Knowledge Engineering in Spielen

Klassische Artikel

Kamill Panitzek und Immanuel Schweizer

Inhaltsverzeichnis

Einleitung

- Entscheidungen im Umfeld von imperfekter Information und Ungewissheit
  - Draw Poker
- Statische Strategien
  - Feste Entscheidungsmuster
  - Zufälliges abweichen davon
- Lernfähige Strategien
  - Anpassung an dynamisches Umfeld
  - Beeinflußung des Umfelds

Draw Poker

- Deal
  - Ante einzahlen
  - Hand mit 5 Karten
- (Predraw) Betting
- Replace (Draw)
  - Ersetze 0 – 3 Karten
- (Postdraw) Betting
- Showdown
  - Verbleibende Spieler legen Karten auf den Tisch

Inhaltsverzeichnis

- Einleitung
- **Generalization Learning Techniques for Automating the Learning of Heuristics**
- Studies in Machine Cognition Using the Game of Poker / Computer Poker
- Flexible Learning of Problem Solving Heuristics through Adaptive Search

Generalization Learning Techniques for Automating the Learning of Heuristics

- Heuristische Lösungsansätze für einen Computergestützten Spieler
- Dieser soll selbstlernend sein
- Verschiedene Lerntechniken werden angesprochen
- Basiert auf den Regeln von Draw Poker
  - Axiome für Draw Poker im Anhang vorhanden
- Ansätze sind aber allgemein anwendbar

Ziele

- Programm muss in der Lage sein
  - Eigene Heuristiken zu überwachen,
  - Ihre Qualität zu messen,
  - Heuristiken gegebenenfalls zu ändern und
  - Neue Heuristiken hinzuzufügen

Maschinelles Lernen

- Wird in zwei Teilprobleme untergliedert

1. Repräsentation der Heuristik
2. Techniken zur Erzeugung, Evaluierung und Modifikation von Heuristiken

Programmrepräsentation

- Programm aufgeteilt in Berechnungsblöcke
- State Vector ξ mit sich den Variablen A, B, C
- Gleichung für einen Block: $\xi' = f(\xi)$

$\xi=(a_1, b_1, c_2) \rightarrow f(\xi)=f(A, B, C) \rightarrow (a_1', b_1', c_1')= \xi'$

Berechnungsblock

- Wird zu einer Production Rule:

$$(A_1, B_1, C_1) \rightarrow (f_1(\xi), f_2(\xi), f_3(\xi))$$

- Mit $A_1 = \{a_1\}$, $B_1 = \{b_1\}$, $C_1 = \{c_1\}$

Repräsentation von Heuristiken

- Production Rules werden unterteilt in

- Heuristic Rules
  - Action Rule: $(A_1, B_1, C_1) \rightarrow (f_1(\xi), f_2(\xi), f_3(\xi))$
- Heuristic Definitions
  - Backward Form Rule: $A_1 \rightarrow A, A > 20$
  - Forward Form Rule: $X \rightarrow K_1 * D$

Entscheidungsfindung mit Production Rules

- Schritt1:
  - Vergleich des State Vectors mit rechten Seiten aller bf rules
  - Linke Seiten zutreffender Regeln werden mit rechten Seite aller bf rules verglichen bis keine Übereinstimmung mehr auffindbar
  - Resultierende Menge ist symbolischer Subvektor
- Schritt 2:
  - Vergleich des Symbolischen Subvektors aus Schritt1 mit linken Seiten aller action rules
  - Bei Übereinstimmung werden Variablen entsprechend der rechten Seite der action rule verändert

Programmablauf

- Bettingphase ist kritisch im Poker
- Kriterien für Entscheidung:
  - Eigene Hand
  - Inhalt des Potts
  - Letzte Erhöhung
  - Wahrscheinlichkeit dass Gegner blufft
  - Anzahl Karten die der Gegner getauscht hat
  - Spielstil des Gegners
- Daraus entsteht folgender State Vector:

ξ=(VHAND, POT, LASTBET, BLUFFO, ORP, OSTYLE)

Trainingsmethoden

- Implizites Training
  - Heuristiken werden durch Deduktion von Axiomen und bereits bestehenden Heuristiken vom Programm selbstständig erlernt
- Explizites Training
  - Heuristiken werden von einem Menschen oder einem Programm überprüft und bewertet.

Programmvarianten

- P[built-in]: fest programmierte Heuristiken
- P[manual]: Heuristiken, die durch Training mit einem Menschen erlernt wurden
- P[automatic]: Heuristiken, die durch Training mit einem anderen Programm erlernt wurden
- P[implicit]: Implizites Training der Heuristiken
- P[]: zufällige Entscheidung

Heuristiken modifizieren und hinzufügen

- Benötigt wird Trainingsinformation
- Bestehend aus Informationen über
  - Akzeptanz: Wie gut (akzeptabel) ist eine Entscheidung in gegebener Situation
  - Relevanz: Welche Variablen des Subvektors sind wichtig zum Treffen der guten Entscheidung
  - Rechtfertigung: Der Grund, aus dem diese Entscheidung getroffen wurde, ergibt sich durch Evaluierung der Relevanten Variablen

Decision Matrix

- Rechtfertigungsinformation für implizites Training wird Decision Matrix entnommen
- Zeile der Matrix steht für Spielentscheidung
- Spalte steht für eine Subvektor Variable

→Eintrag in der Matrix beschreibt wie wichtig eine Variable des Vektors für eine gegebene Spielsituation ist

Hypothesis-Formation

- Relevanzinformation für implizites Testen wird durch Generierung und Testen von Hypothesen gewonnen
- Relevante Variablen werden auf rechter Seite von Regeln nicht mit * ausgedrückt

Beispiel: (A1, *, C1) → (*, X + b, *)

Erlernen von Heuristic Rules

- Extraktion einer action rule (trainings rule) aus der Trainingsinformation
- Akzeptanzinformation bedient rechte Seite der Regel
- Relevanz- und Rechtfertigungsinformation bedient linke Seite der Regel.
- Schwache Regel wird ermittelt
  - Generalisierung: Modifizierung der Regel, damit sie symbolischem Subvektor der training rule entspricht
  - Falls nicht möglich wird training rule direkt nach dieser schwachen Regel eingefügt

Erlernen von Heuristic Definitions

- Wertebereich der Variablen wird partitioniert
  - mutual exclusion
    - A1 → A, A < 15
    - A2 → A, A ≥ 15
  - overlapping
    - A1 → A, A > 10
    - A2 → A, A > 4

Erweiterungen

- Decision Matrix kann auch erlernt werden
- Bei Beginn wird leere Matrix eingegeben
- Programm ermittelt dann entsprechende Einträge der Matrix während Spielablaufs

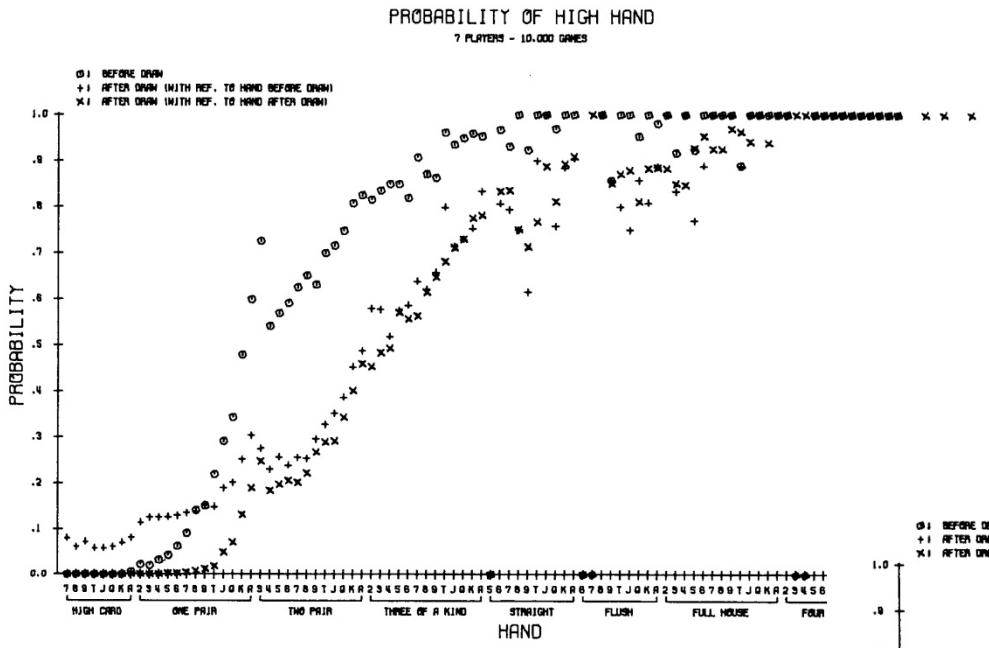
Inhaltsverzeichnis

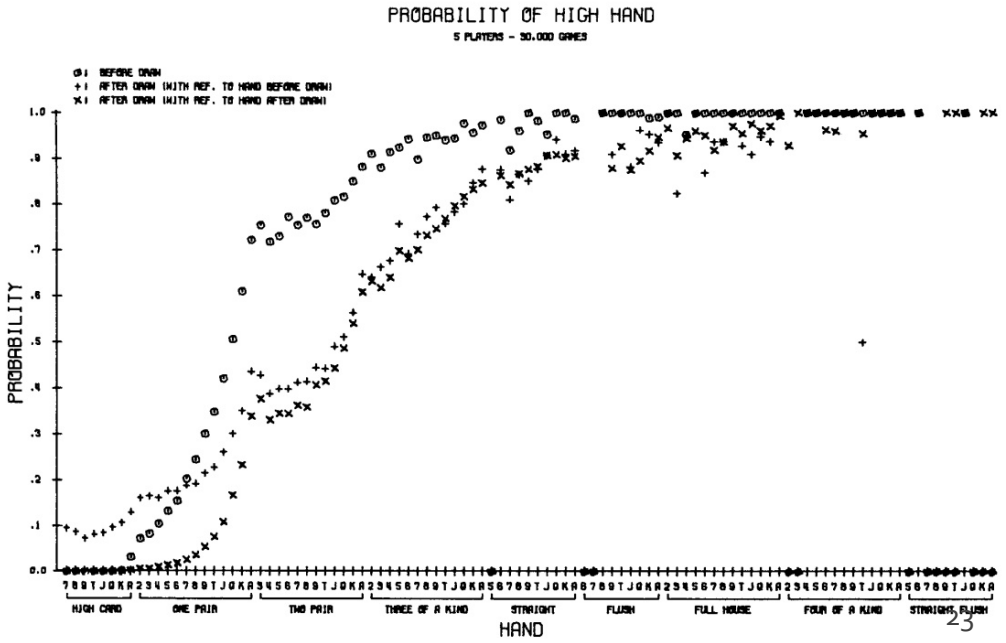
- Einleitung
- Generalization Learning Techniques for Automating the Learning of Heuristics
- **Studies in Machine Cognition Using the Game of Poker / Computer Poker**
- Flexible Learning of Problem Solving Heuristics through Adaptive Search

Studies in Machine Cognition Using the Game of Poker / Computer Poker

- Verschiedenste Strategieansätze am Beispiel von Draw Poker
- Ausgangspunkt oft Monte Carlo Verteilung
  - Vor dem Draw
  - Nach dem Draw im Vergleich zu den Händen vor dem Draw
  - Nach dem Draw im Vergleich zu den Händen nach dem Draw
- Evaluation der Strategien untereinander
  - Verbesserungsrate
  - Asymptotische Spielstärke

Monte Carlo Verteilung





Statische Strategien

- Mathematically Fair Player (MFP)
  - Fair bet

$$p_j * B_0(j,k) = (1 - p_j) * [\sum_{m=1}^{k-1} B_a(j,m) + B_f(j,k)]$$

- RH
  - Verhältnis der positiven zur negativen Motivation

$$RH = \frac{TABLEPOT}{LIVE * (RAISECOUNT + 1) * FOLLOWERS * RAISE}$$

Adaptive Evaluator of Opponents (AEO)

- Einschätzung der Gegner
  - Startet mit grober Einschätzung
- Partitionierung des Lösungsraums
  - 20 gleichverteilte Partitionen
- Verbesserung in drei Dimensionen
  - Anpassung der statistischen Parameter
  - Gewichtung der statistischen Parameter
  - Wahl der statistischen Parameter

Adaptive Aspiration Level (AAL)

- Zwei Zustände
  - Maximierung des Gewinns
  - Verteidigung des Eigenkapitals
- Zustandsübergänge
  - Bei starken Abweichungen
    - Glückssträhne
    - Verlust mit starker Hand
- Anpassung der Fair Bet

Selling and Buying Players‘ Images (SBI)

- Investierung in Spielerinformationen
  - Verkauf falscher eigener Information
  - Kauf von Informationen über die Mitspieler
- Einordnung der Mitspieler
  - Grad der Konsistenz
    - Abweichung von „fair bet“
  - Grad der Zurückhaltung
    - Durchschnittlicher Gewinn/Verlust pro Spiel

Bayesian Strategies (BS)

- Verbesserung durch Vergleich des berechnetem Ausgang eines Spiels mit dem tatsächlichen Ausgang
- Vordefinierte Heuristiken
  - Anpassung der Parameter
  - Neuordnung der Heuristiken
  - Erstellung, Test und Hinzufügen neuer Heuristiken
- Vier Spieler die sich in der Menge der verwendeten Informationen unterscheiden

Bayesian Strategies (BS)

- BS 3 am erfolgreichsten
- Aktionen
  - Folding, Raising, Calling
- Informationen
  - Stärke der Hand und dazugehörige Aktion
  - Summe der Gewinne
  - Häufigkeitsverteilung von Spielsituationen
  - Zeitpunkt des Ereignisses (Predraw, Postdraw etc.)

EPAM-like Player (EP)

- Benutzt EPAM-Bäume, um menschliches Lernen zu simulieren
  - Quantensprünge
- Baum zur Beschreibung jedes Mitspielers
  - Korrelation Verhalten <-> Hand
  - Descriptive Tree
- Baum zur Beschreibung der eigenen Aktionen
  - Quasi-optimum gegenüber Menge der Mitspieler
  - Normative Tree

Quasi-Optimizer (QO)

- Überwacht verchiedene Strategien
  - Zerlegt diese in Einzelkomponenten
  - Setzt diese zu normativer Strategie zusammen
    - Konvergiert gegen theoretisches Optimum
- Kategorisierung von Entscheidungen
  - Zustand des aktuellen Spiel
  - Inhalt
  - Form
- Probleme mit Kategorisierung

Pattern Recognizing Strategy (PRS)

- Menschen nicht zufällig
  - Verhaltensmuster erkennbar
- Verhaltensmuster
  - „landmark“ „trend“„periodicity“„randomness“
- Beobachtbare Elemente
  - Aktionen, Finanzsitutation, Zeitliche Zuordnung
- Nichtbeobachtbare Elemente
  - Wert der gegnerischen Hand, Bluff

Statistically Fair Player (SFP)

- Basiert auf dem MFP
  - Dieser läßt sich Ausbluffen
  - Erweiterung durch Bluffen
- Einordnung der Mitspieler
  - Grad der Konsistenz
    - Hohe Konsitenz <-> weniger Bluffen
  - Grad der Zurückhaltung
    - Hohe Zurückhaltung <-> niedrigere Bluffs

Inhaltsverzeichnis

- Einleitung
- Generalization Learning Techniques for Automating the Learning of Heuristics
- Studies in Machine Cognition Using the Game of Poker / Computer Poker
- **Flexible Learning of Problem Solving Heuristics through Adaptive Search**

Flexible Learning of Problem Solving Heuristics through Adaptive Search

- Unabhängig von der Problemstellung
- Suche im Lösungsraum
  - Genetischer Algorithmus
    - Crossover: c1c2|c3c4 + c5c6|c7c8 => c1c2c7c8 + c5c6c3c4
    - Inversion: c1|c2c3|c4 => c1c3c2c4
    - Mutation
      - Kleine Wahrscheinlichkeit
      - Erreichbarkeit aller Punkte im Suchraum

Flexible Learning of Problem Solving Heuristics through Adaptive Search

- Systemaufbau
  - Knowledge Base
    - m Heuristiken
  - Problem Solving Component
    - k Problemstellungen
  - Critic
    - Bewertung aller Lösungen
  - Genetischer Algorithmus
    - Auswahl und Veränderung der besten Heuristiken

Flexible Learning of Problem Solving Heuristics through Adaptive Search

- Problem Solving Component
  - Problemabhängig:
    - Zustandsvariablen z.B. Wert der eigenen Hand
    - Operationen z.B. Call
- Critic
  - Mißt Performanz
    - Problemabhängige Funktionen
    - Problemunabhängige Teile
      - Strukurelle Eigenschaften
      - Effizienz

Flexible Learning of Problem Solving Heuristics through Adaptive Search

- Anwendung auf Draw Poker
  - Spiel gegen P[built-in]
  - Critic
    - Pokeraxiome
    - Anzahl der erfolgreichen Runden
  - Zustandvariablen
    - Stärke Eigene Hand, Pott, letzter Einsatz etc.
  - Operationen
    - Call, drop, bet low, bet high

Flexible Learning of Problem Solving Heuristics through Adaptive Search

- Sehr gute Performanz
  - Schlägt P[built-in]
  - Gleiche Performanz gegen Mensch
    - Weniger Lernrunden
- Reduzierung der Problemabhängigkeit
  - Problem Solver
    - Zustandsvariablen
    - Operatoren
  - Critic