# Trainieren eines Computer-Pokerspielers

**Reproduktion des Spielverhaltens von Pokerspielern durch algorithmische Ansätze des Maschinellen Lernens**

Bachelor-Thesis von Theo Alois Keiji Kischka aus Darmstadt

Mai 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Fachgebiet Knowledge Engineering
Prof. Dr. Johannes Fürnkranz

Trainieren eines Computer-Pokerspielers
Reproduktion des Spielverhaltens von Pokerspielern durch algorithmische Ansätze des Maschinellen Lernens

Vorgelegte Bachelor-Thesis von Theo Alois Keiji Kischka aus Darmstadt

1. Gutachten: Prof. Dr. Johannes Fürnkranz
2. Gutachten: Dr. Eneldo Loza Mencía

Tag der Einreichung:

# Erklärung zur Bachelor-Thesis

Hiermit versichere ich, die vorliegende Bachelor-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 15. Mai 2014

(Theo Alois Keiji Kischka)

## Inhaltsverzeichnis

# 1 Einleitung

Das Untersuchen von Spielen, wie z.B. Schach oder Dame, stellt eine populäre Disziplin der Forschungsbereiche Maschinelles Lernen und Künstliche Intelligenz dar. Grund dafür ist, dass sich Spiele durch fest definierte Regeln und Ziele auszeichnen und deshalb gute Rahmenbedingungen zur Evaluation bieten. Der Ausgang eines Spielzugs oder Spiels kann oft unmittelbar beurteilt und somit die Qualität der Strategie engeschätzt werden. Auch sind sie leicht zu veranschaulichen und zu verstehen. Die Motivation bei der Untersuchung von Spielproblemen ist, die gewonnenen Erkenntnisse zur Lösung anderer Problemstellungen zu verwenden und die Qualität der eingesetzten Methoden und Verfahren bewerten zu können.

In dieser Arbeit wird das Kartenspiel Texas Hold'em Poker in der Variante drei-Spieler-Fixed-Limit betrachtet. Texas Hold'em ist eine der vielen Spielvarianten von Poker. Poker zeichnet sich durch eine sehr große Anzahl an unterschiedlichen Spielsituationen aus. Es gibt ca. $10^8$ dieser Spielsituationen (Fürnkranz u. a., 2008). Aufgrund dieser Anzahl, welche mitunter der Tatsache, dass Poker ein nichtdeterministisches Spiel ist, denn es werden pro Spiel bis zu fünf Gemeinschaftskarten und jeweils zwei Karten pro Spieler zufällig ausgewählt, zu verdanken ist und der Eigenschaft des Spiels, nicht vollständig beobachtbar zu sein, ist es schwer eine Strategie zu finden, welche langfristig den maximalen Gewinn erzielt. Die Variante Fixed-Limit beschränkt die Anzahl der möglichen Spielsituationen. So ist bei Fixed-Limit-Spielen die Anzahl der möglichen Erhöhungen auf eine bestimmte Anzahl **fixiert** und der Betrag einer Erhöhung auf festgelegte Werte **limitiert**. Bei der Entwicklung von Pokeragenten oder Bots gilt es als Herausforderung eine Strategie zu finden, die so wenig wie möglich ausnutzbar ist und gleichzeitig die Gegner maximal auszunutzt. Vor allem bei drei-Spieler-Poker, die Variante mit der sich diese Arbeit beschäftigt, ist dies der Fall. Bei der Evaluierung müssen wegen der aus dem nichtdeterministischen Charakter des Spiels resultierenden Varianz viele gespielte Hände beobachtet werden. Eine Plattform, welche die Fortschritte in der Entwicklung von intelligenten Pokeragenten zur kompetitiven Schau stellt ist das jährlich stattfindende ACPC (AcpcAktuell, 2014), einem Poker-Bot-Wettkampf. Bei diesen Wettkämpfen nehmen auch Teams der Technischen Universität Darmstadt teil.

## 1.1 Sartre

Einer der Pokeragenten, welcher an den ACPC teilnimmt und gute Resultate vorzuzeigen hat, ist der Bot Sartre. Im Wettkampf von 2010 kann die No-Limit-Variante des Bots SartreNL den zweiten Platz in der Kategorie „Bankroll instant run-off" belegen (Rubin und Watson, 2011). Auch die Mehrspieler-Limit-Variante des Bots schneidet gut bei den ACPC ab (Rubin und Watson, 2012). In dieser Kategorie werden mehrere Runden gespielt. Pro Runde wird der schlechteste Spieler eliminiert. Sartre verwendet keine „herkömliche" Strategie zur Findung einer Strategie, sondern ahmt das Verhalten eines anderen Spielers bzw. Bots nach. Er verwendet das aus dem Maschinellen Lernen bekannte Verfahren des fallbasierten Schließens.

Beim fallbasierten Schließen werden Lösungen fur aktuelle Probleme durch Vergleich mit bereits bekannten Problemen und ihren Lösungen adaptiert und wiederverwendet. Diese Vorgehensweise ist auch als CBR-Zyklus bekannt. Die Sammlung von beobachteten Fällen wird auch als „Case-Base" bezeichnet und kann durch real beobachtete Poker-Spielsituationen oder durch das Spielen von Matches durch Bots erzeugt werden. Fallbasiertes Schließen ist als Top-Down-Ansatz auf die Qualität der Daten stark angewiesen. Vor allem bei einem Spiel wie Poker, welches von nichtdeterministischer Natur und unvollständiger Information ist, da bei der Generalisierung der Fälle durch ungeeignete Metriken Informationen verloren gehen können. Gründe, die dennoch für fallbasiertes Schließen in Limit Hold'em sprechen, sind, dass einerseits Fälle leicht zu beschreiben sind, andererseits diese in Fülle vorhanden bzw. erzeugt werden können. Ein Poker-Fall besteht aus der Spielsituation. Die Lösung dazu stellt die Entscheidung des Pokerspieler bzw. Pokerbots dar. Fälle können durch nicht allzu komplizierte Metriken miteinander ver-

glichen werden.

Sartre verwendet neben einer Abbildung von Zwei-Spieler-Strategien zu Mehrspieler-Strategien wenige, aber aussagekräftige Merkmale zur Beschreibung eines Falls. Wenn bei einem Drei-Spieler-Match einer der Spieler aufgibt und seine Hand weglegt, kann Sartre fortan auch in der Zwei-Spieler-Case-Base nach Lösungen suchen. Die von Sartre verwendeten Merkmale sind $E[HS^2]$ als Maß für die (quadrierte) Handstärke, die Setzverhaltenssequenz und die Gemeinschaftskarten. Die Handstärke kann als numerischer Wert direkt verglichen werden. Sie gibt die Gewinnwahrscheinlichkeit an und ist mit dem in dieser Arbeit verwendeten Begriff All-In-Equity gleichzusetzen (s. Abschnitt 3.5.2). Für die Setzverhaltssequenzen wird eine in drei Ähnlichkeitsstufen unterteilte Metrik angewendet. Diese untersucht die Aggressivität der Gegner, indem die Anzahl der Erhöhungen verglichen wird. Die Einteilung in drei Ähnlichkeitsstufen ist nicht linear. Stark ähnelnde Fälle werden mit Ähnlichkeitswerten von 1.0 bis 0.8 bewertet, nicht ähnliche Fälle mit 0. Die Metrik für die Ähnlichkeit der Gemeinschaftskarten verfährt ebenfalls auf diese Weise, lediglich mit einer breiter gefächerten Auswahl an möglichen Werten. Sartre setzt auf eine spezielle Version der Nächste-Nachbarn-Suche bei der Ermittelung von ähnlichen Fallbeispielen. Nachdem alle nach oben beschriebener Metrik relevanten Fallbeispiele gefunden sind, wird ein Tupel mit einer Zuordnung jeder der Aktionen Erhöhen, Mitgehen und Verwerfen mit Auftrittsgewichtung und summierten Gewinnen erzeugt. Dadurch werden verschiedene Strategien zur Entscheidungsfindung möglich. Eine solche Strategie ist, die Aktion mit der häufigsten Nennung zu verwenden. Eine weitere Strategie ist, die Aktion mit dem größten bisher beobachteten Gewinn auszusuchen.

Sartre ist in der Lage zwei an der Technischen Universität Darmstadt entwickelten Pokeragenten „Akuma“ und „dpp“ zu schlagen. In der ACPC 2011 wird die drei-Spieler-Variante von Sartre in der Kategorie „Total Bankroll“ nur von einem anderen Bot geschlagen (Rubin und Watson, 2012). Die frühe Version von Sartre, die mit Daten aus Spielpartien von Akuma und dem dpp-Bot trainiert wird ist in der Lage diese beiden Bots zu schlagen (Rubin und Watson, 2012).

## 1.2 Motivation und Ziel

Durch Sartre wird erstmals die Verwendung eines Top-Down-Ansatzes demonstriert. Top-Down ist in diesem Zusammenhang als das Entwickeln einer Strategie aus bereits vorhandenen Spielverhaltens-Daten zu verstehen. Es wird erfolgreich gezeigt, dass durch fallbasiertem Schließen das Verhalten eines Pokerspielers nachgeahmt werden kann. Die durch Generalisierung von beobachtetem Experten-Verhalten erzeugten Strategien erweisen sich als robust und zeigen oftmals eine bessere Performanz als die Strategien mit Bottom-Up-Ansatz (Rubin und Watson, 2012).

Nun stellt sich die Frage, ob es auch mit anderen Verfahren des Maschinellen Lernens möglich ist, durch einen Top-Down-Ansatz erfolgreich das Verhalten von Bots zu nachzuahmen. Hierfür soll zunächst Software geschaffen werden, welche diese Untersuchung erleichtern soll. Anders als bei Sartre, welcher lediglich eine kleine Auswahl an Metriken verwendet, sollen viele Spielmerkmale durch die Software erfasst und verwendet werden. Sartre benutzt eine angepasste Version der Nächste-Nachbarn-Suche als Klassifizierer. Es soll untersucht werden, ob sich andere Lernverfahren ebenfalls für diese Aufgabe eignen. Die Reproduzierbarkeit eines Verhaltens durch Verfahren des Maschinellen Lernens steht im Vordergrund dieser Arbeit. Eine Zusatzfragestellung lautet, ob sich durch eine Analyse von Modellen, Rückschlüsse auf die Implementierung einer Strategie treffen lassen.

## 1.3 Aufbau der Arbeit

Zunächst werde ich in Kapitel 2 die Spielregeln der Poker-Variante Texas Hold’em erklären. Dabei gehe ich besonders auf die Variante Fixed-Limit ein. Zusätzlich dazu erkläre und definiere Begriffe, die in dieser Arbeit häufiger gebraucht werden. Der in dieser Arbeit verfolgte Ansatz wird in Kapitel 3.1 erklärt. In Kapitel 3.2 beschreibe ich die Software, welche im Zuge der Arbeit erstellt wird und bereits existierende

Software, die eingesetzt wird. Kapitel 4 beschreibt die Gewinnung der Experimentierdaten und die Aufteilung der Experimente in Voruntersuchung (s. Kapitel 5), Klassifikationsexperimente (s. Kapitel 6) und Praxisexperimente (s. Kapitel 7). Zum Schluss werde ich im Fazit (Kapitel 8) die Ergebnisse diskutieren.

## 2 Spielregeln der Poker-Variante Texas Hold'em

In diesem Kapitel werden die Spielregeln von *Texas Hold'em Poker* erklärt. Es wird auch auf die Spielvariante *Fixed-Limit* eingegangen. Zusätzlich werden noch einige Pokerbegriffe erläutert. Ich werde mich dabei an den Web-Artikeln (Wikipedia, 2014) und (pokerstrategy, 2014) orientieren.

Bei *Hold'em-Pokervarianten* versuchen die Spieler ihre Handkarten mit fünf auf dem Tisch offengelegten *Gemeinschaftskarten* zu einer Pokerhand zu kombinieren. Die Spieler können in mehreren, im Uhrzeigersinn ablaufenden *Wettrunden* auf ihre Karten setzen. Ziel ist es, den Topf mit den darin eingezahlten Wetteinsätzen *(Pot)* zu gewinnen. Dies ist möglich, indem man die beste Pokerhand erhält oder die Gegner zur Aufgabe zwingt.

### 2.1 Spielbeginn

#### 2.1.1 Blinds

Zunächst sind zwei Pflichteinsätze, die sogenannte blinden Einsätze *(Blinds)*, zu leisten. Die Einsatzgrößen werden durch das Spiel-Limit vorgegeben. Beispielsweise beträgt in einem 1$/2$-Spiel die Höhe des kleinen Einsatzes, des sogenannte *Small-Blinds (SB)*, einen Dollar, die Höhe eines ganzen Einsatzes, des *Big-Blinds (BB)*, zwei Dollar. Nach jedem Spiel wird eine Markierung, der sogenannte *Button (BU)*, im Uhrzeigersinn am Tisch weitergereicht. Diese Markierung identifiziert den innehabenden Spieler als Kartengeber. Der Spieler links des Buttons, bzw. des Kartengebers, hat einen kleinen Pflichteinsatz in Höhe eines halben Big-Blinds bzw. eines Small-Blinds zu leisten. Der Spieler links des Small-Blinds hat einen Pflichteinsatz in Höhe eines ganzen Big-Blinds zu leisten.

#### 2.1.2 Karten

Zu Beginn eines Spiels erhält jeder Spieler zwei zufällig aus einem aus 52 Karten des anglo-amerikanischen Blatts bestenden Decks gezogene Handkarten *(Hole Cards)* ausgeteilt. Eine Karte hat einen Kartenrang und eine Farbe. Es gibt dreizehn Kartenränge, beginnend mit der Ziffer Zwei und endend beim Ass. Die Ziffer Zwei stellt den niedrigsten Wert dar, das Ass stellt den höchsten Wert dar. Die vier Farben sind Kreuz, Pik, Herz und Karo. Die Handkarten können mit im Spielverlauf aufgedeckten Gemeinschaftskarten zu stärkeren Händen kombiniert werden. Eine Texas-Hold'em Hand besteht aus genau 5 Karten. Es wird stets die stärkste Handkombination gewertet. Ein Überblick aller möglichen Handstärken findet sich zum Schluss des Kapitels.

### 2.2 Aktionen

Sind die Blinds gesetzt und die Handkarten einmal ausgeteilt, beginnt die erste der insgesamt vier Wettrunden. Mögliche Aktionen der Spieler während der Wettrunden sind das Setzen von Chips *(Bet)*, bzw. eine Erhöhung des Einsatzes *(Raise)*. Ein Spieler kann auch einen Einsatz mitgehen *(Call)*, ohne ihn zu erhöhen. Beträgt der zu leistende Einsatz null Chips, wird das Mitgehen auch Schieben *(Check)* genannt. Möchte man den Einsatz weder erhöhen noch mitgehen, kann man die Hand verwerfen *(Fold)*. Durch einen Fold verliert man jedoch alle bisher in den Pot gezahlten Einsätze, sowie die Chance den Pot oder einen Teil des Pots für sich zu beanspruchen.

### 2.3 Wettrunden

Das Spiel unterteilt sich in die vier Wettrunden *Preflop, Flop, Turn und River* gefolgt von einer weitere Phase, dem *Showdown*. Preflop bezeichnet die Phase, bevor die ersten drei Gemeinschaftskarten aufgedeckt werden. Wetten werden im Urzeigersinn am Tisch abgeschlossen. Preflop hat der Spieler links des Big-Blinds als Erster zu handeln. In den Postflop-Runden beginnt der Small-Blind. Nach Abschluss der

Wetten der Preflop-Phase und dem Offenlegen der Gemeinschaftskarten folgen die drei weiteren Wettrunden Flop, Turn und River während, welcher wieder Wetten platziert werden können. Anders als beim Flop, bei welchem drei Karten aufgedeckt werden, werden auf dem Turn und dem River nur jeweils eine neue Gemeinschaftskarte aus dem Deck gezogen und aufgedeckt. Verbleibt nach dem River mehr als ein Spieler, so kommt es zum Showdown. Beim Showdown wird ermittelt, wer gewonnen hat. Die Spieler decken der Reihe nach ihre Hole-Cards auf. Hält ein Spieler eine schwächere Pokerhand als eine von seinen Gegnern zuvor aufgedeckte Pokerhand, so kann er auch seine Handkarten ohne sie seinen Kontrahenten preiszugeben verwerfen. Es gewinnt der Spieler mit der stärksten Hand.

## 2.4 Besonderheiten Fixed-Limit-Spielvariante

Anders als bei der Spielvariante *No-Limit*, bei welcher die Höhe der Wetteinsätze nur durch die Größe der Chip-Stapel beschränkt ist, sind bei *Fixed-Limit* nur festgelegte Setzgrößen möglich: In der Preflop- und Flop-Phase beträgt eine Erhöhung des Einsatzes einen Big-Blind. Während der Turn- und Riverphase sind es zwei Big-Blind. Zusätzlich sind die Anzahl der Erhöhungen ebenfalls festgelegt. So sind in der Preflop-Phase bis zu drei Erhöhungen, in den Postflop-Phasen bis zu vier Erhöhungen möglich.

## 2.5 Spielerpositionen

Der Name der ersten Preflop-Spielposition, „Under the Gun“ *(UTG)* (engl. für *unter vorgehaltener Schusswaffe*), soll den psychologischen Druck, welchem dieser Spieler ausgesetzt ist, illustrieren: Die übrigen Spieler am Tisch können nach ihm agieren und genießen den Vorteil auf vorangegange Aktionen adäquater reagieren zu können. Der zuerst agierende Spieler muss jedoch das Verhalten aller folgeden Spieler abschätzen. Die Position links von UTG wird UTG+1, die Position links von UTG+1 als UTG+2 bezeichnet. Die mittleren Position erhalten die Bezeichnungen MP1,MP2 und MP3. Der Spieler links von MP3 wird *Cut-Off (CO)* genannt. Zu seiner Linken sitzt der Button *(BU)*. Die Spieler, welche die blinden Einsätze geleistet haben, heißen bezeichnenderweise Small-Blind *(SB)* und Big-Blind *(BB)*. In einem Spiel mit nur drei Spielern existieren lediglich die Positionen Small-Blind, Big-Blind und Button.

## 2.6 Pokerhände

Die folgenden Pokerhände sind nach ihrer Stärke absteigend aufgelistet.

- Straight-Flush: Eine Kombination aus einer Straße und einem Flush. Fünf Karten der gleichen Farbe in einer Reihe sind notwendig. Ein Straight-Flush von Zehn bis Ass stellt die stärkste Hand dar und wird auch *Royal Flush* genannt.
- Vierling *(Four-of-a-Kind, Quads)*: Vier Karten gleichen Rangs
- Full-House: Kombination aus einem Drilling mit einem Paar
- Flush: Fünf Karten der gleichen Farbe
- Straße *(Straight)*: Fünf Karten in auf- bzw. absteigender Reihenfolge ihres Rangs
- Drilling *(Three-of-a-Kind, Set, Trips)*
- Zwei Paare *(Two-Pair)*
- Paar *(Pair)*
- Höchste Karte *(High-Card)*

## 2.7 Draw

Als einen *Draw* (engl. für Zug) bezeichnet man eine Handkombination, welche zwar noch unvollständig ist, sich aber noch zu einer stärkeren Pokerhand, wie z.B. einem Flush oder einer Straße, verbessern kann. Ein Paar hingegen, welches sich noch zu zwei Paaren verbessern kann, wird jedoch nicht als Drawing-Hand bezeichnet, da es als bereits vervollständigt gewertet wird.

## 2.8 Begriffe

Im Poker-Jargon bezeichnet „Hero“ (engl. für Held) den Spieler, aus dessen Sicht das Spielgeschehen präsentiert wird. In der Regel stehen über den Hero-Spieler mehr Informationen als über seine Gegner zur Verfügung. Beispielsweise sind die Handkarten der Gegner für Hero vor dem Showdown nicht einsehbar.

## 2.9 Das Pot-Odds-Kalkül

Mit dem Pot-Odds-Kalkül kann ermittelt werden, welche der Aktionen Mitgehen oder Verwerfen langfristig einen höheren Gewinn erzielt. Die All-In-Equity oder Equity drückt die Gewinnwahrscheinlichkeit als numerischen Wert zwischen 0 und 1 aus. Dieser Wert kann durch vorberechnete Tabellen oder durch eine Monte-Carlo-Simulation bestimmt werden. Die Pot-Odds drücken das Verhältnis des zu leistenden Betrags um Mitzugehen im Verhältnis zum möglichen Gewinnbetrag aus. Die Formel lautet:

$$PotOdds = BetragUmMitzugehen/(Potgrösse + BetragUmMitzugehen)$$

Das Kalkül lautet: Wenn die Equity größer oder gleich den PotOdds ist, dann gehe mit. Ansonsten verwerfe die Hand (Schaffer und Uwira, 2008).

# 3 Lernen von Pokerspielern und technische Umsetzung

## 3.1 Grundidee

Der Ansatz, der in dieser Arbeit verfolgt wird, soll nun kurz dargelegt werden. Es wird versucht Spielsituationen durch Attribute zu beschreiben. Dann soll eine Zuordnung auf die in einem Pokerspiel möglichen Aktionen Erhöhen, Mitgehen und Verwerfen gefunden werden. Es werden, im Vergleich zu Sartre, mehr Attribute definiert. Es werden Modelle aus Spieldaten von Pokerpartien für drei Spieler berechnet. Die resultierende Modelle werden durch Kreuzvalidierung bewertet. Die Übereinstimmung der Verhalten erzeugt durch die Modelle werden mit dem Verhalten der Vorbilder verglichen, indem Bots, die nach den Modellen spielen, sich in einem Fixed-Limit-Texas-Hold'em-Wettkampf für drei Spieler messen.

## 3.2 Technische Umsetzung des Poker-Bot-Trainer-Tools

Im Zuge der Bachelor-Arbeit wird das Java-Tool „Poker-Bot-Trainer“ entwickelt. Es stellt einige Werkzeuge zur Verfügung, welche das Zusammenspiel mit dem Machine-Learning-Framework Weka (Witten und Frank, 2005) erleichtern und dabei auf dem bereits vorhandenen TUD-Poker-Framework aufsetzt (Bank, 2011; Zopf, 2010).

## 3.3 Die Machine-Learning-Software Weka

Bei Weka („Waikato Environment for Knowledge Analysis“) handelt es sich um ein in Java geschriebenes Softwarepaket der University of Waikato (Witten und Frank, 2005). Es stellt eine breites Angebot von Algorithmen des Maschinellen Lernens für Datamining zur Verfügung. Über eine graphische Oberfläche lassen sich bequem Daten manipulieren und zur Analyse visualisieren. Der Kern der Software lässt sich auch im eigenen Java-Code integrieren und verwenden. Neben diverser Filter für die Aufbereitung der Daten gibt es noch Klassifizierer und Meta-Klassifizierer als Hauptanwendungen und schließlich Methoden zur Evaluation der Ergebnisse (Holmes u. a., 1994). Die Verwendung des Frameworks steht unter der Regelung der GNU GPL.

**Klassifizier- und Lernalgorithmen**

Im Folgenden eine Auflistung einiger Weka-Klassifizierer, welche Anwendung in dieser Arbeit finden.

- JRip: JRip ist eine Implementierung des Ripper-Verfahrens (Witten und Frank, 2005, Kap. 10.4).
- Der Naive-Bayes-Lerner.
- Ibk: Dient zur Verwendung des kNN-Algorithmus.
- J48: J48 ist eine Implementierung des C4.5 Algorithmus'. Es erzeugt Entscheidungsbäume (Witten und Frank, 2005, Kap. 6.1, Kap. 10.4).
- SMO: SMO implementiert die „Sequential Minimal Optimization“-Methode für „Support-Vector“-Maschinen (Witten und Frank, 2005, Kap. 10.4).

**Evaluation**

Für die Evaluation ist die Klasse `weka.classifiers.Evaluation` zuständig. Mit ihr kann u.a. eine Kreutzvalidierung durchgeführt und das Resultat neben einiger Metriken wie beispielsweise die Präzision bzw. Korrektklassifikationsrate („Accuracy“) eines Modells auch als Konfusionsmatrix dargestellt werden. Die Aufteilung der Daten in Trainings- und Testinstanzen kann, muss aber nicht automatisch erfolgen. In dieser Arbeit wird fortan der englische Begriff Accuracy für die Korrektklassifikationsrate verwendet.

#### Das Attribute Relation File Format (ARFF)

Zur Repräsentation von Daten verwendet Weka das ARFF. Viele der Klassifizierer erwarten dieses Format zur Einspeisung der Daten. In einer ARFF-Datei wird eine Relation von Attributen bzw. Merkmalen zusammen mit Instanzen dargestellt. Es existieren mehrere unterschiedliche Attribut-Typen in Weka: Numerische, Relationale, Nominale und String-Attribute. Die möglichen Werte eines nominalen Merkmals müssen als Liste definiert werden. In einer ARFF-Datei stehen die Attribute-Definitionen im Kopf. Instanzen sind ab der Markierung „@data“ vorzufinden. Alle Informationen innerhalb einer ARFF-Datei sind in Klartext (Holmes u. a., 1994); (Witten und Frank, 2005, Kap. 2.4).

## 3.4 Bereits vorhandene Software: Das TUD-Poker-Framework

Markus Zopf entwickelt im Rahmen seiner Thesis das TUD-Poker-Framework(Zopf, 2010). Es handelt sich hierbei um ein Java-Programm, welches bei der Entwicklung von Pokeragenten für die ACPC zur Hand geht, indem grundlegende Aufgaben, wie beispielsweise das Parsen der Spielzustands-Strings des ACPC-Pokerservers in umgänglichere Datenformate, den Entwicklern abgenommen wird. Das Framework beinhaltet auch einen Spiele-Server für Testzwecke. Dieser kann über eine GUI bedient werden. Einige simple Pokeragenten sind bereits vorhanden und können als Gegner verwendet werden. Im Hinblick auf meine Arbeit wichtige Klassen des TUD-Poker-Frameworks sollen im Folgenden besprochen werden.

#### Die Klasse `pokertud.gamestate.GameState`

GameStates sind Repräsentationen von Spielzuständen. Sie werden mit einer Factory-Methode aus Spiezustands-Strings, welche vom Server an die Klienten verschickt werden, erzeugt. Ein Spielzustand hält Informationen über den Spieler (`Player`), der Position relativ zum Button, die aktuelle Wettrunde, investierte Chips und die Aktionen der Spieler. Diese Informationen können abgerufen und zur weiteren Analyse des Spiels und zur Ableitung von abstrakteren Metriken genutzt werden.

#### Die Klasse `pokertud.clients.PokerBotClient`

Pokeragenten haben die Klasse `PokerBotClient` zu erweitern, um auf dem TUD-Server und dem ACPC-Server laufen zu können. Änderung des Spielzustands werden über die abstrakte Methode *`handleGameStateChange`*, welche es zu überschreiben gilt, an die Klienten mitgeteilt (Zopf, 2010).

## 3.5 Das Tool Poker-Bot-Trainer

Im Folgenden ein Überblick der wichtigsten Komponenten, welche im Rahmen der Bachelor-Arbeit entwickelt werden.

- **PokerBotTrainer.arfflogger**
  Ermöglicht das Loggen von Spieldaten im Weka-Arff während einer laufenden Partie. Alle im Package PokerBotTrainer.metrics definierten Metriken werden berücksichtigt. Server-Log-Konvertierer und Poker-Bots verwenden diese Klasse für die Konvertierung von Spieldaten in Arff.

- **PokerBotTrainer.converter**
  Stellt Konvertierer und Parser für diverse Formate der Spielserver-Logdateien zur Verfügung. Zielformate sind `pokertud.gameState.GameState` und Weka-Arff.

- **PokerBotTrainer.machinelearning.pokerbotclients** Hier sind Poker-Bots zu finden, welche um die Funktionalitäten des Echtzeit-Arff-Logging und das Spielen nach gelernten Weka-Modellen erweitert sind.

- **PokerBotTrainer.machinelearning.singleevent**
  Stellt Klassen zur Single-Event-Prediction bereit, welche von den Poker-Bots genutzt werden.

- **PokerBotTrainer.metrics** Metriken, welche die Spieldaten analysieren und in als Attribute eines Feature-Vectors (Arff) erfasst werden sind hier definiert. Da sich die Anzahl im Verlauf der Arbeit stetig erhöht hat, sind Mittel zur Kategorisierung und Auswahl vorhanden.

### 3.5.1 Erfassen der Spieldaten als Weka-Attribute-Vektor während einer laufenden Partie

Um Klassifikationsverfahren des Weka-Machine-Learning-Frameworks während einer Spielpartie auf dem TUD-Poker-Server verwenden zu können, ist es notwendig, dass die Spieldaten als Instanzen eines Feature-Vektors vorliegen. Nicht alle Informationen über das Spiel sind jederzeit abrufbar. So sind beispielsweise die Handkarten der Gegner oder der Gewinn bzw. Verlust eines jeden Spielers erst in der Showdown-Phase einsehbar. Andere Informationen, wie beispielsweise die Zuordnung der Spieler zu den Positionen, sind nur zu Beginn einer Partie notwendig zu wissen. Alle Daten zur aktuellen Spielsituation sind in der Klasse `pokertud.gamestate.Gamestate` aggregiert. Da jedoch nicht alle Daten jederzeit wissenswert sind, wird nicht der Ansatz eines Observer-Patterns gewählt, welcher unnötigen Overhead verursacht. Stattdessen werden der Klasse, welche die Spieldaten auswertet, gezielt zu bestimmten Zeitpunkten im Verlauf einer Partie nur die relevanten Daten zugespielt. Hierfür muss eine Kontrollstruktur im Poker-Bot-Klienten geschaffen werden, welche feststellt, welches Ereignis gerade ansteht. Im Folgenden ein knapper Überblick der in der Klasse *`PokerBotTrainer.machinelearning.pokerbotclients.AbstractArffLoggingBot`* implementierten Kontrollstrukturen sowie die Aufrufe der Informationsübermittelungsfunktionen an die Klasse `PokerBotTrainer.arfflogger.ArffLogger`.

- Allererste Spielpartie beginnt: `updateOpponents`
- Eine neue Partie beginnt
- Eine neue Partie beginnt und es ist nicht die allererste Partie: `updateNewHand`
- Hero (s. Abschnitt 2.8) ist am Zug: `updateHeroActionAndGameState`
- Hero ist am Zug und es haben andere Spieler zuvor Aktionen getätigt: `updatePlayerAction`
- Gegner sind am Zug: `updatePlayerAction`
- Gegner sind am Zug und es haben andere Spieler zuvor Aktionen getätigt: `updatePlayerAction`
- Übergang zur nächsten Wettrunde: `updateNewStreet`
- Das Spiel befindet sich in der Showdown-Phase: `updateShowDownGameState`
- Aktuelle Partie-Nr. modulo n ist 0, n ist eine Ganzzahl und kann vor Beginn einer Partie festgelegt werden.

Diese Kontrollstrukturen können von Poker-Agenten, welche *`AbstractArffLoggingBot`* erweitern, für zusätzliche Aktionen ausgenutzt werden.

### 3.5.2 Metriken und Attribute

Der Konstruktor der Klasse `ArffLogger` erwartet als Argumente den Namen des Bots, den es zu beobachten gilt und eine Liste von Attributenamen. Diese Liste kann genauer spezifiziert werden. Zur Unterscheidung der Art der Attribute existieren Präfixe. Als zweite Dimension für eine genaue Festlegung gibt es Kategorien. Sämtliche Attribute sind in der Klasse `PokerBotTrainer.metrics.attributes.EnumFeatures` definiert. Bei der Instanziierung eines `ArffLogger`-Objekts werden vier solcher `EnumFeatures`, eines für jede Wettrunde, nach Angaben abgeleitet aus der Attributenamensliste, erzeugt.

Präfixe dienen zur Unterscheidung, welcher temporärer Natur ein Attribut ist. Die Mehrheit der Attribute besitzen keine Präfixe, was sie als „augenblicklich“ ausweist. Gemeint ist, dass solche Merkmale nur zum Zeitpunkt des Beobachtens gültig sind. Beispiel hierfür sind die Handkarten, die Anzahl der aktiven Spieler, die Potgröße usw. Im Kontrast dazu stehen Attribute, welche über mehrere gespielte Partien aggregiert werden. Dazu zählen Metriken welche Auskunft über das Setzverhalten der Gegner geben. Als dritte Kategorie existieren Merkmale, die „in die Zukunft schauen“. Während eine Partie noch im Gang ist, ist es nicht möglich den Ausgang bereits vorherzusagen. Werden jedoch Server-Logs ausgewertet, stehen solche Informationen zur Verfügung. Es ist daher wichtig, diese „Zukunfts“-Attribute vor einer Weiterverwendung, wie beispielsweise zur Berechnung eines Modells, zu filtern. Sie sollten lediglich für eine händische Kontrolle der Werte herangezogen werden.

Kategorien erfüllen zwei Aufgaben. Durch sie lassen sich bequem Attributelisten erstellen, welche die angegebene Kategorien kombinieren. Beispiel: Durch Angabe der Kategorien „FLOP“ und „TURN“ lässt sich eine Liste aller dieser beiden Kategorien angehörenden Attribute zusammenstellen. Die zweite Aufgabe ist das Festlegung des Attributes auf zulässige Wettrunden. Beispiel: Ein Attribut für die ersten drei aufgedeckten Gemeinschaftskarten ist nicht für die Preflop-Wettrunde zugelassen weil nicht relevant. Eine Beschreibung des Umgangs mit Präfixe und Kategorien findet sich in Abschnitt 3.6.1.

Attribute kommen in vier verschiedenen Typen daher: Relationale, numerische und nominale Attribute sowie String-Attribute. Jedes nominale Attribut muss die Werte, die es annehmen kann als Vektor definieren. Es sind 113 verschiedene Attribute definiert. Wegen der großen Anzahl wird im Folgenden ein grober Überblick der Kategorien aufgelistet. Für eine detailliertere Beschreibung der einzelnen Attribute sei auf Anhang A verwiesen.

- Generelle Spielinformationen: Rohdaten, wie beispielsweise die Anzahl der Spieler, der Betrag der von den Spielern investierten Chips, die Aktionen der Spieler etc.
- Hero-Informationen
- Auf Gegner-bezogene Basis-Daten
- Abgeleitete Merkmale: Durch Abstraktion, Analyse und Anwendung diverser Metriken aus den „Rohdaten“ gewonnene Attribute. Hierzu zählen u.a. All-In-Equity und Pot-Odds.
- Attribute, die das Gegnerverhalten über alle beobachtete Partien beschreiben.
- Attribute, die das Gegnerverhalten zur laufenden Partie beschreiben.

#### Flaschenhals All-In-Equity

All-In-Equity ist ein Maß für die Gewinnwahrscheinlichkeit. Die exakte Bestimmung der All-In-Equity ist aufgrund der vielen Möglichkeiten bei der Verteilung der Hand- und Gemeinschaftskarten technisch zwar durchführbar, aber in der Praxis ungeeignet weil sie zu lange dauert. Während der Matches bei den ACPC existiert ein Zeitlimit für Entscheidungen. Bei nur einem Gegner kann

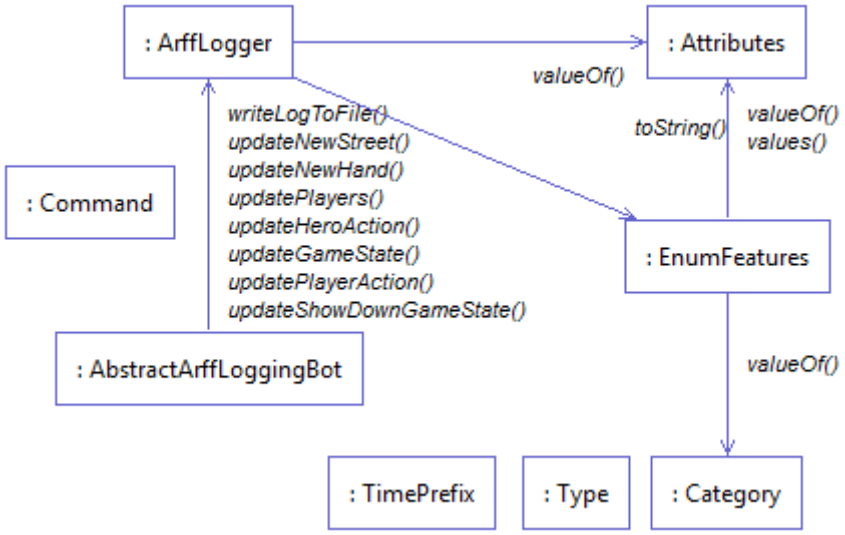


**Abbildung 1:** Interaktionsdiagramm der Klassen, die in den Abschnitten 3.5.1 und 3.5.2 beschrieben werden

auf vorberechnete Tabellen mit Equity-Werten zurückgegriffen werden. Ab zwei Gegner jedoch erhöht sich die Anzahl der möglichen Kombinationen auf 3,788*$10^{13}$ in der Preflop-Phase[1]. Stattdessen wird der Wert mit Monte-Carlo-Simulationen abgeschätzt. Die Methode `getAllInEquity(Cards holeCards, Cards boardCards, int opponentCount, int iterations)` der Klasse `pokertud.divat.MetricsCalculator` erfüllt diesen Zweck. Üblicherweise werden 10000 Iterationen ausgeführt. Um den Prozess zu beschleunigen existiert in der Klasse `PokerBotTrainer.utils.Utils` eine optimierte Version dieser Funktion, die als Kartenbelegung statt Boolean-Arrays einen 64-Bit-Integer verwendet. Auch wird die Tatsache, dass für einen einzelnen Gegner in der River-Wettrunde lediglich 990 Kombinationen verglichen werden müssen, ausgenutzt. Trotzdem bleibt die Berechnung des All-In-Equity-Attributes der Flaschenhals. Ca. 85% bis 90% der Rechenzeit bei der Berechnung aller Attribute wird hierfür veranschlagt.

### 3.5.3 Konvertierung von Logdateien

Um zuvor gespielt und aufgezeichnete Poker-Partien für eine Verwendung durch das Weka-Machine-Learning-Framework zu ermöglichen, ist eine Konvertierung der Poker-Server-Logdateien notwendig. In den Server-Logdateien sind Informationen über den Spielablauf enthalten. Aus diesen Informationen werden durch Analyse diverser Metriken abstraktere Merkmale des Spielablaufs gewonnen. Alle Merkmale werden in einem Array aus Feature-Vektoren zusammengefasst und als „Attribute-Relation File Format“ (ARFF), einem Format, welches vom Weka-Framework gelesen werden kann, gespeichert (Witten und Frank, 2005, Kap. 2.4). Da jedoch durch die minimalistische Repräsentation des Spielablaufs in den Poker-Server-Logs einige Informationen, wie z.B. die Blind-Größen, verlorengehen und um die dadurch resultierende Abweichung bei der Interpretation zu vermeiden, wurde der im TUD-Poker-Framework mitgelieferte Parser verwendet um Instanzen der Klasse `pokertud.gamestate.GameState` zu erzeugen (Zopf, 2010; Bank, 2011). Anschließend werden diese GameStates zu Arff-Dateien konvertiert. Der Vorteil in dieser Vorgehensweise liegt darin, dass nur ein Konvertierer für verschiedene Logdatei-Formate benötigt wird und der Betrag der Blinds, gemessen in Chips, an die im oben erwähnten Parser eingestellten Standard-Werten angepasst werden.

[1] Berechnung: $\binom{50 Deckkarten}{2 Gegnerkarten} * \binom{48 Deckkarten}{2 Gegnerkarten} * \binom{46 Deckkarten}{3 Flopkarten} * 43 Deckkarten * 42 Deckkarten$

Es sind zwei verschiedene Server-Log-Formate der Spielverläufe bekannt. Das Format der Logdateien welches bei den ACPC bis 2009 verwendet wird und das Format, welches für alle neueren Server-Logs verwendet wird. Im Folgenden wird zuerst genanntes Format als „Alt-Format“ und Letzteres als „Neu-Format“ bezeichnet. Das TUD-Poker-Framework speichert Logdateien wiederum in einem eigenen Format ab. Um dennoch diese auswerten zu können müssen Änderungen in den Klassen pokertud.server.Server, pokertud.server.Game und pokertud.server.GameSpecial vorgenommen werden, sodass die Logdateien im Alt-Format gespeichert werden. Die modifizierten Dateien finden sich im Ordner „mods“ des Pokerbot-Trainer-Programms. Eine Betrachtung einiger Beispiele soll den Unterschied der beiden Formate aufzeigen. Das Alt-Format hat die Struktur:

$$< PLAYERS >:< HANDNUMBER >:< BETTING >:< CARDS >:< NETONHAND >$$

Erläuterungen:

- PLAYERS: Die Spielernamen in Reihenfolge beginnend bei dem Small-Blind. Trennzeichen ist „|“
- HANDNUMBER: Wird für jedes Spiel fortlaufend hochgezählt
- BETTING: Das Setzverhalten der Spieler. „r“ steht für eine Erhöhung, „c“ für Schieben oder Mitgehen und „f“ für das Verwerfen der Karten.
- CARDS: Die Handkarten der Spieler in gleicher Reihenfolge wie die Spielernamen, getrennt durch das „|“-Zeichen, gefolgt von den Gemeinschaftskarten für Flop,Turn und River, jeweils getrennt durch ein „/“.
- NETONHAND: Der Ausgang des Spiels gemessen in Big-Blinds in gleicher Reihenfolge wie die Spielernamen.

Ein Beispiel:

$$akuma|Bluechip|dpp : 0 : ccf/rrrc/rc/rrrrc : 2cKs|Jd6h|9cJc/Kc6c2s/7d/9s : 30|-2|-28$$

Das Neu-Format hingegen hat folgende Struktur:

$$< STATE >:< HANDNUMBER >:< BETTING >:< CARDS >:< NETONHAND >:< PLAYERS >$$

STATE ist ein Kontrolltext, welcher sich nicht ändert. Beispiel:

$$STATE : 7 : frc/rc/rc/rc : AdJd|6dQc|5cQs/As6s2c/Ah/8d : 70|-70|0 : Alice|Bob|Charlie$$

Die Formate unterscheiden sich nicht grundlegend. Lediglich die Anordnung der Informationen ist anders. Um die Erweiterbarkeit für zufkünftige Formate zu erleichtern, wird ein Software-Design mit mehreren Konvertierungsstufen gewählt. Es soll nun der Ablauf des Konvertiervorgangs grob beschrieben werden. Um die Informationen aus den oben beschriebenen Zeichenketten, welche die Spiele beschreiben, zu extrahieren, müssen diese durch einen Parser eingelesen werden. Parser für die gezeigten Formate erweitern dazu die abstrakte Klasse `PokerBotTrainer.converter.parser.AbstractGameParser` und müssen die abstrakte Methode `getGameData()` implementieren. Da es erwünscht ist, aus dem Gesamtspielverlauf einzelne, individuelle, d.h. aus Sicht der jeweiligen teilnehmenden Poker-Bots, Situationen des Spielgeschehens wiederherzustellen, werden die durch die Parser eingelesenen Spieldaten in der Klasse `PokerBotTrainer.converter.gamelogic.Game` weiterverarbeitet. Hier werden die Aktionen der *BETTING*-Zeichenkette dem jeweiligen Akteur zugeordnet. Die Klasse `PokerBotTrainer.converter.PreConverter` iteriert alle Spiele in einer Logdatei Zeile für Zeile durch und erstellt wiederum eine Liste aus `Game`-Objekt-Instanzen. Mit der Methode `convert()`, welche wiederum den bereits im TUD-Poker-Framework enthaltene Factory-Methode zur Erzeugung von `GameState`-Objekten der Klasse `pokertud.gamestate.GameStateFactory` aus Situationsbeschreibungen benutzt, werden die Spielsituationsbeschreibungen, welche in Textform vorliegen, in `GameState`-Objekte umgewandelt. Auf höchster Aufrufebene steht die Klasse

`PokerBotTrainer.converter.MainConverter` mit dem Methodenaufruf `convertGameStateToArff()`. Da nun alle Spielzustände vorliegen, können diese noch einmal für einen Poker-Bot, welcher in den Spiele-Logs an den eingelesenen Spielen teilnimmt, „durchlebt“ werden. Das Wieder-Durchleben dient dazu, den bereits vorhandenen Arff-Logger zum Erfassen des Spielgeschehens als Weka-Attribute und Abspeicherns als Arff-Datei wiederzuverwenden. Denn besagter Logger existiert genau für die Aufgabe, während des Spielgeschehens gesehene Situation in Form von `GameState`-Objekten in Arff umzuwandeln. Für zukünftige Formate bedeutet dies, dass lediglich ein Parser, welcher die Klasse `AbstractGameParser` erweitert und die abstrakte Methode *`getGameData()`* implementiert, geschrieben werden muss. Ein Sequenz-Diagramm zum beschriebenen Ablauf des Konvertiervorgangs durch „Wiederdurchleben“ der Spielsituationen findet sich im Anhang D. Abbildung 2 zeigt die wichtigsten Klassen des Konvertiervorgangs.

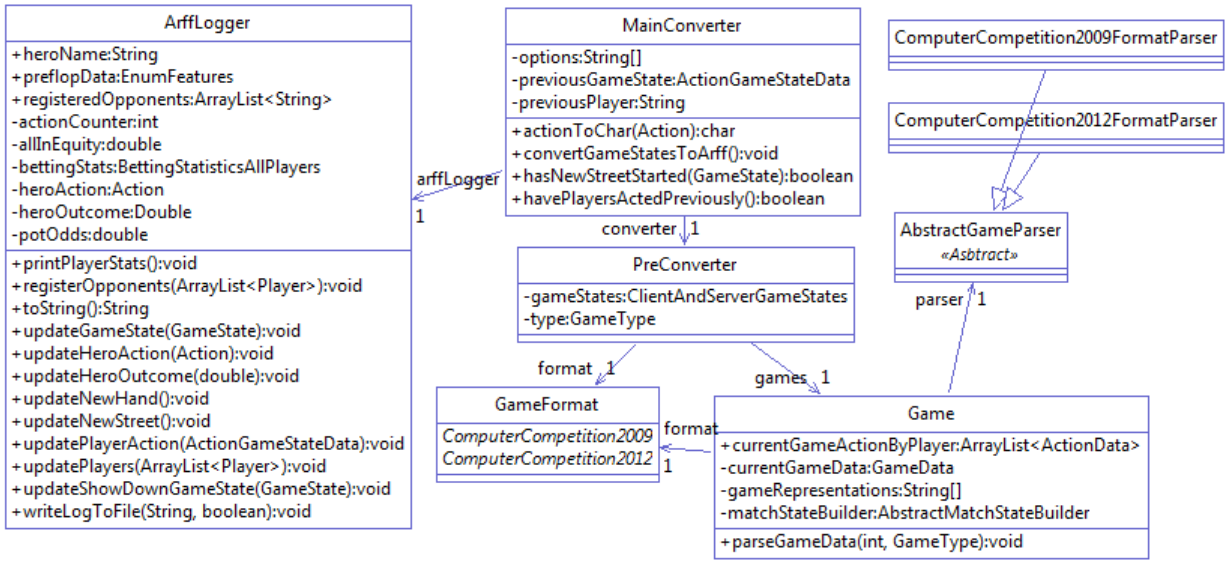


**Abbildung 2:** Klassendiagramm für die für den Konvertiervorgang benötigten Klassen

### 3.5.4 Der Poker-Bot „Trainer-Bot“

Eine alternative Möglichkeit der Erfassung von Spieldaten als Weka-Attribut-Instanzen stellt der „Trainer-Bot“ dar. Er erlaubt eine Echtzeit-Aufzeichnung des Spiels im Arff-Format. Die zugehörige Java-Klasse ist `PokerBotTrainer.machinelearning.pokerbotclients.TrainerBot` und erweitert die abstrakte Klasse *`ArffLoggingBot`* des selbigen Pakets. Erweiterungen des Trainer-Bots müssen die Entscheidungsfindungsfunktion *`public Action makeDecision()`* implementieren. Zum Start des Bots wird eine Liste von aufzuzeichnenden Attributen als Parameter erwartet.

### 3.5.5 Der Poker-Bot „Testing-Bot“

Ähnlich wie sein „Bruder“, dem Trainer-Bot, erfasst der Testing-Bot (`PokerBotTrainer.machinelearning.pokerbotclients.TestingBot`) das aktuelle Spielgeschehen als Weka-Attribut-Instanzen. Allerdings zeichnet er diese nicht in Form eines Arff-Feature-Vektors auf, sondern benutzt die aktuellste Instanz zur Klassifikation. Hierfür wird ein „Event-Predictor“-Objekt (*`PokerBotTrainer.machinelearning.singleevent.AbstractEventPredictor`*) erzeugt. Dieses klassifiziert bei einem passenden vorliegenden Modell mithilfe der Methode *`public Action predict(Instance instance, int classIndex)`* die beobachtete Instanz zu einer der drei Aktionen „Raise“, „Fold“ oder „Call“.

### 3.5.6 „Single-Event-Prediction“

Weka-Modelle werden durch unterschiedliche Arten von Klassifizierern erzeugt. Zum einen existieren Modelle, welche von Klassifikatoren, die das Interface `weka.classifiers.Classifier` implementieren, generiert werden, zum anderen gibt es solche, die von `weka.core.neighboursearch.KDTree` erstellt werden. Es gibt noch weitere Klassifizierer-Typen in Weka. Diese werden aber nicht durch das PokerBotTrainer-Tool unterstützt. Um Modelle beider oben erwähnter Klassifizierer-Typen verwenden zu können, wird ein polymorphes Software-Design gewählt, welches zur Laufzeit den richtigen Klassifizierer-Typ initialisiert. Die Poker-Bot-Clients rufen die abstrakte Methode *`predict`* auf, um eine Einordnung durch gelernte Modelle zu erhalten.

## 3.6 Benutzungsbeispiele

### 3.6.1 Attribute-Liste definieren

Sowohl die Konvertierer, als auch die Poker-Agenten erwarten eine Liste der zu verwendenden Attribute. Diese wird mittels mehrerer Schlüsselwörter, welche im Folgenden vorgestellt werden, zusammengestellt.

- -attributes <Liste von Attributen (s. Anhang A), getrennt durch Kommata ohne Leerzeichen>.
- -categories <Liste von Kategorien: {NO_CATEGORY,GENERAL_GAME_INFO,HERO_INFO, OPPONENT_INFO,DERIVED_INFO,OPPONENT_STATS,OPPONENT_BETTING,HERO_BETTING, ODDS_AND_OUTS,PREFLOP,FLOP,TURN,RIVER,EMPTY_CATEGORY}, getrennt durch Kommata ohne Leerzeichen>. Aus der Auflistung wird die Schnittmenge gebildet.
- -prefix <Einer der Werte: {FUTURE,HISTORY,PRESENT,NOT_SPECIFIED}>
- -noRedundant: Filtert alle redundanten Attribute raus. Für jede unterschiedliche Darstellung einer selben Information existiert ein Attribut, welches als nicht redundant ausgewiesen ist. Es handelt sich bei dieser Präsentation nicht zwangsläufig um die für einen bestimmten Klassifikator am besten geeignetste Darstellung der Information. Beispielsweise existieren für Heros Handkarten neben dem Attribut HERO_HOLE_CARDS, welches Rang und Farbe beider Handkarten erfasst, auch die Attribute HERO_HOLE_CARD_HIGH, HERO_HOLE_CARD_LOW, HERO_HOLE_CARD_HIGH_SUIT und HERO_HOLE_CARD_LOW_SUIT, die gleiche Information jedoch aufgeteilt auf vier verschiedene Attribute.
- -invert: Invertiert die Auswahl der Liste. Um alle möglichen Attribute auszuwählen reicht als Aufruf „-invert“ aus.

Beispielaufrufe: Alle Attribute der Kategorien PREFLOP und RIVER: „-categories PREFLOP,RIVER“.
Alle Attribute außer CLASS_TO_INVEST: „-attributes CLASS_TO_INVEST -invert“.
Alle Kategorien außer TURN: „-categories TURN -invert“.

### 3.6.2 Konvertierung von Logdateien in ARFF

Aufruf der Klasse `PokerBotTrainer.converter.MainConverter` mit Argumenten:

1. Server-Log-Datei
2. Logdatei-Format: {ComputerCompetition2009, ComputerCompetition2012}. „ComputerCompetition2009“ ist für das Alt-Format (s. Abschnitt 3.5.3) und „ComputerCompetition2012“ ist für das Neu-Format zu verwenden.
3. Spieltyp: {ThreePlayerHoldem,ReverseBlindsThreePlayerHoldem}
4. Hero-Name
5. Attribute, beschrieben wie im vorangegangenen Abschnitt

Beispielaufrufe (Argumente): „akuma.log ComputerCompetition2009 ThreePlayerHoldem akuma -attributes CLASS_TO_INVEST -invert“ oder
„richierich.log ComputerCompetition2012 ReverseBlindsThreePlayerHoldem RichieRich -attributes DRAWS,CLASS_TO_INVEST -invert“

### 3.6.3 Starten der Bots

**TrainingBot**

Aufruf der Klasse `PokerBotTrainer.machinelearning.pokerbotclients.TrainingBot` mit folgenden Parametern:

1. IP-Adresse des Servers (z.B. „localhost“ auf Windows-Systemen)
2. Hero-Name
3. Attribute-Liste (s. Abschnitt 3.6.1)

**TestingBot**

Aufruf der Klasse `PokerBotTrainer.machinelearning.pokerbotclients.TestingBot` mit folgenden Parametern:

1. IP-Adresse des Servers
2. Name des Bots
3. Klassifikator-Typ: {CLASSIFIER,NEIGHBOUR_SEARCH}. „CLASSIFIER“ ist für alle Modelle welche aus Klassen, die das Interface `weka.classifiers.Classifier` implementieren zu verwenden. „NEIGHBOUR_SEARCH“ initialisiert eine Instanz der Klasse `weka.core.neighboursearch.KDTree`, auf welcher dann eine Nächste-Nachbarn-Klassifikation vorgenommen werden kann.
4. Pfad zur Preflop-Modelldatei
5. Pfad zur Flop-Modelldatei
6. Pfad zur Turn-Modelldatei
7. Pfad zur River-Modelldatei
8. Klassenattribut
9. Pfad zur Preflop-Attributeliste (s. Abschnitt 3.6.1)
10. Pfad zur Flop-Attributeliste
11. Pfad zur Turn-Attributeliste
12. Pfad zur River-Attributeliste

| Wettrunde | Anzahl Attribute | Instanzen | Dateiname |
|---|---|---|---|
| Preflop | 55 | 4.943.471 | AkumaPreflopFixed.arff |
| Flop | 79 | 2.602.738 | AkumaFlopFixed.arff |
| Turn | 93 | 1.835.214 | AkumaTurnFixed.arff |
| River | 106 | 1.413.702 | AkumaRiverFixed.arff |

**Tabelle 1:** Ungefilterte Experimentierdaten für den Pokerbot Akuma

# 4 Datengrundlage der Experimente

## 4.1 Experimentierdaten

Um das Verhalten eines Pokeragenten durch Methoden des Maschinellen Lernens zu erlernen sind Spieldaten notwendig. Spieldaten können in Form von Server-Logs vorliegen (s. Abschnitt 3.5.3). Für eine Verarbeitung von Spieldaten im Weka-Machine-Learning-Framework sollen diese im ARFF vorliegen. Dazu müssen sie Server-Logs in das ARFF-Format konvertiert werden. Als primäre Quelle von Spieldaten fällt die Wahl auf Logdateien von Partien für drei Spieler, welche während der Annual-Computer-Poker-Competition ausgetragen werden. Gespielt wird Fixed-Limit-Texas-Hold'em. Die Wettkämpfe werden so ausgetragen, dass jeder Pokeragent gegen alle Kombinationen von Gegnern in jeder Position einmal antritt. Diese Vorgehensweise ermöglicht eine objektivere Bewertung der Spielstärke der Bots. Zusätzlich zu den Gegnern und Positionen werden die Handkarten mehrmals ausgetauscht. Dadurch entstehen sehr viele Datensätze. Eine eigene Generierung solcher Spieldaten in gleichem Umfang ist zeitintensiv. Außerdem sind die Daten der ACPC in Hinblick auf die Spielstärke der teilnehmenden Agenten interessant. Bei der Wahl des nachzuahmenden Poker-Bots fällt die Wahl auf den Bot „Akuma“. Akuma nimmt 2009 für die TU Darmstadt an der ACPC für drei-Spieler Limit-Hold'em teil. In der Wettkampf-Struktur „Runoff“ belegt Akuma den vierten Platz und in der Wettkampf-Struktur „Bankroll“ den dritten Platz (Acpc, 2009). Abgesehen vom guten Abschneiden sprechen noch die Verfügbarkeit und Komplexität des Bots für die Nominierung als Ziel der Untersuchungen. Akuma beherrscht Opponent-Modelling. Dies ist eine Technik um Gegnercharakteristika zu erfassen und eine Ausnutzungsstrategie von Schwächen des Gegners zu entwickeln oder bessere Abschätzungen der eigenen Gewinnwahrscheinlichkeiten zu erzielen, um weniger kostspielige Aktionen zu tätigen. Bevor Akuma Gegnerstatistiken ansammelt spielt er „mathematisch fair“. Mathematisch fair bedeutet in diesem Zusammenhang, dass zunächst keine Annahmen über den Gegner getroffen werden, sondern die Gewinnwahrscheinlichkeit gegen ein zufälliges Spektrum an möglichen Handkarten der Gegner berechnet wird. Tabelle 1 zeigt die Experimentierdaten. Rohdaten, aus welchen die Experimentierdaten erzeugt werden, sind 5256 Serverlogs von Spielpartien mit Akuma während der ACPC 2009 (Acpc, 2009). Für die in Abschnitt 5.2.2 beschriebene Kreuzvalidierung werden die Daten, wie in den Tabellen 2, 3, 4 und 5 gezeigt, in Trainings- und Testdaten aufgeteilt.

## 4.2 Ziel und Vorgehensweise

Ziel ist es das Verhalten eines Poker-Agenten mit einer guten Accuracy zu modellieren. Dazu werden zunächst die Server-Logdateien der ACPC von 2009 ausgewertet und in ARFF, unter Hinzunahme abstrahierter Attribute, umgewandelt. Liegen die ARFF vor, werden mit Hilfe des Weka-Machine-Learning-Frameworks Modelle berechnet. Klassenattribut ist das Attribut HERO_ACTION_TAKEN mit den Werten RAISE, FOLD, CALL und INVALID. Neben einer Betrachtung der Accuracy der Modelle werden diese auch Praxis-Tests unterzogen, bei welchen die Spielweise der Bots mit den generierten Modellen mit den Original-Bots verglichen wird. In Kapitel 5 werden zunächst Voruntersuchung zur Generierung von Klassifikationsmodellen behandelt. Kapitel 6 beschreibt Klassifikationsexperimente. Die Praxis-Tests in Form von Pokerspielpartien werden in Kapitel 7 gezeigt.

| Wettrunde | Verwendung | Dateiname | Dateinamen Rohdaten(Serverlogs) | Instanzen |
|---|---|---|---|---|
| Preflop | Training | aptrain | ring10* - ring69* | 4.045.841 |
| Preflop | Training | bptrain | ring0* - ring9* und ring20* - ring69* | 4.059.903 |
| Preflop | Training | cptrain | ring0* - ring19* und ring30* bis ring69* | 4.047.238 |
| Preflop | Training | dptrain | ring0* - ring29* und ring40* bis ring69* | 4.038.669 |
| Preflop | Training | eptrain | ring0* - ring39* und ring50* bis ring69* | 4.360.160 |
| Preflop | Training | fptrain | ring0* - ring49* und ring60* bis ring69* | 4.388.784 |
| Preflop | Training | gptrain | ring0* - ring59* | 4.387.374 |
| Preflop | Test | aptest | ring0* - ring9* | 842.203 |
| Preflop | Test | bptest | ring10* - ring19* | 828.141 |
| Preflop | Test | cptest | ring20* - ring29* | 840.806 |
| Preflop | Test | dptest | ring30* - ring39* | 849.375 |
| Preflop | Test | eptest | ring40* - ring49* | 527.884 |
| Preflop | Test | fptest | ring50* - ring59* | 499.260 |
| Preflop | Test | gptest | ring60* - ring69* | 500.670 |

**Tabelle 2:** Aufteilung der Preflop-Experimentierdaten nach Kartensets (ring0* bis ring69*)

| Wettrunde | Verwendung | Dateiname | Dateinamen Rohdaten(Serverlogs) | Instanzen |
|---|---|---|---|---|
| Flop | Training | aftrain | ring10* - ring69* | 2.131.564 |
| Flop | Training | bftrain | ring0* - ring9* und ring20* - ring69* | 2.139.748 |
| Flop | Training | cftrain | ring0* - ring19* und ring30* bis ring69* | 2.134.621 |
| Flop | Training | dftrain | ring0* - ring29* und ring40* bis ring69* | 2.131.400 |
| Flop | Training | eftrain | ring0* - ring39* und ring50* bis ring69* | 2.320.971 |
| Flop | Training | fftrain | ring0* - ring49* und ring60* bis ring69* | 2.338.657 |
| Flop | Training | gftrain | ring0* - ring59* | 2.340.956 |
| Flop | Test | aftest | ring0* - ring9* | 458.158 |
| Flop | Test | bftest | ring10* - ring19* | 449.974 |
| Flop | Test | cftest | ring20* - ring29* | 455.101 |
| Flop | Test | dftest | ring30* - ring39* | 458.322 |
| Flop | Test | eftest | ring40* - ring49* | 268.751 |
| Flop | Test | fftest | ring50* - ring59* | 251.065 |
| Flop | Test | gftest | ring60* - ring69* | 248.766 |

**Tabelle 3:** Aufteilung der Flop-Experimentierdaten nach Kartensets (ring0* bis ring69*)

| Wettrunde | Verwendung | Dateiname | Dateinamen Rohdaten(Serverlogs) | Instanzen |
|---|---|---|---|---|
| Turn | Training | attrain | ring10* - ring69* | 1.505.715 |
| Turn | Training | bttrain | ring0* - ring9* und ring20* - ring69* | 1.508.957 |
| Turn | Training | cttrain | ring0* - ring19* und ring30* bis ring69* | 1.505.052 |
| Turn | Training | dttrain | ring0* - ring29* und ring40* bis ring69* | 1.501.122 |
| Turn | Training | ettrain | ring0* - ring39* und ring50* bis ring69* | 1.636.176 |
| Turn | Training | fttrain | ring0* - ring49* und ring60* bis ring69* | 1.650.914 |
| Turn | Training | gttrain | ring0* - ring59* | 1.652.055 |
| Turn | Test | attest | ring0* - ring9* | 321.031 |
| Turn | Test | bttest | ring10* - ring19* | 317.789 |
| Turn | Test | cttest | ring20* - ring29* | 321.694 |
| Turn | Test | dttest | ring30* - ring39* | 325.624 |
| Turn | Test | ettest | ring40* - ring49* | 190.570 |
| Turn | Test | fttest | ring50* - ring59* | 175.832 |
| Turn | Test | gttest | ring60* - ring69* | 174.691 |

**Tabelle 4:** Aufteilung der Turn-Experimentierdaten nach Kartensets (ring0* bis ring69*)

| Wettrunde | Verwendung | Dateiname | Dateinamen Rohdaten(Serverlogs) | Instanzen |
|---|---|---|---|---|
| River | Training | artrain | ring10* - ring69* | 1.154.082 |
| River | Training | brtrain | ring0* - ring9* und ring20* - ring69* | 1.159.266 |
| River | Training | crtrain | ring0* - ring19* und ring30* bis ring69* | 1.157.497 |
| River | Training | drtrain | ring0* - ring29* und ring40* bis ring69* | 1.157.196 |
| River | Training | ertrain | ring0* - ring39* und ring50* bis ring69* | 1.257.436 |
| River | Training | frtrain | ring0* - ring49* und ring60* bis ring69* | 1.266.490 |
| River | Training | grtrain | ring0* - ring59* | 1.267.257 |
| River | Test | artest | ring0* - ring9* | 249.303 |
| River | Test | brtest | ring10* - ring19* | 244119 |
| River | Test | crtest | ring20* - ring29* | 245.888 |
| River | Test | drtest | ring30* - ring39* | 245.647 |
| River | Test | ertest | ring40* - ring49* | 145.955 |
| River | Test | frtest | ring50* - ring59* | 136.895 |
| River | Test | grtest | ring60* - ring69* | 136.128 |

**Tabelle 5:** Aufteilung der River-Experimentierdaten nach Kartensets (ring0* bis ring69*)

# 5 Voruntersuchung

## 5.1 Überanpassung

Es ist davon auszugehen, dass die Partien gegen Bots mit vergleichbarer Spielsärke ausgefochten werden. Deswegen sind interessantere Spieldaten zu erwarten, als bei Spielen gegen simple und leicht ausnutzbare Pokeragenten. Allerdings zeigt eine genauere Betrachtung der Daten genau das Gegenteil. Die von den Metriken PREFLOP_ACTION, FLOP_ACTION, TURN-ACTION und RIVER_ACTION (s. Attribute-Definition in Anhang A) erfassten Setzverhaltensmuster der Bots offenbaren, dass aus einer großen Auswahl an möglichen Spielzügen immer wieder nur die gleichen gespielt werden (s. Anhang C). Ursache hierfür ist, dass die Karten nicht bei jedem Match gegen neue getauscht werden, sondern, um die Bots besser vergleichen zu können, mehrmals mit den gleichen Karten gespielt wird. Ein weiterer Faktor ist die Ähnlichkeit der Spielweise vieler Bots, die dann für wenig Varianz in den beobachteten Spielzügen sorgt. In nicht wenigen Fällen werden exakt gleiche Setzverhaltenssequenzen durch vollkommen andere Bots erzeugt.

Dadurch, dass sehr lange auf vielen gleichen Informationen gelernt wird, kann es bei der Erzeugung eines Modells zur Überanpassung kommen. In einer frühen Voruntersuchung werden mit J48 Modelle für Akuma berechnet. Ein Blick auf die Modell-Entscheidungsbäume zeigt, dass den vielwertigen Attributen, die häufig in den Blättern stehen, Beobachtungen bzw. Trainingsbeispiele fehlen. Der Klassifizierer kann aus den wenigen beobachteten Fällen nicht stark genug generalisieren. Als Konsequenz daraus können sich schlechte Modelle ergeben. Ein Beispiel: Nach einem Preflop-Modell der Voruntersuchung wird die Hand 33 verworfen, die schwächere Hand 22 jedoch erhöht. Für die Hand 33 sind 36 beobachtete Fälle verzeichnet, für die Hand 22 sind es null beobachtete Fälle (s. Anhang B.1).

## 5.2 Gegenmaßnahmen

Um einer Überanpassung entgegenzuwirken können folgende Maßnahmen durchgeführt werden. Die Auswirkung dieser Maßnahmen auf die Modelle wird untersucht.

- Attribute-Auswahl
- Abstraktion von Attributen
- Kreuzvalidierung

### 5.2.1 Attribute-Auswahl und Abstraktion von Attributen

Vielwertige Attribute können ein großes Problem darstellen, wenn nicht genügend Lernbeispiele vorliegen (s. Abschnitt 5.1). In diesem Kapitel werden vielwertige Attribute beschrieben und unwichtige Attribute identifiziert. Zum Schluss des Kapitels wird diskutiert, wie Attribute durch Abstraktion stärker generalisierend gestaltet werden können.

**Entfernen von vielwertigen Attributen**

Kartenattribute und Setzverhaltensequenz-Attribute besitzen sehr viele mögliche Werte. Das Attribut für die Handkarten HERO_HOLE_CARDS besitzt 1327 unterschiedliche Werte. Dieses Attribut beschreibt die Ränge und die Farben beider Handkarten. Es existieren parallel dazu weitere Handkarten-Attribute mit unterschiedlicher Darstellung der Informationen. Die gleiche Information kann durch ein Quartett an Attributen, aufgespalten nach Rang und Farbe, dargestellt werden. Attribute, welche das Setzverhalten beschreiben sind PREFLOP_ACTION mit 178 möglichen Werten und FLOP_ACTION, TURN_ACTION, RIVER_ACTION und CURRENT_STREET_ACTION mit 370 möglichen Werten. Auch diese können alternativ durch Aufspaltung in einzelne numerische Attribute, welche die Anzahl der jeweiligen Aktion eines Spielers erfassen, dargestellt werden.

### Unwichtige Attribute

Um unwichtige Attribute zu identifizieren werden maschinelle Attribute-Auswahlen durchgeführt. Angewendet werden dazu folgende Methoden des Weka-Frameworks auf die Datensätze gezeigt in Tabelle 1. Bei der Auswahl der Verfahren steht die Ausführbarkeit und Laufzeit auf größeren Datenmengen im Vordergrund. Ziel ist es eine erste Einschätzung der Wichtigkeit der Attribute zu erhalten:

- `weka.attributeSelection.CfsSubsetEval` mit Parametern:
  1) -M -s „ `weka.attributeSelection.BestFirst` -P 1 -D 2 -N 5 -S 2“
  2) -M -s „`weka.attributeSelection.GreedyStepwise` -T -1.797631348623157E308 -N -1“
  Bei CfsSubsetEval wird die Korrelation eines Sets aus Attributen zum Klassenattribut bewertet. Attribute-Sets mit einer schwachen Korrelation untereinander werden dabei vorgezogen und als geeigneter bewertet. Bei der Suche von Teilmengen (Subsets) können verschiedene Suchmethoden verwendet werden. Die Suchmethoden „BestFirst“ und „GreedyStepwise“ sind Bergsteigeralgorithmen. BestFirst verwendet Backtracking, GreedyStepwise hingegen nicht (Witten und Frank, 2005, Kap. 10.8).

- `weka.attributeSelection.SymmetricalUncertAttributeEval` mit Parameter:
  -M -s „`weka.attributeSelection.Ranker` -T -1.7976931348623157E308 -N -1“

- `weka.attributeSelection.InfoGainAttributeEval` mit Parameter:
  -s „`weka.attributeSelection.Ranker` -T -1.7976931348623157E308 -N -1“
  Die Evaluationsmethoden SymmetricalUncertAttributeEval und InfoGrainAttributeEval bewerten keine Teilsets, sondern erstellen ein Ranking aller Attribute anhand von verschiedener möglicher Metriken. „Symmetrical Uncertainty“ ist eine solche Metrik und beschreibt die Korrelation zweier nominaler Attribute (Witten und Frank, 2005, Kap. 7.1, Kap. 10.8).

Tabelle 6 zeigt Attribute, welche eine häufige Nennung erfahren oder eine hohe Wertung erhalten.

Es zeigt sich, dass Handkarten-Attribute für die Preflop-Phase wichtig sind, jedoch in den Postflop-Wettrunden an Bedeutung verlieren. Setzverhaltenssequenzen für die aktuelle und vorangegangene Wettrunden sind wichtig und werden mehrfach genannt. Die häufige Erwähnung der Merkmale, die von den „rohen“ Setzverhaltenssequenzen eine Abstraktion vornehmen, ist ein Indiz dafür, dass die relevante Information dabei nicht verlorengeht. Es kann eine Reduktion durch Auswahl eines einzigen, vorzugsweise abstrahierten, Setzverhaltenssequenz-Merkmals erreicht werden. Weiterhin sind Pot-bezogene Größen wie Pot-Odds und das Investment wichtig. Ebenfalls wichtig sind Merkmale über die Gewinnwahrscheinlichkeit. Das Attribut POT_ODDS_TO_ALL_IN_EQUITY_RATIO, welches das Verhältnis der Pot-Odds zur All-In-Equity ausdrückt ist sehr ausdrucksstark und wird entsprechend häufig hoch bewertet. Die Attribute POT_SIZE und ALL_IN_EQUITY sind als Teilattribute der Ratio anzusehen. Im Abschnitt 6.3.2 wird untersucht, wie der Verzicht dieser Teil- oder Basisattribute sich auf die Qualität der Modelle auswirkt. Alle Attribute, die in der Tabelle keine Erwähnung erfahren sind weniger bedeutend. Es handelt sich bei diesen vor allem um Gegner-Statistiken. Diese Werte werden über den Verlauf eines Matches mit mehreren gespielten Partien erhoben. Daher sind diese Werte zu Beginn eines Matches noch nicht besonders aussagekräftig. Da bei der ACPC während eines Matches nur jeweils eintausend Hände gespielt werden, sind für viele Merkmale nur ungenügend Beispiele vorhanden, sodass diese statistisch gesehen nicht „reifen“. Daher werden all diese Attribute für alle folgende Experimente gefiltert. Tabelle 7 listet diese auf.

### Abstrahierung

Abstrahierung kommt als weitere Maßnahme gegen Überanpassung zum Einsatz. Die Setzverhaltens-Attribute und die Handkarten-Attribute werden beibehalten, die Anzahl der möglichen Werte werden jedoch, durch stärkere Abstraktion reduziert.

| Attribute-Selektionen nach Wettrunde, aufgeteilt nach Kategorien | | | |
|---|---|---|---|
| | Setzverhalten | Handkarten | Investment/Equity/Position |
| Preflop | preflopAction, preflopActionAbstr, currentStreetAction, currentStreetActionAbstr, preflopActionAbstrAll | heroHoleCardHigh, heroHoleCardLow, abstractedHoleCards, heroHoleCards | potSize, allInEquity2Players, potOddsToAllInEquityRatio, heroInvestedThisStreet, investedThisStreet, potOdds, allInEquity, heroPosition |
| Flop | flopRaises, flopAction, currentStreetAction, currentStreetActionAbstrAll, flopActionAbstr | handType, potentialOuts | maxBetSizeThisStreet, allInEquity2Players, potOddsToAllInEquityRatio, investedThisStreet, allInEquity, potOdds, callAmount, potSize, investedThisStreetOpponent0, investedThisStreetOpponent1, lastRaisePosition |
| Turn | flopRaises, turnRaises, turnAction, currentStreetAction, turnActionAbstrAll, currentStreetActionAbstr, turnActionAbstr, preflopActionAbstr, flopActionAbstr, flopAction | | maxBetSizeThisStreet, allInEquity, potOddsToAllInEquityRatio, investedThisStreet, callAmount, potOdds, investedThisStreetOpponent0, investedThisStreetOpponent1, activePlayer |
| River | riverAction, currentStreetAction, riverActionAbstrAll, riverActionAbstr | | maxBetSizeThisStreet, potOddsToAllInEquityRatio, investedThisStreet, potOdds, callAmount, heroInvestedThisStreet, investedThisStreetOpponent0, investedThisStreetOpponent1 |

**Tabelle 6:** Zusammengefasste Ergebnisse der Attribute-Selektionen nach Wettrunde: Wichtige Attribute

| | |
|---|---|
| Preflop | historyVpipOpponent0, historyVpipOpponent1, historyPreflopRaiseOpponent0, historyPreflopRaiseOpponent1, historyW$sdOpponent0, historyW$sdOpponent1, historyWtsdOpponent0, historyWtsdOpponent1, historyFoldToBetOpponent0, historyFoldToBetOpponent1, historyWinPercentageOpponent0, historyWinPercentageOpponent1, raisesPreflopOpp0, raisesPreflopOpp1, callsPreflopOpp0, callsPreflopOpp1, raisesPreflopHero, callsPreflopHero, foldsPreflopHero |
| Flop | Für Preflop aufgelistete Attribute und zusätzlich diese: raisesFlopOpp0, raisesFlopOpp1, callsFlopOpp0, callsFlopOpp1, raisesFlopHero, callsFlopHero, foldsFlopHero |
| Turn | Für Preflop und Flop aufgelistete Attribute und zusätzlich diese: raisesTurnOpp0, raisesTurnOpp1, callsTurnOpp0, callsTurnOpp1, raisesTurnHero, callsTurnHero, foldsTurnHero |
| River | Für Preflop, Flop und Turn aufgelistete Attribute und zusätzlich diese: raisesRiverOpp0, raisesRiverOpp1, callsRiverOpp0, callsRiverOpp1, raisesRiverHero, callsRiverHero, foldsRiverHero |

**Tabelle 7:** Unwichtige Attribute

Anstatt jede mögliche Setzsequenz als Wert zu verwenden, wird stattdessen gezählt, wie häufig während einer Wettrunde Erhöhungen des Einsatzes durch Gegner gemacht werden. Es existieren anstatt der zuvor 178 Preflop- bzw. 370 Postflop-Werte nur noch sechs Werte (s. Anhang A.7). Die Attribute, welche die abstrahierte Setzverhaltenssequenzen der Gegner widerspiegeln sind PREFLOP_ACTION_ABSTR_ALL, FLOP_ACTION_ABSTR_ALL, TURN_ACTION_ABSTR_ALL und RIVER_ACTION_ABSTR_ALL. Bei der Abbildung der Werte geht die Information, welcher Spieler auf welche Weise reagiert, verloren. Allerdings ist diese Information redundant, wenn zusätzliche Attribute, welche die Anzahl der jeweiligen Aktionen pro Spieler mitzählen, ebenfalls berechnet werden.

Für das im Abschnitt 5.2.1 beschriebene Attribut für die Handkarten HERO_HOLE_CARDS existieren einige alternative, weniger vielwertige, Darstellungen. In Abschnitt 6.3.1 wird untersucht welche der Darstellungen sich am besten eignen.

**Ergebnisse**

Folgende Ergebnisse der Untersuchungen aus den Abschnitten 6.3.1, 6.3.2 und 6.4 sind vorweggenommen. Auswertungen von J48-Modellbäumen haben gezeigt, dass für die Darstellung der Handkarten-Informationen in Hinsicht auf Blätteranzahl das Duo aus den Attributen HERO_HOLE_CARDS_HIGH und HERO_HOLE_CARDS_LOW, welche lediglich die Ränge der beiden Handkarten abbilden, sich als effizienteste Darstellung erwiesen. Die Attribute-Auswahl hilft, in unserem Fall, die Anzahl der verwendeten Attribute drastisch zu senken und somit auch die Modelle zu vereinfachen. Eine signifikante Verbesserung der Modelle durch Identifikation und Auswahl von Attribute mit gleicher Information, aber unterschiedlicher Präsentation, kann nicht festgestellt werden. Anscheinend verwendet J48 bei redundanten Informationen von sich aus die am besten geeignetsten Darstellungen der Informationen.

### 5.2.2 Siebenfache Kreuzvalidierung

Als weitere Maßnahme gegen Überanpassungen kann eine Kreuzvalidierung angewendet werden. Verwendet wird eine spezielle Kreuzvalidierung um eine saubere Trennung von Training- und Testdaten sicherzustellen. Bei den ACPC 2009 spielt der Pokeragent Akuma in einem Match mit den gleichen Handkarten mehrmals gegen die gleichen Gegner. Die Position wird dabei lediglich rotiert und die Gegner alle sechs Rotationen bzw. Permutationen abgewechselt, sodass gegen jede Kombination der Positionen der Bots gespielt wird. Insgesamt werden 70 Sets aus Handkarten gespielt. Um eine Überanpassung zu vermeiden, ist es notwendig bei der Konvertierung darauf zu achten, dass diese Sets nicht durchmischt werden. Denn dadurch wird sichergestellt, dass in den Trainingsdaten keine Lösungen für Testdaten enthalten sind und dadurch die Accuracy nicht verfälscht wird. Die 70 Kartensets werden in sieben Teil-Datensätze eingeteilt. Dann wird eine Kreuzvalidierung durchgeführt. Dabei dient stets ein Datensatz zum Testen und die Konkatenation der restlichen Teil-Datensätze als Trainingsbeispiele. Anschließend werden die Konfusionsmatritzen der Evaluation der Teildaten aufsummiert, um die durchschnittlichen Werte zu bestimmen. Tabellen 2, 3, 4 und 5 zeigen die Aufteilung der Daten anhand der Kartensets in Trainings- und Testdaten.

## 6 Klassifikationsexperimente

### 6.1 Vergleich der Klassifikatoren

Es wird der Fragestellung nachgegangen, welcher Weka-Klassifikator die besten Ergebnisse in Hinsicht der Accuracy liefert. Dazu werden Modelle mit den fünf Klassifizierern J48, JRip, Naive-Bayes, Nächste Nachbarnsuche mit k=5 und SMO (s. Abschnitt 3.3) gelernt und verglichen. Die Wahl fällt beim kNN-Algorithmus deshalb auf 5, weil bei k=10 die Ausführungszeit bereits deutlich ansteigt. Es werden stets die Standardeinstellungen der Lernverfahren verwendet. Diese sind:

- J48: C=0.25, M=2, N=3, Q=1
- JRip: F=3, N=2.0, O=2, D=false, S=1, E, P

- SMO: C=1, N=0, L=1.0e-3, P=1.0e-12, V=-1, W=1,
  K=weka.classifiers.functions.supportVector.PolyKernel wiederum mit folgenden Werten: C=250007, E=1.0 (Option D, no-checks und L nicht aktiviert).

Zweck der Untersuchung ist es einen geeigneten Klassifikator für weitere Experimente auszusuchen. Während der Ausführung dieser Untersuchung zeigen sich bereits einige Schwächen. Alle Klassifikatoren außer J48 haben einen zu großen Speicherverbrauch und sehr lange Trainingszeiten. Aus diesem Grund werden die Trainings- und Testdatensätze je nach Effizienz des Klassifikators verkleinert (s. Tabelle 9). Für SMO liegen für die Turn-Wettrunde wegen Speichermangel keine Resultate vor.

| Gegenüberstellung der Klassifikatoren | | | | | |
|---|---|---|---|---|---|
| **Accuracy in %** | 5NN | J48 | JRip | Naive Bayes | SMO |
| Preflop | 99,7 | 99,94 | 99,38 | 85,89 | 99,89 |
| Flop | 85,45 | 97,32 | 94,55 | 83,2 | 91,21 |
| Turn | 90,44 | 95,4 | 95,35 | 83,07 | fehlt |
| River | 93,14 | 96,2 | 96,31 | 87,2 | 94,65 |

**Tabelle 8:** Gegenüberstellung der Klassifikatoren

| Prozentsatz der verwendeten Trainings- und Testdaten | | | | | |
|---|---|---|---|---|---|
| **Teilmenge d. Daten in %** | 5NN | J48 | JRip | Naive Bayes | SMO |
| Preflop | 1 | 10 | 1 | 10 | 1 |
| Flop | 1 | 10 | 1 | 10 | 0,5 |
| Turn | 10 | 10 | 10 | 10 | (1) |
| River | 10 | 10 | 10 | 10 | 1 |

**Tabelle 9:** Verwendete Prozentsätze der Daten nach Lernalgorithmus

**Wahl des Lernverfahrens und Attribute-Auswahl für alle folgenden Experimente**

Der J48-Klassifikator liefert meistens die besten Accuracy-Werte (s. Tabelle 8). Auch kommt er mit großen Datensätzen zurecht und die Trainingszeiten sind viel kürzer als beispielsweise JRip, Nächste Nachbarnsuche, SMO und Naive-Bayes. Aus diesen Gründen beschränken sich weitere Untersuchungen auf den J48-Klassifikator. Wenn nicht anders angegeben, wird das Lernverfahren mit den in diesem Abschnitt aufgeführten Standard-Einstellungen gestartet. Es werden die Datensätze gezeigt in den Tabellen 2, 3, 4 und 5 verwendet. Unwichtige Attribute, gezeigt in Tabelle 7, werden gefiltert.

## 6.2 Accuracy in Abhängigkeit der Anzahl von Trainingsbeispiel-Instanzen

Es wird überpruft, wie sich die Anzahl der Trainingsbeispiele auf die Accuracy und die Größe der Modelle auswirkt. Dazu wird der J48-Klassifizierer auf ein Prozent, einem Zehntel und einhundert Prozent der Daten trainiert. Für eine Kreuzvalidierung werden die Daten in sieben Teile aufgespalten. Durchschnittlich gibt es 4.189.709 Trainingsinstanzen für die Preflop-Phase, 2.219.711 Instanzen für die Flop-Phase, 1.565.713 Instanzen für die Turn-Phase und 1.202.745 Instanzen für die River-Phase. Es werden jeweils 1 Prozent, 10 Prozent und 100 Prozent der Trainings-Daten (s. Tabellen 2, 3, 4 und 5) verwendet um Vergleichsmodelle zu erzeugen. Evaluiert wird mit einer siebenfachen Kreuzvalidierung (s. Abschnitt 5.2.2) auf 100 Prozent der Testdaten.

Die Kurve für die Preflop-Accuracy (s. Abbildung 4) ist bereits bei wenigen Trainingsinstanzen nahe einhundert Prozent. Sie flacht bereits ab ca. 418.000 Instanzen ab. Die Postflop-Modelle sind etwas ferner der Hundert-Prozent-Marke. Hier ist jedoch außer bei den Turn-Modellen keine Abflachung der

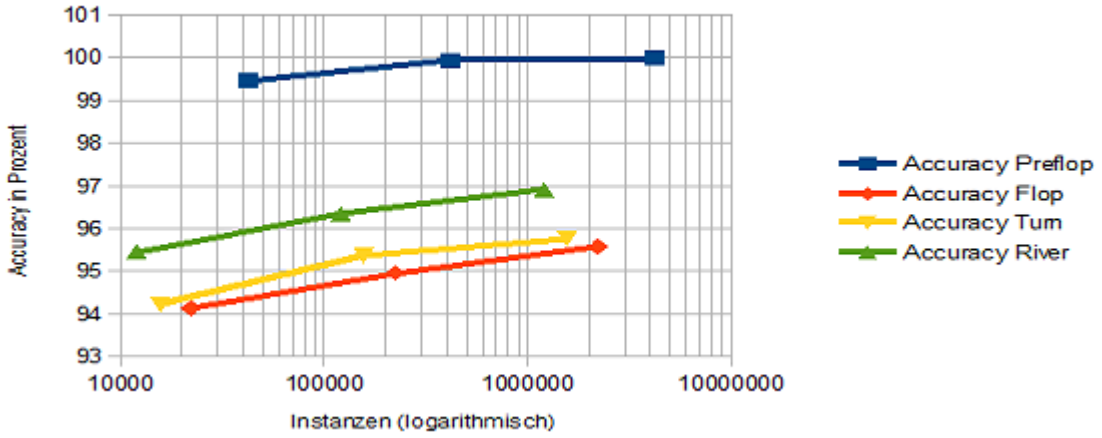


**Abbildung 3:** Accuracy der J48-Akuma-Modelle über Anzahl der Trainingsinstanzen

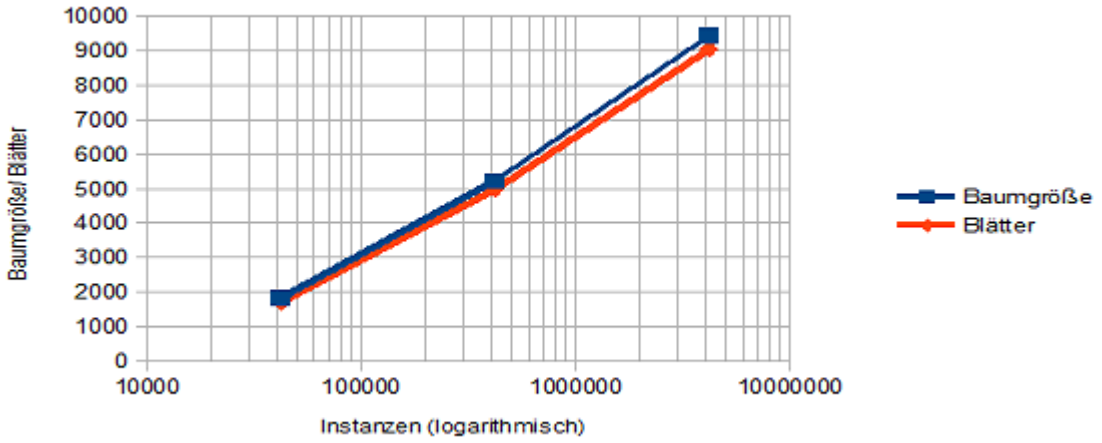


**Abbildung 4:** Baumgröße und Blätterzahl der J48-Entscheidungsmodellbäume über Anzahl der Preflop-Trainingsinstanzen

Kurve festzustellen. Es ist daher anzunehmen, dass durch eine Erhöhung der Trainingsbeispiele hier die Accuracy noch weiter verbessert werden kann. Zwar verbessert sich mit wachsender Anzahl an Trainingsbeispielen die Accuracy stetig, jedoch explodiert zugleich die Modellgröße, wie auf den Abbildungen 4, 5, 6 und 7 zu sehen ist. Bei einem Einsatz mit begrenzten Mitteln, wie beispielsweise während eines Pokerbot-Wettkampfes, kann die Übergröße einiger Modelle hinderlich sein. In Abschnitt 6.4 wird eine Modellvereinfachung vorgestellt.

## 6.3 Geeignete Attribute-Darstellungen finden

Es soll herausgefunden werden, welche Darstellung von Spielinformationen als Weka-Attribute am besten für J48 geeignet sind. Als Performanzmaß wird die Accuracy des jeweiligen Modells des Akuma-Bots verwendet.

### 6.3.1 Hero-Hole-Cards

Folgende Kombinationen von Attributen konkurrieren gegeneinander. Eine Beschreibung der Attribute findet sich in Anhang A.

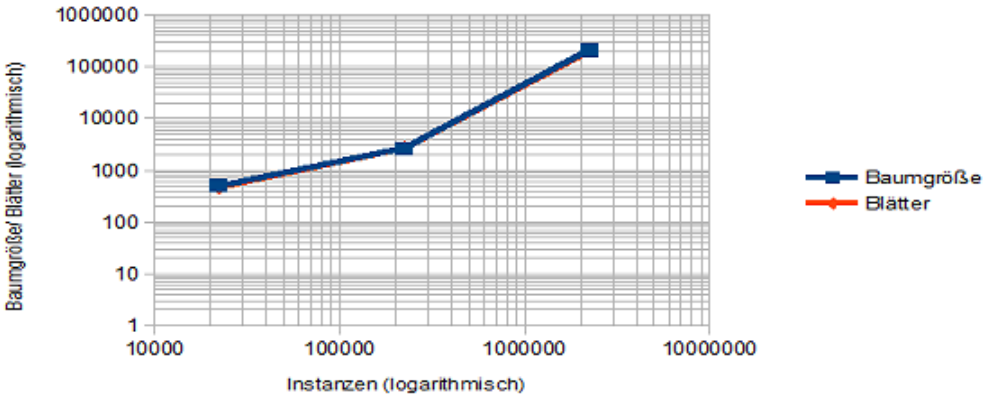


**Abbildung 5:** Baumgröße und Blätterzahl der J48-Entscheidungsmodellbäume über Anzahl der Flop-Trainingsinstanzen

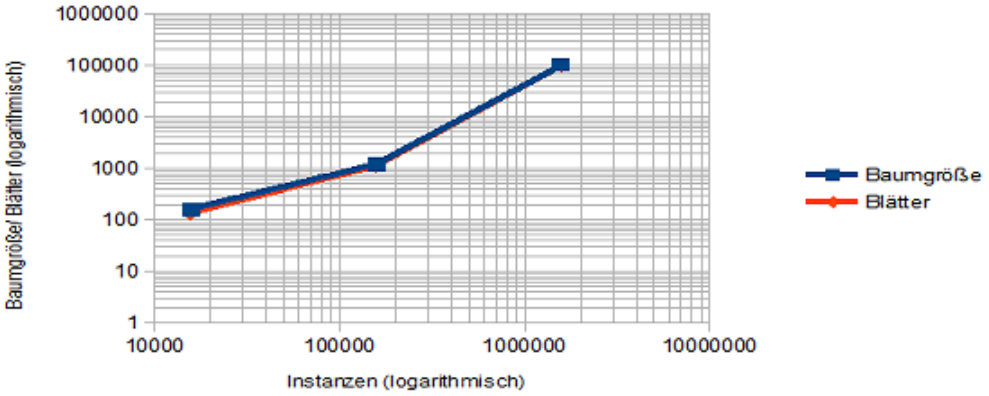


**Abbildung 6:** Baumgröße und Blätterzahl der J48-Entscheidungsmodellbäume über Anzahl der Turn-Trainingsinstanzen

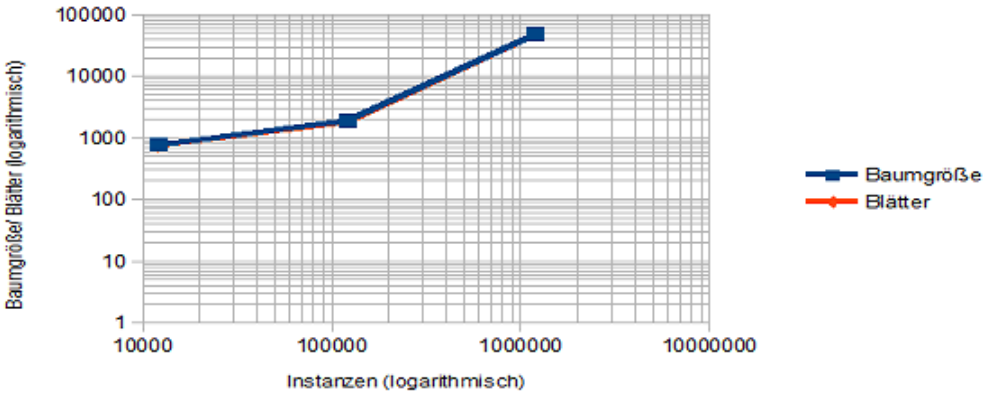


**Abbildung 7:** Baumgröße und Blätterzahl der J48-Entscheidungsmodellbäume über Anzahl der River-Trainingsinstanzen

**a)** HERO_HOLE_CARDS: Entält Informationen über den Rang und Farbe beider Handkarten.

**b)** ABSTRACTED_HOLE_CARDS (Rang beider Handkarten ohne Farbe, 92 nominale Werte) mit dem Duo HERO_HOLE_CARDS_HIGH_SUIT und HERO_HOLE_CARDS_LOW_SUIT (die Farbe beider Handkarten, 4 nominale Werte).

**c)** HERO_HOLE_CARD_HIGH(der höhere Rang der beiden Karten, 13 nominale Werte), HERO_HOLE_CARD_LOW(der niedrigere Rang der beiden Karten, 13 nominal Werte) mit dem Duo HERO_HOLE_CARDS_HIGH_SUIT und HERO_HOLE_CARDS_LOW_SUIT.

Die Ergebnisse dieser Untersuchung sind unter Vorbehalt zu betrachten, denn sie wurde in einem frühen Experiment auf einem unvollständigen Datensatz durchgeführt. Sowohl Preflop als auch Postflop ist die Darstellung c) am besten geeignet. Die vier Kartenfarben spielen noch keine Rolle in der Preflop-Wettrunde. Lediglich die Information, ob die Handkarten gleichfarbig sind, ist von Bedeutung.

### 6.3.2 Das Attribut POT_ODDS_TO_ALL_IN_EQUITY_RATIO

Mit dem Attribut POT_ODDS_TO_ALL_IN_EQUITY_RATIO wird das Verhältnis der Pot-Odds zur All-In-Equity erfasst. Es wird untersucht wie sich das Weglassen der Attribute, aus welchen sich dieses Verhältnis berechnet, für resultierende Modelle auswirkt. Dazu werden folgende Attribute aus den Datensätzen entfernt und Modelle mit dem J48-Klassifikator für alle Wettrunden berechnet: maxBetSizeThisStreet, potSize, investedThisStreet, callAmount, heroInvestedThisStreet, investedThisStreetOpponent0, investedThisStreetOpponent1, allInEquity, allInEquity2Players, potOdds. Die Modelle werden auf 50 Prozent der Trainingsdaten (s. Tabellen 2, 3, 4 und 5) berechnet. Evaluiert wird mit einer siebenfachen Kreuzvalidierung (s. Abschnitt 5.2.2) auf 100 Prozent der Testdaten.

| Wettrunde | Accuracy ohne Basisattribute in % | Baumgröße | Blätter |
|---|---|---|---|
| Preflop | 99,99 | 11051,9 | 10698,7 |
| Flop | 94,01 | 269705 | 266437 |
| Turn | 94,45 | 101369,7 | 99775,5 |
| River | 95,5 | 21980,3 | 21566,7 |
| | Accuracy mit Basisattributen in % | | |
| Preflop | 99,98 | 7757,3 | 7409,4 |
| Flop | 95,5 | 366956 | 358237 |
| Turn | 95,7 | 29776,6 | 28587 |
| River | 96,82 | 12544 | 12216 |

**Tabelle 10:** Vergleich der Accuracy, Baumgröße und Blätterzahl der Modelle jeweils mit und ohne der Basisattribute von Ratio

Durch das Weglassen der Basisattribute wird außer für die Preflop-Phase keine Verbesserung der Accuracy erreicht. Allerdings bricht durch den Wegfall diese nicht dramatisch ein. Die resultierende Modelle ohne Basisattribute sind mit Ausnahme des Flop-Modells in ihrer Baumgröße und Blätterzahl kleiner. Das Weglassen der Basisattribute hat keinen Vorteil. Allerdings ist der Verzicht auf diese nicht besonders schmerzhaft. Wenn der Rechenzeit für die Generierung der Modelle Grenzen gesetzt sind, kann der durchschnittliche Verlust von etwa einem Prozentpunkt Accuracy in Kauf genommen werden.

## 6.4 Modellvereinfachung

In Abschnitt 5.2.1 werden als mögliche Vereinfachungen der Modelle eine Abstraktion von vielwertigen Attributen und eine Attribute-Auswahl zur Identifizierung unwichtiger Merkmale beschrieben. Der Abschnitt 6.3.2 beschreibt, wie sich durch das Weglassen von bestimmten Basisattributen des Attributes

POT_ODDS_TO_ALL_IN_EQUITY_RATIO die Anzahl der zu verwendenden Merkmale reduzieren lässt. Diese Erkenntnisse sollen nun Anwendung finden. Ziel ist es, eine Vereinfachung der Modelle ohne, oder mit nur leichten, Einbußen der Genauigkeit zu erreichen. Dazu werden die als unwichtig identifizierten Attribute aus dem Trainingsdatensatz entfernt (s. Tabelle 7). Statt den vielwertigen Vertretern der Setzverhaltenssequenzen werden abstrahierte Merkmale verwendet. Handkartenattribute werden vollständig entfernt, da bereits bekannt ist, dass diese insgesamt bei der Klassifizierung eine untergeordnete Rolle spielen (s. Tabelle 6). Allerdings wird das Attribut HAND_TYPE beibehalten (s. Anhang A). Es werden auch auf die Basisattribute von POT_ODDS_TO_ALL_IN_EQUITY_RATIO verzichtet. Tabelle 11 zeigt die Zusammenstellungen von Attributen nach Wettrunden. Tabelle 12 zeigt den Vergleich der vereinfachten Modelle zu Modellen, die unter Einbezug aller Attribute berechnet werden. Trainiert wird auf 100 Prozent der Trainingsdaten (s. Tabellen 2, 3, 4 und 5). Evaluiert wird mit einer siebenfachen Kreuzvalidierung (s. Abschnitt 5.2.2) auf 100 Prozent der Testdaten.

**Ergebnis**

Durch das Fokussieren auf wenige Attribute können die Baumgrößen und die Blätterzahl der resultierenden Modelle stark verkleinert werden. Allerdings sind dafür ein bis zwei Prozentpunkte Verlust der Accuracy in den Preflop-, Flop- und Rivermodellen festzustellen. Einzige Ausnahme ist das vereinfachte Turnmodell, welches nur geringe Verluste bei der Accuracy hinnehmen muss. Beachtenswert bei diesem Modell ist auch, dass es gegen den Trend der zunehmenden Komplexität der Modelle mit Fortschreiten der Wettrunden, kleiner ist als das vereinfachte Flop-Modell. Zwar kann eine Verkleinerung der Modelle erreicht werden, allerdings haben diese eine schlechtere Accuracy, sodass diese für den Einsatz in der Praxis nicht sehr empfehlenswert sind. Die Modelle sind auch nicht stark genug verkleinert, um effektiv händisch analysiert werden zu können. Im Abschnitt 6.2 wird festgestellt, dass kleinere Trainingsdatensätze zu kleineren Modellen führen. Als weiteren Schritt für eine Vereinfachung der Modelle zur manuellen Analyse ist daher zusätzlich eine Reduktion der Trainingsdaten zu empfehlen.

| Wettrunde | Attribute |
|---|---|
| Preflop | activePlayerCount, preflopActionAbstrAll, classHeroActionTaken, allInEquity, potOddsToAllInEquityRatio |
| Flop | activePlayerCount, preflopActionAbstrAll, flopActionAbstrAll, classHeroActionTaken, handType, allInEquity, potOddsToAllInEquityRatio |
| Turn | activePlayerCount, maxBetSizeThisStreet, investedThisStreet, preflopActionAbstrAll, flopActionAbstrAll, turnActionAbstrAll, classHeroActionTaken, handType, allInEquity, allInEquity2Players, potOddsToAllInEquityRatio |
| River | activePlayerCount, preflopActionAbstrAll, flopActionAbstrAll, turnActionAbstrAll, riverActionAbstrAll, classHeroActionTaken, handTye, allInEquity, potOddsToAllInEquityRatio |

**Tabelle 11:** Zur Vereinfachung der Modelle ausgewählte Attribute-Zusammenstellungen aufgeteilt nach Wettrunden.

## 6.5 Modelle der Praxistests

Die Modelle für die Praxistest werden für Akuma mit J48 auf 50 Prozent der Daten (s. Tabelle 1), unter Verwendung aller Attribute, mit Ausnahme von Attributen gezeigt in Tabelle 7, berechnet.

In den Praxistests wird ein weiterer Bot, TunedFixedRichieRich (s. Abschnitt 7.1), untersucht. Die Modelle für diesen Bot werden auf folgenden Daten unter Verwendung aller Attribute, mit Ausnahme der Attribute, die in Tabelle 7 gezeigt werden, berechnet. Tabelle 13 zeigt die verwendeten Datensätze. Das Preflopmodell wird auf 66 Prozent der Daten, das Flopmodell auf 36 Prozent, das Turnmodell auf 41 Prozent und das Rivermodell auf 52 Prozent der Daten trainiert.

| **Modelle mit allen Attributen** | | | | |
|---|---|---|---|---|
| Wettrunde | Trainingsinstanzen | Accuracy in % | Baumgröße | Blätterzahl |
| Preflop | 4189709 | 99,99 | 9441,7 | 9036,2 |
| Flop | 2219711 | 95,58 | 207149,2 | 202450,6 |
| Turn | 1565713 | 95,77 | 100913,86 | 97693,57 |
| River | 1202745 | 96,91 | 48645,72 | 47709,57 |
| **Vereinfachte Modelle** | | | | |
| Preflop | 4189709 | 97.19 | 665 | 355,14 |
| Flop | 2219711 | 93.56 | 3632,2 | 2393,86 |
| Turn | 1565713 | 95.32 | 2384,57 | 1542,43 |
| River | 1202745 | 94.82 | 12880 | 8664,43 |

**Tabelle 12:** Vergleich der vereinfachten Modelle zu den Referenzmodellen mit allen Attributen.

| Wettrunde | Instanzen | Dateiname |
|---|---|---|
| Preflop | 25.433 | TFRRPreflop.arff |
| Flop | 19.907 | TFRRFlop.arff |
| Turn | 18.885 | TFRRTurn.arff |
| River | 17.294 | TFFRiver.arff |

**Tabelle 13:** TunedFixedRichieRich-Datensätze

## 7 Spielpartien

In diesem Abschnitt wird die Praxistauglichkeit der zuvor beschriebenen Methoden zur Modellierung der Spielweise von Pokerbots getestet. Dazu werden Spielpartien (Matches) ausgetragen. Es wird mit den gelernten Modellen gespielt. Die Übereinstimmung der Spielweise wird mit den Originalen verglichen. Dies geschieht über die Betrachtung der Verhaltensmuster durch den im Poker-TUD-Framework enthaltenen Tools `GameAnalyszer`. Dieses Tool erlaubt zusätzlich eine Veranschaulichung einer Partie als Gewinn/Verlust-Kurve. Betrachtet werden zwei Pokerbots: Der im Poker-TUD-Framework enthaltene Agent *TunedFixedRichieRich* (s. Abschnitt 7.1). Dieser Bot hat eine relativ einfache Implementierung. Er spielt nach einer Form der Pot-Odds (s. Abschnitt 2.9) und verwendet kein Opponent-Modelling. Das bedeutet, dass bei er bei der Berechnung seiner Gewinnwahrscheinlichkeiten keine Annahmen über die möglichen Hände seiner Gegner fällt, um somit seine Vorhersage zu verbessern. Stattdessen geht er von einer rein zufälligen Verteilung von Gegnerhänden aus. Der zweite Bot, der hier betrachtet wird, ist der bereits zuvor untersuchte Bot *Akuma*. Akuma ist komplexer als TunedFixedRichieRich. Er verwendet Opponent-Modelling. Je länger Akuma gegen die gleichen Gegner spielt, umso besser kann er seine Gewinnwahrscheinlichkeit berechnen, denn er zieht Rückschlüsse aus beobachteten Partien auf die Stärke seiner Gegner.

Die Spielpartien werden wie folgt ausgetragen. Es werden eintausend Hände, die zuvor festgelegt werden, gespielt. Nach tausend Händen wird die Position rotiert. Dies wird solange wiederholt, bis jeder Bot einmal von jeder Position beginnt. Da als Gegner zwei identische Bots ausgewählt werden, verringert sich die Zahl der zu vergleichende Kombinationen von Anfangspositionen von sechs auf drei.

### 7.1 TunedFixedRichieRich

Im Folgenden ein Code-Auszug der Entscheidungsfindung von TunedFixedRichieRich:

```
1:double equity = getAllInEquity();
```

```
2:if(equity * (double) activePlayerCount > 1.0) raise();
3:else if(equity * potSize > callAmount + 0.01 ||
callAmount == 0) call();
4:else fold();
```

Die Variablen activePlayerCount speichert die Anzahl der Spieler, die noch aktiv am aktuellen Spiel beteiligt sind. Die Variable „potSize“ steht für den Betrag, der insgesamt bereits gesetzt wurde. In „potSize“ sind also auch die zuvor getätigte Einsätze der aktuellen Wettrunde mit eingerechnet. callAmount gibt den Betrag an, den ein Spieler zum Mitgehen zu leisten hat.

Sind noch drei Spieler aktiv wird bei einer Gewinnwahrscheinlichkeit bzw. Equity von mindestens einem Drittel eine Erhöhung getätigt. Bei nur zwei verbleibenden Spielern muss für eine Erhöhung mindestens eine Gewinnwahrscheinlichkeit von 0,5 gegeben sein. Bei der Entscheidung, ob erhöht werden soll, spielt callAmount noch keine Rolle. Nun eine Betrachtung der Fallunterscheidung, ob statt zu erhöhen bloß  mitgeganen werden soll (s. Zeile 3 des Code-Auszugs): Der zweite Teil der Fallunterscheidung

$$callAmount == 0$$

ist trivial. Wenn der Betrag, den man zu leisten hat um mitzugehen gleich null ist, soll mitgegangen werden. Der erste Teil

$$equity * potSize > callAmount + 0.01$$

wird vereinfacht auf

$$equity * potSize > callAmount$$

und umgeformt, sodass nach equity aufgelöst wird:

$$equity > callAmount/potSize \tag{1}$$

Die Formel zur Berechnung der Pot-Odds lautet (s. Abschnitt 2.9):

$$potOdds = callAmount/(potSize + callAmount) \tag{2}$$

Der Vergleich der Gleichung (1) mit (2) liefert die Erkenntnis, dass TunedFixedRichieRich nicht exakt nach Pot-Odds spielt aber eine Abwandlung der Formel (2) verwendet. Einige Zahlenbeispiele zu Formel (1): Equity = 0,34; PotSize = 25; CallAmount = 10; Ist 8,5 $\geq$ 10 ? Nein, es wird nicht mitgegangen. Equity = 0,51; PotSize = 25; CallAmount = 10; Ist 12,5 $\geq$ 10? Ja, es wird mitgegangen.

Da nun die Spielweise des Bots bekannt ist, soll hier das Preflop-Modell, ein Entscheidungsbaum, analysiert werden. Das Modell wird mit dem J48-Klassifikator unter Verwendung aller Attribute, mit Ausnahme der auf Tabelle 7 gezeigten Attribute, berechnet.

```
1:ALL_IN_EQUITY <= 0.333867
2:| POT_ODDS <= 0.222222: CALL (42.0)
3:| POT_ODDS > 0.222222: FOLD (3969.0/90.0)
4:ALL_IN_EQUITY > 0.333867
5:| ALL_IN_EQUITY_2PLAYERS <= 0.49635
6:| | POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.739827: CALL (3266.0/32.0)
7:| | POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.739827
8:| | | ACTIVE_PLAYER_COUNT <= 2: CALL (31.0/9.0)
9:| | | ACTIVE_PLAYER_COUNT > 2: RAISE (48.0/14.0)
10:| ALL_IN_EQUITY_2PLAYERS > 0.49635
11:| | MAX_BET_SIZE_THIS_STREET <= 30: RAISE (7326.0/138.0)
```

12:| | MAX_BET_SIZE_THIS_STREET > 30: CALL (2104.0)

Bemerkenswert ist, dass aus allen verfügbaren Attributen (s. Anhang A) nur Attribute, mit Ausnahmen des Attributes Max-Bet-Size-This-Street, durch den Klassifikator verwendet werden, welche im Zusammenhang zur tatsächlichen Implementierung der Entscheidungsfindung stehen. Zunächst eine Betrachtung der Blätter, die zu einer Erhöhung führen. Diese stehen in der neunten und elften Zeile des Modell-Entscheidungsbaumes. Der vollständige Pfad von der Wurzel des Baumes bis zum Blatt in Zeile neun bedingt, dass All-In-Equity > 0.333867, All-In-Equity-2Players <= 0.49635, Pot-Odds-To-All-In-Equity-Ratio > 0.739827 und Active-Player-Count > 2 sind. Die erste Fallunterscheidung All-In-Equity > 0.333867 entspricht fast exakt dem theoretischen Wert von einem Drittel. Auch der Wert für All-In-Equity-2Players mit 0,49635 ist sehr nahe dem theoretischen Break-Even-Wert der Equity für zwei Spieler. Wie bereits oben gezeigt liegt dieser Wert bei 0,5. Das Pot-Odds-Kalkül verlangt als Break-Even-Punkt eine Gleichheit der Werte für All-In-Equity und Pot-Odds. Das Verhlltnis der beiden Werte zueinander sollte gleich Eins sein. Stattdessen liegt im Modell der Wert bei 0,739827. Dies lässt sich durch die Implementierung des TunedFixedRichieRich, welche sich nicht strikt an Formel (2) für Pot-Odds hält erklären.

Der zweite Fall für eine Erhöhung tritt ein, wenn folgende Werte vorliegen: All-In-Equity > 0.333867, All-In-Equity-2Players > 0.49635 und Max-Bet-Size-This-Street <= 30. Ein Blick in den Code-Ausschnitt offenbart, dass für alle Fälle, wenn die Equity bei drei aktiven Spielern größer einem Drittel und die Equity gegen zwei aktive Spieler größer 0,5 ist, erhöht wird. Stattdessen wird im Modell eine weitere Fallunterscheidung mit dem Attribut Max-Bet-Size-This-Street, ein Maß für den maximalen Betrag, der in der aktuellen Runde gesetzt wird, durchgeführt. Die Verwendung dieses Attributes ist auf dem ersten Blick ungewöhnlich, da es in der Original-Entscheidungsfindung keine Rolle spielt. Bei Fixed-Limit Hold'em sind in der Preflop-Wettrunde bis zu drei Erhöhungen möglich. Die für die Berechnung des Modells verwendeten Logdateien sind Pokerpartien mit den Blind-Größen 5/10. Das bedeutet, dass nach dreimaliger Erhöhung der Wert für Max-Bet-Size-This-Street 30 ist und danach keine weiteren Erhöhungen mehr möglich sind. Während des Spiels wird also, wenn TunedFixedRichieRich nach dreimaliger Erhöhung eine weitere Erhöhung tätigen will, stattdessen vom Spieleserver ein Mitgehen erzwungen, weil Erhöhen nicht länger zulässig ist. Der Klassifikator kann wegen diesem Umstand niemals die wahre Entscheidung des Originals mit einer Trefferquote von 100% modellieren. Die Modelle der Wettrunden Flop, Turn und River sind im Anhang B.2 zu finden. Diese sind ebenfalls relativ klein. Durch Analyse kann bei besagten Modellen ebenfalls auf die Implementierung des Bots geschlossen werden.

### Spielpartien

Folgende Partien werden Gespielt in den Startpositionen (TFRR: Abkürzung für TunedFixedRichieRich):

**Small-Blind|Big-Blind|Button**

Partie Nr. 0: TFRR(Model)|TFRR(Original)|TFRR(Original)
Partie Nr. 1: TFRR(Original)|TFRR(Modell)|TFRR(Original)
Partie Nr. 2: TFRR(Original)|TFRR(Original)|TFRR(Modell)
Die Resultate sind auf den Tabellen 14, 15 und 16 gezeigt.

In der Spielpartie, welche als Referenz zum Vergleich dienen soll, spielt TunedFixedRichieRich gegen zwei Kopien seiner selbst. Um die Modelle zu evaluieren tritt in weiteren Spielpartien der modellierte TunedFixedRichieRich gegen zwei Originale an. Die Tabellen 14, 15 und 16 veranschaulichen den Unterschied der Modelle zum Original. Tabelle 14 stellt die Gewinnresultate des Modells den Resultaten des Originals gegenüber. Es sind Unterschiede in der Ausprägung der Gewinne bzw. der Verluste festzustellen. Gemeinsamkeiten finden sich, aus welcher Startposition aus insgesamt ein Gewinn oder ein Verlust erzielt wird.

Ein Vergleich des Setzverhaltens, gezeigt in Tabelle 15, soll Klarheit schaffen, ob die Modelle dem Verhalten des Originals nahekommen. Das Augenmerk sei auf die Abweichungen der absoluten Anzahl der

| Gegenüberstellung der Gewinne des Modells und des Originals | | | | | | | |
|---|---|---|---|---|---|---|---|
| Partie-Nr. | Spieler | Nettogewinn | Preflop | Flop | Turn | River | Showdown |
| 0 | Original | -1445 | -5 | -20 | 345 | 190 | -1955 |
| 0 | Modell | -1305(+140) | -25(+20) | -5(+15) | 305(-40) | 415(+225) | -1995(-40) |
| 1 | Original | 2502,5 | 5 | 10 | 190 | 590 | 1707 |
| 1 | Modell | 1877,5(-625) | 20(+15) | -10(-20) | 210(+20) | 665(+75) | 992(-715) |
| 2 | Original | -1057,5 | 0 | 10 | -535 | -780 | 247,5 |
| 2 | Modell | -1447,5(-390) | -20(-20) | 20(+10) | -500(+30) | -825(-45) | -122,5(125) |

**Tabelle 14:** Gewinne des Original-TunedFixedRichieRich im Vergleich zum modellierten TunedFixedRichieRich nach Wettrunde. Die Abweichung des Modells zum Original ist geklammert angegeben.

Erhöhungen, des Mitgehens und des Verwerfens (Spalten Raises, Calls und Folds) gerichtet. Die größte Abweichung liegt Preflop bei 2,7%. Für den Flop beträgt die größte Abweichung 5,6%. Der Unterschied fällt hier vermeintlich großaus. Tatsächlich aber verwirft das Original 18 der Hände, das Modell 19 Hände. Auch in den Turn- und Riverwettrunden ist der relative Fehler bei den verworfenen Händen am größten. Auf dem River werden 9,1% weniger Hände verworfen. Wegen kleine Anzahl an Händen, die auf den Postflop-Wettrunden verworfen werden, wirken sich hier Unterschiede als relative Abweichung besonders stark. Tatsächlich verwirft das Original 22 Hände, das Modell „nur“ 20 Hände. Auffallend ist jedoch, dass das Modell auf dem River etwas aggressiver als das Original spielt. Für die River-Wettrunde stehen am wenigsten Trainingsbeispiele zur Verfügung. Eine Erhöhung der Trainingsbeispiele könnte den Fehler verbessern.

Tabelle 16 zeigt, wie die Showdowns ausfallen. Es kann eine hohe Übereinstimmung der Werte festgestellt werden. Dass ist in sofern etwas überraschend, weil aufgrund der nichtdeterministischen Bestimmungsart der All-In-Equity eine gewisse Varianz in der Spielweise der Bots vorprogrammiert ist.

Abbildung 8 zeigt die Gewinnkurve der Modell-Bots und des Originals. Die zugehörigen Werte stehen in den Tabellen. Das Teilbild links oben zeigt Spielpartie Nr.0. Rechts oben ist Spielpartie Nr. 1 zu sehen. Unten links wird Spielpartie Nr.2 dargestellt. Als Referenz dient das Teilbild rechts unten. Es zeigt den Spielverlauf, wenn drei TunedFixedRichieRichs gegeneinander spielen.

| Showdown-Vergleich des Modells und des Originals | | | | |
|---|---|---|---|---|
| Partie-Nr. | Spieler | Gespielt | Sieg | Sieganteil in Prozent |
| 0 | Original | 543 | 246 | 45,3 |
| 0 | Modell | 542(+1) | 243(+3) | 44,8(-0,5) |
| 1 | Original | 517 | 251 | 48,5 |
| 1 | Modell | 519(+2) | 249(-2) | 48,0(-0,5) |
| 2 | Original | 507 | 248 | 48,1 |
| 2 | Modell | 517(+10) | 244(-4) | 48,0(-0,1) |

**Tabelle 16:** Showdown-Vergleich Original-TunedFixedRichieRich zum modellierten TunedFixedRichieRich. Die Abweichung des Modells zum Original ist geklammert angegeben.

| Vergleich des Preflop-Verhaltens des Modells mit dem Original | | | | | | | |
|---|---|---|---|---|---|---|---|
| Partie-Nr. | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
| 0 | Preflop | Original | 1000 | 4 | 606 | 449 | 299 |
| 0 | Preflop | Modell | 1000(0) | 3(-1) | 606(0) | 444(5) | 298(-1) |
| 1 | Preflop | Original | 1000 | 5 | 563 | 465 | 301 |
| 1 | Preflop | Modell | 1000(0) | 5(0) | 578(+15) | 470(+5) | 298(-3) |
| 2 | Preflop | Original | 1000 | 3 | 557 | 427 | 334 |
| 2 | Preflop | Modell | 1000(0) | 0(-3) | 547(-10) | 421(-6) | 328(-6) |

| Vergleich des Flop-Verhaltens des Modells mit dem Original | | | | | | | |
|---|---|---|---|---|---|---|---|
| Partie-Nr. | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
| 0 | Flop | Original | 697 | 22 | 553 | 518 | 18 |
| 0 | Flop | Modell | 699(+2) | 23(+1) | 562(+9) | 517(-1) | 19(+1) |
| 1 | Flop | Original | 694 | 24 | 549 | 493 | 25 |
| 1 | Flop | Modell | 697(+3) | 23(-1) | 553(+4) | 496(+3) | 26(+1) |
| 2 | Flop | Original | 663 | 19 | 493 | 524 | 22 |
| 2 | Flop | Modell | 672(+9) | 21(+2) | 506(+13) | 532(+8) | 23(+1) |

| Vergleich des Turn-Verhaltens des Modells mit dem Original | | | | | | | |
|---|---|---|---|---|---|---|---|
| Partie-Nr. | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
| 0 | Turn | Original | 657 | 43 | 521 | 475 | 31 |
| 0 | Turn | Modell | 657(0) | 43(0) | 517(-4) | 477(+2) | 31(0) |
| 1 | Turn | Original | 645 | 39 | 502 | 454 | 40 |
| 1 | Turn | Modell | 648(+3) | 40(+1) | 504(+2) | 462(+8) | 39(-1) |
| 2 | Turn | Original | 622 | 27 | 464 | 453 | 45 |
| 2 | Turn | Modell | 628(+6) | 27(0) | 471(+7) | 465(+12) | 44(-1) |

| Vergleich des River-Verhaltens des Modells mit dem Original | | | | | | | |
|---|---|---|---|---|---|---|---|
| Partie-Nr. | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
| 0 | River | Original | 583 | 18 | 482 | 433 | 22 |
| 0 | River | Modell | 583(0) | 21(+3) | 489(+7) | 424(-9) | 20(-2) |
| 1 | River | Original | 566 | 27 | 495 | 395 | 22 |
| 1 | River | Modell | 569(+3) | 28(+1) | 497(+2) | 405(+10) | 22(0) |
| 2 | River | Original | 550 | 12 | 446 | 398 | 31 |
| 2 | River | Modell | 557(+7) | 10(+2) | 466(+20) | 395(+3) | 30(-1) |

**Tabelle 15:** Setzverhalten des Original-TunedFixedRichieRich im Vergleich zum modellierten TunedFixed-RichieRich. Aus Platzgründen werden die englischen Begriffe Win für Sieg, Raise für Erhöhen, Call für Mitgehen und Fold für Verwerfen verwendet.

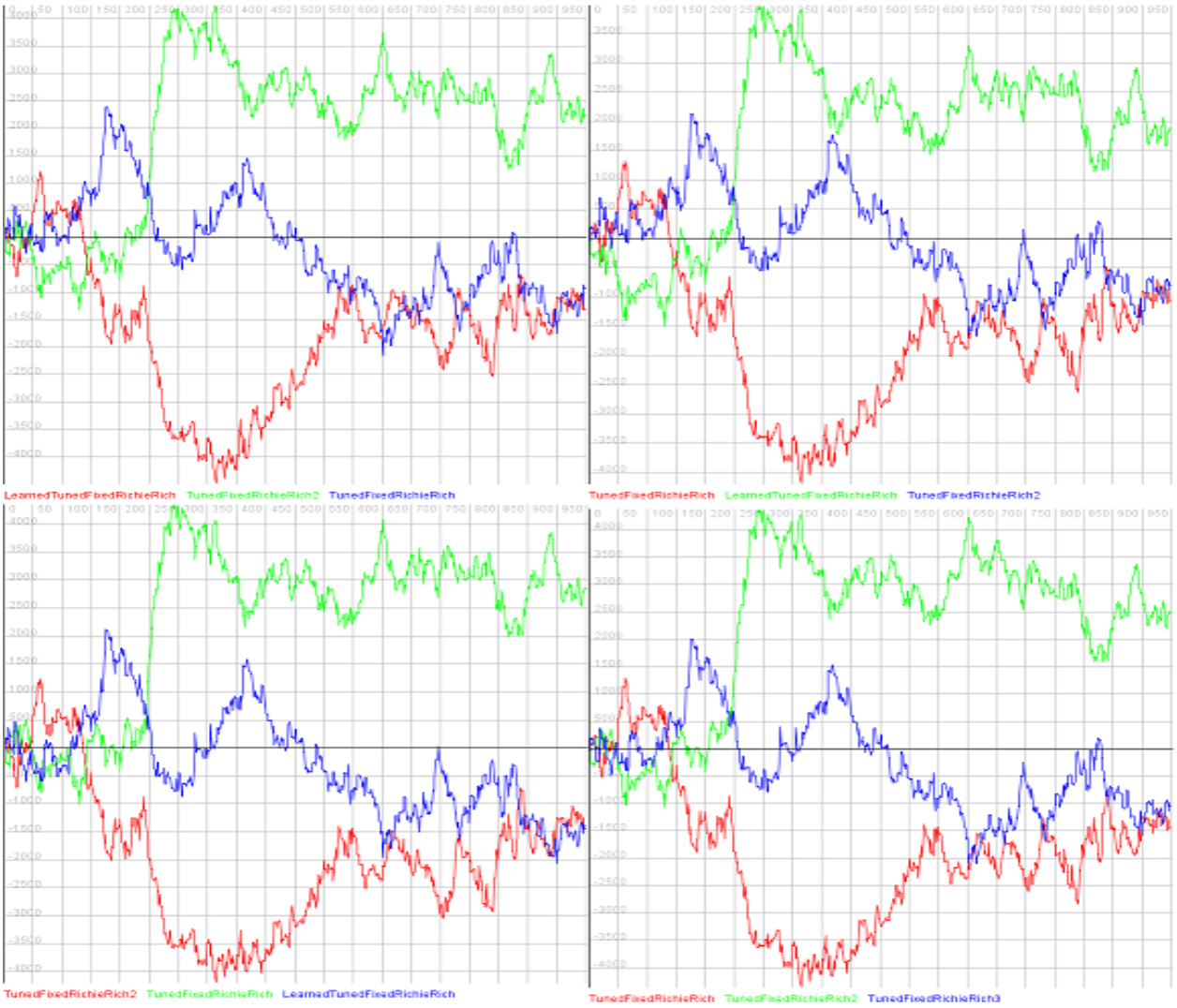


**Abbildung 8:** Gewinnkurve des nach Modell spielenden Bots im Vergleich zum Original-TunedFixedRichieRich (Original im Teilbild rechts unten). Die Farben des Modell-Bot-Graphen sind im Bild links oben rot, im Bild rechts oben grün und im Bild links unten blau.

## 7.2 Akuma

In diesem Abschnitt wird das Spielverhalten des nach dem Akuma-Modell spielenden Bots mit dem des Originals verglichen. Untersucht wird dabei das Verhalten gegen die Gegner CMURingLimit und Bluechip. Beides sind Bots, die bei der ACPC 2009 teilnehmen. Diese Bots stehen eigentlich nicht zur Verfügung um in Matches gegen den modellierten Akuma anzutreten. Daher wird das Spielverhalten dieser aus einer Logdatei der ACPC extrahiert und rekreiert, sodass zwei spezielle Bots genau nach Anweisung der extrahierten Daten spielen, um so das Verhalten der Gegner der aus der ACPC zu simulieren. Sollte es aufgrund einer Abweichung in der Spielweise des Modell-Akumas im Vergleich zum Original dazu kommen, dass Situationen entstehen, in welchen die simulierten Gegner keine Anweisung parat haben, so gehen diese mit. Es sollen die gleichen Handkarten und Gemeinschaftskarten wie bei der ACPC gespielt werden. Diese sind jedoch nur dann vollständig in der Logdatei gespeichert, wenn ein Spiel den River sieht. Aus diesem Grund können nur Spiele, die bis zum River gegangen sind betrachtet werden. Je nach Spielweise der Bots sind das etwa ein Drittel aller gespielten Hände einer

Partie. Das Setzverhalten der Gegner von Akuma sowie die Hand- und Boardkarten werden mit dem `PokerbotTrainer.converter.LogfileDataExtractor` aus der Logdatei extrahiert.

Folgende Partien werden Gespielt in den Startpositionen:
**Small-Blind|Big-Blind|Button**
Partie Nr. 0: Akuma(Modell)|Bluechip|CMURingLimit
Partie Nr. 1: Akuma(Modell)|CMURingLimit|Bluechip
Partie Nr. 2: Bluechip|Akuma(Modell)|CMURingLimit
Partie Nr. 3: Bluechip|CMURingLimit|Akuma(Modell)
Partie Nr. 4: CMURingLimit|Akuma(Modell)|Bluechip
Partie Nr. 5: CMURingLimit|Bluechip|Akuma(Modell)

Die Tabellen 17, 18, 19 und 20 zeigen den Vergleich der Spielweise der Modelle mit dem Original in den Wettrunden Preflop, Flop, Turn und River. Tabelle 22 zeigt den Vergleich des Gewinns. Tabelle 21 veranschaulicht den Unterschied der Showdowns. Die in Abschnitt 6.2 bestimmte gute Trefferquote des Preflop-Modells kann experimentell bestätigt werden. Lediglich in Spielpartie Nr.5 sind geringe Abweichungen festzustellen (s. Tabelle 17). Für die beiden Postflop-Wettrunden Flop und Turn ändert sich das Bild. Die Anzahl der verworfenen Hände stimmt noch gut überein. Es werden jedoch weniger Erhöhungen und stattdessen mehr Calls vom modellierten Akuma getätigt (s. Tabellen 18 und 19). Das passivere Auftreten des modellierten Bots kann die weniger stark ausfallenden Gewinne erklären (s. Tabelle 22). Für die River-Wettrunde kehrt sich die Passivität der Flop- und Turn-Wettrunde um. Es werden etwas mehr Erhöhungen getätigt. Die Anzahl der Calls zeigt nun eine gute Übereinstimmung. Die verworfenen Hände zeigen bei einigen Spielpartien stärkere Abweichungen (s. Tabelle 20). Insgesamt ist jedoch, wie durch die Gewinnkurven der Abbildung 9, welche als Beispiel die Partie Nr.0 zeigt, verdeutlicht wird, eine hohe Übereinstimmung der Spielweise zu erkennen. Bei der Datenerhebung werden zwangsläufig weniger River-Daten als Daten der vorangegangenen Wettrunden erzeugt. Eine Erhöhung der Trainingsbeispiele für die Postflop-Wettrunden könnte das Ergebnis verbessern. Hier nicht gezeigte Untersuchungen haben ergeben, dass durch Manipulation der zu verwendenden Attribute die Precision- und Recall-Werte der Modelle zugunsten einer Klasse, beispielsweise für „Fold“, verbessert werden können.

| **Partie-Nr.** | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
|---|---|---|---|---|---|---|---|
| 0 | Preflop | Original | 372 | 0 | 177 | 80 | 142 |
| 0 | Preflop | Modell | 372(0) | 0(0) | 177(0) | 80(0) | 142(0) |
| **1** | Preflop | Original | 423 | 0 | 177 | 130 | 156 |
| 1 | Preflop | Modell | 423(0) | 0(0) | 177(0) | 130(0) | 156(0) |
| **2** | Preflop | Original | 366 | 0 | 154 | 110 | 148 |
| 2 | Preflop | Modell | 366(0) | 0(0) | 154(0) | 110(0) | 148(0) |
| **3** | Preflop | Original | 375 | 0 | 198 | 81 | 119 |
| 3 | Preflop | Modell | 375(0) | 0(0) | 198(0) | 81(0) | 119(0) |
| **4** | Preflop | Original | 381 | 0 | 167 | 110 | 132 |
| 4 | Preflop | Modell | 381(0) | 0(0) | 167(0) | 110(0) | 132(0) |
| **5** | Preflop | Original | 441 | 0 | 179 | 112 | 164 |
| 5 | Preflop | Modell | 441(0) | 0(0) | 180(+1) | 111(+1) | 164(0) |

**Tabelle 17:** Vergleich der Spielweise Modell-Akuma zum Original. Die Abweichung des Modells gegenüber dem Original ist geklammert angegeben. Aus Platzgründen werden die englischen Begriffe Raise für Erhöhen, Call für Mitgehen und Fold für Verwerfen benutzt.

| **Partie-Nr.** | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
|---|---|---|---|---|---|---|---|
| 0 | Flop | Original | 230 | 0 | 173 | 82 | 8 |
| 0 | Flop | Modell | 230(0) | 0(0) | 171(-2) | 84(+2) | 8(0) |
| 1 | Flop | Original | 267 | 0 | 191 | 97 | 16 |
| 1 | Flop | Modell | 267(0) | 0(0) | 191(0) | 99(+2) | 16(0) |
| 2 | Flop | Original | 218 | 0 | 165 | 80 | 9 |
| 2 | Flop | Modell | 218(0) | 0(0) | 153(-12) | 88(+8) | 11(+2) |
| 3 | Flop | Original | 256 | 0 | 185 | 98 | 12 |
| 3 | Flop | Modell | 256(0) | 0(0) | 180(-5) | 103(+5) | 12(0) |
| 4 | Flop | Original | 249 | 0 | 166 | 111 | 13 |
| 4 | Flop | Modell | 249(0) | 0(0) | 162(-4) | 116(5) | 13(0) |
| 5 | Flop | Original | 247 | 0 | 186 | 92 | 11 |
| 5 | Flop | Modell | 247(0) | 0(0) | 182(-4) | 97(+5) | 11(0) |

**Tabelle 18:** Vergleich der Spielweise Modell-Akuma zum Original. Die Abweichung des Modells gegenüber dem Original ist geklammert angegeben.

| **Partie-Nr.** | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
|---|---|---|---|---|---|---|---|
| 0 | Turn | Original | 222 | 0 | 196 | 46 | 5 |
| 0 | Turn | Modell | 222(0) | 0(0) | 194(-2) | 46(0) | 5(0) |
| 1 | Turn | Original | 251 | 0 | 221 | 46 | 4 |
| 1 | Turn | Modell | 251(0) | 0(0) | 213(-8) | 52(+6) | 4(0) |
| 2 | Turn | Original | 209 | 0 | 195 | 35 | 4 |
| 2 | Turn | Modell | 207(-2) | 0(0) | 191(-4) | 35(0) | 3(-1) |
| 3 | Turn | Original | 244 | 0 | 221 | 39 | 3 |
| 3 | Turn | Modell | 244(0) | 0(0) | 216(-5) | 45(+6) | 3(0) |
| 4 | Turn | Original | 236 | 0 | 222 | 43 | 3 |
| 4 | Turn | Modell | 236(0) | 0(0) | 206(-16) | 57(+14) | 5(+2) |
| 5 | Turn | Original | 236 | 0 | 217 | 46 | 3 |
| 5 | Turn | Modell | 236(0) | 0(0) | 218(+1) | 44(2) | 3(0) |

**Tabelle 19:** Vergleich der Spielweise Modell-Akuma zum Original. Die Abweichung des Modells gegenüber dem Original ist geklammert angegeben.

| **Partie-Nr.** | Wettrunde | Spieler | Gespielt | Sieg | Raises | Calls | Folds |
|---|---|---|---|---|---|---|---|
| 0 | River | Original | 219 | 14 | 216 | 11 | 11 |
| 0 | River | Modell | 217(-2) | 13(-1) | 215(-1) | 12(+1) | 7(-3) |
| 1 | River | Original | 247 | 16 | 249 | 12 | 10 |
| 1 | River | Modell | 247(0) | 16(0) | 251(+2) | 12(0) | 9(-1) |
| 2 | River | Original | 205 | 12 | 210 | 10 | 8 |
| 2 | River | Modell | 204(-1) | 12(0) | 207(-3) | 11(+1) | 7(-1) |
| 3 | River | Original | 241 | 21 | 232 | 14 | 11 |
| 3 | River | Modell | 241(0) | 20(-1) | 239(+7) | 12(-2) | 7(-4) |
| 4 | River | Original | 233 | 22 | 225 | 15 | 17 |
| 4 | River | Modell | 231(-2) | 22(0) | 230(+5) | 15(0) | 10(-7) |
| 5 | River | Original | 233 | 21 | 232 | 14 | 11 |
| 5 | River | Modell | 233(0) | 21(0) | 235(+3) | 16(+2) | 7(-4) |

**Tabelle 20:** Vergleich der Spielweise Modell-Akuma zum Original. Die Abweichung des Modells gegenüber dem Origina ist geklammert angegeben.

| Partie-Nr. | Spieler | Gespielt | Sieg | Sieganteil in % |
|---|---|---|---|---|
| 0 | Original | 194 | 106 | 54,6 |
| 0 | Modell | 197(+3) | 105(-1) | 53,3 |
| 1 | Original | 221 | 122 | 55,2 |
| 1 | Modell | 222(+1) | 125(+3) | 56,3 |
| 2 | Original | 185 | 110 | 59,5 |
| 2 | Modell | 185(0) | 111(+1) | 60,0 |
| 3 | Original | 209 | 109 | 52,2 |
| 3 | Modell | 214(+5) | 110(+1) | 51,9 |
| 4 | Original | 194 | 101 | 52,1 |
| 4 | Modell | 199(+5) | 103(+2) | 51,8 |
| 5 | Original | 201 | 111 | 55,2 |
| 5 | Modell | 205(+4) | 114(+3) | 55,6 |

**Tabelle 21:** Showdown-Vergleich Modell-Akuma zum Original. Die Abweichung der Modells gegenüber dem Original ist geklammert angegeben.

| Partie-Nr. | Spieler | Nettogewinn | Preflop | Flop | Turn | River | Showdown |
|---|---|---|---|---|---|---|---|
| 0 | Original | 2580 | 0 | 0 | 0 | 135 | 2445 |
| 0 | Modell | 2325(-255) | 0(0) | 0(0) | -70(-70) | 605(+470) | 1790(-655) |
| 1 | Original | 3150 | 0 | 0 | 0 | 635 | 2515 |
| 1 | Modell | 3567,5(+417,5) | 0(0) | 0(0) | 0(0) | 815(+180) | 2687,5(+172,5) |
| 2 | Original | 3290 | 0 | 0 | 0 | 155 | 3135 |
| 2 | Modell | 3107,5(-182,5) | 0(0) | 0(0) | -90(-90) | 345(+190) | 2852,5(-282,5) |
| 3 | Original | 2945 | 0 | 0 | 0 | 310 | 2635 |
| 3 | Modell | 2747,5(-197,5) | 0(0) | 0(0) | 0(0) | 760(+450) | 1987,5(-647,5) |
| 4 | Original | 1375 | 0 | 0 | 0 | -310 | 1685 |
| 4 | Modell | 1800(+425) | 0(0) | 0(0) | -50(-50) | 620(+930) | 1230(-455) |
| 5 | Original | 2505 | 0 | 0 | 0 | -180 | 2685 |
| 5 | Modell | 2402,5(-102,5) | 0(0) | 0(0) | 0(0) | 460(+640) | 1942,5(-742,5) |

**Tabelle 22:** Gewinn-Vergleich Modell-Akuma zum Original. Die Abweichung des Modells gegenüber dem Original ist geklammert angegeben.


**Abbildung 9:** Gewinnkurve der Spielpartie Nr.0 (rechts) im Vergleich zur Original-Spielpartie der ACPC 2009. Die rote Kurve im Teilbild rechts steht für den modellierten Bot.

## 8 Fazit und Ausblick

Es wird der Fragestellung nachgegangen, ob sich durch Verfahren des Maschinellen Lernens, die nicht vom Pokerbot Sartre (s. Abschnitt 1.1) eingesetzt werden, mit Hilfe eines Top-Down-Ansatzes das Verhalten eines Pokerspielers nachgeahmt werden kann. Es wird der Ansatz verfolgt, viele Merkmale einer Spielsituation zur Vorhersage der zu tätigenden Aktion zu verwenden.

In Kapitel 2 werden zunächst die Regeln von Fixed-Limit-Poker, sowie Konzepte, die Anwendung als Attribute finden sollen, wie beispielsweise die Pot-Odds (s. Abschnitt 2.9), erklärt.

Um eine Beschreibung der Spielsituationen zu erhalten wird die Software Poker-Bot-Trainer entwickelt (s. Kapitel 3.2 ff.). Die Software setzt auf dem bereits vorhandenen TUD-Poker-Framework (s. Abschnitt 3.4) auf und setzt für die Klassifikation das Weka-Machine-Learning-Framework ein (s. Abschnitt 3.3).

Als Datengrundlage dienen Serverlogs von Spielpartien für drei Spieler des an der TU-Darmstadt entwickelten Pokerbots Akuma bei der ACPC von 2009 (s. Kap. 4). Aus den Serverlogs werden, mit Hilfe der zuvor beschriebenen Software, Datensätze für die Klassifikation erzeugt.

Kapitel 5 befasst sich mit der Voruntersuchung der Daten. Durch eine maschinelle Attribute-Auswahl werden unwichtige Attribute identifiziert (s. Abschnitt 5.2.1) und anschließend entfernt. Da während der ACPC mehrmals mit gleichen Karten, jedoch mit anderen Gegnern in anderen Startpositionen gespielt wird, ist bei der Trennung der Daten in Trainingsdaten und Testdaten darauf zu achten, dass nach Kartensets getrennt wird (s. Abschnitt 5.2.2).

Kapitel 6 befasst sich mit Klassifikationsexperimenten. Ein Vergleich der Klassifizierer zeigt, dass der J48-Klassifizierer, in Hinblick auf Trainingszeit, Speicherverbrauch und Trefferquote, die besten Resultate liefert. Deshalb wird er für alle folgenden Experimente ausgewählt wird (s. Abschnitt 6.1). In Abschnitt 6.2 wird untersucht, wie sich die Anzahl der Trainingsinstanzen auf die Modellgröße und die Accuracy auswirkt. Es stellt sich heraus, dass bereits Modelle, welche auf nur einem Prozent der Trainingsinstanzen trainiert werden, gute Trefferquoten haben und eine Erhöhung der Trainingsbeispiele diese noch weiter verbessert. Allerdings explodiert die Modellgröße mit zunehmender Anzahl an Trainingsinstanzen, was eine händische Analyse der Modelle erschwert. Durch Filterung von Basisgrößen eines abgeleiteten Attributs und durch Abstraktion von vielwertigen nominalen Attributen kann eine starke Verkleinerung der Modelle erreicht werden (s. Abschnitt 6.4). Jedoch verschlechtert sich die Trefferquote durch diese Maßnahme geringfügig. Deshalb werden die Modelle für die Praxistests lediglich mit einer Filterung unwichtiger Attribute, nicht jedoch mit einer stärkeren Reduktion, berechnet (s. Abschnitt 6.5).

Die Praxistests in Kapitel 7 werden in Form von Spielpartien abgehalten, bei welchen das Verhalten der modellierten Bots mit dem Verhalten der Orginale verglichen wird. Betrachtet werden die Bots Akuma und TunedFixedRichieRich, ein dem TUD-Poker-Framework beiliegender Bot. Bei Akuma zeigt sich eine hohe Übereinstimmung der Spielweise, besonders in der Preflop-Phase. In den Flop- und Turnwettrunden ist der modellierte Bot etwas passiver. Auf dem River hingegen etwas aggressiver (s. Abschnitt 7.2). Der Bot TunedFixedRichieRich zeigt insgesamt ebenfalls eine gute Ähnlichkeit zum Original. In Abschnitt 7.1 wird gezeigt, dass sich durch eine Analyse des Modells Rückschlüsse auf die Implementierung des simplen TunedFixedRichieRichs treffen lassen und somit ein Reverse-Engineering in Aussicht stellen.

Zusammenfassend lässt sich sagen, dass der verfolgte Ansatz zur Nachahmung von Pokerspielerverhalten gut funktioniert. Sowohl die Kreuzvalidierungen als auch die Praxistests untermauern dies.

### Ausblick

Die Modelle des Akuma-Bots waren trotz der vorgestellten Vereinfachung immer noch zu groß für eine manuelle Analyse. Hier könnte man die Zahl der Trainingsinstanzen solange verkleinern, bis eine Analyse möglich wird und die Resultate dann unter Vorbehalt betrachten, um die grobe Spielweise des Spielers zu verstehen. Die Experimente zeigten auch, dass einige Attribute, wie beispielsweise die All-In-Equity (s. Abschnitt 3.5.2), sehr wichtig sind. Das liegt auch daran, weil die untersuchten Bots die Equity-Berechnung beherrschen. Im Umkehrschluss bedeutet dies jedoch, dass Bots, die anhand derzeit noch

nicht vom Poker-Bot-Tool erfassbaren Spielesituationsattribute Entscheidungen treffen, möglicherweise nicht so gut modelliert werden können. Die Arbeit hat gezeigt, dass durch Hinzunahme vieler mittelmäßig wichtiger Attribute zusätzlich zu den wichtigsten Merkmalen die Modelle verfeinern können. Möglicherweise lassen sich durch Implementierung weiterer Merkmale die Resultate noch weiter verbessern. Man könnte auch untersuchen, wie sich die Verwendung anderer Klassenattribute, wie beispielsweise den Betrag, den Hero investieren wird, auswirkt. Ein weiterer Ansatz wäre, den zu erwartenden Gewinn vorherzusagen, um dann die Aktion auszuwählen, die den höchsten Betrag verspricht.

## Literatur


[Acpc 2009] *The Annual Computer Poker Competition*. 2009. – URL http://webdocs.cs.ualberta.ca/~pokert/2009/. – Zugriffsdatum: 11/5/14. – Website

[pokerstrategy 2014] *Strategie: Die Spielregeln von Texas Hold'em*. 2014. – URL http://de.pokerstrategy.com/strategy/others/244/1/. – Zugriffsdatum: 12/03/14. – Website

[Wikipedia 2014] *Texas Hold'em - Wikipedia*. 2014. – URL https://de.wikipedia.org/wiki/Texas_Hold'em. – Zugriffsdatum: 12/03/14. – Website

[AcpcAktuell 2014] *The Annual Computer Poker Competition*. 2014. – URL http://www.computerpokercompetition.org/. – Zugriffsdatum: 11/5/14. – Website

[Bank 2011] Bank, Max: *Eine Grafische Benutzeroberfläche für ein Poker-Spiel*, TU Darmstadt, Knowledge Engineering Group, Diplomarbeit, 2011. – URL http://www.ke.tu-darmstadt.de/lehre/arbeiten/bachelor/2011/Bank_Max.pdf. – Bachelor-Thesis

[Fürnkranz u. a. 2008] Fürnkranz, Johannes ; Lorenz, Ulf ; Janssen, Frederik ; Park, Sang-hyeun ; Loza, Eneldo ; Sulzmann, Jan-Nikolaus: Proceedings of the 1st Poker Challenge / TU Darmstadt, Knowledge Engineering Group. URL http://www.ke.tu-darmstadt.de/lehre/archiv/ss08/challenge/Ausarbeitungen/Proceedings%20Poker%20Challenge%20SS08.pdf, 2008. – Forschungsbericht

[Holmes u. a. 1994] Holmes, G. ; Donkin, A. ; Witten, Ian H.: WEKA: a machine learning workbench. In: *Intelligent Information Systems,1994. Proceedings of the 1994 Second Australian and New Zealand Conference on Intelligent Information Systems*, IEEE, Nov 1994, S. 357–361. – URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=396988. – ISBN 0780324048

[Rubin und Watson 2011] Rubin, Jonathan ; Watson, Ian: Successful Performance via Decision Generalisation in No Limit Texas Hold'em. In: Ram, Ashwin (Hrsg.) ; Wiratunga, Nirmalie (Hrsg.): *Case-Based Reasoning Research and Development* Bd. 6880. Springer Berlin Heidelberg, 2011, S. 467–481. – URL http://dx.doi.org/10.1007/978-3-642-23291-6_34. – ISBN 978-3-642-23290-9

[Rubin und Watson 2012] Rubin, Jonathan ; Watson, Ian: Case-based Strategies in Computer Poker. In: *AI Communications* 25 (2012), Januar, Nr. 1, S. 19–48. – URL http://dl.acm.org/citation.cfm?id=2350131.2350136. – ISSN 0921-7126

[Schaffer und Uwira 2008] Schaffer, Hendrik ; Uwira, Oliver: Limit Texas Hold'em - Einführung in Konzepte menschlicher Strategie. In: Fürnkranz, Johannes (Hrsg.) ; Lorenz, Ulf (Hrsg.) ; Janssen, Frederik (Hrsg.) ; Park, Sang-hyeun (Hrsg.) ; Loza, Eneldo (Hrsg.) ; Sulzmann, Jan-Nikolaus (Hrsg.): *Proceedings of the 1st Poker Challenge*, TU Darmstadt, Knowledge Engineering Group, 2008, S. 4–29. – URL http://www.ke.tu-darmstadt.de/lehre/archiv/ss08/challenge/Ausarbeitungen/Proceedings%20Poker%20Challenge%20SS08.pdf

[Witten und Frank 2005] Witten, Ian H. ; Frank, Eibe ; Gray, Jim (Hrsg.): *Data Mining: Practical Machine Learning Tools and Techniques*. 2nd. San Francisco, CA : Morgan Kaufmann Publishers, 2005 (The Morgan Kaufmann Series in Data Management Systems). – ISBN 0-12-088407-0

[Zopf 2010] Zopf, Markus: *Ein Framework zur Entwicklung und Evaluation intelligenter Pokeragenten*, TU Darmstadt, Knowledge Engineering Group, Diplomarbeit, 2010. – URL http://www.ke.tu-darmstadt.de/lehre/arbeiten/bachelor/2010/Zopf_Markus.pdf. – Bachelor-Thesis

## Tabellenverzeichnis

## Abbildungsverzeichnis

# A Attribute-Definitionen

Im Folgenden werden alle Attribute angegeben. Dabei steht bei immer der Attribut-Typ bzw. die möglichen Werte bei nominalen Attributen geklammert hinter den Namen.

## A.1 Klassenattribute

- CLASS_HERO_ACTION_TAKEN (RAISE,FOLD,CALL,INVALID)
- CLASS_TO_INVEST numerisch
  Dient zur Vorhersage des Betrags den Hero in der aktuellen Wettrunde investieren wird.

## A.2 Anzahl der (aktiven) Spieler

- ACTIVE_PLAYER_COUNT numerisch
- PLAYER_COUNT numerisch
- FOLDS numerisch
- OPPONENT_HAS_FOLDED_0, OPPONENT_HAS_FOLDED_1 (true,false)

## A.3 Investment und Pot-bezogene Attribute

- MAX_BET_SIZE_THIS_STREET numerisch
- POT_SIZE numerisch
- CALL_AMOUNT numerisch
- INVESTED_THIS_STREET numerisch
- HERO_INVESTED_THIS_STREET numerisch
- INVESTED_THIS_STREET_OPPONENT_0, INVESTED_THIS_STREET_OPPONENT_1 numerisch

## A.4 Position

- LAST_RAISE_POSITION (SB,BB,UTG_1,UTG_2,UTG_3,MP_1,MP_2,MP_3,CO,BU,INVALID)
- HERO_POSITION (SB,BB,UTG_1,UTG_2,UTG_3,MP_1,MP_2,MP_3,CO,BU,INVALID)
- OPPONENT_POSITION_0, OPPONENT_POSITION_1 (SB,BB,UTG_1,UTG_2,UTG_3,MP_1,MP_2,
  MP_3,CO,BU,INVALID)

## A.5 Handkarten

- HERO-HOLE-CARDS: (3c2c,3c2d,3c2h,3c2s,4c2c,4c3c,4c2d,..., AsTs, AsJs,AsQs,AsKs,INVALID) (insgesamt 1327 Werte)
- HERO-HOLE-CARDS-SUITEDNESS: (true,false)
- ABSTRACTED-HOLE-CARDS: (22,33,32,44,42,43,55,52,...,AT,AJ,AQ, AK,INVALID) (insgesamt 92 Werte)
- HERO-HOLE-CARD-HIGH, HERO-HOLE-CARD-LOW: (2,3,4,5,6,7,8,9,T,J,Q,K,A,INVALID)
- HERO_HOLE_CARD_HIGH_SUIT, HERO_HOLE_CARD_LOW_SUIT: (c,d,h,s,INVALID)

## A.6 Gemeinschaftskarten

- FLOP_CARD1_RANK,FLOP_CARD2_RANK,FLOP_CARD3_RANK,TURN_CARD_RANK,
  RIVER_CARD_RANK: (2,3,4,5,6,7,8,9,T,J,Q,K,A,INVALID)
- FLOP_CARD1_SUIT,FLOP_CARD2_SUIT,FLOP_CARD3_SUIT,TURN_CARD_SUIT,
  RIVER_CARD_SUIT: (c,d,h,s,INVALID)
- FLOP_CARD_1,FLOP_CARD_2,FLOP_CARD_3,TURN_CARD,RIVER_CARD: 2c,3c,4c,5c,6c,7c,8c,
  9c,Tc,Jc,Qc,Kc,Ac,2d,3d,4d,5d,6d,7d,8d,9d,Td,Jd,Qd,Kd,Ad,2h,3h,4h,5h,6h,7h,8h,9h,Th,Jh,Qh,
  Kh,Ah,2s,3s,4s,5s,6s,7s,8s,9s,Ts,Js,Qs,Ks,As,INVALID

## A.7 Setzverhalten

r = Raise, f = Fold, c = Call

- PREFLOP-ACTION: ”,r,rr,rrr,rrrf,...,ccrcf,ccrcc,ccf,ccc (insgesamt 178 Werte)
- FLOP-ACTION, TURN-ACTION, RIVER-ACTION, CURRENT_STREET_ACTION: ”,r,rr,rrr,rrrr,rrrrf, ...,ccrcf,ccrcc,ccf,ccc (insgesamt 370 Werte)[2]
- PREFLOP_ACTION_ABSTR, FLOP_ACTION_ABSTR, TURN_ACTION_ABSTR, RIVER_ACTION_
  ABSTR, CURRENT_STREET_ACTION_ABSTR: (ACTIVE_ACTIVE, ACTIVE_PASSIVE, PASSIVE
  _PASSIVE,FIRST_TO_ACT, ACTIVE,PASSIVE, INVALID)
- PREFLOP_RAISES, FLOP_RAISES, TURN_RAISES, RIVER_RAISES: numerisch
- PREFLOP_ACTION_ABSTR_ALL, FLOP_ACTION_ABSTR_ALL, TURN_ACTION_ABSTR_ALL, RIVER_ACTION_ABSTR_ALL:
  (ACTIVE_ACTIVE,ACTIVE_PASSIVE,PASSIVE_PASSIVE,FIRST_TO_ACT,
  ACTIVE,PASSIVE,INVALID)

[2] Der Unterschied ergibt sich aus dem Umstand, dass Postflop bis zu vier, Preflop jedoch nur bis zu drei Erhöhungen möglich sind.

## A.8 Spieler-bezogene Attribute

- HISTORY_VPIP_OPPONENT_0,HISTORY_VPIP_OPPONENT_1: numerisch, Anteil der Spiele mit freiwilligen Investment
- HISTORY_PREFLOP_RAISE_OPPONENT_0HISTORY_PREFLOP_RAISE_OPPONENT_1: numerisch
- HISTORY_W$SD_OPPONENT_0,HISTORY_W$SD_OPPONENT_1: numerisch, Anteil der gewonnenen Showdowns
- HISTORY_WTSD_OPPONENT_0,HISTORY_WTSD_OPPONENT_1: numerisch, gesehene Showdowns
- HISTORY_FOLD_TO_BET_OPPONENT_0,HISTORY_FOLD_TO_BET_OPPONENT_1: numerisch
- HISTORY_WIN_PERCENTAGE_OPPONENT_0,HISTORY_WIN_PERCENTAGE_OPPONENT_1: numerisch
- RAISES_PREFLOP_OPP_0,RAISES_PREFLOP_OPP_1,RAISES_FLOP_OPP_0,RAISES_FLOP_OPP_1, RAISES_TURN_OPP_0,RAISES_TURN_OPP_1,RAISES_RIVER_OPP_0,RAISES_RIVER_OPP_1: numerisch
- CALLS_PREFLOP_OPP_0,CALLS_PREFLOP_OPP_1,CALLS_FLOP_OPP_0,CALLS_FLOP_OPP_1, CALLS_TURN_OPP_0,CALLS_TURN_OPP_1,CALLS_RIVER_OPP_0,CALLS_RIVER_OPP_1: numerisch
- RAISES_PREFLOP_HERO,RAISES_FLOP_HERO,RAISES_TURN_HERO,RAISES_RIVER_HERO: numerisch
- CALLS_PREFLOP_HERO,CALLS_FLOP_HERO,CALLS_TURN_HERO,CALLS_RIVER_HERO: numerisch
- FOLDS_PREFLOP_HERO,FOLDS_FLOP_HERO,FOLDS_TURN_HERO,FOLDS_RIVER_HERO: numerisch

## A.9 Abstrahierte Attribute

- BOARD_TEXTURE: (NO_SALIENT,S,S+,F,F+,F_S,F_S+,F+_S,F+_S+,H,S_H,S+_H,F_H,F+_H,F_S_H,F_S+_H,F+_S_H,F+_S+_H,H+, S_H+,S+_H+,F_H+,F+_H+,F_S_H+,F_S+_H+,F+_S_H+,F+_S+_H+) (F steht für „Flush ist wahrscheinlich“; S steht für „Straße ist wahrscheinl.“; H bedeutet, dass das Board min. zwei Karten gleichen Rangs hat; „+“-Modifikator bedeutet „sehr“, d.h. F+ bedeutet „Flush sehr wahrscheinl.“)
- SIMPLE_BOARD_TEXTURE: (NO_SALIENT,F,F+,S,S+,F_S,F+_S,F_S+,F+_S+)
- HAND_RANK: (HIGH_CARD,PAIR,TWO_PAIR,THREE_OF_A_KIND,STRAIGHT,FLUSH, FULL_HOUSE,FOUR_OF_A_KIND,STRAIGHT_FLUSH)
- HAND_TYPE: (NOTHING,PAIR,TOPPAIR,OVERPAIR,TWOPAIR,TRIPS,STRAIGHT,FLUSH, FULLHOUSE,QUADS,STRAIGHTFLUSH)
- ALL_IN_EQUITY, ALL_IN_EQUITY_2PLAYERS: numerisch
- POT_ODDS: numerisch
- POT_ODDS_TO_ALL_IN_EQUITY_RATIO: numerisch
- POTENTIAL_OUTS: numerisch
- DRAWS: (STRAIGHT_DRAW,FLUSH_DRAW,STRAIGHT_AND-FLUSH_DRAW,NO_DRAW)

## A.10 Zukunftsattribute

- FUTURE_HERO_OUTCOME: numerisch
- FUTURE_HOLE_CARDS_OPPONENT_0,FUTURE_HOLE_CARDS_OPPONENT_1: numerisch
- FUTURE_OUTCOME_OPPONENT_0,FUTURE_OUTCOME_OPPONENT_1: numerisch

# B Modelle

## B.1 Beispiel für ein überangepasstes Modell (Auszug)

```
J48 pruned tree
------------------

PREFLOP_ACTION =
|   POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 1.188884
|   |   POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 1.164766: RAISE (6862.0/4.0)
|   |   POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 1.164766
|   |   |   ABSTRACTED_HOLE_CARDS = 22: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 33: FOLD (36.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 32: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 44: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 42: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 43: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 55: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 52: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 53: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 54: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 66: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 62: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 63: RAISE (0.0)
|   |   |   ABSTRACTED_HOLE_CARDS = 64: RAISE (0.0)
```

## B.2 TunedFixedRichieRich-Modelle

### B.2.1 Flop-Modell

```
ALL_IN_EQUITY_2PLAYERS <= 0.49895
| POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.736752: CALL (2814.0/21.0)
| POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.736752
| | POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.847099
| | | PREFLOP_RAISES <= 0: FOLD (21.0/1.0)
| | | PREFLOP_RAISES > 0
| | | | POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.813786: CALL (25.0)
| | | | POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.813786: FOLD (10.0/3.0)
```

| | POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.847099: FOLD (96.0)
ALL_IN_EQUITY_2PLAYERS > 0.49895
| MAX_BET_SIZE_THIS_STREET <= 30
| | ALL_IN_EQUITY <= 0.329167: CALL (77.0)
| | ALL_IN_EQUITY > 0.329167: RAISE (3512.0/45.0)
| MAX_BET_SIZE_THIS_STREET > 30: CALL (612.0)

### B.2.2 Turn-Modell

POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.77235
| MAX_BET_SIZE_THIS_STREET <= 60
| | CALLS_TURN_HERO <= 0
| | | ALL_IN_EQUITY <= 0.51305
| | | | ALL_IN_EQUITY <= 0.3329: CALL (1180.0)
| | | | ALL_IN_EQUITY > 0.3329
| | | | | ALL_IN_EQUITY_2PLAYERS <= 0.5047: CALL (1236.0/41.0)
| | | | | ALL_IN_EQUITY_2PLAYERS > 0.5047: RAISE (233.0/11.0)
| | | ALL_IN_EQUITY > 0.51305: RAISE (3576.0/1.0)
| | CALLS_TURN_HERO > 0
| | | ALL_IN_EQUITY <= 0.4997: CALL (489.0/3.0)
| | | ALL_IN_EQUITY > 0.4997: RAISE (11.0/4.0)
| MAX_BET_SIZE_THIS_STREET > 60: CALL (741.0)
POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.77235
| POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.83212
| | PREFLOP_RAISES <= 2: FOLD (37.0/14.0)
| | PREFLOP_RAISES > 2: CALL (14.0)
| POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.83212
| | POT_ODDS <= 0.111111
| | | POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.914547: CALL (4.0/1.0)
| | | POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.914547: FOLD (13.0)
| | POT_ODDS > 0.111111: FOLD (209.0)

### B.2.3 River-Modell

ALL_IN_EQUITY_2PLAYERS <= 0.494949

| POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.836453

| | POT_ODDS <= 0.190476: CALL (2991.92)

| | POT_ODDS > 0.190476

| | | POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.702963: CALL (18.0)

| | | POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.702963

| | | | LAST_RAISE_POSITION = SB: CALL (2.0)

| | | | LAST_RAISE_POSITION = BB: FOLD (10.0/1.0)

| | | | LAST_RAISE_POSITION = UTG_1: FOLD (0.0)

| | | | LAST_RAISE_POSITION = UTG_2: FOLD (0.0)

| | | | LAST_RAISE_POSITION = UTG_3: FOLD (0.0)

| | | | LAST_RAISE_POSITION = MP_1: FOLD (0.0)

| | | | LAST_RAISE_POSITION = MP_2: FOLD (0.0)

| | | | LAST_RAISE_POSITION = MP_3: FOLD (0.0)

| | | | LAST_RAISE_POSITION = CO: FOLD (0.0)

| | | | LAST_RAISE_POSITION = BU: FOLD (0.0)

| | | | LAST_RAISE_POSITION = INVALID: FOLD (0.0)

| POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.836453

| | POT_ODDS_TO_ALL_IN_EQUITY_RATIO <= 0.937251

| | | POT_SIZE <= 140: FOLD (22.01/2.01)

| | | POT_SIZE > 140: CALL (10.0)

| | POT_ODDS_TO_ALL_IN_EQUITY_RATIO > 0.937251: FOLD (219.07/0.07)

ALL_IN_EQUITY_2PLAYERS > 0.494949

| MAX_BET_SIZE_THIS_STREET <= 60

| | CALLS_RIVER_HERO <= 0

| | | ALL_IN_EQUITY <= 0.509091

| | | | ALL_IN_EQUITY <= 0.326768: CALL (43.0/1.0)

| | | | ALL_IN_EQUITY > 0.326768: RAISE (212.0/37.0)

| | | ALL_IN_EQUITY > 0.509091: RAISE (4524.0)

| | CALLS_RIVER_HERO > 0: CALL (29.0/1.0)

| MAX_BET_SIZE_THIS_STREET > 60: CALL (912.0)

## C Verteilung der Werte bei Setzverhalenssequenzen auf dem Datensatz der Voruntersuchungen

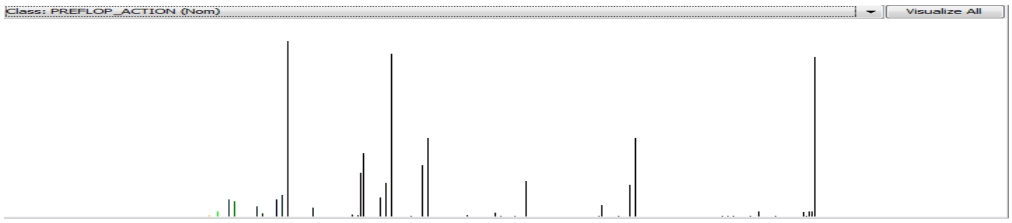


**Abbildung 10:** Verteilung der Werte des Attributes PREFLOP-ACTION (223 Werte). Die Werte der drei großen Balken sind (von links nach rechts): *rfc,frc,ccc*

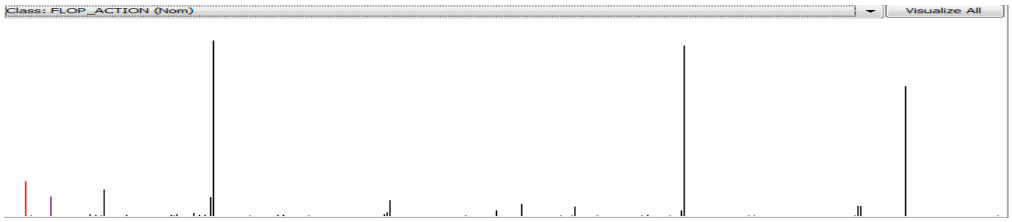


**Abbildung 11:** Verteilung der Werte des Attributes FLOP-ACTION (463 Werte). Die Werte der drei großen Balken sind (von links nach rechts): *rcc,crc,cc*

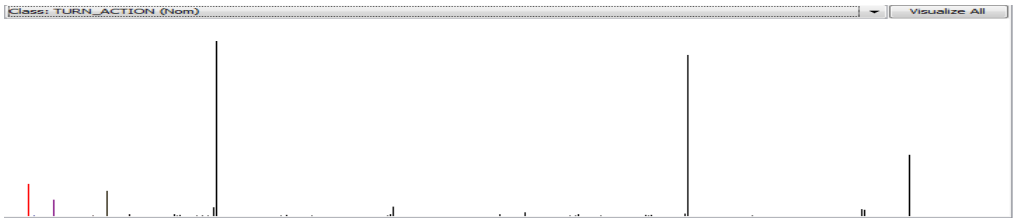


**Abbildung 12:** Verteilung der Werte des Attributes TURN-ACTION (463 Werte). Die Werte der drei großen Balken sind (von links nach rechts): *rc,crc,cc*

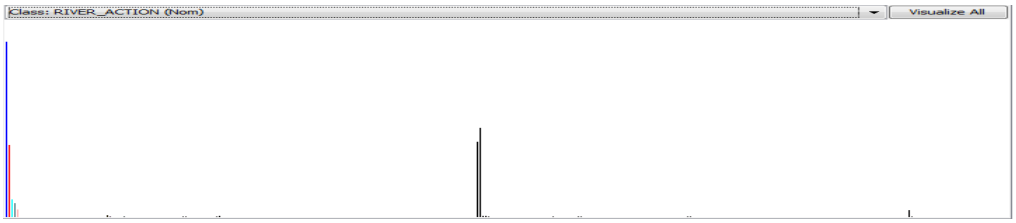


**Abbildung 13:** Verteilung der Werte des Attributes RIVER-ACTION (463 Werte). Die Werte der vier großen Balken sind (von links nach rechts): *<leer>,r,crc,cr*. Der Unterschied zur Flop- und Turnwettrunde lässt sich dadurch erklären, dass der Pokeragent den Ausgang der River-wettrunde nicht beobachten kann, den Ausgang der Flop- bzw. Turn-Wettrunde dagegen schon.

# D Sequenz-Diagramme

## D.1 Übertragung der Spielinformationen durch den `MainConverter`