プログラミング課題 第４回

Michigan型ファジィ遺伝的機械学習

大阪府立大学

計算知能工学研究室

1. **はじめに**

課題４もファジィ識別器の設計である．与えられたパターンに対してファジィルールを生成することで，パターン識別問題に対するファジィ識別器に関する知識を深めることを目的とする．

遺伝的機械学習（Genetics-based machine learning: GBML ）は機械学習に進化型最適化手法を適用した学習手法であり，新しい知識の獲得や，データマイニング行う方法として広く研究されている．本課題で使用するアルゴリズムは，ルール1つの個体として表現し，学習終了時の個体群1つの識別器として出力するMichigan型アルゴリズムにおいて条件部にファジィ集合を適用したMichigan 型ファジィGBML を用いる．

本課題では３つのデータを用いてファジィ識別器を生成する．その後，生成された識別器の識別率やルール数を比較し正しいプログラムができたか確認する．

1. **Michigan型ファジィGBML**
   1. **ファジィ識別器**

*n*属性*M*クラスのパターンを持つ*m*個の学習用パターン**x***p* = (*xp*1, ..., *xpn*) が与えられているパターン識別問題を考える．*xpi*は*p*番目の学習用パターンにおける*i*番目の属性値 (*i* = 1, 2, …, *n*) を表す．この問題に対して，ファジィ識別器を設計する際には，以下に示されるIf-then形式のファジィルールを生成する．

Rule *Rq*: If *xp*1 is *Aq*1 and ... and *xpn* is *Aqn*

then Class is *Cq* with *CFq* (1)

ここで，*Rq*は*q*番目のファジィルールを表すラベルであり，**A***q* = (*Aq*1, ..., *Aqn*) は条件部ファジィ集合，*Cq*は結論部クラス，*CFq*はルールの重みを表す．ファジィ集合には様々な粒度や形状を持つものが提案されているが，ここでは比較的計算が容易で，一般的に広く用いられている三角型のファジィ集合を利用する(図1)．



図1: 6種類のファジィ集合

今回の課題では，全ての属性に対して，全てのメンバシップ値を1とする “*don’t care*” と，2分割，3分割のファジィ分割を同時に使用する．すなわち，各ルールの各属性には合計6 (1+2+3) 種類のファジィ集合から1つが選択され条件部として与えられる．

条件部を決定したルール*Rq*の学習用パターンに対する適合の度度合いを測るためには，学習用パターンに対するルール*Rq*の適合度を算出する必要がある．三角型メンバシップ関数を持つルール*Rq*を用いてパターン**x***p* = (*xp*1, ..., *xpn*) の適合度を導出するには，まず，パターン**x***p*の全ての属性において次の式によりメンバシップ値を算出する．

(2)



(3)



(4)

ここで，*K*はファジィ集合*Aqi*におけるファジィ分割数を示しており，*k*はその分割数におけるファジィ集合の番号である．例えば，図1に示すファジィ集合A23は属性の分割数が3 (*K* = 3) で，2番目 (*k* = 2) のファジィ集合にあたる．

全ての属性におけるメンバシップ値を算出した後，メンバシップ値を乗算することでルール*Rq*の学習用パターンに対する適合度を導く．

(5)



学習用データから生成されたファジィルール*Rq*の評価尺度として，ルールの信頼度が用いられる．*μ***A***q*(**x***p*) を**x***p*に対する*Rq*の適合度とすると，信頼度*c*(**A***q*⇒Class *Cq*) は次の式で表される．

　　(6)



ファジィルールの結論となる結論部クラス*Cq*は学習用データに含まれるクラスの中で，ルール*Rq*の信頼度 *c*(**A***q*⇒Class *Cq*) が最も高くなるクラスを用いる．

(7)



また，生成されたファジィルールの識別における重要性を表す尺度として，ルール重み*CFq*が用いられる．ルール重みは信頼度の値を用いて次の式で与えられる．

(8)



ここで，ルール重み*CFq*は0より大きな値をとる必要がある．ルール重みの値が0以下になるファジィルール，つまり，ファジィルールの信頼度が0.5以下になるルールは，生成不可能なルールとする．

* 1. **ファジィ識別器による未知パターンの推論法**

生成されたファジィルールを用いて未知パターンのクラスを推論する方法を説明する．未知パターン**x**の推論は，各ルールの未知パターン**x**に対する適合度とルール重みの積を用いて行う．結論となるクラスを決定する際には，単一勝利ルールによる推論法を用いる．単一勝利ルールによる推論は，適合度とルール重みの積が最も高いルールを勝者ルール*Rw*とし，未知パターン**x**に対する推論結果を勝者ルール*Rw*の結論部*Cw*に決定する．ここで，異なる結論部を持つ複数のルールが等しく最大値をとり，勝者ルールを一意に決定することができない場合，未知パターン**x**を識別不能とする．

* 1. **個体表現**

Michigan型ファジィGBMLでは，1つのルールを1つの個体で表現する．属性数が*n*であるパターン識別問題を考える場合においては，1つのルールは*n*個の条件部集合によって表されるため，1つの個体は長さ*n*の整数列として扱うことができる．

式 (1) で示した様に，本稿で用いるファジィルールは結論部クラスとルール重みを持つIf-then形式のルールである．しかし，結論部クラスとルール重みは条件部ファジィ集合の組合せにより学習用データから自動的に算出される．よって，算出された結論部クラスとルール重みをファジィルール毎に保持する．一度算出された値をファジィルール毎に保存しておくことで，計算時間の短縮に繋げる．学習により前件部が変化（条件部ファジィ集合が別のファジィ集合に変化）した場合，ファジィルールの結論部クラスとルール重みの再計算を行う．ここで，新たに生成されたファジィルールの信頼度が0.5より小さくなる場合，ルール重みが0以下になり，意味を持たないルールとなる．この様なファジィルールはルール重みを0に設定し，結論部クラスを与えず，前件部だけのファジィルールとして保持する．

* 1. **初期個体群の生成**

初期個体群を生成する際には，個体群*N*pop個のファジィルールを生成する．ルールの各属性に対して5種類のファジィ集合をランダムに割り当てる．このとき生成されたファジィルールは全ての属性にファジィ集合が割り当てられており，パターン空間を細かく覆うルールになる．そこで，ファジィルールの覆う範囲を広げ，汎化性能を上げるために “*don*’*t care*” の適用を行う．“*don*’*t care*” の適用は“*don*’*t care*” 適用確率に従って行われ，選択された属性のファジィ集合を “*don*’*t care*” に変更する．全ての属性にファジィ集合を割り当て，“*don*’*t care*” 適用操作を実行した後，ファジィルールの結論部クラスとルール重みの計算を行う．

* 1. **個体の評価**

ルール集合で構成される識別器を用いて学習用パターンを識別する際には，単一勝利ルールによる推論法を用いる．そのため，学習用パターン1つにつき1つの勝利ルールが決定されることになる．ここで，複数のパターンが勝利ルールの候補として選択され，かつそれらルールの結論部クラスが異なる場合はそのパターンを識別不能として扱う．勝利ルールとして選択されたルールが，そのパターンを正しく識別した場合，ルールの評価値を1増加させる．したがって，ルールの評価値である適応度は正しく識別したパターン数となる．

また，識別器の中に，全く同じファジィルールが複数存在する場合があるが，その場合に適応度を得られるのはいずれか1つのファジィルールのみである．

* 1. **新規ルールの生成方法**

新規ルールの生成には本来，遺伝的生成法とヒューリスティック生成法の両方を用いるが本課題では簡略化のため遺伝的生成法のみを用いる．

遺伝的生成法は，まず復元抽出によるバイナリトーナメント選択を用いて，個体群内から親個体を2つ選択する．そして，選択された親個体に一様交叉を適用し，2つの子個体を生成する．2つの個体から一方をランダムに選択し，突然変異操作を適用する．突然変異操作では，突然変異確率に従い，条件部ファジィ集合をランダムに別のファジィ集合 (“*don’t care*” を含む) に変更する．ファジィルールの結論部とルール重みはルールの条件部によって自動的に決定されるため，突然変異によりファジィ集合が変更された場合，学習用データを用いて再計算を行う．

* 1. **新規ルールの生成と個体群の更新**

生成される子個体の数*N*repは本課題では，10個とする．これらを遺伝的生成法で生成する．

新規ルールの生成後，個体群の更新を行う．個体群*N*pop個のうち，適応度の小さいものから*N*rep個を新しく作られた子個体と無条件で置き換える．

* 1. **識別器の獲得**

本課題ではアルゴリズムの終了条件として世代数を用いる．また，最終的に獲得する識別器は全世代を通して学習用データに対する識別率が最も高かった個体群を選択する．つまり，最終的に獲得される識別器は，最終世代の個体群ではなく最も学習用データに対する識別性能の高い個体群となる．これは，本研究で用いているMichigan型ファジィGBMLにおいて，進化が，ルール集合である識別器には関係なく，個々のルールの評価に基づいて行われるため，最終世代の個体群が必ずしも最良の識別器になるとは限らないからである．

* 1. **Michigan型ファジィGBMLの流れ**

Michigan型ファジィGBMLのアルゴリズムを示す．

1. 初期個体群を生成し，初期評価を行う．
2. 遺伝的操作を行い子個体を生成する．
3. 選択操作により親個体のペアを選択する．
4. 交叉操作により子個体を生成する．
5. 生成された子個体に対して突然変異操作を適用する．
6. 現在の個体群の一部を子個体群と置換し，次世代個体群を作成する．
7. 終了判定を行い，終了条件を満たしていれば，Step. 5へ．満たしていなければStep. 2へ戻る．
8. 全世代の個体の中から学習用識別率が最も大きかった世代を最終的な識別器とする．またことのき不要ルールの削除操作を行う．

Step. 1では，初期個体を*N*pop個生成し，初期個体群を作る．ここで*N*popは現在の個体群，子個体群の個体群サイズを表す．初期個体群を生成した後，学習用データを識別し，各ルールの使用回数をもとに初期評価を行う．

Step. 2では，現在の個体群に対して遺伝的操作を適用し，子個体群を生成する．まず，バイナリトーナメント選択を用いて親個体群のペアを選択する．バイナリトーナメント選択は，現在の個体群の中からランダムに二つの個体を選択し，個体の適応度が大きい一方を親個体とする手法である．この操作を2回繰り返し，親個体のペアを得る．

次に，交叉確率に従い，2つの親個体を用いて一様交叉により子個体を生成する．一様交叉とは，親個体の各要素を独立に入れ替える交叉操作である．各要素に対して，二つの親個体間で対立要素を1/2の確率で入れ替えることにより，2種類の子個体を生成する．生成された二つの子個体のうち，ランダムに一方を選択し，子個体とするため，1回の交叉操作で得られる子個体の数は一つである．ここで，交叉確率により交叉を行わない場合は，親個体のうちランダムに一方を選択し，その複製を子個体として生成する．子個体が生成された後，生成された子個体に対して突然変異操作を適用する．

Step. 3では，現在の個体群と子個体群から次世代の個体群を作成する．本課題では生成した全ての子個体を現個体群のなかで評価値が小さいものから置換する．

Step. 4では，遺伝的アルゴリズムにおける終了判定を行う．終了条件を満たしている場合は，Step. 5において最終解の選択を行い，満たしていない場合は，Step. 2へ戻り，終了条件を満たすまでStep. 2-4の操作を繰り返し行う．

Step. 5における最終解の選択では，全ての個体群の中から最も学習用識別率が大きい世代を最終的な識別器として獲得する．その後，学習用データの識別で一度も使われなかったルールを削除した識別器を用いて評価用データを識別することで，獲得された識別器の汎化性能を評価する．

1. **数値実験**
   1. **数値実験設定**

本課題では，数値実験におけるベンチマーク問題として，KEEL-data set repositoryにより提供されているデータを用いる．今回は全ての属性値を[0, 1]で正規化したデータを用いる．

識別器生成の際には，識別器の汎化性能を評価するため，10-fold cross validation (10CV) を用いる．10CVでは，まず初めに，数値実験に用いるデータをランダムに10分割する．そして，10個に分割されたデータの部分集合のうち，9つを学習用データとして識別器の生成に用い，残りの1つを評価用データとして得られた識別器の評価に用いる．この操作を，学習に用いる9つの部分集合と，評価に用いる部分集合を順々に変更しながら10回繰り返し行い，最後に10回の平均値を算出することで，識別器の性能を得る．今回の課題では，10CVを5回繰り返し (合計50回試行) ，その平均値を算出する．乱数にはMersenne Twisterを使用し，各々の試行において乱数のシードは毎回変更するものとする．また，10CVにおけるデータの分割は，既に分割済みのデータ(a\*\_*name*-10tra.dat, a\*\_*name*-10tst.dat) を用いる．\*印は10CVにおける試行回数 (0 ~ 9) を表し，*name*はデータ名を表す．以下に今回の実験で用いる実験設定を示す．

表１　使用するデータ

|  |  |  |  |
| --- | --- | --- | --- |
| **データ名** | **パターン数** | **属性数** | **クラス数** |
| **Pima** | 768 | 8 | 2 |
| **Vehicle** | 846 | 18 | 4 |
| **Australian** | 690 | 14 | 2 |

試行回数: 10CV×5

個体群サイズ*N*pop: 30

ルール交換数 *N*rep : 10

終了条件: 10,000世代

交叉確率: 0.9

突然変異確率: 1 / *n* (*n*は属性数)

*“don’t care”* 適用確率: (*n*-5) / *n* (*n*は属性数)

* 1. **数値実験準備**

まず初めに2章で解説した各ルールに対するメンバシップ値，適合度，信頼度，結論部クラス，ルール重みが正しく計算されるかどうか確かめる．Pimaの１つ目の分割データ(a0\_0\_pima-10tra.dat)を使い，あるルールの結論部クラス，ルール重みを計算する．結果を合わせるために下記のルール1つについて計算する．

Rule: 0 4 0 3 3 0 0 0

結論部クラス：0，重み：0.504となれば正解である．

次に以下のルール集合を使ってPimaの１つ目の分割データ(a0\_0\_pima-10tra.dat)を識別し，誤識別率と各ルールの適応度（学習用データ識別時にそのルールが使われた回数）を計算する．この場合も結果を合わせるために下記のルール集合をプログラムに与え計算する．

Rule: 0 0 0 2 0 0 0 0, Fitness: 179

Rule: 0 4 0 0 3 0 0 4, Fitness: 136

Rule: 0 1 0 3 1 2 0 0, Fitness: 134

このとき誤識別率: 35.02，各ルールの適応度は上記のようになれば正解である．

* 1. **数値実験結果**

表２に結果を示す．アルゴリズムにランダム性が含まれるため，表２と自分の平均値が合うことはまず無い．特に評価用誤識別率は分散が大きいため参考程度で良い．よって3.4章においては学習用誤識別率を検定する．

表2　各データの数値実験50回の平均結果

|  |  |  |
| --- | --- | --- |
|  | **学習用**  **誤識別率(%)** | **評価用**  **誤識別率(%)** |
| **Pima** | 31.43 | 31.90 |
| **Vehicle** | 34.76 | 36.73 |
| **Australian** | 12.72 | 14.55 |

* 1. **検定**

作成したプログラムがあっているかどうかの検定にはウィルコクソンの符号順位検定（wilcoxon signed rank test）を用いる．今回はPython-Scipyのwilcoxon関数を用いて有意確率*p*値を計算し，その値が0.05を下回っていれば統計学的に優位な差が無いと判断する．

検定を行うために各試行の学習用識別率50個を一行ごとにテキストファイルに保存し，自分の出力したファイルと解答のファイルを検定のプログラムで読み込み検定する．

* 1. **参考**

学習用データ

a0\_0\_[dataname]-10tra.dat

a0\_1\_[dataname]-10tra.dat

…

a4\_9\_[dataname]-10tra.dat

評価用データ

a0\_0\_[dataname]-10tst.dat

a0\_1\_[dataname]-10tst.dat

…

a4\_9\_[dataname]-10tst.dat

各データにつき学習用，評価用で50個ずつのファイルを使用する．各データは１行目に（パターン数，属性数，クラス数）が示されており，２行目以降に各パターンの属性値，最終列にクラス番号が示されている．

乱数生成のメルセンヌ・ツイスタ（置き場所）

\\157.16.63.75\disk1\labowiki\mersennetwister