SVMs und Kategorisierung der Spieler

Benjamin Herbert BHERBERT@RBG.INFORMATIK.TU-DARMSTADT.DE

Abstract

1. Einleitung

Texas Hold'em ist eine sehr populäre Pokervariante, die in den letzten Jahren vermehrt Gegenstand der Forschung geworden ist. Viele Forscher aus dem Gebiet der künstlichen Intelligenz forschen an der Entwicklung von Programmen, die Poker spielen können, so genannten *Pokerbots* oder kurz *Bots*. Dabei werden verschiedene Ansätze gewählt, unter anderem regelbasierte Systeme, Simulationen und Neuronale Netzwerke. Ein Pokerbot spielt nach einer bestimmten Strategie und reagiert auf bestimmte Spielsituationen mit bestimmten Aktionen. Ein interessanter Ansatz ist es, die verfügbaren Spieldaten eines erfolgreichen Pokerspielers oder Pokerbots zu verwenden, um Anhand der in ähnlichen Spielsituationen gewählten Entscheidung die eigene Entscheidung zu treffen. Spielplattformen im Internet bieten oft die Möglichkeit Spiele zu beobachten und Spieldaten zu extrahieren. Dabei ist es von Interesse aus Spieldaten Rückschlüsse zu ziehen, wie man Spieler einordnen kann oder ob bestimmte Verhaltensweisen erfolgreiches Pokerspielen bedingen.

2. Klassifikation

Das Einordnen von Objekten anhand ihrer Merkmale in vorgegebene Klassen bezeichnet man als Klassifikation. Im Maschinellen Lernen wird mit Hilfe von Beispielen ein Klassifikator erlernt, mit dem sich die Klassen von unbekannten Beispielen vorhersagen lassen. Ein Beispiel besteht dabei aus einer Menge von Merkmalen und der zugehörigen Klasse. Oft wird die so genannte *binäre Klassifikation* mit den Klassen + und − betrachtet, diese sind jedoch symbolhaft zu verstehen und werden in der Anwendung durch andere Bezeichnungen ersetzt.

2.1 Beispiele und Klassen

Ein Beispiel wird durch Merkmale beschrieben. In Klassifikationsalgorithmen wird häufig ein sogenannter Featureraum verwendet, in dem sich Beispiele durch Punkte oder Vektoren darstellen lassen. Die Merkmale der Beispiele werden in diesen Featureraum abgebildet. Dazu ist es notwendig, dass nicht diskrete Werte in diskrete Werte überführt werden. Beispiele lassen sich also als Tupel darstellen: (**Feature**,Klasse) wobei es sich bei **Feature** um einen Vektor von Features handelt.

2.2 Anwendung im Pokerspiel

Im Pokerspiel gibt es eine Vielzahl an Merkmalen und Klassifikationsaufgaben. Es folgen eine Auswahl von Merkmalen und eine kurze Erläuterung.

2.2.1 Direkte Merkmale

Direkte Merkmale sind Merkmale, die man direkt aus dem Spiel oder einer aktuellen Hand ableiten kann. Die folgenden Merkmale sind an (Blank, 2004) angelehnt.

Handstärke Die Bewertung der Stärke einer Hand kann dazu dienen Spielentscheidungen zu treffen. So ist es bei einer sehr schwachen Hand ratsam zu passen, bei einer sehr starken Hand sollte auf Gewinnmaximierung wert gelegt werden.

Pot Odds Der Anteil zwischen aktuellem Gesamtbetrag im Spiel und dem Einsatz den man bringen muss um mitzuspielen. Wenn das Verhältnis hoch ist, so kann man auch mit schwächeren Karten mitspielen.

Positives Potential Die Fähigkeit im späteren Spielverlauf eine sehr starke Hand zu erzielen wird Positives Potential genannt. Dabei handelt es sich oft um Flush- und Straight Draws, also Möglichkeiten einen Flush oder einen Straight zu erzielen.

Für weitere Merkmale und eine tiefergehende Beschreibung sei auf (Miller, Sklansky, & Malmuth, 2004; Harrington & Robertie, 2004) verwiesen.

2.2.2 Indirekte Merkmale

Anhand von Daten, die man von vergangenen Spielen zur Verfügung hat, kann man im Nachhinein indirekte Merkmale berechnen. Einige Merkmale kann man für einzelne Spieler berechnen, diese können helfen einen Spieler zu charakterisieren.

Anteil der gespielten Hände Der Anteil der gespielten Hände eines Spielers kann ein Indiz dafür sein, dass dieser zu viele schwache Hände spielt.

Cold Calls Falls ein Spieler mit einer vorausgegangenen Erhöhung mitgeht, ohne vorher Geld investiert zu haben, nennt man das Cold-Calling. Dies sollte man normalerweise vermeiden, sofern man keine gute Hand hat, da ein aggressives Spiel meist Zeichen für eine starke Hand ist.

Diese und weitere Merkmale sind in (Johansson, Sonstrod, & Niklasson, 2006) zu finden.

2.2.3 Klassifikationsaufgaben in Poker

Eine Vielzahl an Klassifikationsaufgaben in Poker sind denkbar. Es ist denkbar den Gegner anhand seiner Spielweise als bestimmten Spielertyp zu erkennen und sein eigenes Spiel anzupassen. So ist es gegen Spieler, die ausschließlich gute Starthände spielen, notwendig sehr solide zu spielen und sich nicht auf unnötiges Risiko einzulassen. Nach Möglichkeit sollte man ein Aufeinandertreffen meiden und frühzeitig passen, sofern man keine gute Hand hält(Harrington & Robertie, 2004).

3. Support Vector Machines

Bei den Suppor Vector Machines (SVM) handelt es sich um eine Klasse von Algorithmen, die zur Klassifikation eingesetzt werden können. In der ursprünglichen Form waren SVMs lineare Klassifikatoren, die Punkte in einem Featureraum durch eine Hyperebene linear separieren.

3.1 Theorie

Im folgenden wird der Fall von linear separierbaren Klassen betrachtet, wie von Vapnik ursprünglich betrachtet. Eine Support Vector Machines beschreibt die optimal separierende Hyperebene, die die gegebenen Beispiele linear separiert. Dabei wird die Hyperebene berechnet, die den größten Abstand zu den Beispielpunkten hat. Diese Ebene kann durch folgende lineare Gleichung beschrieben werden:

$$H = \mathbf{w}\mathbf{x} + b = 0$$

wobei $\mathbf{w}$ eine Normale ist und $\frac{|b|}{|\mathbf{w}|}$ den Abstand zum Ursprung entlang der Normalen beschreibt.

Die parallelen Hyperebenen H_1 und H_2 zu H, ergeben sich zu

$$H_1 : \mathbf{w}\mathbf{x} + b = +1$$
$$H_2 : \mathbf{w}\mathbf{x} + b = -1$$

Für eine Beispielmenge $(\mathbf{x_i}, y_i)$ werden H_1 und H_2 betrachtet und deren Abstand $\frac{2}{|\mathbf{w}|}$ wird maximiert. Dazu muss $|\mathbf{w}|$ minimiert werden. Da keine Punkte in den Raum zwischen H_1 und H_2 fallen sollen, gelten folgende Randbedingungen:

$$\mathbf{w}\mathbf{x_i} + b \geq +1 \text{ if } y_i = +1$$
$$\mathbf{w}\mathbf{x_i} + b \leq -1 \text{ if } y_i = -1$$

die man kompakter als

$$y_i(\mathbf{w}\mathbf{x_i} + b) - 1 \geq 0 \text{ für alle } 1 \leq i \leq n$$

schreiben kann(Burges, 1998). Daraus lässt sich folgendes Optimierungsproblem ableiten, welches die optimal separierende Hyperebene H findet.

Wähle $(\mathbf{w}, b)$ und minimiere $|\mathbf{w}|$ mit der Randbedingung
$$c_i(\mathbf{w}\mathbf{x_i} - b) \geq 1 \text{ für alle } 1 \leq i \leq n$$

Dies kann auf ein *Quadratic Programming* Optimierungsproblem überführt werden

$$\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j x_i x_j$$

mit den Randbedingungen

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$

Eine effiziente Lösung dieses Problems wird in (Joachims, 1999) besprochen, welche in SVMlight implementiert wurde.

3.2 Klassifikation mit SVMs

Das Klassifizieren von unbekannten Beispielen $(x_i, ?)$ erfolgt dann mittels

$$Klasse(x_i) = \begin{cases} +1 & \text{if} \mathbf{w x_i} + b > 0, \\ -1 & \text{if} \mathbf{w x_i} + b \leq 0 \end{cases}$$

Das heisst, es wird für Beispiele anhand ihrer Repräsentation im Featureraum berechnet, auf welcher Seite der Hyperebene H sie sich befinden und die jeweilige Klasse wird zugewiesen.

3.3 Nicht linear separierbare Beispiele

Um auch Klassifikationsproblemen lösen zu können, die nicht linear separierbar sind, ist es möglich, den sogenannten *Kernel-Trick* anzuwenden. Nicht linear separierbare Probleme sind in höherdimensionalen Vektorräumen lösbar. Dabei kommt eine Abbildung ϕ zum Einsatz, die die Vektoren x in diesen höherdimensionalen Featureraum F abbildet. Dies führt jedoch zu nicht mehr handhabbaren Problemen, wenn die Vektoren in diesen Räumen zu groß werden.
In oben genannter dualen Repräsentation erscheinen die Vektoren nur als Skalarprodukte. An den Featureraum angepasst muss die Gleichung also lauten

$$\sum_{i=1}^{n} \alpha_i + \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j \alpha_i \alpha_j \phi(\mathbf{x_i}) \phi(\mathbf{x_j}).$$

Da in dieser Formulierung nur das Skalarprodukt steht ist die Dimensionalität von F nebensächlich. Eine Kernfunktion, oder kurz Kern, ist eine Funktion K, die den Wert des Skalarprodukts von zwei Punkten in F zurückgibt

$$K(x_i, x_j) = \phi(\mathbf{x_i}) \phi(\mathbf{x_j})$$

Das heißt in den Formulierungen kann das Skalarprodukt durch eine geeignete Kernfunktion ersetzt werden. Im folgenden Kapitel werden einige Kernfunktionen aufgeführt.

3.4 Einige Kernfunktionen

Einige geeignete Kernfunktionen sind

lineare Kernfunktionen $K(x_i, x_j) = \langle x_i, x_j \rangle$

polynomielle Kernfunktionen $K(x_i, x_j) = \langle x_i, x_j \rangle^d$

RBF-Kernfunktionen(Radiale Basis Funktion) $K(x_i, x_j) = \exp(-\gamma |x_i - x_j|^2)$

sigmoide Kernfunktionen $K(x_i, x_j) = \tanh(\kappa \langle x_i, x_j \rangle + \nu)$.

Für weitere Information über Kernfunktionen sei auf (Cristianini, 2004), (Schoelkopf, 1997) und (Schoelkopf & Smola, 2002) verwiesen.

3.5 Parameterwahl

Die Auswahl einer geeigneten Kernfunktion kann experimentell geschehen. In (Blank, 2004) wurden dazu auf einem Traingsset für unterschiedliche Kernel Tests mit verschiedenen Parametern durchgeführt und mit einem Testset überprüft.

3.6 SVM Implementierungen

Es gibt zahlreiche freie SVM Implementierungen. In (Blank, 2004) wurde SVM^{light} verwendet, eine OpenSource SVM Implementierung, die für Mustererkennung, Regressionsprobleme und Ranking eingesetzt werden kann. Sie unterstützt die unter Punkt 3.4 genannten Kernel und diese können durch Parameter konfiguriert werden.

4. Entscheidungsbäume

Ein weiterer Algorithmus zur Klassifikation von Beispielen sind Entscheidungsbäume. Ein Entscheidungsbaum beschreibt ein Konzept als Disjunktion von Konjunktionen.

Dabei wird eine hierarchische Struktur von Entscheidungsknoten gelernt. Diese haben den Vorteil, dass das gelernte Modell bzw. die Entscheidungsfindung sehr gut von Menschen zu verstehen ist.

4.1 Klassifikation mit Entscheidungsbäumen

An jedem Knoten wird ein Attribut eines Beispiels untersucht und abhängig von dessen Wert wird das Beispiel mit den Entscheidungsknoten im jeweiligen Teilbaum weiter untersucht. In den Blättern eines Entscheidungsbaumes wird dann eine Klasse zugeordnet.

4.2 Lernen von Entscheidungsbäumen

Das Lernen eines Entscheidungsbaumes besteht darin, zu entscheiden welches Feature man im Wurzelknoten betrachtet. Dabei kommen verschiedene Heuristiken zum Einsatz. Ein Beispiel dafür ist die *Information Gain* Heuristik, die zum Beispiel auch in ID3 und dessen Nachfolger C4.5 zum Einsatz kommt. Im binären Fall ergeben sich somit nach der Entscheidung für ein Attribut im Wurzelknoten zwei Teilbäume. In jedem dieser Teilbäume wird

dann mit den aufgeteilten Beispielen nach dem nächsten Attribut gesucht nach welchem Unterschieden werden soll. Der Entscheidungsbaumalgorithmus von Quinlan, C4.5 ist in (Quinlan, 1993) besprochen.

4.3 Pruning

Entscheidungsbäumen ist es möglich, auf dem Trainingsset eine Genauigkeit von 100% zu erzielen. Der Entscheidungsbaum ist dann jedoch sehr verästelt und generalisiert schlecht, das heisst, dass ungesehene Beispiele wohlmöglich falsch eingeordnet werden und die Genauigkeit auf einem unbekannten Testset somit niedrig ist. Dies nennt man auch *Overfitting* zu deutsch etwa *Überanpassung*.

Eine Mittel um diese Genauigkeit auf unbekannten Daten zu erhöhen ist das sogenannte *Pruning*, bei dem Knoten aus dem Baum entfernt werden um oben genanntes Overfitting zu vermeiden. Pruning hat auch den Vorteil, die Lesbarkeit eines Entscheidungsbaums zu verbessern, da weniger Knoten im Baum vorkommen.

5. Weka

Weka ist eine Sammlung von Algorithmen für Data Mining Aufgaben. Es ist ein Open-Source Software Projekt unter der GNU General Public License. Die Software ist in Java geschrieben und kann als JAR Paket heruntergeladen werden. Es kann in eigenen Javaprogrammen eingesetzt werden um verschiedenste Data Mining Aufgaben zu bewältigen.

5.1 ARFF Dateien

Beispiele, die bestimmte Merkmale haben, lassen sich in eine sogenannte ARFF Datei schreiben. ARFF steht für Attribute-Relation File Format. In dieser Datei werden zunächst im *Header* Attribute definiert. Im *Data* Teil der Datei kommagetrennte Werte der Beispiele.

5.2 Beispiele

Beispiele werden in Weka durch die Klasse `weka.core.Instances.` repräsentiert.

5.3 J48

Weka bietet eine eigene C4.5 Implementierung an, die auch in einem der Artikel verwendet wurde. Die Zugehörige Javaklasse ist `weka.classifiers.trees.J48`.

6. Zusammenfassung: Creating an SVM to Play Strong Poker

Im Paper von Blank, Soh und Scott(Blank, 2004) ist ein Experiment beschrieben, in dem versucht wurde einen erfolgreichen Pokerspieler zu imitieren. In den folgenden Abschnitten wird das Vorgehen und das Ergebnis kurz zusammengefasst.

6.1 Ansatz

Im Experiment kam eine SVM zum Einsatz, die mit Spieldaten eines Spielers trainiert wurde. Die Spieldaten stammen vom Gewinner eines Spiels von mehreren Pokerbots, einem *Simbot* namens Hari. Es wurden die in Kapitel 2.2.1 genannten Merkmale betrachtet. Man betrachtete drei Klassen, die die Entscheidungen in bestimmten Situationen darstellen.

- Fold
- Check/Call
- Bet/Raise

Für die jeweils vorliegende Spielsituation wurde die Klasse der Entscheidung festgestellt und vermerkt. Anhand dieser Daten wurden drei SVMs mit sigmoider Kernfunktion gelernt und verwendet um die Entscheidung vorauszusagen, die Hari gewählt hätte, wenn er in einer ähnlichen Situation gewesen wäre. Diese SVMs wurden dann in einem eigenen Pokerbot implementiert und gegen weitere Pokerbots getestet.

6.2 Ergebnis

Um eine Entscheidung für eine bestimmte Spielsituation zu erhalten wurden oben genannte Features berechnet und die Ergebnisse der drei SVMs für die Spieldaten betrachtet. Die Entscheidung, die von der Mehrheit der SVMs getroffen wurde, wurde im Spiel durchgeführt. Bei eventuellem Gleichstand wurde die sicherste Entscheidung übernommen. Das Ergebnis des Tests war, dass der implementierte Pokerbot im Schnitt einen Verlust von 0,53 Small Bets pro Hand erzielte. Besonders negativ war, dass er sehr passiv spielte und keinerlei Aggression zeigte. Somit war es nicht möglich selbst mit den stärksten Händen genug Profit zu erzielen um etwaige Verluste schwächerer Hände auszugleichen. Stellt man den Anteil der Aktionen des implementierten Pokerbots und Hari gegenüber, so ist zu bemerken, dass beide Pokerbots ungefähr gleich oft passen.

7. Zusammenfassung: Explaining Winning Poker

Johansson, Sönströd und Niklasson zeigen in (Johansson et al., 2006), wie man mit Machine Learning Methoden Regeln aus Daten von Pokerspielen extrahieren kann. Die Experimente und die Ergebnisse werden nun zusammengefasst.

7.1 Ansatz

Zwischen März und Mai 2006 wurden alle Spieltische mit sechs Spielern auf Ladbroke's Onlinepokerplattform beobachtet und mittels einer Pokersoftware namens PokerOffice wurden die Spieldaten extrahiert. Dabei wurden Spieler betrachtet, die mehr als 500 Hände gespielt hatten. Aus den Daten der 105 Spieler, die genug Hände gespielt hatten, wurden verschiedene Features für die jeweiligen Spieler berechnet (siehe Kapitel 2.2.2)und die Daten wurden in einer *MySQL* Datenbank gespeichert. Es wurde mit Hilfe von Wekas J48 und dem Regellernalgorithmus G-REX untersucht, ob sich aus den so gewonnenen Daten herausfinden lässt, woran man einen gewinnenden oder einen verlierenden Spieler erkennen kann.

Dazu wurden verschiedene Sortierungen der Beispiele untersucht und jeweils die obersten 25 Spieler zu einer Klasse gezählt. Mit den so erstellten Beispielen wurde für jede Sortierung versucht das Konzept mit den erstellten Entscheidungsbäumen zu beschreiben.

7.2 Ergebnis

Die Regeln von J48 waren aufgrund eingeschaltetem Pruning (siehe Kapitel 4.3) relativ kompakt hatten aber eine weniger gute Genauigkeit als G-REX. Die mit G-REX gefundenen Regeln waren in der Lage mit relativ kompakten Regeln eine gute Genauigkeit zu erzielen. Es zeigte sich, dass es anscheinend eine Korrelation zwischen verschiedenen Merkmalen und dem Gewinn bzw. Verlust von Spielern gibt und man mit Hilfe von Entscheidungsbäumen eventuell eine grundlegende Einteilung von Spielertypen erreichen kann, wenn man genug Daten eines Spielers gesammelt hat.

8. Fazit

Der Ansatz von (Blank, 2004) die Strategie eines Pokerspielers zu imitieren ist interessant, da bei erfolgreicher Anwendung lediglich die Spieldaten eines guten Pokerspielers oder -bots benötigt würden. Das schlechte Abschneiden des implementierten Pokerbots zeigt, dass es mit den untersuchten Merkmalen allein nicht möglich ist erfolgreich zu spielen. Hier wäre es interessant weitere Merkmale zu betrachten. Die Autoren wollen in weiteren Studien auch auf die Position eines Spielers eingehen. Das Experiment von (Johansson et al., 2006) basiert auf einer relativ kleinen Datengrundlage. Die Ergebnisse zeigen, dass durch die mit G-REX gefundenen Entscheidungsbäume die jeweiligen Spielergruppen gut klassifiziert werden können. Ein denkbarer Einsatz wäre das Finden von grundlegenden Regeln für das Spielverhalten oder das Klassifizieren von Gegnern auf Grund beobachteter Daten. Hiefür müssten allerdings die Daten einer größeren Spielerzahl zur Verfügung stehen.

Literatur

Blank, T. Leen-Kiat Soh Scott, S. (2004). Creating an SVM to Play Strong Poker.. pp. 150– 155.

Burges, C. J. C. (1998). A Tutorial on Support Vector Machines for Pattern Recognition. *Data Mining and Knowledge Discovery*, *2*(2), 121–167.

Cristianini, J. S.-T. . N. (2004). *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Harrington, D., & Robertie, B. (2004). *Harrington on Hold 'Em, Volume 1: Expert Strategy for No Limit Tournaments: Strategic Play: Expert Strategy for No Limit Tournaments: 1*. Two Plus Two Publishing LLC.

Joachims, T. (1999). Making large-scale SVM learning practical. Advances in Kernel Methods - Support Vector Learning, B. Schoelkopf and C. Burges and A. Smola (ed.), MIT-Press.

Johansson, U., Sonstrod, C., & Niklasson, L. (2006). Explaining Winning Poker–A Data Mining Approach. In *ICMLA '06: Proceedings of the 5th International Conference*

on Machine Learning and Applications, pp. 129–134, Washington, DC, USA. IEEE Computer Society.

Miller, E., Sklansky, D., & Malmuth, M. (2004). *Small Stakes Hold'em: Winning Big with Expert Play* (1 edition). Two Plus Two Publishing LLC.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers.

Schoelkopf, B. (1997). Support Vector Learning. Tech. rep..

Schoelkopf, B., & Smola, A. J. (2002). *Learning with Kernels*. The MIT Press, Cambridge, MA.