競艇データベースに対するSVMによる解析

2005MT060 前川 猛 2006MI154 澤田 英明 2006MI179 竹内 秀樹
指導教員　河野 浩之

1　はじめに

近年, データマイニング技術が, 野球の配球のデータ解析や, 競馬の着順予測で活用されている.

しかし競艇のデータ解析ソフトである, Boat Advisor の手法・精度が有料ソフトであるが故に不明である [1]. そこで我々は Kyotei Official Web[2] にある過去 13 年分の過去データベースに対して, データマイニングを用いた着順予測を行う.

まずどのデータマイニング手法を用いるか, サポートベクターマシン (以下 SVM), SVR, SMO, 決定木, k 近傍法の速度・精度の比較検討を行い, その結果, 速度は遅いが精度が非常に高い SVM を採用する.

次に 1996 年から 2005 年までの過去 10 年分のデータを PostgreSQL に格納し, データマイニングツールで解析する. データマイニングツールとして, SVM-light, LibSVM, TinySVM, Weka から, 決定木や EM アルゴリズムと言ったアルゴリズムでも実験可能な Weka を採用する. Weka を用いた実験結果と 2006 年から 2009 年までの結果の着順一致率を比較する.

しかし, 我々の所持する PC ではメモリを最大に使用しても不足していたため, 今回はテスト実験として 2005 年のデータを格納し解析を行い, 2006 年のデータと比較して考察を行う.

2　パターン分析による未来予測の実例

先行研究として競艇と比較的ルールなどが似ている競馬の着順予測を紹介する.

競馬では, 決定木によりレースを「荒れないレース/荒れるレース」に分類して, より利益を出すためのシミュレーションが行われている. まず始めに, 荒れないレースというのは人気馬が絡んでいるレースのことで, いくつかの通説として, 1 番人気の馬の人気が高い時は荒れない, 少頭数のレースは荒れない, 良馬場のレースは荒れない, 牝馬のレースは荒れるといったものなどがある. これらの通説について過去のデータを用いてそれぞれ検証している. この検証により 1 番人気馬の人気が高い時は荒れないというのが, 最も重要な要因になっていたことが分かった. そしてこの検証を基に決定木により競馬の予測シミュレーションが行なわれている. シミュレーション結果として, 回収率 0.8 から 1.4 という安定した結果が得られていた [3].

3　競艇の過去データにおける分析の提案

3.1　分析までの流れ

本研究では使用するデータとして登番, 着, レースタイム, スタートタイミング, 展示, 艇番, モーター, ボートのデータを使用する.

これらのデータの収集のために Kyotei Official Web より過去のデータを利用する [2]. Kyotei Official Web の過去のデータを PostgreSQL に格納する. データを格納する際, Kyotei Official Web からダウンロードした状態のままだとデータ型が text 型になっている. このままではデータを扱いづらいので, csv 型に変換するために gawk を利用する. 格納してあるデータベースとデータマイニングフリーソフトである Weka とを結合させてデータの解析及び考察を行う.

解析したデータを元に着順予想の一致率を本研究のテーマとする. 本研究における分析の流れをフローチャートで表すと図 1 のようになる. 以下が図 1 の解説である.

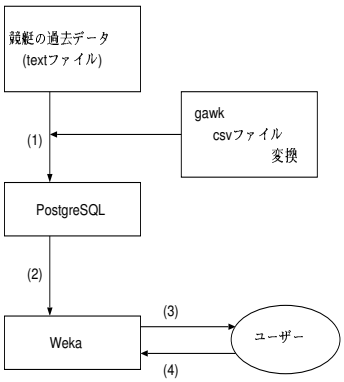

図 1　実装までのフローチャート

(1)gawk を用いて csv ファイルに変換した競艇の過去データを PostgreSQL に格納する.
(2)PostgreSQL のデータを Weka に学習させる.
(3) ユーザーが次に予測したいレースのデータを Weka に読み込ませる.
(4)Weka を用いて過去のデータから得られた SVM での解析結果をユーザーに表示する.

3.2　アルゴリズムの選択

代表的な学習有り未来予測には, サポートベクターマシン (以下 SVM), SVR, SMO, 決定木, k 近傍法などがある.

表 1　アルゴリズムの比較表

	SVM	SVR	SMO	決定木	k 近傍法
速度	×	△	△	◎	△
精度	◎	○	○	×	×
特徴	データが多ければ精度がかなり高い	大規模な問題を高速に解ける	部分問題を作成し, 逐次最適化を行う	規則を人間が見付けやすい	多数決で決めるため精度が悪い
短所	データが多いと最適化が困難	正則化パラメータを適切に決定しなければならない	サンプルが多い場合は時間がかかる	精度が悪い	更新の激しいデータには向かない

それぞれのアルゴリズムの特徴は表 1 にまとめた. 本研究では速度よりも精度を第一に考えているので, 検証した結果 SVM を利用することとした.

3.2.1　SVM の学習方法

SVM は線形入力素子を利用して 2 クラスのパターン識別器を構成するアルゴリズムである. 様々なデータは線形入力素子を通すことで平面上に点として表現される. 構成された点は平面上で図 2 のようになる. このような線形のデータに対してクラス分けを最も誤差の少ない方法により行いより良いデータを得ることができる．SVM は線形分離可能であるかどうかでカーネルトリックを使用するかを決定する [4][5].

識別器は, 識別関数 $f(x)$ の形で表される. 本研究におけるレースタイムなどの要素を識別関数によって識別した場合, x の f が正なら図 2 の四角の集合, f が負なら図 2 の丸の集合に識別される．$f(x) = 0$ を満たす点 x の集合は, 二つのクラスの境界面をなしているが, この面を識別面と呼ぶ. 関数にはパラメータの良さを評価する w を定義しこの値を変えることによって識別面の位置をコントロールできるようになる．次に, 学習法について触れる.

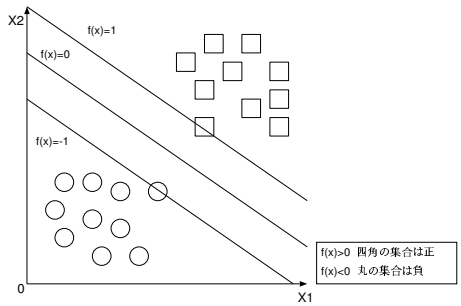

図 2　線形識別面のクラス分け

SVM は線形識別器の一つであり, テストサンプル式 (1)

$$x = (x_1, \cdot\cdot\cdot, x_d)^T \tag{1}$$

の識別関数を式 (2) に表す.

$$f(x) = \sum_{j=1}^{d} w_j x_j + b \tag{2}$$

ここで w_j は線形識別器の重みと呼ばれるパラメータでベクトル表示したもの w を重みベクトルと呼ぶ. また, b はバイアス項と呼ばれるパラメータである. この識別器の $f(x) = 0$ は超平面となる．SVM では, 訓練サンプルを完全に識別する超平面の中で最適解を探さなければならない. 最適解は超平面と訓練サンプルとの最小距離 (マージン) を評価関数として用いて, これを最大にするような超平面を選ぶ. これをマージン最大化という. マージン最大化とはクラス分けを行う際に, 最も誤差を少なくする方法のことである. 線形識別面でクラス分けを行う際に, どの位置が最適な解なのかがわからない. そのため, マージンと呼ばれる線形識別面からサンプル点への最近傍までの距離を最適な選択を図 3 のようにしなければならない. 最適な選択を行うためのには式 (2) の条件の元で式 (3) を満たせば良い．これを満たすことが出来ればマージンが最大となる.

$$2/|w| = 1/|w| + 1/|w| \tag{3}$$

非線形データに対して SVM は必ずしも良い性能の識別

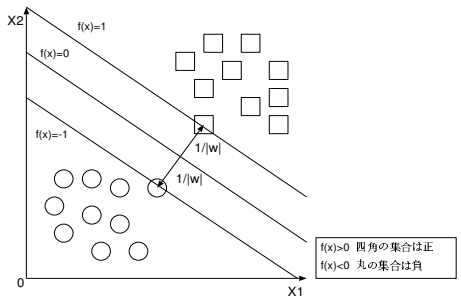

図 3　線形識別面のマージン最大化

器を構成できるとは限らないので, 本質的に非線形のデータに対応する方法として, 特徴ベクトルを非線形変換して, その空間で線形の識別を行うカーネルトリックがある. この方法を用いることで SVM は線形分離不可能な場合に対してより良い最適解を得ることが可能である.

3.3 ソフトウェアの比較

SVM を扱えるソフトウェアの代表的なものとして, SVM-light，LibSVM，TinySVM，Weka がある. それぞれのライセンスなどの特徴は表 2 にまとめてある通りである. なお精度は, SVM-light の公式サイトにあるサンプルファイルを元に実験を行った結果である. ここで, 精度も良く, フリーソフトでもあり，SVM 以外のアルゴリズムを扱うことの出来るのは Weka のみである. 本実験では, 他のアルゴリズムでも Weka を用いて実験を行い, それぞれの結果からの精度の差などで比較なども行うので Weka で実験を行う.

4 競艇データベースを SVM に基づく解析システムの実装

4.1 gawk を用いて対象データを変換

今回, 解析を行うにあたって必要なデータが 1996 年から現在に至るまで, ほぼ毎日 12 レースが Kyotei Official Web にテキストファイルとして掲載され続けている. しかし, このままでは PostgreSQL に格納出来ないので, gawk を使用して変換を行う. そのために以下のプログラムを作成し, テキストファイルから必要な部分だけを変換し抜き出し, csv ファイルへの変換が完了し, データベースへの格納が可能となった.

最適なパラメータの値

```
BEGIN {FS = " ";}
 if($1=="01"||$1=="02"||$1=="03"
  ||$1=="04"||$1=="05"||$1=="06"){
  if($10=="0.00.0"||$10=="."){
  printf("%s,%s,%s,%s,%s,%s,%s,%s,%s,""\n",
              $1,$2,$3,$4,$5,$6,$7,$8,$9 );}
  else{a = $10;
  split(a,b,".");
  printf("%s,%s,%s,%s,%s,%s,
          %s,%s,%s,%d.%d\n",
          $1,$2,$3,$4,$5,
          $6,$7,$8,$9,b[1]*60+b[2],b[3]);}
}
}
```

4.2 PostgreSQL に csv ファイルを格納

本研究では競艇の過去データ (2005 年) をデータベースに格納しなければならないため PostgreSQL を使用することとした. バージョンは日本語対応と文字セットサポートで SJIS のある PostgreSQL v8.4.1 を利用する [6].

図 4 が競艇に対する ER 図である. 競艇の要素に対するデータ型は全て, text 型で行った.

格納したデータ量は 10 年分のデータでは約 177MB,

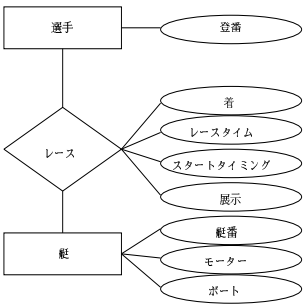

図 4　競艇に対する ER 図

1260584 行, 1 年分のデータでは約 19MB, 300529 行となった.

4.3 Weka と PostgreSQL の結合

Weka と PostgreSQL の結合は PostgreSQL の JDBC ドライバと構成ファイルの変更と, データベースの設定ファイルの変更, 環境変数の ClassPath の設定が必要となる.

JDBC ドライバは JAVA とデータベースの接続のために必要な API である．Weka は JAVA で動いているため JDBC ドライバが必要となってくる．PostgreSQL の JDBC ドライバは pgAdmin のスタックビルダを利用することでインストール可能である.

次に, Weka の構成ファイルの変更である．Weka の構成ファイルは RunWeka.ini であるがこれがデフォルトのままでは, 解析の際に大量のデータを読み込むために使用メモリが足りないため, 使用メモリを変更し大量のデータを読み込むことを可能にした.

5 競艇に対する SVM の性能評価

5.1 競艇の解析に対する評価方法

本研究では, データマイニングソフト Weka に実装されているアルゴリズムの 1 つである SVM での解析結果を元に評価を行う.

具体的には 2005 年のデータを Weka で学習させ, 2006 年の競艇の着順予測を行い, 予測した着順と実際の着順の一致率を比較する. 比較した結果から本研究でのシステムが着順の未来予測としての精度を何%出力するかを評価基準とする． また, 他のアルゴリズムとして, EM アルゴリズムでの精度と測定速度の比較も評価基準の対

表 2　ソフトウェアの比較表

	SVMlight	LibSVM	TinySVM	Weka
データベース	MySQL	MySQL	MySQL	PostgreSQL
ライセンス	thorsten-Joachims	GPL	GNU	GNU
SVM 以外のアルゴリズム	×	×	×	○
言語	C	C++,JAVA	C++	JAVA
Linux	○	○	○	○
Windows	○	○	○	○
精度 (%)	79.87	84.81	84.98	84.98

象とする. なお, 今回の実験をする PC 環境は, CPU が AMD Athlon 64 3500+, システムメモリは 2GB となっている.

5.2　SVM のパラメータの決定

パラメータをデフォルトの状態と, cost と gamma の設定を Grid Search という自動で最適なパラメータを設定するシステムを利用したものとを比較した結果, Grid Search を用いたものの方が 5〜8 %ほど精度が良かったので Grid Search の採用を決定した. cost と gamma は以下のようになった.

最適なパラメータの値

```
X property: classifier.cost
Y property: classifier.gamma
Coordinates: [20.0 , 5.0]
```

5.3　SVM での未来予測に対する評価

2005 年と, それを学習させた 2006 年の実験結果は以下のようになった.

2005 年の解析結果

```
Correctly Classified Instances
                    126827      42.920%
Incorrectly Classified Instances
                    168673      57.080%
Kappa Statistic                   0.42
Total Number of Instances         295500
```

2006 年の解析結果

```
Correctly Classified Instances
                    184854      51.9984%
Incorrectly Classified Instances
                    170646      48.0015%
Kappa Statistic                   0.51
Total Number of Instances         355500
```

2005 年のデータを学習させるために要した時間は 1676254s, 2006 年のデータの解析に対して要した時間は 1898897s とかなり長い時間を要した.

5.4　SVM と EM アルゴリズムとの比較

他のアルゴリズムとの比較として, EM アルゴリズムと SVM とを精度・速度の点から 1 ヶ月分のデータで比較を行った. EM アルゴリズムを取り入れて行った実験では 2006 年の結果との一致率が 20.0926 %で, SVM と比較して極めて低い数値となった. 要因として, 学習データの量や, パラメータの設定が不十分だったことが考えられる. 一方, 実験時間は 3929s で, これは SVM の 24638s に対し非常に速い結果となった.

6　まとめ

本研究の結果から SVM は精度が 51.9984 %となり, 実際に使えるモデルを得るまでには至らなかった. その要因として, データの量や, レースの開催場所などの要素の数が不十分だったことや, Grid Search で求めたパラメータよりも良いパラメータの設定の方法があったのではないか, などが考えられる.

参考文献

[1] “競艇ソフト Boat Advisor,” http://boat-advisor.com/(accessed 2009.8).

[2] “Kyotei Official Web,” http://www.kyotei.or.jp/(accessed 2009.8).

[3] 月本洋, “実践データマイニング,” オーム社, pp.27-187, 1999.

[4] 森泉豊栄, 中本高道, “鼻を使わず匂いを嗅ぐ,” LANDFALL, Vol.39, pp.20-22, 2000.

[5] 佐々木裕, 磯崎秀樹, 鈴木潤, 国領弘治, 平尾努, 賀沢秀人, 前田英作, “SVM を用いた学習型質問応答システム SAIQA-2,” 情報処理学会論文誌, Vol.45, No.2, pp.635-646, 2004.

[6] “NPO 法人 日本 PostgreSQL ユーザ会,” http://www.postgresql.jp/(accesed 2008.11).