ランキングSVMを用いたコンピュータ将棋における評価関数の学習

九州大学大学院システム情報科学府
修士２年　末廣 大貴

京都と将棋

京都将棋

京都将棋

* 1976年，田宮克哉氏が発表
* 京都銀閣将棋，京都銀閣金鶏秘譜将棋とも
* 盤面は縦横５マス
* 大山康晴十五世名人が絶賛？

（Wikipedia より）

参考：５五将棋

京都将棋

ルール

- 自陣，敵陣はない
- 香車の裏はと金
- 銀将の裏は角将
- 金将の裏は桂馬
- 飛車の裏は歩兵
- 一手動かすごとに駒を裏返す
- 取った駒は表裏どっちで打ってもよい

初期配置

目次

* コンピュータ将棋の歴史と研究背景
* 評価関数
* SVM と多項式カーネル
* ランキング SVM
* 実験
* まとめと今後の課題

コンピュータ将棋の歴史

1974年　コンピュータ将棋開発開始？

～ 探索と評価関数の試行錯誤 ～

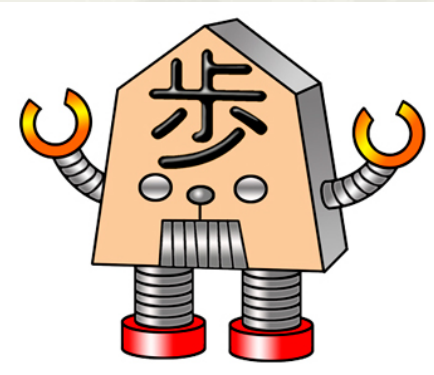
2005年 「激指」がアマ竜王戦でベスト16

2006年 「Bonanza」の登場

～評価関数の自動調整化～

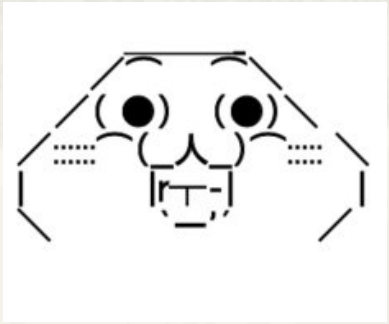
2010年 「あから2010」が清水市代女流王将（当時）に勝利

STR 誕生



STR

* 第20回世界コンピュータ将棋選手権で世界デビュー？
* 現在世界最弱と思われる
* 序盤はたまにまともな将棋を指す
* イメージキャラクターはやる夫



背景

* 強いコンピュータ将棋ソフトを作るには
  * 探索アルゴリズム
  * 評価関数
  * 高速化

背景

* 強いコンピュータ将棋ソフトを作るには
  * 探索アルゴリズム
  * 評価関数
  * 高速化

評価関数 f

入力：局面の特徴を表現するベクトル $\boldsymbol{x}$

出力：実数（評価値）

$\boldsymbol{w}$ を特徴の重みベクトルとしたとき,

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \boldsymbol{x}$$

特徴

将棋において重要そうな（？）ものを用意

* 駒の価値
* 玉に対する金，銀の位置
* 玉の危険度（玉周辺の駒の利き）
* 各駒の働き
  * 駒同士の位置関係
  * 駒が動けるか
  * 利きの数
  * ひもがついているか

  などなど・・・

評価関数の作り方

	従来	近年	提案手法	
特徴	作成者が用意	作成者が用意	自動生成	多項式カーネル
重み	手作業で調整	自動調整	自動調整	SVM

Bonanza Method [保木 '09]など

機械学習の分野ではかなり一般的な手法であり，
基礎データをとる意味でもやってみる価値はある

SVMと多項式カーネル

SVM（サポートベクトルマシン）

* ２値クラス分類に用いられる線形学習システム

* マージン最大化の原理に基づいて重みを求める

  ➡ 分類精度が高いことが保証されている

サンプル集合

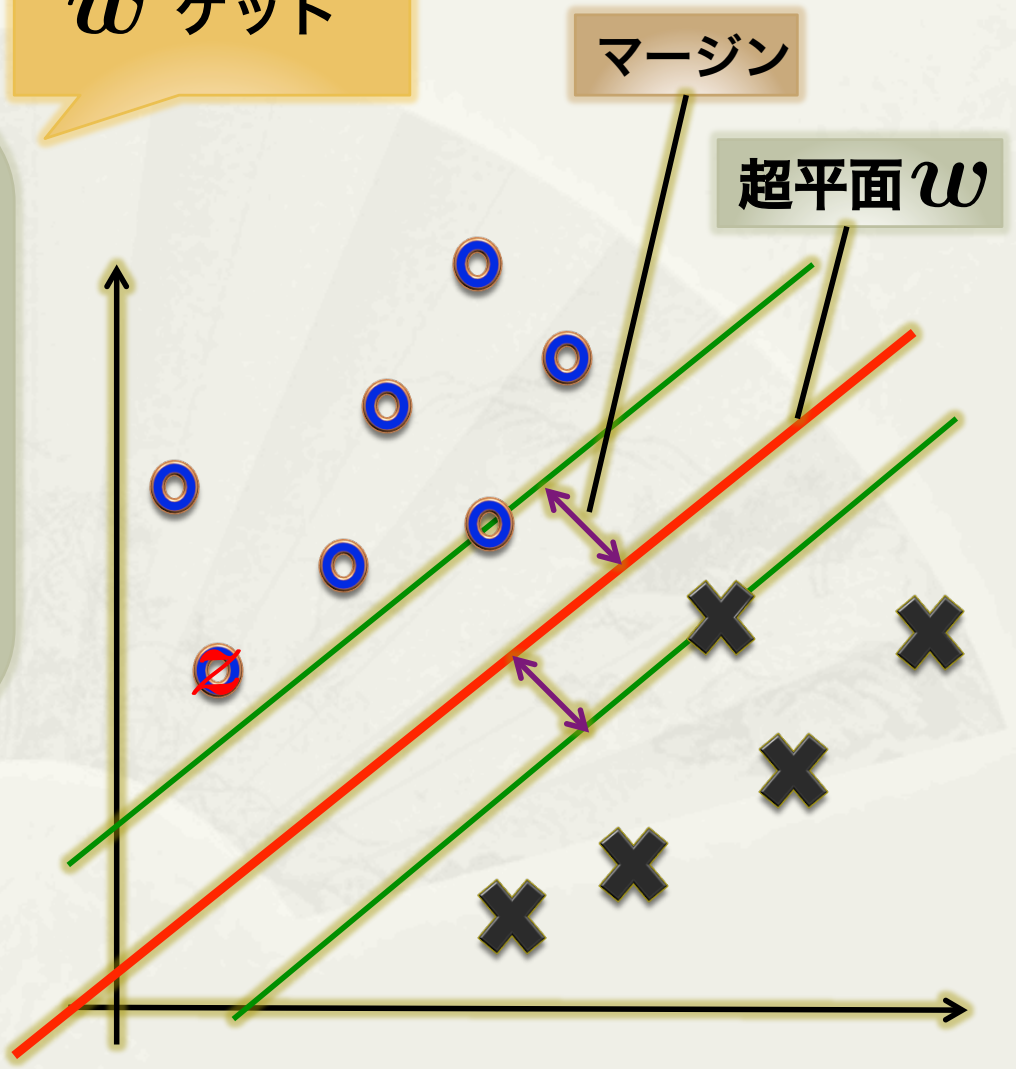
$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$$

$$\boldsymbol{x}_i \in \mathbb{R}^n, \quad y_i \in \{1, -1\}$$

$$\begin{bmatrix} \boldsymbol{x}_1 \cdot \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_1 \cdot \boldsymbol{x}_m \\ \vdots & \ddots & \vdots \\ \boldsymbol{x}_m \cdot \boldsymbol{x}_1 & \cdots & \boldsymbol{x}_m \cdot \boldsymbol{x}_m \end{bmatrix}$$

マージン最大化

$\boldsymbol{w}$ ゲット



未知の例 $\boldsymbol{z}$ が与えられる

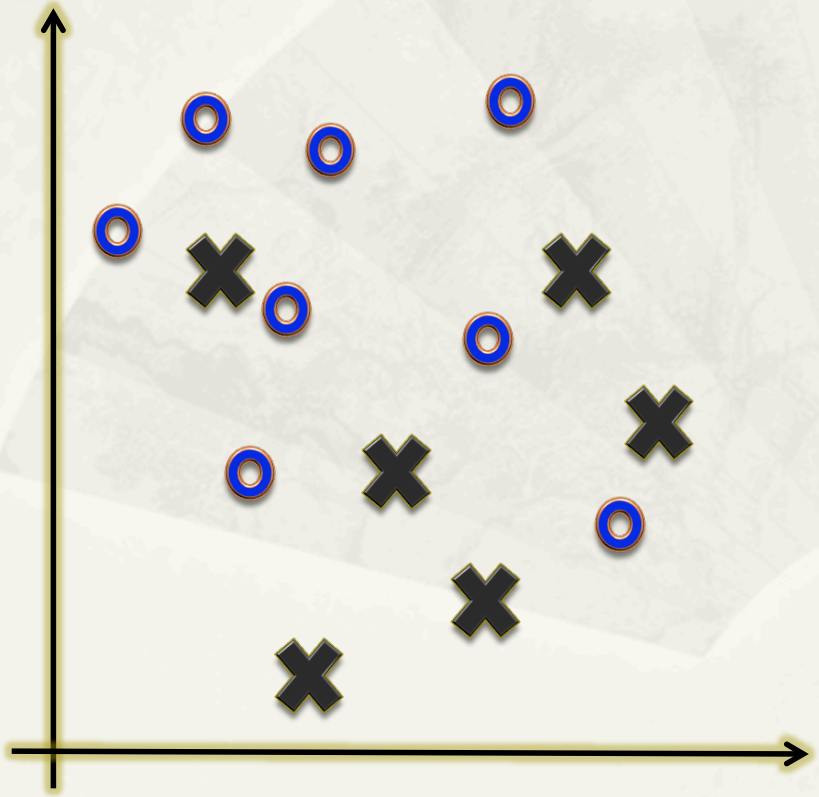
$$f(\boldsymbol{z}) = \boldsymbol{w} \cdot \boldsymbol{z}$$

超平面に応じて（f の符号）予測

非線形写像 ϕ による学習精度の向上

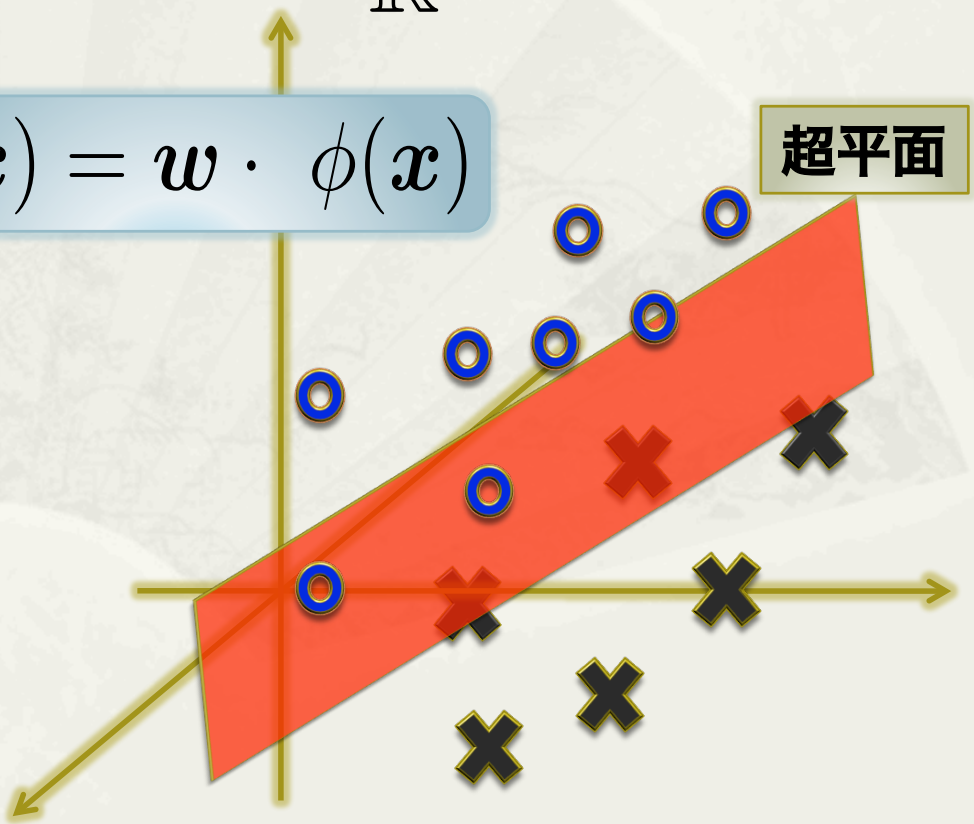
$$\phi : \mathbb{R}^n \rightarrow \mathbb{R}^N \qquad (N >> n)$$

$\mathbb{R}^n$



$\mathbb{R}^N$

$$f(x) = w \cdot \phi(x)$$



カーネルトリック

$$(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_m, y_m)$$

$$\boldsymbol{w} \in \mathbb{R}^N$$

$$\begin{bmatrix} \phi(\boldsymbol{x}_1) \cdot \phi(\boldsymbol{x}_1) & \cdots & \phi(\boldsymbol{x}_1) \cdot \phi(\boldsymbol{x}_m) \\ \vdots & \ddots & \vdots \\ \phi(\boldsymbol{x}_m) \cdot \phi(\boldsymbol{x}_1) & \cdots & \phi(\boldsymbol{x}_m) \cdot \phi(\boldsymbol{x}_m) \end{bmatrix}$$

$$= \sum_i^m \alpha_i \phi(\boldsymbol{x_i})$$

$$f(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x})$$

$$= \sum_i^m \alpha_i K(\boldsymbol{x}, \boldsymbol{x_i})$$

マージン最大化

$$K(\boldsymbol{a}, \boldsymbol{b}) = \phi(\boldsymbol{a}) \cdot \phi(\boldsymbol{b})$$

カーネル関数

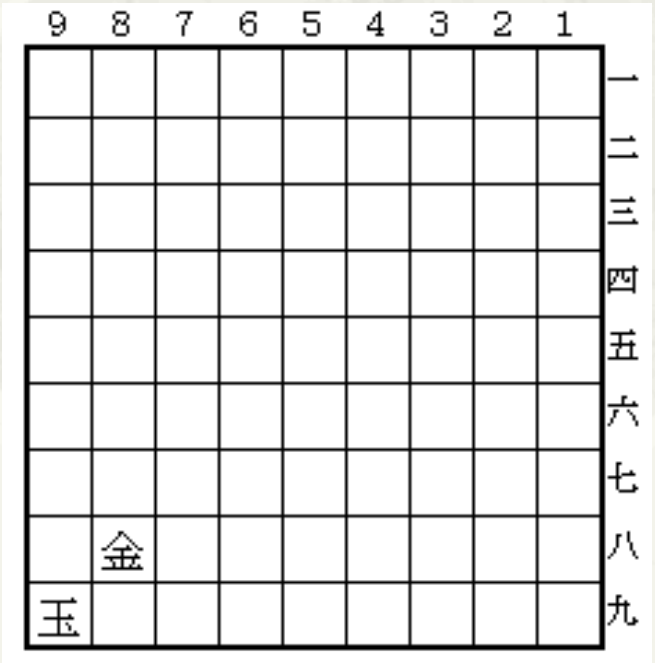
次元が増えているので計算時間が増大しそうだが・・・

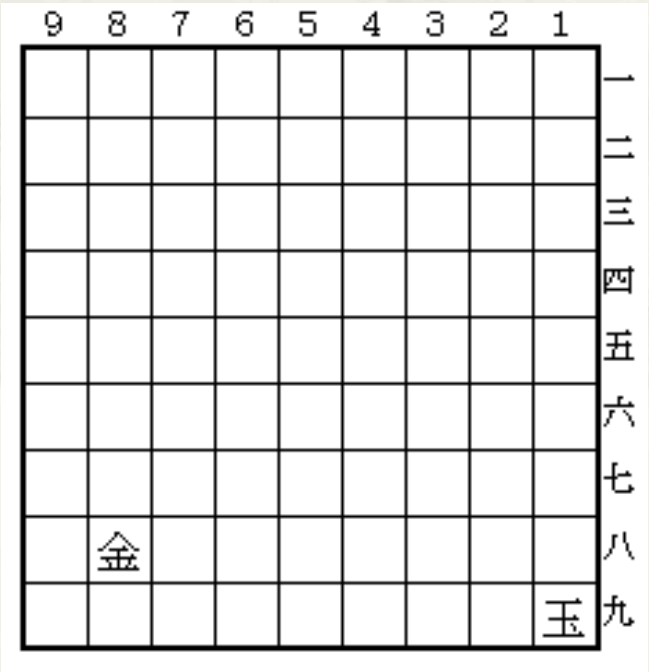
例：将棋

「金の位置」という特徴だけだと８八金の価値の違いが表せない
「玉に対する金の位置」（駒の位置の二項関係）という特徴が必要

$$x_{i,j,p} = \begin{cases} 1 & \text{駒 p がある} \\ 0 & \text{駒 p がない} \end{cases}$$

$$\boldsymbol{x}_{i,j,k,l,\mathrm{Kin},\mathrm{Ou}} = \boldsymbol{x}_{i,j,\mathrm{Kin}} \wedge \boldsymbol{x}_{k,l,\mathrm{Ou}}$$





特徴写像 ϕ の選択に向けて

このように将棋における複雑な特徴の多くが，局面を表現する
基本的で単純な特徴の d 項関係として表現可能な点に着目

基本的で単純な特徴（低次の特徴）
（１）盤上にある各駒の数
（２）持ち駒の種類とその数
（３）盤上の各枡に位置している駒の種類
（４）盤上の各枡に対する利きの有無とその駒の種類

（約 4000 次元）

ちなみに 2009 年度世界コンピュータ将棋選手権優勝の GPS 将棋が
学習に用いたものは 300万次元

* 玉に対する金，銀の位置　→ 玉の位置 ∧ 金銀の位置
* 玉の危険度（玉周辺の駒の利き）
  → 玉の位置 ∧ 敵駒の利き ∧ 敵駒の利き ∧ ・・・
* 各駒の働き
  * 駒同士の位置関係　→ 駒の位置 ∧ 駒の位置 ∧ ・・・
  * 駒が動けるか　→ 駒の利き ∧ 味方の駒の位置
  * 利きの数　→ 駒の利き ∧ 駒の利き ∧ ・・・
  * ひもがついているか　→ 駒の位置 ∧ 駒の利き

全てではないが，多くのものは特徴同士の d 項関係で表現可能

ナイーブにやると・・・

例：２次元の高々２項の関係を考慮　$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

$$\phi_{2,2}(\boldsymbol{x}) = (x_1 x_2, x_1, x_2, 1)$$

例：４０００次元の高々１０項の関係を考慮

$$\phi_{4000,10}(\boldsymbol{x}) = (x_1 x_2 x_3 \cdots, x_2 x_3 x_4 \cdots, \ldots)$$

約 3×10^{29} 次元！

アボガドロさん

単純に d 項関係を考えると，次元数が膨大になってしまう

多項式カーネル

$$K(\boldsymbol{a}, \boldsymbol{b}) = (\boldsymbol{a} \cdot \boldsymbol{b} + 1)^d = \phi_{n,d}(\boldsymbol{a}) \cdot \phi_{n,d}(\boldsymbol{b})$$

（本研究で用いた多項式カーネル）

対応する特徴空間 $\mathbb{R}^N$, $N = \binom{n+d}{d}$ の各次元が，

高々 d 次の単項式に対応する　※n は低次の特徴の次元数

低次の特徴の k 項関係（ $k \leq d$ ）で，単項式として表すことのできるものを全て特徴として含んでいる

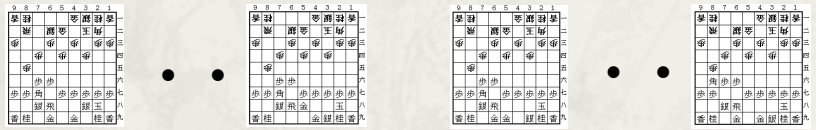
カーネルの計算量は次数 d を上げても
ほとんど変わらない（$\mathcal{O}(n)$ 時間で計算可能）

学習サンプルの作り方

ある局面 x

棋譜データ

合法手後の局面を作成



$P(x)$　　$N(x)$

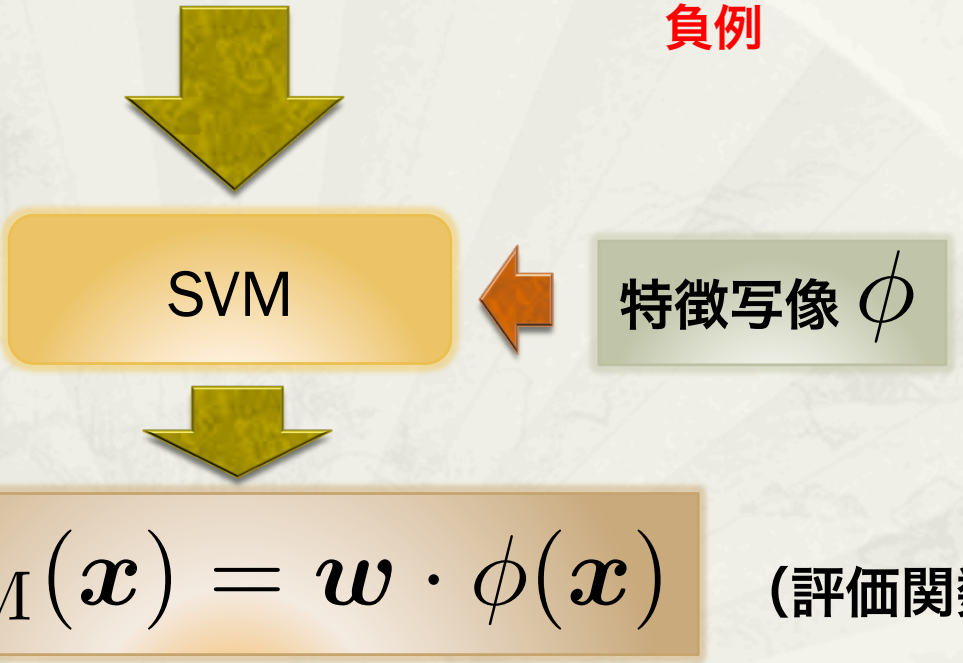
$P(x)$: x の次局面で棋譜データに現れるものの集合

$N(x)$: x の次局面で棋譜データに現れないものの集合

学習の流れ

$$S = \{(\boldsymbol{x}^+, 1) \mid \boldsymbol{x}^+ \in \bigcup_{\boldsymbol{x}} P(\boldsymbol{x})\} \cup \{(\boldsymbol{x}^-, -1) \mid \boldsymbol{x}^- \in \bigcup_{\boldsymbol{x}} N(\boldsymbol{x})\}$$

正例　　負例



$$f_{\mathrm{SVM}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x})$$

（評価関数）

棋譜に現れる局面を正例，現れないものを負例とする２値分類

問題点

局面 $\boldsymbol{x}$ は・・・

$f_{\mathrm{SVM}}(\boldsymbol{x}) > 0$ ⟶ プロ棋士が選びそうな局面

$f_{\mathrm{SVM}}(\boldsymbol{x}) < 0$ ⟶ そうでない局面

評価関数の値そのもの $f_{\mathrm{SVM}}(\boldsymbol{x})$ が局面 $\boldsymbol{x}$ の優劣を表しているわけではない

例：次局面の評価値が全て負の場合，評価値の高い局面が最善とは言えない

局面集合を順位付け（ランキング）する
評価関数を作成するべき！

正例と負例の２値ラベルからランキング学習ができる
ランキングSVM（Joachims ’02）を用いる

ランキング SVM と評価関数

ランキングSVM

$$S = \{(\langle \boldsymbol{x}^+, \boldsymbol{x}^- \rangle, 1) \mid (\boldsymbol{x}^+, \boldsymbol{x}^-) \in \bigcup_{\boldsymbol{x}} P(\boldsymbol{x}) \times N(\boldsymbol{x})\}$$

サンプルは正例と負例のペアでワンセット，全て正例

サンプルを正例と負例のペアとして与えることで，ランキング問題を２値分類問題に還元

SVM

$$\psi(\langle \boldsymbol{x}^+, \boldsymbol{x}^- \rangle) = \phi(\boldsymbol{x}^+) - \phi(\boldsymbol{x}^-)$$

ランキングから２値分類への還元を行うための写像

$f_{\mathrm{RSVM}}(\boldsymbol{x}^+) > f_{\mathrm{RSVM}}(\boldsymbol{x}^-)$

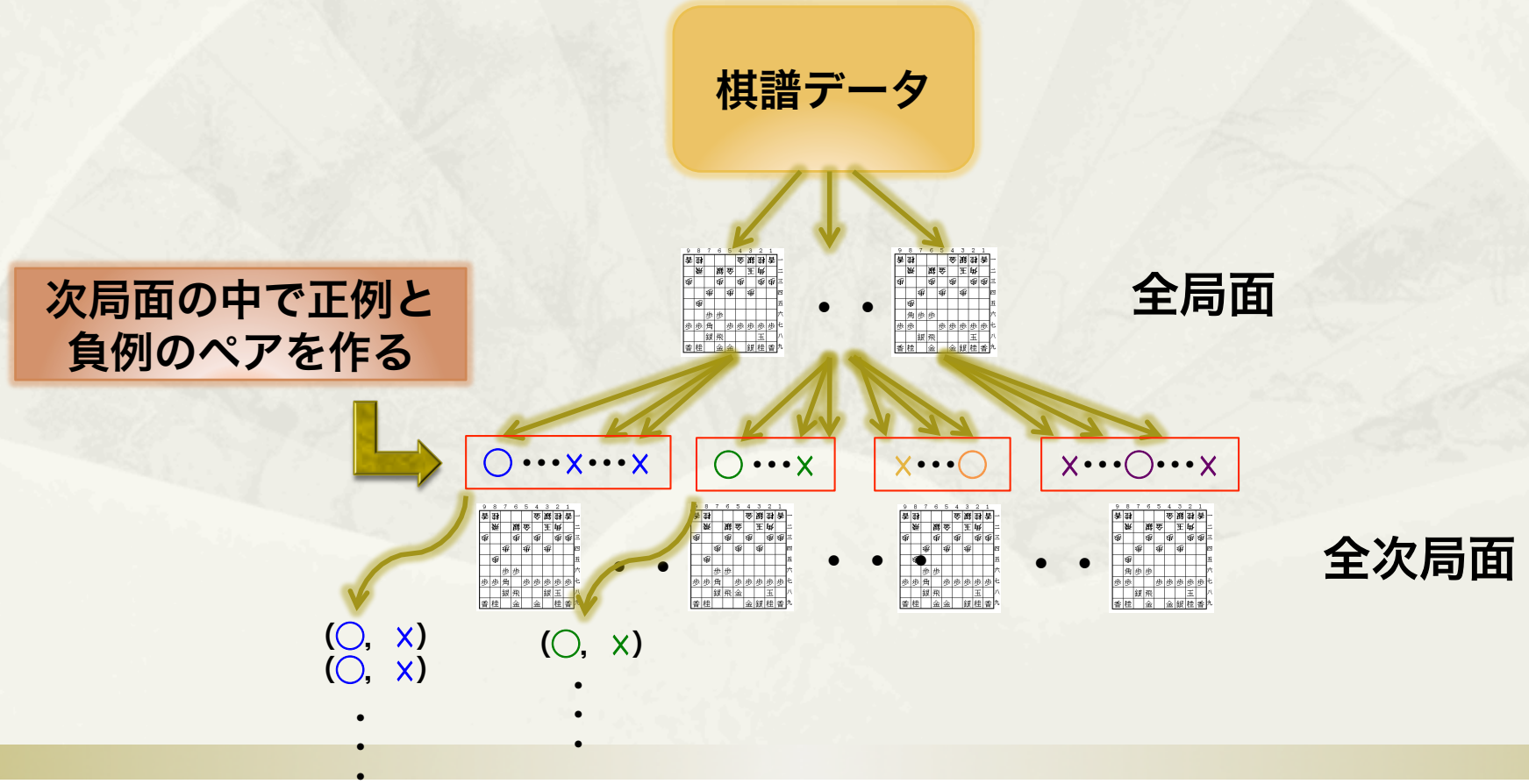
となるような学習が行われる

$$\hat{f}(\langle \boldsymbol{x}, \boldsymbol{x}' \rangle) = \boldsymbol{w} \cdot (\phi(\boldsymbol{x}) - \phi(\boldsymbol{x}'))$$

$$f_{\mathrm{RSVM}}(\boldsymbol{x}) = \boldsymbol{w} \cdot \phi(\boldsymbol{x})$$

ランキングと将棋

ランキングから２値分類に還元する際，一般には全ての正例と負例のペアからなるサンプルを作るため**サンプル数が２乗に増大**するが，将棋においては**次局面の集合内でのみペアを作成すればよいのでほぼ増大なし**



実装 & 実験

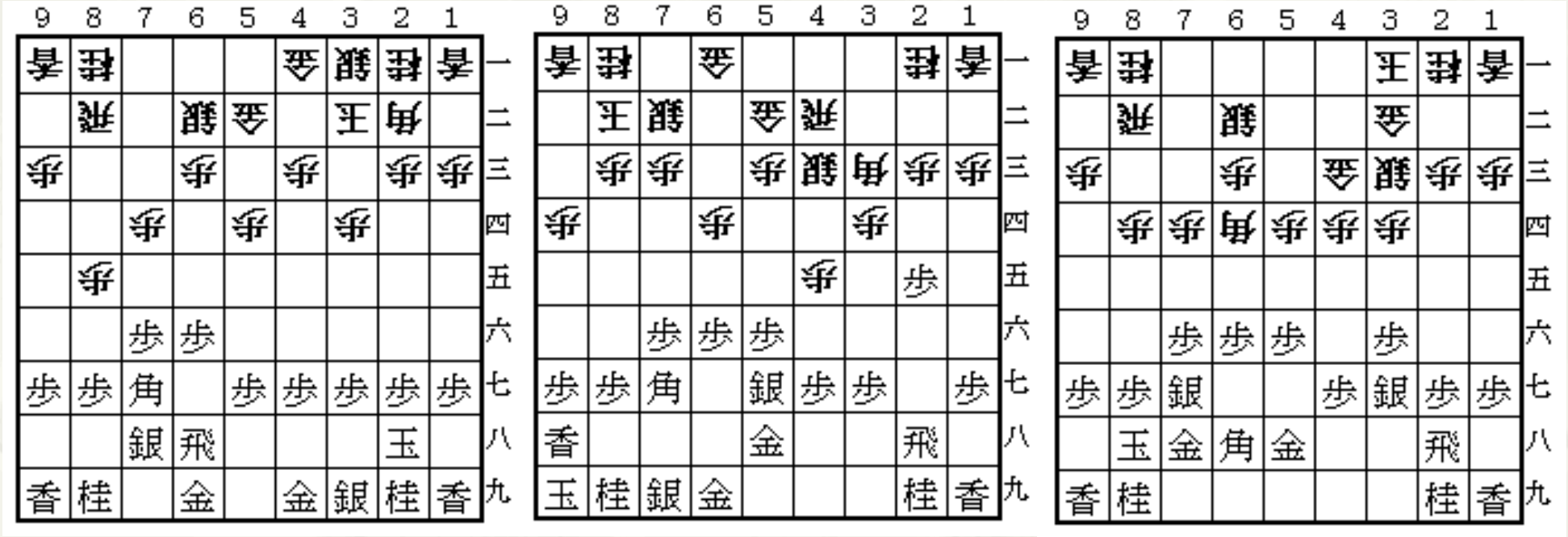
Pegasos (Shai-Shwartz ら '08)

* Primal Estimated sub-GrAdiend SOlver for SVM
* ランダムサンプリングと勾配法を用いたシンプルなアルゴリズム
* 逐次学習（大規模データでも大丈夫）
* 計算時間がサンプル数に依存しないので高速

今回実験で用いたデータ（棋譜５万局分）は正例負例合わせて４億例（750GB）

実験（駒組みの観察）

- ５万局の棋譜をサンプルとした $f_{\mathrm{SVM}}, f_{\mathrm{RSVM}}$
- 指定局面を３つ用意し, f_{SVM}, f_{RSVM} の上位３手を比較



多項式カーネルは１０次, λ は0.001, ϵ は0.01
逐次学習における更新回数は１０万回として学習を行った

実験結果

	評価順位	SVM	ランキングSVM
局面A	1	６五歩	８八角
	2	４六歩	４八銀
	3	３八飛	３八銀
局面B	1	３六歩	７八金
	2	８八玉	８八銀
	3	６八金左	７八銀
局面C	1	６七金	２六歩
	2	４八金	４六角
	3	５九金	４六銀

第一候補手にあまり差はないが，ランキング学習を用いたほうが，
まともな手を多く上位に含んでいる

実験（棋譜との一致率）

- 1000 局の棋譜をサンプル
- Pegasos のパラメータ λ を 0.01，0.05，0.1，ϵ を 0.001，0.01と変化させて f_{SVM}, f_{RSVM} を作成（全 16 個）
- 更新回数は $1/\lambda\epsilon$
- テストデータは 100 局分（10521 局面）
- それぞれの評価関数により選択された第 n 候補までの次局面の中に，棋譜における局面と同一の局面含まれていれば一致

実験結果

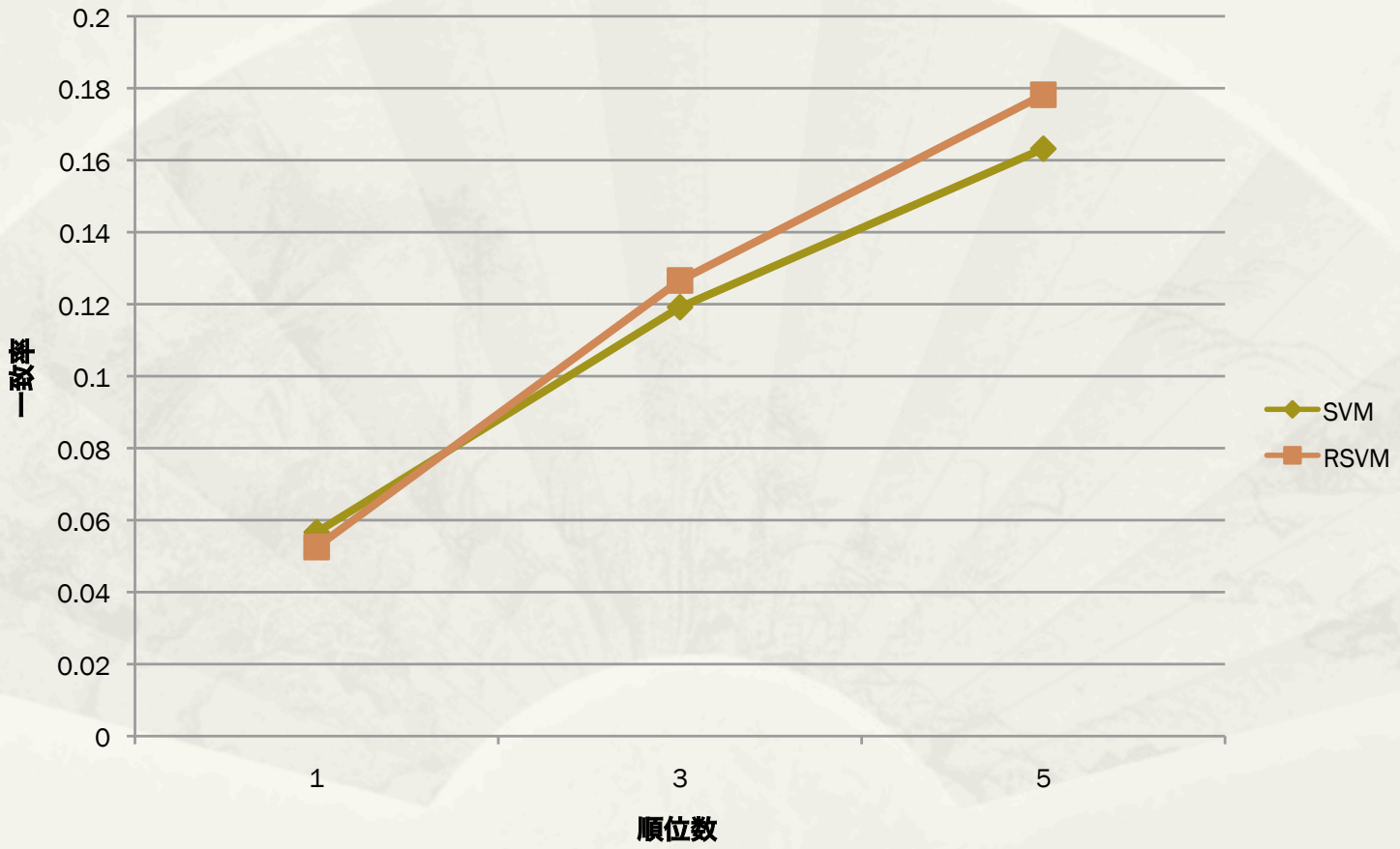
$\epsilon = 0.001$

λ	n	SVM	RSVM
0.01	1	0.05646	0.05826
	2	0.11910	0.14560
	3	0.16319	0.20141
0.05	1	0.05655	0.06321
	2	0.11833	0.13687
	3	0.16291	0.19428
0.1	1	0.05646	0.04857
	2	0.11891	0.11948
	3	0.16253	0.17185

$\epsilon = 0.01$

λ	n	SVM	RSVM
0.01	1	0.05589	0.05057
	2	0.11795	0.11643
	3	0.16206	0.17166
0.05	1	0.05646	0.03840
	2	0.12090	0.10959
	3	0.16415	0.15597
0.1	1	0.05807	0.05617
	2	0.11976	0.13117
	3	0.16434	0.17394

実験結果



まとめ & 今後の課題

まとめ

- 多項式カーネルを用いたことによる特徴の自動生成
- ランキング SVM を用いた評価関数の学習

今後の課題

- minimax 探索を組み込んだ学習
- 内積計算の高速化