルを初期化
s を初期化
各ステップに対して繰り返し
　　方策 π により行動 a を得る
　　行動 a をとり、報酬 r と次の状態 s' を観測
　　$V(s) \leftarrow V(s) + \alpha[r + \gamma V(s') - V(s)]$
　　$s \leftarrow s'$
試行が終端するまで繰り返し

Sarsa (1)

次に、制御問題について考える。制御問題においては値関数 $V^\pi(s)$ よりも、行動価値関数 $Q^\pi(s,a)$ を学習するほうが都合が良い。

もし、$Q^\pi(s_t,a_t)$ がわかっているならば、

$$Q^{(i+1)}(s_t,a_t) = Q^{(i)}(s_t,a_t) + \alpha[Q^\pi(s_t,a_t) - Q^{(i)}(s_t,a_t)]$$

とすればよいが、実際は $Q^\pi(s_t,a_t)$ は不明。そこで次の関係を使う。

$$Q^\pi(s,a) = E\{r_t + \gamma V^\pi(s_{t+1}) \,|\, s_t = s, a_t = a\}$$

の代わりに、$s_{t+1} = s'$ と $r_{t+1} = r$ がわかっているとき

$$\begin{aligned} E\{r_{t+1} + \gamma V^\pi(s_{t+1}) \,|\, s_t = s, a_t = a, s_{t+1} = s', r_{t+1} = r\} \\ = r + \gamma V^\pi(s') = r + E\{Q^\pi(s_{t+1}, a') \,|\, s_{t+1} = s'\} \end{aligned}$$

Sarsa (2)

Sarsa:

$$Q^{(i+1)}(s_t, a_t) = Q^{(i)}(s_t, a_t) + \alpha_t[r_{t+1} + \gamma Q^{(i)}(s_{t+1}, a_{t+1}) - Q^{(i)}(s_t, a_t)]$$

- ✔ ここでは、a_{t+1} を 1 ステップ前に既に求めていることが前提。
- ✔ 一般の π に対して $Q^{\pi}(s, a)$ を求めてもあまり意味は無いので、制御を考えると、グリーディな方策を取って Q^* を求めたい。しかし、グリーディな方策では全ての s と a の組を学習できないので、ランダム性を取り入れ ϵ-グリーディ方策を用いればよい。そして、ϵ は学習回数に応じて徐々に減らしてゼロに近づければ、Q^* を求められることが期待できる。
- ✔ 求めている Q は方策に依存するので、方策 on 型 TD 学習といわれる。

Sarsa (3)

Sarsa のアルゴリズム:

$Q(s,a)$ のテーブルを初期化
s を初期化
ある方策により行動 a を得る
各ステップに対して繰り返し
　　行動 a をとり、報酬 r と次の状態 s' を観測
　　s' に対し、ある方策 a' を求める
　　$Q(s,a) \leftarrow Q(s,a) + \alpha_t[r + \gamma Q(s',a') - Q(s,a)]$
　　$s \leftarrow s'$, $a \leftarrow a'$
試行が終端するまで繰り返し

Q 学習 (1)

次に、最適な Q^* を直接学習する方法を考えよう。

もし、$Q^*(s_t, a_t)$ がわかっているならば、

$$Q^{(i+1)}(s_t, a_t) = Q^{(i)}(s_t, a_t) + \alpha[Q^*(s_t, a_t) - Q^{(i)}(s_t, a_t)]$$

とすればよいが、実際は $Q^*(s_t, a_t)$ は不明。そこで次の関係を使う。

$$Q^*(s, a) = E\{r_t + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a\}$$

であるが、$s_{t+1} = s'$ と $r_{t+1} = r$ がわかっているとき

$$\begin{aligned} &E\{r_{t+1} + \gamma V^*(s_{t+1}) \,|\, s_t = s, a_t = a, s_{t+1} = s', r_{t+1} = r\} \\ &= r + \gamma V^*(s') = r + \max_{a'} Q^*(s', a') \end{aligned}$$

s' と r は、ある「標本過程」のデータということになる。

Q 学習 (2)

Q 学習:

$$Q^{(i+1)}(s_t, a_t) = Q^{(i)}(s_t, a_t) + \alpha_t [r_{t+1} + \gamma \max_{a'} Q^{(i)}(s_{t+1}, a') - Q^{(i)}(s_t, a_t)]$$

- ✔ a_t を与えている実際の方策とは無関係に、最適な $Q^*(s_t, a_t)$ を学習できる。⇒ 方策 off 型 TD 学習
- ✔ 全ての s と a の組み合わせが十分な回数現れ、かつ

$$\sum_{t=0}^{\infty} \alpha_t \to \infty, \qquad \sum_{t=0}^{\infty} \alpha_t^2 < \infty$$

と言う仮定のもとで、確率 1 で最適な $Q^*(s_t, a_t)$ に収束。

- ✔ a_t を与える方策としてはランダムでも理論的には収束するが、収束を早めるために、ϵ-グリーディ方策や、ボルツマン分布を利用したソフトマックス手法などが使用されている。

Q 学習 (3)

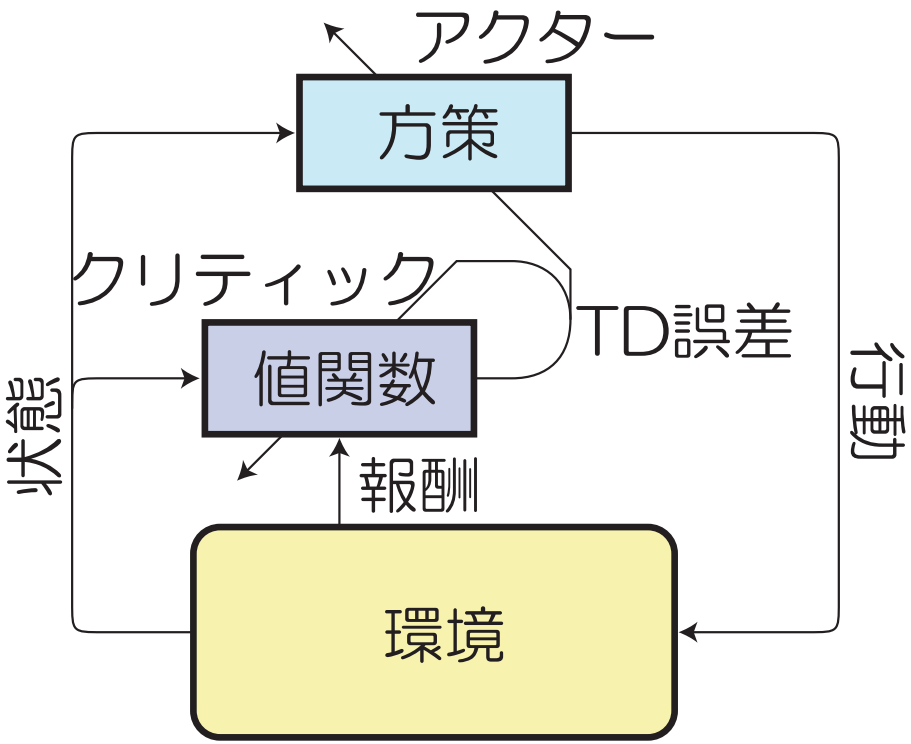
Q 学習のアルゴリズム:

$Q(s,a)$ のテーブルを初期化
s を初期化
各ステップに対して繰り返し
　　ある方策により行動 a を得る
　　行動 a をとり、報酬 r と次の状態 s' を観測
　　全ての a' に対し
　　　　$Q(s',a')$ のテーブルを検索。最大値 $\max_{a'} Q(s',a')$ を探す
　　$Q(s,a) \leftarrow Q(s,a) + \alpha_t[r + \gamma \max_{a'} Q(s',a') - Q(s,a)]$
　　$s \leftarrow s'$
試行が終端するまで繰り返し

アクター・クリティック手法 (1)

アクター・クリティック手法は様々なバリエーションがあるので、ここで示すのはあくまで１つの方法である。

- ✔ **アクター**: 方策をつかさどる部分
- ✔ **クリティック**: 値関数の評価をする部分



アクター・クリティック手法 (2)

クリティック: TD 学習を採用
TD 誤差: $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

- ✔ TD 誤差が正: r_{t+1} が比較的大きい $= a_t$ は選ばれるべき。
- ✔ TD 誤差が負: r_{t+1} が比較的小さい $= a_t$ を選ぶ確率を下げるべき。

アクター: ソフトマックス手法 (Gibbs 分布)

$$\pi_t(s,a) = \Pr\{a_t \mid s_t = s\} = \frac{e^{p(s,a)}}{\sum_b e^{p(s,b)}}$$

行動優先度: $p(s,a)$

$$p(s_t,a_t) \leftarrow p(s_t,a_t) + \beta\delta_t$$

TD(λ) 法 (1)

1 ステップの TD 法:

$$R_t = r_{t+1} + \gamma V_t(s_{t+1})$$

n ステップの TD 法:

$$R_t^{[n]} = r_{t+1} + \gamma r_{t+2} + \cdots + \gamma^{n-1} r_{t+n} + \gamma^n V_t(s_{t+n})$$

ただし、モンテカルロ法と同じく事後学習となってしまう。
s_t だけを固定すれば、$E[R_t^{[n]}] = E[R_t]$。

λ 収益 (λ return): $R_t^\lambda = (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} R_t^{[n]}$

✔ $\lambda = 0$ のとき、TD(0)
✔ $\lambda = 1$ のとき、(無限回試行後の) モンテカルロ法

s_t だけを固定すれば、$E[R_t] = E[R_t^\lambda]$

TD(λ) 法 (2)

λ 収益アルゴリズム:

$$V^{(i+1)}(s_t) = V^{(i)}(s_t) + \alpha[R_t^{\lambda} - V^{(i)}(s_t)]$$

この方法を素直に考えれば事後学習となる。

TD(λ) 法 (3)

適格度トレースを用いた TD(λ) 法:

適格度トレース (eligibility trace): 全ての s に対し、

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & (s \neq s_t) \\ \gamma\lambda e_{t-1}(s) + 1 & (s = s_t) \end{cases}$$

s_t が "訪問" すれば 1 増加。それ以外は徐々に減衰。

TD(λ) 法 (4)

TD(λ) 学習のアルゴリズム (適格度トレースの表を用いた実装):

$V(s)$ のテーブルを初期化
全ての s に対し適格度トレース $e(s)$ をゼロに初期化
s を初期化
各ステップに対して繰り返し
　　方策 π により行動 a を得る
　　行動 a をとり、報酬 r と次の状態 s' を観測
　　$\delta_t = r + \gamma V(s') - V(s)$
　　$e(s) \leftarrow e(s) + 1$
　　全ての σ に対して:
　　　　$V(\sigma) \leftarrow V(\sigma) + \alpha\delta e(\sigma)$
　　　　$e(\sigma) \leftarrow \gamma\lambda e(\sigma)$
　　$s \leftarrow s'$
試行が終端するまで繰り返し

TD(λ) 法 (5)

- ✔ s_t だけではなく適格度トレースがゼロでない状態に対する V も同様に変更
- ✔ 実は、λ 収益アルゴリズムと等価。(証明は省略)
- ✔ 完全にオンラインで実行可能

Sarsa(λ) 法

$Q(s,a)$ を学習する場合は適格度トレース $e(s,a)$ も s と a の関数。

Sarsa(λ) のアルゴリズム:

$Q(s,a)$ のテーブルを初期化し、また、全ての s, a に対し $e(s,a)=0$
s を初期化
ある方策により行動 a を得る
各ステップに対して繰り返し
　　行動 a をとり、報酬 r と次の状態 s' を観測
　　s' に対し、ある方策 a' を求める
　　$\delta \leftarrow r + \gamma Q(s',a') - Q(s,a)$
　　$e(s,a) \leftarrow e(s,a) + 1$
　　全ての $\bar{s}$, $\bar{a}$ に対して:
　　　　$Q(\bar{s},\bar{a}) \leftarrow Q(\bar{s},\bar{a}) + \alpha\delta e(\bar{s},\bar{a})$
　　　　$e(\bar{s},\bar{a}) \leftarrow \gamma\lambda e(\bar{s},\bar{a})$
　　$s \leftarrow s'$, $a \leftarrow a'$
試行が終端するまで繰り返し

同様に Q(λ) も考えられるが、Watkins の Q(λ) はあまり良くない。

サポートベクターマシン

識別問題

識別問題: 入力ベクトル x を 2 つ (以上) のクラスに分類する問題。サポートベクターマシンを使う場合は、通常 2 クラス分類問題を考える。

学習サンプル

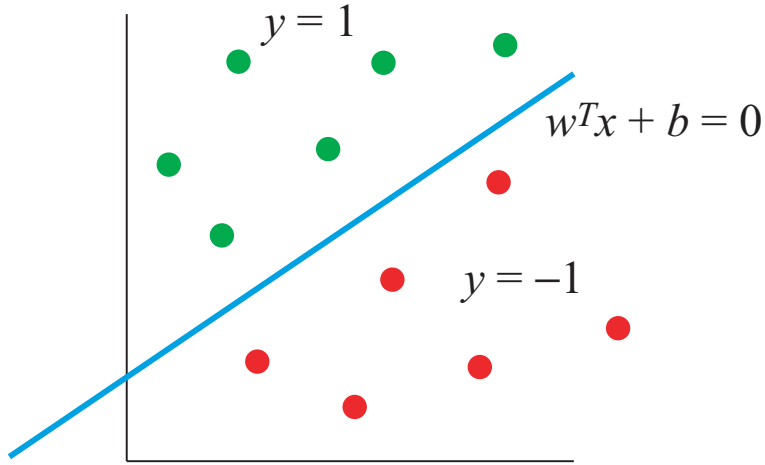
$$(x_1, y_1), (x_2, y_2), (x_3, y_3), \ldots$$

から学習。x_i はベクトル、y_i はクラス (± 1)。

線形識別器:

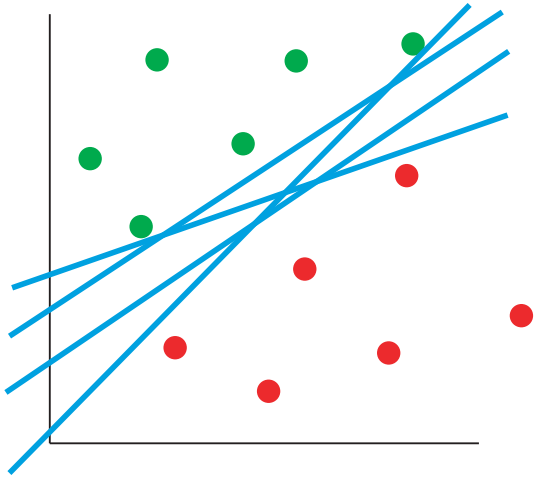
$$\begin{aligned} y &= \mathrm{sgn}[w^T x + b] \\ &= \mathrm{sgn}[w_1 x_1 + \cdots + w_n x_n + b] \end{aligned}$$

$$\mathrm{sgn}[u] = \begin{cases} 1 & (u \geq 0) \\ -1 & (u < 0) \end{cases}$$

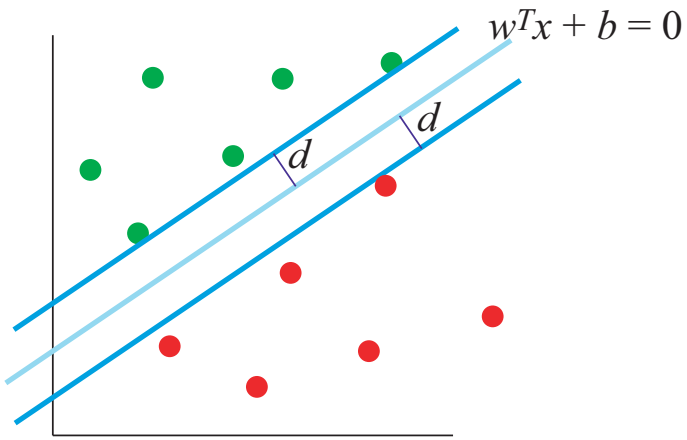


マージンの導入

正しい識別を行う超平面は一意に決まらない。



$\Rightarrow$ 真ん中を通る平面が望ましい



$$d = \min_i \frac{|w^T x_i + b|}{\|w\|}$$

d を最大化する w と b を求めたい。
このようにして求めた識別器をハードマージンの線形サポートベクタマシン (Support Vector Machine; SVM) という。

2 次計画問題への帰着

- ✔ とりあえず線形分離可能性を仮定。つまり、学習データを完全に分類できる超平面が存在するとする。
- ✔ $y = 1$ のクラスとの分離平面を $w^T x + b = 1$, $y = -1$ のクラスとの分離平面を $w^T x + b = -1$ とおいても、一般性を失われない。
- ✔ このとき、マージンは、

$$d = \frac{1}{\|w\|}$$

ハードマージン・線形 SVM のパラメータを求める問題 (主問題):

$$L(w) = \frac{1}{2}\|w\|^2 \to \min \qquad \text{subject to } y_i(w^T x_i + b) \geq 1$$

$\Rightarrow$ 2 次計画問題

凸制約を持つ 2 次計画問題は様々な方法で解くことができる。(内点法など)

ハードマージン SVM の双対問題 (1)

ラグランジュ乗数 α_i (≥ 0) の導入 (拡張評価関数):

$$L(w, b, \alpha) = \frac{1}{2}\|w\|^2 - \sum_{i=1}^{N} \alpha_i \{y_i(w^T x_i + b) - 1\} \to \min, \quad \alpha_i \geq 0$$

w, b による偏微分より、

$$\left(\frac{\partial L(w, b, \alpha)}{\partial w}\right)^T = w - \sum_{i=1}^{N} \alpha_i y_i x_i = 0$$

$$\left(\frac{\partial L(w, b, \alpha)}{\partial b}\right)^T = -\sum_{i=1}^{N} \alpha_i y_i = 0$$

これを拡張評価関数に代入

ハードマージン SVM の双対問題 (2)

ハードマージン・線形 SVM のパラメータを求める問題 (双対問題):

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

$$\text{subject to } \sum_{i=1}^{N} \alpha_i y_i = 0,\ \alpha_i \geq 0$$

双対問題の解 α_i から、パラメータを求める式は、

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$b = y_i - w^T x_i, \quad \text{such that } \alpha_i \neq 0$$

ハードマージン SVM の双対問題 (3)

Karush-Kuhn-Tucker 条件:

$$\sum_{i=1}^{N} \alpha_i y_i = 0$$

$$\alpha_i \geq 0, \quad i = 1, \ldots, N$$

$$y_i(w^T x_i + b) \geq 1, \quad i = 1, \ldots, N$$

$$w = \sum_{i=1}^{N} \alpha_i y_i x_i$$

$$\alpha_i \{y_i(w^T x_i + b) - 1\} = 0, \quad i = 1, \ldots, N$$

代数方程式・不等式の組に変換された。

最後の式より、ほとんどの α_i はゼロ。マージンを表す超平面上のデータ点に対応する α_i のみ、非ゼロ。

非ゼロな α_i に対応する x_i を**サポートベクター**という。

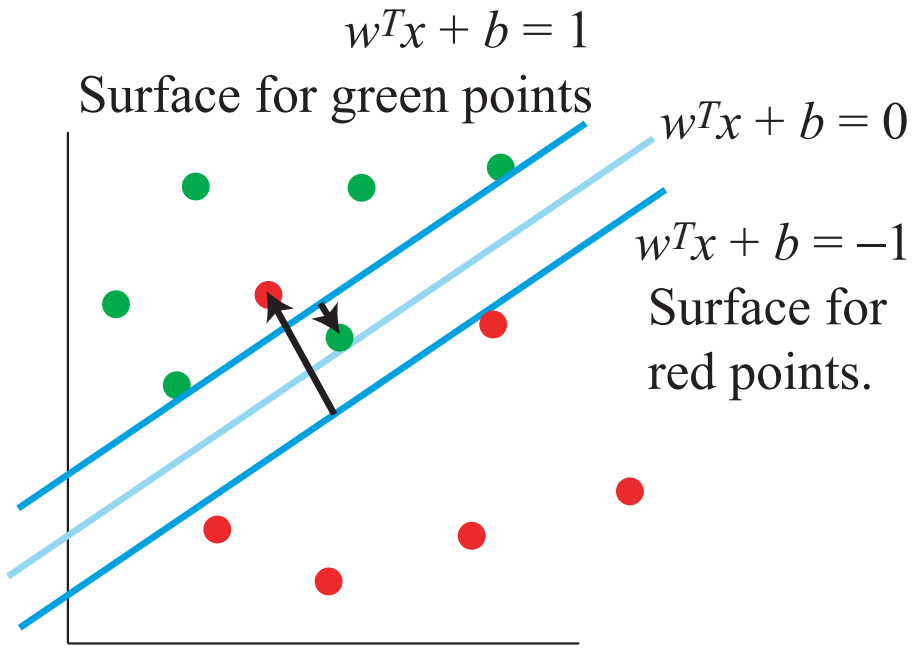
ソフトマージン SVM (1)

「厳密に線形分離」でない場合について考える。
制約を満たさないデータ点に関して、それを許すかわりにペナルティを評価関数に加える。

$$C \cdot \mathrm{Plus}[1 - y_i(w^T x + b)]$$

がペナルティ。ここで、

$$\mathrm{Plus}[s] = \frac{1}{2}(s + |s|)$$
$$= \begin{cases} s & (s \geq 0) \\ 0 & (s < 0) \end{cases}$$



ソフトマージン SVM (2)

ソフトマージン線形 SVM の主問題: 以下の w, b を見つけること

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \mathrm{Plus}[1 - y_i(w^T x + b)] \to \min$$

$\Downarrow$ スラック変数の導入

以下のような w, b, ξ_i を見つける。

$$\frac{1}{2}\|w\|^2 + C\sum_{i=1}^{N} \xi_i \to \min$$

$$\text{subject to } \xi_i \geq 1 - y_i(w^T x_i + b)$$

$$\xi_i \geq 0$$

ソフトマージン SVM の双対問題

Lagrange 乗数の導入:

$$L = \frac{1}{2}\|w\|^2 + C\sum_i \xi_i - \sum_i \alpha_i(\xi - 1 + y_i(w^T x_i + b)) - \sum_i \beta_i \xi_i$$

偏微分すると、

$$w = \sum_i \alpha_i y_i x_i, \qquad \sum_i y_i \alpha_i = 0, \qquad \alpha_i + \beta_i = C \quad \Rightarrow \quad \alpha_i \leq C$$

ソフトマージン SVM の双対問題:

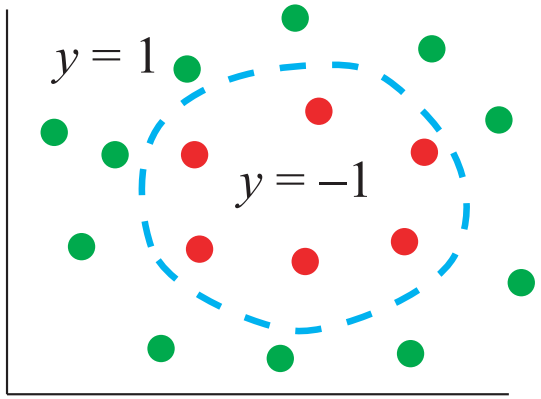
$$\max_\alpha \sum_i \alpha_i - \frac{1}{2}\sum_{i,j} \alpha_i \alpha_j y_i y_j x_i^T x_j$$

$$\text{subject to } 0 \leq \alpha_i \leq C$$

$$\sum_i y_i \alpha_i = 0$$

高次元への射影

そもそも、線形分離というより非線形分離が適している場合がある。



そのような場合、ベクトル x を高次元のベクトルに非線形写像で写して考えればよい。

$$z = \phi(x)$$

[例]

$x = (x_1, x_2)^T$ を射影して、$z = (x_1^2, x_1x_2, x_2^2, x_1, x_2, 1)^T$ のように取って、2 次の関数による判別器を作ることができる。

この例では、定数 1 をベクトルに含んでいるので、b は不要。
よって、このとき、$\sum_i y_i\alpha_i = 0$ の条件も不要。

カーネルトリック

高次元に射影した場合の SVM パラメータ決定:

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

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0$$

ここで、高次元同士の内積を 1 つのカーネルで書き表す。

$$k(x_i, x_j) = \phi(x_i)^T \phi(x_j)$$

→ カーネルトリック (計算量の削減になる)

正定値カーネル

正定値カーネル:

✔ (対称性) $k(x_i, x_j) = k(x_j, x_i)$

✔ (正定性) 任意の x_1, x_2,... に対して、グラム行列

$$[k(x_i, x_j)]_{(i,j)} = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_p) \\ \vdots & & \vdots \\ k(x_p, x_1) & \cdots & k(x_p, x_p) \end{bmatrix}$$

が準正定

非線形分離をする SVM は、内積を正定値カーネルに置き換える。

マーセルの定理により、正定値カーネルは $k(x, y) = \phi(x)^T \phi(y)$ のように分解できる。

カーネルの例

- ✔ **多項式カーネル**:
$$k(x, y) = (1 + x^T y)^p$$
- ✔ **Gaussian カーネル**:
$$k(x, y) = \exp\left(\frac{-\|x - y\|^2}{2\sigma^2}\right)$$
- ✔ **シグモイドカーネル**: (正定値カーネルではないが、使われることもある)
$$k(x, y) = \tanh(ax^T y - b)$$

カーネル化 SVM

カーネル化 SVM は以下の手順で解くことができる。

1. 最適化問題 (双対問題) を解く

$$\max_{\alpha} \sum_i \alpha_i - \frac{1}{2} \sum_{i,j} \alpha_i \alpha_j y_i y_j k(x_i, x_j)$$

$$\text{subject to } 0 \leq \alpha_i \leq C, \quad \sum_i \alpha_i y_i = 0$$

2. 次のようにサポートベクトルを見つける

$$S = \{i | 0 < \alpha_i < C\}, \quad O = \{i | \alpha_i = C\}, \quad I = \{i | \alpha_i = 0\}$$

3. 識別関数は、以下のように求まる。

$$y = \text{sgn}\left[\sum_{i \in S \cup O} \alpha_i k(x_i, x) + b\right], \quad b = y_i - \sum_j \alpha_j k(x_j, x_i) \quad (i \in S)$$

ニューラルネットワークとの関連

- ✔ シグモイドカーネルを使った判別関数は 3 層のニューラルネットワーク (出力層は sign 関数) になる。
  ただし、$I \to H$ の重みは学習データのベクトルそのもので、
  $H \to O$ の重みは双対問題の最適化から決定される。つまり、通常の学習ではない。
- ✔ 同様に、Gaussian カーネルを使った判別関数は、RBF ネットワーク (ニューラルネットワークの一種) に sign 関数を付けたものとして実現される。