Beitrag zum Proseminar *Komplexe Adaptive Systeme*
PD Dr. Rüdiger Brause, J. W. Goethe-Universität Frankfurt / M., WS 2004/05

Discovering Fuzzy Classifiers by Genetic Algorithms

Christoph Knopp ⟨ckwon@gmx.de⟩

7. Januar 2005

Vorstellung einer auf *Maschinelles Lernen* basierten Methode zur Bestimmung von unscharfen Klassifikatoren mittels genetischen Algorithmen.

Inhaltsverzeichnis

1 Einführung

Heutzutage kann man einen Großteil der Probleme in Anwendungen auf Probleme bei der Einteillung und Kategorisierung von Elementen in Klassen, der sogenannten Klassifikation, zurückführen. Man sucht ein Verfahren, dass bekannte und unbekannte Muster mit hoher Genauigkeit erkennen und einer Klasse zuordnen kann. Neben den herkömmlichen statistischen Verfahren werden Techniken des *maschinellen Lernens* und des *Data-Mining* (Datengewinnung) zur Lösung dieser Probleme eingesetzt. Die so gefundenen Lösungen können durch *IF...THEN...*-Regeln dargestellt werden. In einem *evolutionären Prozeß* werden dann mit Hilfe von genetischen Algorithmen die besten Regeln ausgewählt und optimiert.

Genetische Algorithmen haben den Vorteil, dass sie eine globale Suche auf den gesamten Raum der Lösungen durchführen. Ein Großteil der beim Data-Mining angewendeten Algorithmen verwenden nur eine lokale *Greedy* Suche auf den vorliegenden Lösungen. Auf diese Weise können genetische Algorithmen Lösungen finden, die durch *Greedy* Suche unter Umständen ausgelassen werden.

Ein weiteres großes Problem der meisten Anwendungen besteht darin, dass die Trennung der einzelnen Klassen voneinander nicht eindeutig definiert ist – diese *Unschärfe* verursacht oft Fehler. In solchen Fällen kommt die *Fuzzy Logik* (Unscharfe Logik) zum Einsatz, bei der die Zugehörigkeit eines Elementes zu einer Klasse mit Werten im Intervall $[0, 1]$, dem sogenannten *Zugehörigkeitsgrad*, angegeben wird. Ein Klassifikationverfahren, dass durch Anwendung der Fuzzy Logik eine Einteillung der Klassen vornimmt, wird *Fuzzy Klassifikator* genannt.

Zur Bestimmung solcher *Fuzzy Klassifikatoren* setzen viele Anwendungen auf genetischen Algorithmen. Hier wird eine Variante vorgestellt, die durch verbesserte Methoden bei *Fuzzifikation*, Auswertung der Klassifikatoren und genetische Operationen eine höheren Erkennungsrate erreicht. Die Auswertung und ein Vergleich zu ähnlichen Verfahren erfolgt anhand einer Analyse von Standarddatensätzen.

2 Konzepte der Fuzzy Logik

Für die meisten komplexen Probleme formulieren wir Menschen Regeln, die wir, obwohl sie keine genaue Vorgehensweise vorschreiben, sehr gut verstehen können. Solche Regeln sind für einen Computer wenig hilfreich. Ein Beispiel für so eine Regel ist die Einstellung einer Heizung. Wir sagen „Wenn es ungefähr 19°C in der Wohnung ist, dann muss die Heizung ein wenig aufgedreht werden“, können diese Anweisung leicht verstehen und danach handeln. Der Computer kann mit den unscharfen Begriffen „ungefähr 19°C“ und „ein wenig aufdrehen“ nicht arbeiten.

2.1 Fuzzy Logik

Die Fuzzy Logik, also die unscharfe Logik, hat das Ziel, dem Computer den Umgang mit unscharfen Reglen zu ermöglichen. Man nutzt dafür die Theorie der unscharfen Mengen (fuzzy set theory), die 1965 von Lotfi A. Zadeh an der Universität von Kalifornien entwickelt wurde. Dabei kann ein Element, anders als bei der klassischen Mengenlehre, bis zu einem bestimmten Grad einer Menge angehören.

Klassische Mengen	**Fuzzy Mengen**
Ein Element ist völlig in einer Menge oder gar nicht.	Ein Element kann teilweise zu einer Menge gehören.
Der Zugehörigkeitsgrad nimmt nur die Werte 0 und 1 an.	Der Zugehörigkeitsgrad nimmt Werte zwischen 0 und 1 an.
1 bedeutet volle Zugehörigkeit zur Menge, 0 bedeutet gar nicht in der Menge. Andere Werte sind nicht zugelassen.	1 bedeutet volle Zugehörigkeit, 0 bedeutet gar nicht in der Menge. Andere Werte bedeuten teilweise Zugehörigkeit zu einer Menge

Tabelle 1: Unterschiede zwischen klassischen und Fuzzy Mengen

Mit der klassischen Mengenlehre würde man z.B. die Menge der angehmen Raumtemperaturen nur durch ein scharf begrenztes Intervall angeben können, etwa den Breich von $19°C$ bis $24°C$ (Abbildung 1a). Temperaturen, die etwas unterhalb dieses Intervalls $(18,9°C)$ liegen, würden damit nicht mehr zu den angenehmen Raumtemperaturen gehören, obwohl wir als Mensch diese durchaus noch als angenehm empfinden. Mit der unscharfen Mengenlehre könnte man die Temperatur $19°C$ mit einem Zugehörigkeitsgrad von $0{,}8$ zu der Menge der angenehmen Temperaturen gehören lassen. Die Temperaturen knapp unter $19°C$ würden dann noch mit Zugehörigkeitsgraden $< 0{,}8$ im Bereich der angenehmen Temperaturen liegen (Abbildung 1b).

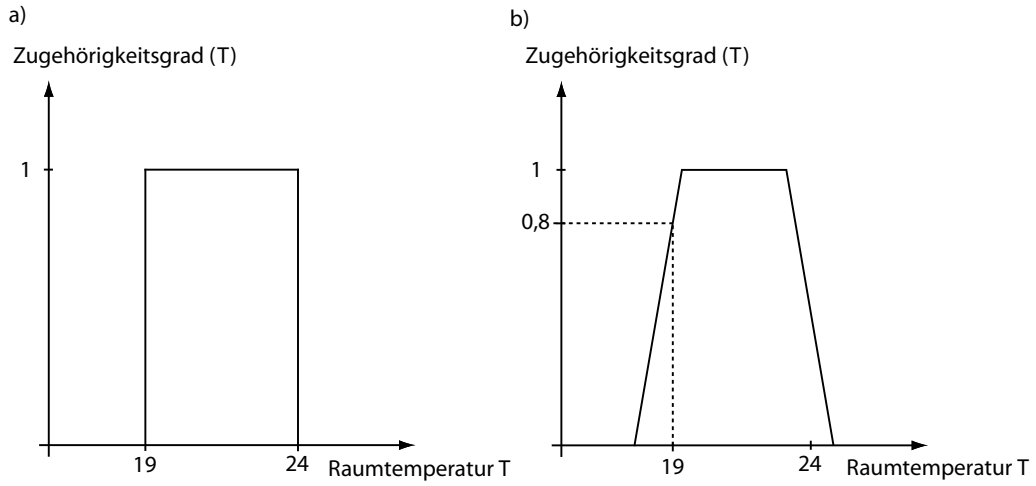

Abbildung 1: Angenehme Raumtemperaturen im scharfen a) und unscharfen Intervall b)

Mit Hilfe der Fuzzy Logik werden linguistische Begriffe wie „angenehm“, „warm“, „kalt“, „hoch“ und „tief“ durch unscharfe Mengen beschrieben. Die *Zugehörigkeitsfunktion* bestimmt den Grad der Zugehörigkeit eines Elementes zur Menge. Der Definitionsbereich ist das *Sprachuniversum* und der Zugehörigkeitsgrad liegt im Intervall $[0,1]$. Die Zugehörigkeitsfunktion ist meistens eine möglichst einfache Funktion. Oft sind dies Dreickeck-, Trapez- oder Gauss-Funktionen (Abbildung 2), da diese durch wenige Parameter beschrieben werden können und daher gut in Computern berechnet und gespeichert werden können.

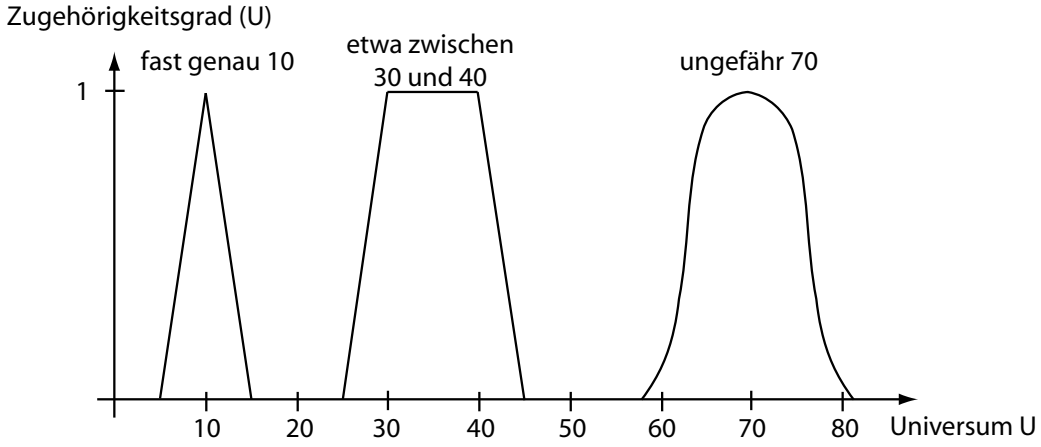

Abbildung 2: verschiedene Zugehörigkeitsfunktionen

2.2 Fuzzy Klassifikator

Ein auf der Fuzzy Mengen Theorie basiertes Klassifikationsverfahren nennt man *Fuzzy Klassifikator*. Dieser gibt für Elemente aus einer Datenmenge die Zugehörigkeit zu einer Klasse an. Einen Fuzzy Klassifikator kann man mit *Fuzzy Regeln* darstellen.

Fuzzy Regeln haben die Form:

IF Bedingung THEN Konsequenz [Gewicht]

Bedingung ist ein komplexer Fuzzy Ausdruck, der Fuzzy Logik Operatoren und atomare Fuzzy Ausdrücke benutzt. *Konsequenz* ist eine atomarer Ausdruck, *Gewicht* ist eine reelle Zahl, die die Vertrauenswürdigkeit der Regeln angibt.

2.3 Fuzzy C-Mean Clustering

Um in einer Datenmenge eine Struktur zu erkennen, verwendet man die Cluster Analyse. Dazu wird die Datenmenge in Cluster bzw. Klassen aufgeteilt. Ähnliche Daten werden dann jeweils zu einer Klasse zusammengefaßt. Da wir hier aber mit unscharfen Mengen arbeiten, müssen wir Fuzzy Clustering verwenden. Diese Clustering-Funktion teilt die Daten nicht mehr in scharfe Klassen ein, sondern berechnet den Zugehörigkeitsgrad zur jeweiligen Klasse.

Das Ergebnis des Fuzzy Clusterings wird in einer Zugehörigkeitsmatrix U angegeben. Dort sind die Zugehörigkeiten der Daten zu den jeweiligen Klassen enthalten. Diese Einteilung in die Klassen wird mit einer Bewertungsfunktion ermittelt, die wie folgt definiert ist:

$$J(U,V) = \sum_{k=1}^{n}\sum_{i=1}^{c}(u_{ik})^m||x_k - v_i||^2, \qquad u_{ik} \in [0,1], \qquad \sum_{i=1}^{c} u_{ik} = 1 \forall k \tag{1}$$

Diese Funktion verwendet zur Bestimmung der Zugehörigkeit u_{ik} den euklidischen Abstand zwischen dem k-ten Datum x_k zur i-ten Klasse v_i. Der Exponent m gibt die Gewichtung der Zugehörigkeit an und wird *Fuzzifikator* genannt. Mit $m = 1$ erreicht man eine harte Clusterung, d.h. dass

Einteilungen eine hohe Bewertung erhalten, bei denen die Daten mit maximaler Zugehörigkeit (also 1) der Klasse zugeordnet werden, zu der sie den geringsten Abstand haben. Wählt man $m \to \infty$ dann erhalten alle Daten die gleiche Zugehörigkeit zu jeder Klasse. In der Praxis hat sich für den Fuzzifikator ein Wert von $m = 2$ bewährt, weil dadurch viele Berechnungen vereinfacht werden können.

Eine gute Klaseneinteilung erhält man, indem man ein Minimum der Bewertungsfunktion bestimmt. Diese Minimierung übernimmt der *Fuzzy C-Mean Algorithmus*. Das „C“ im Namen bedeutet, dass man die Anzahl c der Klassen im Vorfeld festlegen muss. „mean“ deutet darauf hin, dass man den euklidischen Mittelwert (mean) zur Bestimmung des Abstands verwendet. Der Algorithmus nähert sich dann iterativ der Lösung an, bis die Abbruchbedingung ε erfüllt ist.

3 Entwicklung der Fuzzy Klassifikatoren

3.1 Genetische Algorithmen

Ein genetischer Algorithmus funktioniert ähnlich wie der natürliche Evolutionsprozeß. Aus einer Menge von *Individuen* (Population) entwickeln sich durch Vermehrung und Mutationen mit der Zeit neue Individuen. Jedes dieser Individuen beinhaltet Information zur Lösung des Problems, *Chromosomen* genannt. Individuen, deren Lösung ein bestimmtes Kriterium am ehesten entspricht, haben eine größere Chance den Evolutionsprozeß zu überleben (Survival of the Fittest). Die Chromosomen dieser Individuen werden sich bevorzugt auf die nächste Generation übertragen. Auf diese Weise entstehen sukzessive neue Individuen, deren Chromosomen immer bessere Lösungen codieren.

In einem genetischen Algorithmus wird jedes Individuum aus der zufällig generierten Grundpopulation mittels einer *Fitness-Funktion* bewertet. Diese Bewertung gibt die Effizienz der Lösung des jeweiligen Individuums an. Individuen mit einer hohen Bewertung werden mit größerer Wahrscheinlichkeit von einem der genetischen Operatoren *Selektion*, *Kreuzung* und *Mutation* ausgewählt. Die so entstandenen Individuen bilden die neue Generation, die dann in den nächsten Evolutionvorgang übergeht. Dieser Evolutionsprozeß wiederholt sich bis eine bestimmte Abbruchbedingung erfüllt wird, wie z.B. „keine Änderung der Fitness der Individuen in den letzten Generationen“ oder „eine bestimmte Anzahl von Generationen wurde durchlaufen“.

Möchte man einen genetischen Algorithmus zur Bestimmung eines Fuzzy Klassifikator verwenden, muss man zunächst eine Methode zur Codierung der Fuzzy Regeln in den Chromosomen bestimmen. Außerdem müssen die genetischen Operatoren und die Fitness-Funktion definiert werden.

3.2 Codierung der Regeln

Grundsätzlich gibt es zwei unterschiedliche Vorgehensweisen wie man Regeln in den Chromosomen der Indivuduen codieren kann. Beim Michigan-Ansatz codiert jedes Individuum eine Regel, der Pittsburgh-Ansatz hingegen codiert eine Menge von Regeln in dem Indivduum.

Den *Bedingung*-Teil der Regeln kann man auf verschiedenster Weise codieren. Beispiele hierfür sind Konjunktion/Disjunktion von einfachen Termen, festgelegte Kontroll-Strukturen, lineare Bäume mit Vorrang oder komplette Ausdrucks-Bäume.

Für das vorgestellten Verfahren wird eine Darstellung mit Hilfe der disjunktiven Normalform (DNF) verwendet. Der *IF*-Teil wird aus Konjunktionen (Verundung) einfacher Terme gebildet. Durch Disjunktion (Veroderung) dieser Teile entsteht eine zusammengesetze Regeln:

DNF − k:

$$(A_1 \wedge A_2 \wedge \ldots \wedge A_k) \vee (A_{k+1} \wedge A_{k+2} \wedge \ldots \wedge A_{2k}) \vee \ldots \vee (A_p \wedge A_{p+1} \wedge \ldots \wedge A_{p+(k-1)}) \quad (2)$$

Die Verwendung der DNF zur Darstellung der Fuzzy Regeln hat den Vorteil, dass man leicht eine geeignete Fitness-Funktion definieren kann. Zudem fällt dadurch auch die Störung in den genetischen Operatoren gering aus.

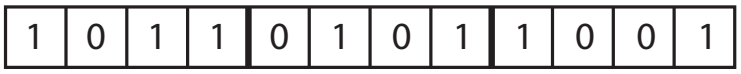

Abbildung 3: Beispiel für ein Chromosom

Die Darstellung des *THEN*-Teils in der Regel eines Chromosoms hängt stark von der verwendeten Evolutionmethode ab und wird daher später behandelt.

3.3 Genetische Operatoren

Die genetischen Operatoren sind wichtig für die Weiterentwicklung der Population. Das Ziel ist möglichst unterschiedliche Cromosomen zu produzieren um ein weites Spektrum an Individuen und damit auch viele Lösungen zu erhalten. Andererseits versucht man aber auch schon bestehende Individuen mit guter Fittness weiter zu optimieren, indem man sie untereinander kombiniert.

- Selektion:

  Die Selektion filtert aus der Population die Individuen heraus, die sich kreuzen dürfen. Dabei werden Individuen mit guten Chromosomen bevorzugt ausgewählt, da diese von der Fitness-Funktion eine hohe Wahrscheinlichkeit zugeordnet bekommen haben. Individuen mit schlechten Chromosomen sollen, wenn auch nur mit einer geringen Wahrscheinlichkeit, ausgewählt werden können, damit der Chromosomenpool möglichst groß bleibt.

  Die Individuen werden hier durch *Wettkampfselektion* ausgewählt, bei der jeweils n (*Wettkampfgröße*) Individuen aus der Population verglichen werden. Das Individuum mit der besten Fitness wird dann einer zweiten Population hinzugefügt, bleibt aber auch noch in der ursprünglichen Population erhalten. Dieser Vorgang wird wiederholt, bis die zweite Population die gleiche Größe wie die Ausgangspopulation besitzt.

- Kreuzung:

  Die Kreuzung simuliert die Fortpflanzung zweier Individuen. Dabei wird versucht durch Mischung beider Chromosomen neue Individuen mit optimierten Lösungen zu erhalten. Wie sich die Chromosomen des neuen Individuums zusammensetzen wird durch die *Single Point Crossover* Methode bestimmt. Dazu wird zufällig ein Punkt in den Chromosomen bestimmt. Der Nachkomme erhält nun von dem einen Individuum den Chromosomenteil bis zu diesem Kreuzungspunkt und den restlichen Chromosomenteil vom anderen Individuum.

- Mutation:

  Durch Mutation werden zufällige Änderungen an den Chromsomen vorgenommen, etwa durch Hinzufügen, Ersetzen oder Löschen zufälliger Teile. Dabei entstehen Lösungen, die alleine durch Kobination zweier bestehender Lösungen nicht entstanden wären. Dennoch sollte die Mutation nur mit einer geringen Wahrscheinlichkeit auftreten, da sonst gute Lösungen nicht mehr weitervererbt werden.

3.4 Fitness Funktion

Die Fitness-Funktion ordnet jedem Individuum einen Zahlenwert zu. Je großer dieser Wert ist, desto effizienter ist die Lösung für das gegebene Problem.

Die Vorhersagegenauigkeit einer Regel kann in einer 2×2 Matrix, der sogenannten Konfusionsmatrix, dargestellt werden. Die Elemente dieser Matrix sind *TRUE positiv (TP)*, *FALSE positiv (FP)*, *TRUE negativ (TN)* und *FALSE negativ (FN)*. Sie geben an mit welcher Wahrscheinlichkeit eine Regel eine korrekte Zuordnung durchführt. Dabei bedeutet *TP*, dass die Auswertung der Regel zu *TRUE* zutrifft. *FN* heißt, dass der Wert *FALSE* der Regel nicht zutreffend ist.

		Aktuelle Klasse	
		C	Nicht C
Vorhergesagte Klasse	C	TP	FP
	Nicht C	FN	TN

Tabelle 2: Konfusionsmatrix

Für die hier vorgestellte Methode werden die Elemente der Konfusionsmatrix mit den folgenden Gleichungen berechnet:

$$\mathrm{TP} = \sum_{i=1}^{p} \mathrm{predicted}(\mathrm{class_1 data_i})^{\frac{1}{2}} \qquad \mathrm{TN} = \sum_{i=1}^{q} (1 - \mathrm{predicted}(\mathrm{class_2 data_i}))^{\frac{1}{2}} \tag{3}$$

$$\mathrm{FP} = \sum_{i=1}^{q} \mathrm{predicted}(\mathrm{class_2 data_i})^{\frac{1}{2}} \qquad \mathrm{FN} = \sum_{i=1}^{p} (1 - \mathrm{predicted}(\mathrm{class_1 data_i}))^{\frac{1}{2}} \tag{4}$$

Diese Verhältnisse sind für Klassifikatoren von class_1 in einer binären Klassifikation. $\mathrm{predicted}$ ist der Fuzzy-Wert des Bedingungsteils der codierten Regel. $\mathrm{predicted}(\mathrm{class_j data_i})$ gibt den Vorhersehbarkeitsgrad des i-ten Datums zur der j-ten Klasse mit diesem Klassifikator an. p und q beschreibt die Anzahl der Proben aus dem Trainings-Datensatz.

Zur Berechnung der Elemente der Konfusionsmatrix wird die Summe aus den Wurzeln der Vorhersagen anstatt nur die Summe der Vorhersagen verwendet, damit die Regeln eine generelle Vorhersage auf alle ihre Proben haben. Die so gebildeten Regeln können bei allen Mustern mit mittlerer Genauigkeit vorhersagen (ca. 0,5) und sind damit besser als spezialisierte Regeln, die nur bei manchen Proben mit sehr guter (ca. 1) und allen anderen nur mit sehr schlechter (ca. 0) Genauigkeit vorhersagen können.

Mit der Konfusionsmatrix kann man nun den Konfidenzkoeffizienten (CF) und den Vollständigkeitsfaktor (Comp) brechnen, die wie folgt definiert sind:

$$\mathrm{CF} = \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FP})} \tag{5}$$

$$\mathrm{Comp}_1 = \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FN})} \tag{6}$$

$$\mathrm{Comp}_2 = \frac{\mathrm{TN}}{(\mathrm{TN}+\mathrm{FP})} \tag{7}$$

Für den Vollständigkeitsfaktor betrachtet man das Verhältnis der sich überdeckenden verwandten Mustern (Comp_1) und das Verhältnis der nicht überdenkenden nicht verwandeten Mustern (Comp_1).

Aus der Konfusionsmatrix kann man verschiedene Fitness-Funktionen definieren, so z.B. eine Kostenfunktion aus zwei Vollständigkeitsfaktoren oder das Produkt aus Konfidenzkoeffizient und Vollständigkeitsfaktor. Hier wird die Fitness-Funktion wie folgt berechnet:

$$\mathrm{Fitness} = \mathrm{Comp}_1 \cdot \mathrm{Comp}_2 = \frac{\mathrm{TP}}{(\mathrm{TP}+\mathrm{FN})} \cdot \frac{\mathrm{TN}}{(\mathrm{TN}+\mathrm{FP})} \tag{8}$$

Da man an Regeln interessiert ist, die sich mit dem Muster einer Klasse aber nicht mit dem Muster anderer Klasse decken, muss hier die *T-Norm* des Vollständigkeitsfaktors verwendet werden. Mit Hilfe der T-Norm kann man gleichzeitig beide Vollständigkeitsfaktoren optimieren und senkt durch diese Näherung das Risiko gegen ein lokales Optimum (etwa leere oder allgemeine Regeln) zu konvergieren.

Um die Verständlichkeit der Regeln zu verbessern fügt man der Fitness-Funktion noch einen weiteren Faktor hinzu – die *Einfachheit*. Die Darstellung der Regeln wurde hier so gewählt, dass man den Einfachheitsfaktor durch die Anzahl der einfachen Regeln (Rc) im Individuum und die gesamte Anzahl der Bedingungen (Cc) in der Regel wie folgt berechnen kann:

$$\mathrm{Simp} = \frac{1}{\mathrm{Cc}} + \frac{\mathrm{Rc}}{\mathrm{Cc}} \tag{9}$$

Die endgültige Fitness-Funktion sieht dann wie folgt aus:

$$\mathrm{Fitness} = w_1 \cdot (\mathrm{Comp}_1 \cdot \mathrm{Comp}_2) + w_2 \cdot \mathrm{Simp} \tag{10}$$

Die Varibalen w_1 und w_2 geben an in welchem Verhältnis die Vollständigkeitsfaktoren und die Einfachheit in die Fitness-Funktion eingehen sollen.

3.5 Evolutionprozeß

Der Evolutionsprozeß hängt stark von der verwendeten Dartstellung des *Then*-Teils der Regeln, also der vorhergesagten Klasse, ab. Zum einem bietet sich die Möglichkeit an, diesen Teil mit in die Chromosomen des Indivduums zu codieren und es so von der Evolution abhängig zu machen. Man könnte auch alle Individuen einer Population mit der gleichen vorhergesagten Klasse zusammenfassen, müßte dann aber für jede Klasse den Evolutionsprozeß neu starten. Sobald eine Anwendung mehrere Klassen verwendet führt diese Darstellung zu schlechten Laufzeiten. Für solche Anwendung kommt die Methode in Frage, die in einem Evolutionsprozeß eine Regeln der zutreffenden Klasse

zuordnet. Für die Auswertung der Testdatensätze kann hier die zweite Methode verwendet werden, da die Anzahl der Klassen gering ist.

Zur Erkennung eines Musters wurde hier der Konfidenzkoeffizient der Regel verwendet. Dazu wird dieser Faktor mit der Vorhersage der Regel multipliziert. Gemäß dem Ergebnis kann dann die Klasse des Musters festgelegt werden. Der Erkennungsgrenzwert zur Erkennung eines Muster durch einen einzigen Klassifikator wird auf $0{,}5$ gesetzt. Werden alle Klassifikatoren zusammen verwendet, legt die am ehesten zutreffende Regel die Klasse des Musters fest. Beide Methoden können auch in der binären Klassifikation angewendet werden.

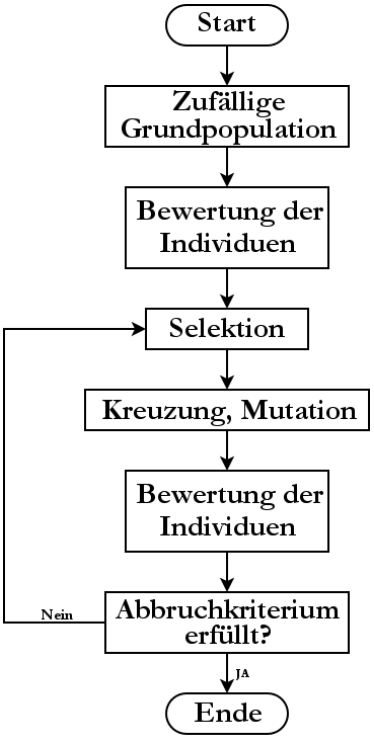

Abbildung 4: Evolutionsprozeß

4 Ergebnisse

Um die Performance des vorgestellten Verfahrens (System1) zu ermitteln, werden die Datensätze *IRIS*, *WINE* und *VOTE* verwendet. Dabei handelt es sich um Standarddaten, die oft zur Analyse von Methoden des maschinen-basierten Lernens verwendet werden. Bei dem *WINE*-Datensatz handelt es sich beispielsweise um die Daten einer chemischen Analyse von Weinen zur Bestimmung ihrer Herkunft. Dieser Datensatz besteht aus 178 Instanzen, 13 kontinuierlichen Attributen und 3 Klassen. Die Eigenschaften der anderen Datensätze sind in Tabelle 3 beschrieben.

Datensätze	Datensatz Eigenschaften			Testparameter	
	Größe	**Anzahl Klassen**	**Anzahl Attribute**	**Mutationsrate**	**Wettkampfgröße**
WINE	178	3	14	0,9	4
IRIS	150	3	4	0,5	2
VOTE	435	2	16	0,65	4

Tabelle 3: Eigenschaften der Datensätze und Testparameter

Zunächst werden die numerischen Attribute normalisiert, so dass sie zwischen 0 und 1 liegen. Für das C-Mean Clustering wird eine 3-Dreieckfunktion als Zugehörigkeitsfunktion für die kontinuierlichen Atrribute definiert. Indem man Fuzzy-Mengen ohne Überschneidungen bildet, kann man diese wie scharfe Mengen für nicht-numerischen Attribute mit katgegorischen Werten benutzen.

Zum Vergleich werden zwei weitere evolutionäre Systeme ausgewertet. System2 verwendet komplette Ausdrucksbäume zur Bestimmung der Fuzzy Klassifikatoren, System3 setzt auf Co-Evolution, bei der sich zwei unterschiedliche Populationen getrennt voneinander entwickeln.

Die Parameter für den Testlauf werden auf eine Populationsgröße von 200 Individuen und 200 Iterationen festgelegt. Die Wettkampfgröße für die Selektion und die Mutationsrate werden in der Tabelle 3 angegeben. Die Fitness-Funktion wird mit den Parametern $w_1 = 0{,}999$ und $w_2 = 0.001$ ausgewertet.

Nach 10 Iterationen im Evolutionsprozeß werden einfache doppelte Regeln in einem Klassifikator entfernt. Um bei diesem Schritt die Laufzeitkosten gering zu halten, kann man diesen Prozeß von der *Elitism* Strategie durchführen lassen. Dazu bildet man die neuen Generationen jeweils aus ein paar der besten Individuen der Population und füllt dann mit den Nachkommen auf. Auf diese Weise können gute Lösungen nicht verloren gehen.

Jeder Datensatz wurde zufällig in 10 Gruppen aufgeteilt. Jede Gruppe wurde dann mit dem aus den restlichen 9 Gruppen ermittelten Klassifikator getestet. Dieser Vorgang wurde jeweils fünfmal wiederholt. Die vorliegenden Ergebnisse entsprechen dem Druchschnitt aus drei Testdurchläufen.

System	***IRIS***	***WINE***	***VOTE***
System1	**97,11**	**94,06**	**95,33**
System2	94,84	92,22	95,42
System2 : DNF	93,3	90,55	95,43
System3	95,3	–	–

Tabelle 4: Erkennungsraten

Das vorgestellte System (System1) liefert die besten Ergebnisse mit den IRIS und den WINE Datensätzen. Da diese Datensätze kontinuierliche Attribute besitzen, profitieren sie von den Fuzzy Regeln und dadurch, dass man die Summe der Wurzeln in der Fitness-Funktion berechnet. Der VOTE Datensatz hingegen kann diese Vorteile nicht nutzen, da er nur aus diskreten Attributen besteht.

5 Fazit

Die vorgestellte Methode zur Bestimmung von Fuzzy Klassifikatoren mit genetischen Algorithmen liefert gute Ergebnisse und übertrifft damit sogar vergleichbare Systeme. Ein guter Ansatzpunkt für Verbesserungen besteht in der Fitness-Funktion, da es sich nicht einfach gestaltet, eine geeignete Funktion zu finden, die eine gute Näherung für die Vorhersehbarkeit einer Regeln berechnet. Außerdem könnte man die Berechnung der Elemente der Konfusionsmatrix durch Verwendung von *Aggregation* Funktionen verbessern.

In diesem Verfahren wurden die Klassifikatoren nur in einem einzelnen Evolutionsprozeß ermittelt. Das Ziel wird nun sein, dass man ähnliche Fortschritte auch bei der binären Klassifikation durch Verwendung eines Co-Evolutionsprozeß erreicht. Damit könnte man auch widersprüchliche Klassifikatoren im Falle von mehr als zwei Klassen vermeiden.

Literatur

[1] Hasanzade, M., Begheri, S., Lucas, C.: *Discovering Fuzzy Classifiers by Genetic Algorithms,* Proceedings of 4th International ICSC Symposium on Engineering of Intelligent Systems (EIS2004), Portugal, März 2004.