Untersuchung und Entwicklung von Verfahren zum kooperativen Maschinellen Lernen in Multi-Agenten Systemen

Michael Wurst

25. September 2001

Inhaltsverzeichnis

Abbildungsverzeichnis

1 Einleitung

Maschinelles Lernen spielt schon seit dem Beginn der Forschung im Bereich der Künstlichen Intelligenz (KI) eine zentrale Rolle. Wissenschaftliche Arbeit auf dem Gebiet des Maschinellen Lernens wurde bislang hauptsächlich von Vertretern der klassischen KI betrieben. In letzter Zeit wird aber versucht, diese Forschung auch in den Bereich der Verteilten Künstlichen Intelligenz und damit vor allem in den Bereich der Multi-Agenten Systeme (MAS) zu tragen.

Die vorliegende Arbeit hat zum Ziel, systematisch Untersuchungen im Bereich des Multi-Agenten Lernens durchzuführen. Der Schwerpunkt soll dabei auf dem Vergleich von Verfahren liegen, die es Agenten in einem Multi-Agenten System erlauben, sich gegenseitig beim Lernen eines Konzepts zu unterstützen, um damit die Gesamtleistung des Systems zu steigern. Dazu wird zunächst ein Szenario entworfen, welches zum einen genau festlegt, wie diese Leistung ermittelt wird, zum anderen eine Anzahl von Parametern enthält, welche eine Variation des Szenarios zulassen. Auf dieser Grundlage sollen drei Ansätze des kooperativen Lernens verglichen werden: der Austausch von Erfahrung, der Austausch von Hypothesen und das Abgleichen von Entscheidungen zwischen Agenten. Neben einer theoretischen Analyse sollen dabei mit Hilfe einer Simulation Experimente durchgeführt werden, die einen quantitativen Vergleich der Verfahren ermöglichen. Dabei steht die Frage im Mittelpunkt, wie sich eine Änderung der Parameter des Szenarios auf den Erfolg der einzelnen Verfahren auswirkt.

Die Arbeit ist folgendermaßen aufgebaut:

Abschnitt 2 gibt einen kurzen Überblick zum Thema Multi-Agenten Systeme und stellt das Stuttgarter Agentenmodell vor.

Der dritte Abschnitt enthält eine Einführung in die Thematik des Maschinellen Lernens, sowie des Multi-Agenten Lernens. Zusätzlich werden verschiedene Aspekte des Konzeptlernens und Verfahren zum Aufbau von Entscheidungsbäumen erörtert.

In Abschnitt 4 wird das Lernszenario vorgestellt, welches die Grundlage der weiteren Analyse einzelner Kooperationsverfahren bildet.

Abschnitt 5 beschäftigt sich mit der Frage, wie eine Gruppe nicht-kooperierender Agenten für dieses Szenario aufgebaut sein könnte.

Der zentrale Abschnitt dieser Arbeit ist Abschnitt 6. An dieser Stelle werden die einzelnen Verfahren zum kooperativen Lernen vorgestellt und analysiert.

An diese Analyse schließt sich eine experimentelle Untersuchung an. Die Ergebnisse dieser Untersuchung, die mit Hilfe einer Simulation gewonnen werden, sollen in Abschnitt 7 dargestellt werden.

Abschnitt 8 gibt schließlich eine Zusammenfassung und ein Fazit der gesamten

Arbeit, sowie einen Ausblick auf offene Fragen und Probleme, die im Rahmen dieser Arbeit nicht behandelt werden konnten.

Folgende Bezeichnungen werden in der gesamten Arbeit durchweg verwendet: n für die Anzahl von Attributen, m für die Anzahl an Trainingsbeispielen und k für die Anzahl von Agenten.

2 Multi-Agenten Systeme

2.1 Begriffsbestimmung

Wie der Name schon sagt, handelt es sich bei Multi-Agenten Systemen zunächst um eine Ansammlung von Agenten. Was ist ein Agent? Obwohl besonders in den letzten Jahren eine Vielzahl verschiedenster Definitionen für diesen Begriff gegeben wurde, lassen sich einige zentrale Charakteristika angeben, die den meisten dieser Definitionen zu Grunde liegen. Ein Agent ist demnach ein intelligentes System mit eigenem Wissen und eigenen Zielen, welches zu einem gewissen Grad *rational*, *autonom* und *sozial* ist.

Rationalität bedeutet in diesem Zusammenhang meist, dass die Agenten ihre Handlungen so wählen, dass diese möglichst optimal zu Erreichung ihrer Ziele sind.

Autonomie wird oft als Fähigkeit eines Agenten erklärt, in einem gewissen Grade unabhängig von seiner Umwelt Entscheidungen zu fällen und Aktionen auszuführen. Tatsächlich bedeutet aber Autonomie mehr: es ist die Fähigkeit sich selbst ein Gesetz zu geben. Bezogen auf intelligente Agenten bedeutet dies, dass sie über die Fähigkeit verfügen, Handlungsregeln zu wählen, sie zu repräsentieren und ihr Verhalten an diesen Regeln zu orientieren. Auf diese Weise können sie beispielsweise Vereinbarungen mit anderen Agenten eingehen, und die anderen Agenten haben zumindest eine Grundlage anzunehmen, dass sie sich an die Vereibarung halten.

Sozialität bezieht sich auf die Fähigkeit des Agenten mit anderen Agenten zu kommunizieren und zu kooperieren.

2.2 Das Stuttgarter Agentenmodell

In [8] beschreibt P. Levi ein abstraktes Agentenmodell, welches in diesem Abschnitt grob dargestellt werden soll. Kern des Konzepts bildet der sogenannte Autonomiezyklus, der in Abbildung 1 dargestellt ist.

Ein Agent besteht aus vier Komponenten:

1. *Die Weltmodelle*
   Hier werden Informationen über die möglichen Objekte der Umwelt gespeichert, über Zustände dieser Objekte und mögliche Aktionen. Zusätzlich sind hier Informationen über die Ziele des Agenten gespeichert. Dies ist die Wissensbasis des Agenten.

2. *Die Pläne*
   Diese Menge enthält alle möglichen Pläne des Agenten.

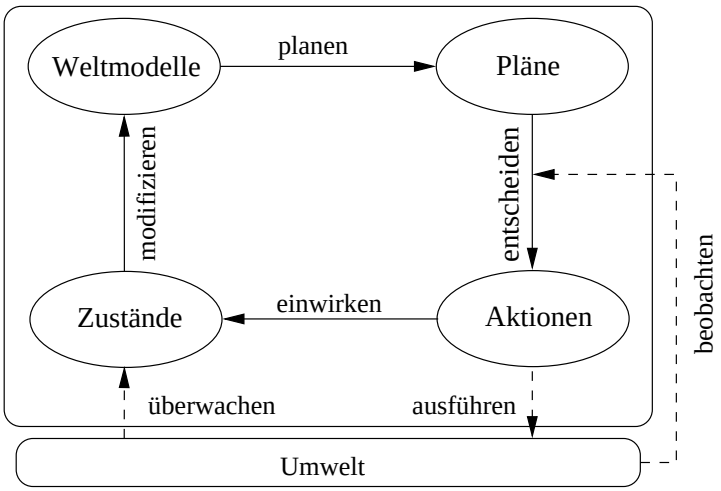


Abbildung 1: Der Autonomiezyklus

3. *Die Aktionen*
   Diese Menge umfasst alle möglichen Aktionen, die der Agent ausführen kann.

4. *Die Zustände*
   Dies sind alle möglichen externen Zustände, also Zustände welche die Welt annehmen kann. Im Gegensatz zum Weltmodell, das im Prinzip statisch ist, gibt diese Menge die veränderlichen Eigenschaften der Umwelt wieder.

Diese vier Mengen sind durch eine Reihe von Funktionen verbunden. Diese kann man sich als asynchron ablaufende Prozesse vorstellen:

1. *planen*
   Auf Grundlage seines Weltmodells und der Zustände der Umwelt, entwirft der Agent verschiedene Pläne.

2. *beobachten*
   Dann beobachtet er die Umwelt, wobei er eventuell nur einen Teil der externen Zustände wahrnehmen kann.

3. *entscheiden*
   Auf Grundlage dieser Beobachtung, wählt er den geeignetsten Plan, und davon ausgehend eine konkrete Aktion aus.

4. *ausführen*
   Die gewählte Aktion wird ausgeführt.

5. *einwirken*
   Auf Grundlage seines Wissens über die Wirkung einer bestimmten Aktion, bei bestimmten Zuständen der Umwelt, ermittelt der Agent, wie sich die Umwelt verändert haben müsste.

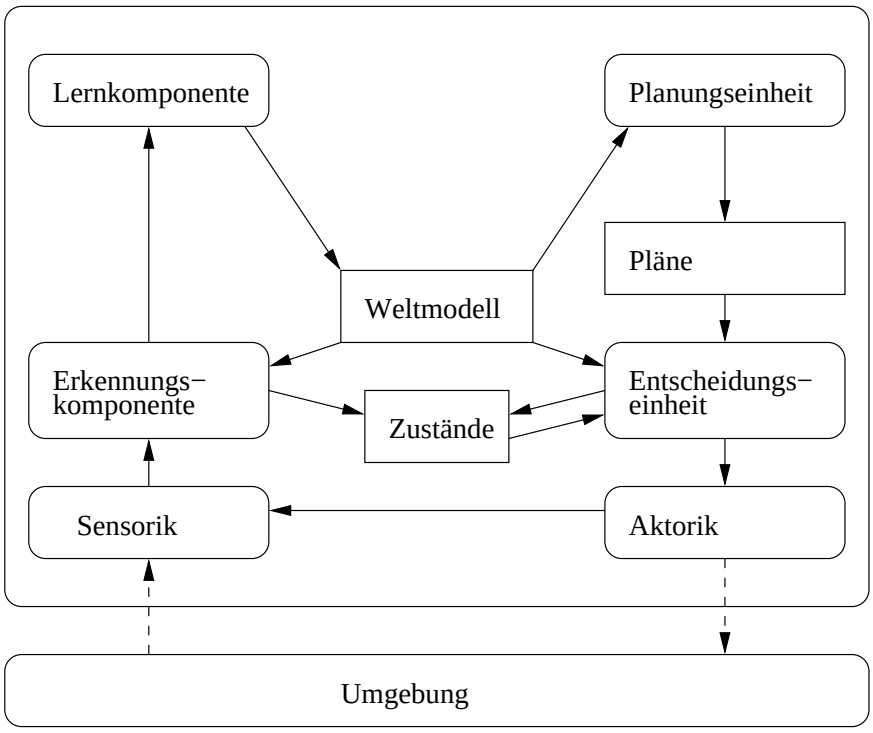


Abbildung 2: Abstraktes Agentenmodell

6. *überwachen*
   Da es aber nicht sicher ist, dass die Aktion erfolgreich war, wird nun nochmals die tatsächliche Veränderung der Umwelt beobachtet. Dies macht die Funktion *einwirken* normalerweise nicht überflüssig, denn der Agent kann oft nur einen Teil der Zustände der Umwelt wahrnehmen. Vor allem aber wird es erst so möglich zu ermitteln, ob eine Aktion wirklich erfolgreich war, d.h. ob Soll- und Ist-Zustand übereinstimmen.

7. *modifizieren*
   Wenn die Aktion nicht die gewünschte Wirkung gezeigt hat, dann muss das Weltmodell modifiziert und eventuell ein neuer Plan gestartet werden. Wie das Weltmodell modifiziert wird, legt das so genannte *Adaptionswissen* fest.

Um ein etwas konkreteres Bild eines Agenten zu erhalten, der auf dem Autonomiezyklus basiert, muss man das abstrakte Schema etwas konkretisieren. Dies zeigt beispielsweise Abbildung 2. Zunächst wird mit Hilfe des Weltmodells von der Planungseinheit eine Menge von Plänen erzeugt. Die Entscheidungseinheit wählt mit Hilfe der Zustände einen Plan aus, und gibt die entsprechende Aktionsanweisung an die Aktorik weiter. Außerdem wird festgehalten, welcher Zustand sich durch diese Aktion einstellen müsste. Durch die Sensorik und die Erkennungskomponente wird nun der tatsächliche Zustand der Umwelt festgestellt. Dies dient zum einen dazu, die Information über den Zustand der Umwelt

für weiteres Planen festzuhalten. Zum anderen kann es, vor allem wenn die Aktion ihr Ziel nicht erreicht hat, dazu dienen, das Weltmodell zu verändern. Dafür ist die Lernkomponente verantwortlich, sie verändert das deklarative Wissen im Weltmodell.

Befinden sich mehrere Agenten in einer gemeinsamen Umwelt, so muss das Modell eines Agenten entsprechend erweitert werden. Das Weltwissen enthält nun nicht nur Wissen über die Objekte der Umwelt, sondern auch über die anderen Agenten. Außerdem enthält es Informationen über Möglichkeiten des verteilten Problemlösens, der Kommunikation und Organisation.

Die Menge der Aktionen eines Agenten wird um die gruppenbezogenen Aktionen erweitert, zu denen auch das Kommunizieren gehört.

Die Menge der Pläne enthält nun nicht nur eigene Pläne, sondern auch gruppenbezogene Pläne.

Es tritt eine neue Funktion *kommunizieren* hinzu, welche es ermöglicht über eine festgelegte Sprache mit den anderen Agenten zu kommunizieren. Die Funktionen *planen* und *entscheiden* werden so erweitert, dass sie jeweils die Möglichkeit der Auswahl zwischen eigenen und gruppenbezogenen Plänen und Aktionen ermöglichen.

Letztlich wird auch das *Adaptionswissen*, also das Wissen, welches beschreibt, wie das Weltmodell geändert wird, so erweitert, dass es zum einen erlaubt auch gruppenspezifisches Wissen anzupassen, zum anderen den Wissensaustausch zwischen den Agenten koordiniert.

3 Maschinelles Lernen

3.1 Begriffsbestimmung

Den Begriff "Lernen" näher zu bestimmen ist ähnlich schwierig, wie den Begriff "Intelligenz" klar zu bestimmen. Diese Schwierigkeit überträgt sich auf die Definition von "Maschinellem Lernen". Vom Ergebnis her betrachtet, kann man Maschinelles Lernen beispielsweise als automatisierten Erwerb von Wissen oder von Fertigkeiten sehen. Legt man den Schwerpunkt mehr auf den Vorgang des Lernens, so kann Maschinelles Lernen z.B. als automatische Generalisierung aus Erfahrungen gefasst werden. In technischen Systemen wiederum bedeutet Maschinelles Lernen oft das Anpassen einiger interner Parameter an äußere Umstände. Aus dieser Vielzahl an Aspekten folgt auch eine Vielzahl an verschiedenen Definitionen. In dieser Arbeit soll ein Ansatz zur Definition von Maschinellem Lernen verfolgt werden, der vollständig von der Frage *wie* gelernt wird abstrahiert. Das lernende Programm wird als Black-Box behandelt und nur nach seinem nach außen hin sichtbaren Verhalten beurteilt. Dem entspricht die folgende Definition von T. Mitchell.

> "A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E"([9] S.10)

Nach dieser Definition müssen also, wenn man über lernende Systeme spricht, zunächst die drei Komponenten T, E und P genau angegeben werden. Während die Lernaufgabe, also T, meist durch die Zielsetzung des Systems festgelegt ist, und die Erfahrung E durch die dem System zugänglichen Informationen, kommt der genauen Festlegung der Leistungsbewertung P eine besondere Rolle zu. Erst wenn diese Leistungsbewertung feststeht, ist eine Beurteilung des Erfolgs des Systems möglich. Ebenso können verschiedene Lernverfahren erst dann sinnvoll verglichen werden. Letztlich kann sich nach einer Festlegung der drei Komponenten sogar zeigen, dass das Lernproblem nicht lösbar ist, wenn sich nämlich herausstellt, dass mit der zur Verfügung stehenden Erfahrung E, gar nichts gelernt werden kann, was zu einer Verbesserung der Leistung in T, gemessen mit P führt.

Die Definition von T. Mitchell stellt somit ein äußerst wichtiges Instrument zur Analyse von Lernproblemen dar, und wird für die weitere Arbeit zu Grunde gelegt[1].

[1] Die Definition von T. Mitchell ist insofern umstritten, da sie gewisse Vorgänge umfasst, die nach alltäglichem Sprachgebrauch nicht als Lernen bezeichnet werden, wie beispielsweise das Einspeichern von Werten in eine Datenbank. Außerdem verfehlt die Definition nach Ansicht einiger Wissenschaftler gerade dadurch, dass sie die Frage nach dem *wie* des Lernens ausklammert, den Kern dessen, was mit Lernen gemeint ist.

3.2 Lernen eines einzelnen Agenten

3.2.1 Was ist ein lernender Agent?

Nimmt man zum einen das in Abschnitt 2 vorgestellte Agentenmodell zur Hand, zum anderen die Definition Maschinellen Lernens von T. Mitchell, dann kann man das Lernen eines Agenten als selbstständiges Verändern seiner Wissensbasen auffassen, welches die Leistung des Agenten bezüglich seiner Aufgaben steigert und welches auf Grundlage der Erfahrung des Agenten erfolgt.

Um also sinnvollerweise von einem lernenden Agenten sprechen zu können, muss man zum einen festlegen, was die Aufgaben sind, in denen er sich durch Lernen verbessern soll, zweitens wie seine Leistung in diesen Aufgaben bewertet wird und drittens welche Erfahrung ihm dabei zur Verfügung steht. Das Agentenmodell stellt dann in einem zweiten Schritt ein Instrument dar, mit dem analysiert werden kann, welche Information in den Wissensbasen auf welche Weise verändert werden muss, damit es zu einer solchen Leistungssteigerung kommt.

3.2.2 Warum sollten Agenten lernfähig sein?

Man kann sich fragen, warum es überhaupt interessant ist, Agenten mit der Fähigkeit zu lernen auszustatten. Ein Grund dafür liegt sicherlich darin, dass die Entwickler eines intelligenten Agenten zur Zeit seiner Entwicklung noch nicht das Einsatzgebiet des Agenten vollständig voraussehen können. Beispielweise muss ein Agent der Waren auf einem digitalen Markt handelt mit Agenten verhandeln, deren Verhalten er nicht kennen kann. Indem er die Fähigkeit besitzt, die Strategie des Gegners zu lernen, kann er seine eigene Strategie dynamisch an die Situation anpassen. Ein anderes Beispiel dieser Art ist ein Agent der einen fremden Planten explorieren soll, von dem nicht bekannt ist, welche Bedingungen dort herrschen. In all diesen Fällen ist die Fähigkeit zu lernen zentral für den Erfolg des Systems.

Selbst wenn das Verhalten der Umwelt, in welcher der Agent eingesetzt wird, bekannt ist, kann es von Vorteil sein diesen Agenten mit Lernfähigkeit auszustatten. Der Grund dafür liegt darin, dass es schwierig ist die Stabilität und Optimalität eines komplexen Systems zu gewährleisten, da die Anzahl der möglichen Situationen, die eintreten können, zwar vielleicht endlich, aber schwer überblickbar ist. In diesen Fällen kann es einfacher sein, das System so zu entwerfen, dass es sich zur Laufzeit selbst optimiert und stabilisiert. Dies ist nun aber oft an Adaptions- und Lernprozesse gekoppelt.

Letztlich legt es das Bild eines autonomen Agenten nahe, dass Agenten fähig sein sollten, ihr Wissen selbständig dynamisch zu verändern und zu erweitern. In diesem Sinne ist Lernfähigkeit eine natürliche Fortführung des Agenten-Ansatzes überhaupt.

3.2.3 Möglichkeiten des Lernens anhand des Agentenmodells

Die Möglichkeiten, Maschinelles Lernen in einem Agenten einzusetzten, lassen sich anhand dessen charakterisieren, was gelernt werden soll, wie es gelernt werden soll und welche Art von Erfahrung dazu zur Verfügung steht. Darüber hinaus kann man noch verschiedene Lernstrategien unterscheiden.

Was wird gelernt?

Es wird hier angenommen, dass der Agent durch Lernen nur sein deklaratives Wissen anpasst, also nicht sein prozedurales Wissen. Dies stellt keine wesentliche Einschränkung dar, da das Wissen über mögliche Pläne und Strategien in dem hier zu Grunde liegenden Agentenmodell als deklaratives Wissen gefasst ist. Was der Agent im einzelnen lernen kann, lässt sich anhand der verschiedenen Wissensbasen einteilen (wobei diese Aufzählung keinen Anspruch auf inhaltliche Vollständigkeit erhebt):

- Weltwissen
  - Welche Objekte gibt es in der Umgebung, welche Eigenschaften haben sie, wo befinden sie sich und wie stehen sie in Relation zueinander?
  - Welche Ressourcen gibt es, wie lassen sie sich nutzen?
  - Wie verändert sich die Umwelt (bzw. einzelne Objekte der Umwelt) dynamisch?
  - Wie lässt sich die Umwelt in Situationen einteilen und wie lassen sich diese charakterisieren?
- Planungs- und Aktionswissen
  - Welche Operatoren lassen sich auf welche Objekte anwenden, was sind die Bedingungen für die Anwendung eines Operators?
  - Welche Interaktionsmuster können mit Objekten der Umwelt auftreten?
  - Welche Strategien sind am besten geeignet, um das Gesamtziel zu erreichen?
  - Welche Unterziele müssen erreicht werden, um das Hauptziel zu erreichen ?
- Adaptionswissen

  Da gelernt werden soll, was oder wie gelernt werden soll, handelt es sich bei Änderungen am Adaptionswissen um sogenanntes Meta-Lernen. Beispiele für Meta-Lern-Aufgaben sind:
  - Welche Lernmethode funktioniert bei der gegebenen Erfahrung am besten?

– Wann soll gelernt werden und wann nicht?

Wie wird gelernt?

Das Lernen eines Agenten wurde definiert als das selbständige Verändern der Wissensbasen des Agenten. Also geht es bei der Frage, wie gelernt wird, um das Problem, wie ausgehend von Erfahrung die Wissensbasen des Agenten geändert werden. Meistens werden folgende grundlegende Methoden unterschieden:

- direkte Beobachtung von Fakten (z.B.: Bestimmung der eigenen Position durch GPS)
- Beobachtung und Transformation durch weiteres Wissen (z.B.: Agent lernt die Größe eines Objekts durch seine sensorischen Daten und sein Wissen über dreidimensionale Objekte)
- Lernen durch Generalisierung (z.B.: der Agent lernt Objekte zu unterscheiden und zu erkennen, indem ihm eine Reihe von Mustern zusammen mit den jeweiligen Objektbezeichnungen vorgelegt wird)
- Lernen durch Analogie (z.B.: der Agent kann lernen, wie auf ein Objekt eingewirkt werden kann, indem er sein Wissen von einem ähnlichen Objekt auf jenes überträgt)

Auf Grundlage welcher Erfahrung wird gelernt?

Oft wird die Art der Erfahrung danach eingeteilt, wie das zu lernende Konzept aus der Erfahrung gewonnen werden kann. Vor allem werden zwei Arten des Lernens unterschieden:

- *supervised learning*
  Der Agent erhält Beispiele des zu lernenden Wissens mit einer direkten Bewertung und kann aus diesen durch Generalisierung oder Analogie lernen. In manchen Fällen besteht die Möglichkeit, dass der Agent die Beispiele selbst aussucht. Beispielsweise könnte ein Agent der geschriebene Buchstaben erkennt, seinem Lehrer Muster vorlegen und von diesem dann die Information erhalten, um welchen Buchstaben es sich handelt.
- *re-inforcement learning*
  Der Agent erhält nur Rückmeldung über den Erfolg seiner Aktionen im Ganzen. Er muss selbständig auf die Ursachen von nicht erfolgreichem Verhalten kommen. Wenn er eine Reihe von Aktionen ausführt so erhält er nur die Information, ob diese Aktionen im Ganzen zum gewünschten Erfolg geführt haben, nicht welche der einzelnen Aktionen dafür verantwortlich war. Das führt dazu, dass er seine Wissensbasen nicht direkt anpassen kann, sondern zunächst analysieren muss, welche Information in den Wissensbasen zu dem schlechten Ergebnis seiner Handlungen geführt hat.

Lernstrategien

Der Agent kann seine Umwelt entweder passiv wahrnehmen und daraus lernen, oder diese aktiv explorieren, d.h. aktive "Experimente" in Form von Handlungen vornehmen. Im letzteren Fall ergeben sich einige interessante Probleme. Die Exploration neuer, noch nicht bekannter Möglichkeiten kann nämlich in vielen Fällen kostspielig oder sogar gefährlich sein und es ist oft nicht klar welchen Nutzen sie bringt. Wenn ein Agent sich andererseits zu vorsichtig verhält, kann er eventuell auf einem "lokalen Maximum" stehenbleiben und sich so langfristig suboptimal verhalten.

3.3 Multi-Agenten Lernen

3.3.1 Grundlagen

Durch die Anwesenheit mehrerer Agenten in einer Umwelt ergeben sich neue Probleme, aber auch Möglichkeiten für die Anwendung Maschinellen Lernens. Das Forschungsgebiet, das sich mit Fragen des Multi-Agenten-Lernens auseinandersetzt ist relativ jung und besteht hauptsächlich aus Untersuchungen zu Einzelproblemen.

Dabei ist schon der Begriff "Multi-Agenten Lernen" selbst nicht einfach zu fassen. In [18](S. 5) wird ein schwacher und ein starker Begriff von "Multi-Agenten-Lernen" unterschieden.

Der starke Begriff besagt, dass man nur dann von Multi-Agenten Lernen reden kann, wenn die Agenten ein gemeinsames Lernziel verfolgen. Der schwache Begriff besagt hingegen nur, dass sich die Agenten beim Lernen gegenseitig beeinflussen. Für diese Arbeit wird der schwache Begriff zu Grunde gelegt.

In jedem Fall gilt dasselbe wie im Fall des Lernens eines einzelnen Agenten: bevor man sinnvollerweise von einem lernenden MAS reden kann, muss man genau festlegen, was die Aufgabe der Agenten ist, welche durch Lernen verbessert werden soll, wie ihre Leistung bezüglich dieser Aufgabe gemessen wird und welche Erfahrung ihnen dabei zur Verfügung steht. Beispielsweise kann die Gesamtleistung einfach die Summe der Einzelleistungen der Agenten sein. Wenn die Agenten zusammen an einer Aufgabe arbeiten, dann wird vielleicht nur die Leistung bezüglich der Gesamtaufgabe interessieren, nicht die Einzelleistung der Agenten, usw.

In diesem Abschnitt soll analog zum vorhergehenden Abschnitt anhand des Agentenmodells untersucht werden, welche weiteren Möglichkeiten es zum Lernen in einem MAS gegenüber dem Lernen einzelner Agenten gibt. Danach soll eine kurze Übersicht zu bestehenden Forschungsarbeiten auf dem Gebiet des Multi-Agenten Lernens gegeben werden.

3.3.2 Möglichkeiten des Multi-Agenten Lernens anhand des Agentenmodells

Ebenso wie die Wissensbasen eines Agenten in einer Umwelt, in der sich mehrere Agenten befinden, erweitert werden können, erweitern sich auch die Möglichkeiten, was gelernt werden kann. Außerdem ergeben sich weitere Möglichkeiten dafür, wie gelernt werden kann und damit auch für Lernstrategien.

Was wird gelernt?

Aus dem hier zu Grunde gelegten Agentenmodell ergeben sich unter anderem folgende Möglichkeiten:

- Weltwissen
  - Welche Eigenschaften und Fähigkeiten haben die anderen Agenten, wie verhalten sie sich?
  - Welche Absichten bzw. Ziele haben die anderen Agenten?
  - Welches Wissen haben die anderen Agenten?
  - Welche Ressourcen benötigen die anderen Agenten?
- Planungswissen
  - Lernen von Kooperation: gemeinsame Operatoren, Restriktionen, Koordination von Aktionen, gemeinsame Strategien
- Adaptionswissen

  Von welchen Agenten kann was gelernt werden ? Wie verlässlich sind die Informationen eines bestimmten Agenten?

Wie wird gelernt?

Auch die Möglichkeiten, wie gelernt wird, erweitern sich in einer MAS-Umwelt. Dabei sind Lernmethoden und Art der Erfahrung nicht sinnvoll trennbar, so dass beide Punkte in der folgenden Aufzählung gemeinsam behandelt werden.

- Lernen durch direkten Informationsaustausch, eventuell mit einer zusätzlichen Transformation (z.B. kann ein Agent einem anderen seine Position relativ zu einem festen Punkt mitteilen, woraus dann dieser die absolute Position von jenem errechnen kann)
- Lernen durch Beobachtung anderer Agenten (z.B. kann ein Agent einen anderen bei einer Aktion beobachten und dieses Wissen in sein Planungswissen einbauen)

- Lernen durch Feedback anderer Agenten (z.B. können Agenten nach Durchführung einer gemeinsamen Aktion den Beitrag der anderen Agenten bewerten)

Lernstrategie

Ein Agent kann beim Lernen mehr oder weniger kooperationsbereit sein. Vor allem wenn es sich um ein System handelt, in welchem die Agenten miteinander in Konkurrenz stehen, ist es für die Agenten wichtig, zu entscheiden, welche Informationen oder Hilfe sie an andere Agenten weitergeben. Sind sie zu "hilfsbereit", dann sinkt eventuell ihre Leistung relativ zu den anderen Agenten. Sind sie zu verschlossen, dann entgehen ihnen vielleicht wichtige Informationen, die sie von anderen Agenten erhalten hätten.

3.3.3 Einzelne Forschungsarbeiten

Ein guter Ausgangspunkt zur Literatursuche im Bereich des Multi-Agenten-Lernens sind zwei "proceedings" von Konferenzen zu diesem Thema [18] und [19], sowie das entsprechende Kapitel in [20].

Die größte Aufmerksamkeit wurde Problemen im Zusammenhang mit re-inforcement-Lernen in MAS gewidmet. Zum einen führt die Anwesenheit von mehreren Agenten in einer gemeinsamen Umwelt zu Problemen, die eine Erweiterung der klassischen Algorithmen für re-inforcement-Lernen nötig machen, zum anderen kann re-inforcement-Lernen in MAS eingesetzt werden, um gemeinsame Aktionen zu koordinieren ohne dabei zu kommunizieren.

Ein Grund dafür, dass die klassischen Algorithmen in einer Multi-Agenten Umwelt nicht mehr ausreichen, ist, dass ein einzelner Agent nicht mehr einer mehr oder weniger gleichförmig reagierenden Umwelt gegenüber steht, sondern selbst wieder anderen Agenten, welche ebenso lernfähig sind. Beispielsweise könnte ein Agent nach der Kollision mit einem Objekt, die nicht sehr vorteilhaft für ihn war, immer nach links ausweichen, wenn er sich einem Objekt nähert. War dieses Objekt nun aber ein anderer Agent, der auch in Zukunft Kollisionen vermeiden will, der aber unglücklicherweise jetzt immer nach rechts ausweicht, dann haben beide ihr Ziel nicht erreicht, da es beim nächsten Zusammentreffen wieder zu einer Kollision kommt. Beide haben nicht einmal die Möglichkeit, die Ursache für das Problem zu entdecken, solange sie nicht kommunizieren.

Ein weiteres Problem, welches auftreten kann, ist folgendes: wenn mehrere Agenten gemeinsam eine Aktion ausführen, so ist oft nur der Erfolg oder Misserfolg der Gesamtaktion bekannt, und nicht, wie die einzelnen Agenten dazu beigetragen haben. Es muss eine Möglichkeit gefunden werden, kollektives in individuelles feedback umzuwandeln, um ein re-inforcement Lernen der einzelnen Agenten zu ermöglichen (siehe z.B. [17]).

Neben diesen Problemen, bietet re-inforcement-Lernen in MAS die Möglichkeit,

dass Agenten Aktionen koordiniert ausführen, ohne dabei explizit zu kommunizieren. Durch Versuch und Irrtum können sie sich so aufeinander einstellen, dass sie sich nicht gegenseitig behindern, sondern sich vielleicht sogar gegenseitig unterstützen. Man muss dabei zwischen Szenarien unterscheiden in denen die Agenten gegeneinander arbeiten (z.B. auf einem digitalen Marktplatz), Szenarien in denen sie miteinander arbeiten und so genannte "semi-competitive"-Szenarien (wie z.B. dem Gefangenendilemma, siehe [14]).

Ein weiteres Problem, welches untersucht wurde ist, wie ein Agent Wissen über andere Agenten in dem MAS erlangen kann. In [11] wird dies anhand des Problems erläutert, an welche Agenten eine Ausschreibung im Rahmen des contract-net-Protokolls gesendet werden soll. Indem der Agent lernt, welcher der anderen Agenten für eine gegebene Aufgabe am vielversprechendsten ist, kann er gezielter Anfragen stellen. In [10] wird ein simuliertes Roboterfußballsystem vorgestellt, in welchem die Agenten die Fähigkeit ihrer Mitspieler und Gegner lernen sollen. Besondere Aufmerksamkeit wurde auch dem Problem gewidmet, die Strategie eines Gegners bei Verhandlungen zu erlernen und seine eigene Strategie entsprechend anzupassen. Ausführungen dazu finden sich beispielsweise in [3].

Wissensaustausch zwischen Agenten ist ein weiteres Gebiet, in welchem Probleme des Multi-Agenten-Lernens diskutiert werden. Zum einen stellt sich die Frage, wie sich die Wissensbasen von verschiedenen Agenten aufeinander abstimmen lassen. Beispielsweise könnte ein Agent absichtlich oder versehentlich falsche Informationen verbreiten. Wie reagieren die anderen Agenten auf diese Information? Stellt sich ein stabiler Zustand der Wissensbasen ein? Solche Fragen werden unter anderem in [5] diskutiert. Allgemein mit dem Problem des Wissensaustauschs befasst sich ein Projekt namens "Knowledge Sharing Effort", welches unter anderem die Kommunikationssprachen KIF und KQML hervorgebracht hat. Während im Fall von KQML und KIF meist schon ein gemeinsames Verständnis (eine gemeinsame Ontologie) vorausgesetzt wird, gibt es auch Ansätze, dieses gemeinsame Verständnis zu erlernen (siehe [6]).

3.4 Konzeptlernen

3.4.1 Problemstellung

Klassifikation, und als Spezialfall davon Konzeptlernen, stellen eine wichtige Klasse von Lernproblemen dar. Diese Klasse zeichnet sich folgendermaßen aus: eine Menge von Entitäten soll in endlich viele disjunkte Klassen $K_1, .., K_m$ eingeteilt werden. Die Entitäten werden durch eine Anzahl von Attributen beschrieben $A_1, ...A_n$. Formal ist also das Ziel eine Funktion f, mit

$f : A_1 \times A_2 \times ... \times A_n \rightarrow \{K_1, K_2, ...K_m\}$

zu lernen.[2].

[2]Die Funktion f stellt dabei nur eine Annahme dar, d.h. man geht davon aus, dass eine solche

Im Fall von Konzeptlernen werden nur zwei Klassen zugelassen, die als Zugehörigkeit einer Entität zu einem Konzept interpretiert werden können, z.B. das Konzept aller Säugetiere. Es gibt dann nur die Klassen "nicht enthalten" und "enthalten", oder abgekürzt *false* und *true* . $A_1 ... A_n$ können im allgemeinen beliebige Mengen sein. Hier sollen aber aus Gründen der Einfachheit hauptsächlich Mengen angenommen werden, die genau zwei Elemente enthalten. Um eine einheitliche Schreibweise zu erhalten, werden diese beiden Elemente auch immer mit *true* und *false* bezeichnet.

Im Fall des überwachten Konzeptlernens erhält der Lernalgorithmus eine Anzahl von Trainingsbeispielen, welche alle die Form haben:

$((a_1, a_2, ..., a_n), c), c = f(a_1, a_2, ..., a_n)$

also den Verlauf der Funktion f an einzelnen Stellen angeben.

Würde der Lernalgorithmus für jede mögliche Kombination von Attributen ein solches Trainingsbeispiel erhalten, dann wäre die Funktion f vollständig bestimmt und Lernen wäre nichts anderes als Werte in eine Tabelle zu speichern. Normalerweise sind aber nicht alle möglichen Kombinationen von Attributen gegeben. In diesem Fall muss der Algorithmus die "Lücken" zwischen den gegebenen Werten schließen. Dabei wird er aber normalerweise nicht zu einer eindeutigen Funktion kommen, sondern zu verschiedenen Hypothesen über den tatsächlichen Verlauf von f. Diese Hypothesen haben dieselbe Form wie f also:

$h_i : A_1 \times A_2 \times ... \times A_n \rightarrow \{K_1, K_2, ...K_m\}$

Es wird angenommen, dass der Lernalgorithmus eine Menge von Trainingsbeispielen erhält, die zufällig aber nach einer festen Verteilung D gezogen werden. Daraufhin generiert der Lernalgorithmus eine Hypothese h. Der echte Fehler einer solchen Hypothese h ist die Wahrscheinlichkeit p, dass ein zufälliges nach Verteilung D gezogenes Beispiel mit h nicht übereinstimmt, also:

$P(((a_1, a_2, ..., a_n), c) \wedge h(a_1, a_2, ...a_n) \neq c))$

Neben dem echten Fehler einer Hypothese gibt es noch den Fehler, der sich als Anteil der falsch klassifizierten Beispiele aus der Menge der Trainingsbeispiele ergibt, den "sample error". Lernalgorithmen, die immer einen "sample error" von Null erzeugen, sich also perfekt an die Trainingsbeispiele anpassen, werden als konsistente Lernalgorithmen bezeichnet.

Funktion existiert und sich lernen lässt. Dies muss allerdings nicht immer der Fall sein. Der gesuchte Zusammenhang zwischen Attributen und Klasse könnte auch nur statistisch sein, d.h. die Funktion könnte für dieselbe Attributkombination verschiedene Werte annehmen, wäre also keine Funktion mehr. Im folgenden sollen allerdings nur Fälle betrachtet werden in denen diese Funktion existiert, also ein eindeutiger deterministischer Zusammenhang zwischen Attributen und Klasse besteht, so dass dieselbe Kombination von Attributen immer zu derselben Klasse führt.

3.4.2 Das Hypothesenraum-Modell

Jede Hypothese muss repräsentiert werden. Dies kann beispielsweise durch eine logische Formel geschehen, wie $h_1 \equiv a_1 \wedge a_2$. Die Menge aller Hypothesen, die ein bestimmtes Lernverfahren repräsentieren kann, wird als Hypothesenraum bezeichnet. Dieser kann endlich sein, wie es beispielsweise der Fall ist, wenn die Hypothesen als aussagenlogische Formeln repräsentiert werden, oder unendlich, wie beispielsweise im Fall vieler Künstlicher Neuronaler Netze.

Lernen kann aus dieser Sicht als Suche im Hypothesenraum betrachtet werden, denn die Aufgabe des Lernalgorithmus ist es, diejenige Hypothese aus dem Hypothesenraum auszuwählen, welche nach gewissen Kriterien am besten zu den gegebenen Trainingsbeispielen passt.

Aus dieser Sicht besteht ein Lernverfahren aus drei Komponenten: dem Hypothesenraum, einer Bewertungsfunktion für Hypothesen in Abhängigkeit der Trainingsbeispiele und einer Strategie in welcher Reihenfolge die Hypothesen des Hypothesenraums betrachtet werden.

Je mehr Elemente der Hypothesenraum enthält, desto aufwendiger gestaltet sich das Lernen. Andererseits muss sichergestellt sein, dass der Hypothesenraum eine Hypothese enthält, durch welche sich die gesuchte Funktion darstellen lässt. Wird beispielsweise als Hypothesenraum die Menge aller Konjunktionen von Literalen gewählt, so dass sich Hypothesen darstellen lassen wie $a_1 \wedge \neg a_3 \wedge a_4$, das tatsächliche Konzept lautet aber $(a_1 \vee a_2) \wedge \neg a_3 \wedge a_4$, dann wird sich diese Funktion nicht korrekt lernen lassen.

3.4.3 Konzeptlernen aus Sicht der Lerntheorie

Analog zur Komplexitätstheorie in der theoretischen Informatik, gibt es eine Theorie der Komplexität des Lernens. Dabei geht es nicht nur um Zeit- und Speicherplatzaufwand, sondern vor allem um die Frage, wieviele Trainingsbeispiele benötigt werden, damit eine bestimmte Lernaufgabe gelernt werden kann. Hier soll nur ein zentrales Ergebnis der Lerntheorie für Klassifikationsprobleme vorgestellt werden. Es geht darum, Aussagen zu treffen wie: "Um mit Wahrscheinlichkeit 95% eine Hypothese zu lernen, die in 80% aller Fälle ein Beispiel korrekt klassifiziert, werden mindestens 40 Trainingsbeispiele benötigt". Entscheidend für die Berechnung dieses Zahlenwerts ist die Anzahl der Elemente im Hypothesenraum H, wobei in diesem Zusammenhang nur endliche Hypothesenräume betrachtet werden.

Der Zusammenhang wird wie folgt hergeleitet[3]:

Es werden nur konsistente Lernalgorithmen betrachtet. Wenn ein solcher Lernalgorithmus eine Reihe D von m Trainingsbeispielen erhält, dann betrachtet

[3] siehe [9], Kapitel 7

er nur noch diejenigen Hypothesen $h \in H$ als möglich, die mit diesen Beispielen konsistent sind. Diese Menge von Hypothesen wird als "Version-Space" $VS_{H,D} \subseteq H$ bezeichnet.

Enthält dieser "Version-Space" nur Hypothesen deren echter Fehler kleiner als ε ist, dann ist sichergestellt, dass der Algorithmus keine Hypothese ausgibt, welche einen Fehler größer als ε hat, denn der Algorithmus wählt garantiert eine Hypothese aus $VS_{H,D}$. Die entscheidende Frage ist nun, wieviele Trainingsbeispiele (unabhängig vom benutzten Lernalgorithmus) mindestens nötig sind, um dies zu garantieren.

Die Idee zur Berechnung dieses Wertes besteht darin, dass man annimmt, es gäbe k Hypothesen $H_{err} \subseteq H$, die einen echten Fehler größer als ε haben. Eine dieser Hypothesen ist dann in $VS_{H,D}$ enthalten, wenn sie alle Trainingsbeispiele aus D korrekt klassifiziert, denn $VS_{H,D}$ enthält nur mit D konsistente Hypothesen. Die Wahrscheinlichkeit, dass ein $h \in H_{err}$ mit einem zufälligen Trainingsbeispiel konsistent ist, ist kleiner als $(1-\varepsilon)$, da angenommen wurde, dass h einen Fehler größer als ε hat. Da nun die Beispiele als unabhängig angesehen werden, ist die Wahrscheinlichkeit, dass h alle m Trainingsbeispiele aus D korrekt klassifiziert kleiner als $(1-\varepsilon)^m$. Die Wahrscheinlichkeit, dass $VS_{H,D}$ eine beliebige Hypothese aus H_{err} enthält lässt sich nach oben abschätzen, indem man die Einzelwahrscheinlichkeiten für jede dieser Hypothesen aufaddiert[4].

Dies führt für die Wahrscheinlichkeit, dass $VS_{H,D}$ mindestens eine Hypothese mit Fehler größer ε enthält, zu einer oberen Schranke von,

$k * (1-\varepsilon)^m$

da $H_{err} \subseteq H$ gilt $k \leq |H|$

weiterhin gilt die allgemeine Beziehung $(1-\varepsilon) \leq e^{-\varepsilon}$, für $0 < \varepsilon < 1$.

und somit: $|H| * e^{-\varepsilon m}$

Dies ist die obere Schranke für die Wahrscheinlichkeit, dass $VS_{H,D}$ eine Hypothese mit Fehler größer ε enthält. Man kann nun eine Schranke δ für diese Wahrscheinlichkeit vorgeben und damit die Anzahl der mindestens nötigen Trainingsbeispiele m_0 berechnen:

$|H| * e^{-\varepsilon * m_0} \leq \delta$

$m_0 \geq \frac{1}{\varepsilon} * (ln|H| + ln(\frac{1}{\delta}))$

Dies lässt sich interpretieren als: wenn die Wahrscheinlichkeit, dass $VS_{H,D}$ mindestens eine Hypothese mit Fehler größer als ε enthält, kleiner ist als δ, dann enthält $VS_{H,D}$ mit Wahrscheinlichkeit größer als $(1-\delta)$ keine Hypothese mit Fehler größer als ε. Um dies in jedem Fall sicherzustellen, benötigt ein beliebiger konsistenter Lernalgorithmus mindestens m_0 Trainingsbeispiele.

[4] $P(A \cup B) \leq P(A) + P(B)$

Beispiel: Der Hypothesenraum enthalte alle Konjunktionen von Literalen aus fünf verschiedenen binären Attributen. Dann ist $|H| = 3^5$. Um mit Wahrscheinlichkeit 95% sicherzustellen, dass ein Algorithmus eine Hypothese ausgibt, die einen Fehler kleiner als 20% hat, benötigt man mindestens

$\frac{1}{0.2} * (ln(3^5) + ln(\frac{1}{0.05})) \sim 43$

Beispiele.

Der Wert der Formel liegt vor allem darin, dass man Klassen von Lernproblemen aufstellen kann. Beispielsweise gibt es Lernprobleme, die sich nur durch eine exponentielle Anzahl von Beispielen, in Bezug auf die Anzahl der Attribute, lernen lassen. Dies gilt beispielsweise für aussagenlogische Formeln. Die Größe des Hypothesenraums beträgt hier $|H| = 2^{2^n}$. Eingesetzt in die Formel ergibt sich:

$m_0 > \frac{1}{\varepsilon} * (2^n + \ln(\frac{1}{\delta}))$

Die Anzahl der mindestens benötigten Beispiele steigt exponentiell mit der Anzahl der Attribute.

3.4.4 Einzelne Verfahren zum Konzeptlernen

Wie bereits oben beschrieben, kann (Konzept-)lernen als Suche in einem Hypothesenraum aufgefasst werden. Die verschiedenen Verfahren zum Konzeptlernen unterscheiden sich nun in drei Hinsichten:

1. Der Hypothesenraum selbst kann verschieden sein.
2. Die Funktion, welche die Hypothesen aus diesem Hypothesenraum daraufhin bewertet, wie gut sie zu den gegebenen Trainingsbeispielen passen, kann verschieden sein.
3. Die Suchstrategie, welche entscheidet in welcher Reihenfolge die Hypothesen des Hypothesenraums untersucht werden, kann verschieden sein.

Da das zweite und dritte Element vom ersten abhängen, lassen sich die verschiedenen Ansätze zum Konzeptlernen am besten anhand der verschiedenen Hypothesenräume einteilen:

1. *Regeln oder Regelmengen*
   Das gesuchte Konzept wird durch eine logische Formel bzw. Regel beschrieben. Diese Formeln können sowohl aussagenlogisch, wie auch prädikatenlogisch sein. Im letzteren Fall spricht man von ILP(Inductive Logic Programming).

Es gibt nun verschiedene Verfahren, die möglichen Regeln bei gegebenen Trainingsbeispielen zu untersuchen und zu bewerten. Eine Möglichkeit ist den "version space" explizit zu machen. Die Idee besteht darin, alle Elemente des Hypothesenraums aufzulisten. Danach werden nacheinander die Beispiele betrachtet und in jedem Schritt werden alle Hypothesen, die mit diesem Beispiel nicht konsistent sind (die also zu einer anderen Klassifizierung führen als die Klasse des Beispiels) aus der Liste der möglichen Hypothesen gelöscht, bis nur noch eine Hypothese übrig bleibt, welche dann die gesuchte Hypothese ist.
Ein weitere wichtige Klasse von Verfahren folgt dem Prinzip des "sequential covering". Die Idee besteht darin, eine Konjunktion aus Literalen zu finden, welche eine möglichst große Anzahl von positiven Beispielen abdeckt, aber keine negativen. Ist eine solche Regel gefunden, werden die positiven Beispiele, welche diese Formel abdeckt, aus der Trainingsmenge entfernt und das Verfahren wird wiederholt, bis keine positiven Beispiele mehr in der Regelmenge vorkommen. Daraus ergibt sich eine Menge von Konjunktionen, welche zu einer DNF zusammengefasst werden können.
Eine etwas speziellere Form von Regelmengen liegt den Verfahren zum Entscheidungsbaumaufbau zu Grunde. Diese werden im nächsten Abschnitt ausführlich besprochen.

2. *Künstliche Neuronale Netze* (KNN)
   Künstliche Neuronale Netze sind Strukturen, welche der Physiologie des menschlichen Gehirns nachempfunden sind. Eine Anzahl von Neuronen ist durch gewichtete Kanten verbunden. Jedes Neuron hat eine Schwellwertfunktion, welche in Abhängigkeit der Eingänge des Neurons entscheidet, ob der Ausgang des Neurons aktiviert ist. Die Gewichtungen der Kanten an den Eingängen entscheiden dabei über den Einfluss dieser Eingänge auf die Aktivierung. Jedes künstliche Neuronale Netz besitzt Eingänge und Ausgänge. Wenn KNN zum Konzeptlernen eingesetzt werden, dann haben sie für jedes verfügbare Attribut mindestens einen Eingang. Das gesamte Netz hat für jede Klasse einen Ausgang. Werden nun an den Eingängen bestimmte Attributwerte angelegt, so ergibt sich, nachdem alle künstlichen Neuronen geschaltet haben, ein Wert am Ausgang, welcher die Klasse des Beispiels angibt, dessen Attribute am Eingang angelegt wurden.

   Der Hypothesenraum eines KNN ist also ein Vektor von reellen Zahlen, welche die Gewichtungen der Kanten im Netz beschreiben. Lernen besteht darin, diese Gewichtungen anzupassen. Bei einer sehr populären Klasse von KNN, den "feed-forward"-Netzen, funktioniert das folgendermaßen: die Attribute eines Trainingsbeispiels werden an die Eingänge des Netzes angelegt und geprüft, ob das Netz die korrekte Klassifizierung am Ausgang liefert. Ist dies nicht der Fall dann wird untersucht, welche Gewichtungen dafür verantwortlich sind und diese werden entsprechend angepasst. Dieser Vorgang wird mit allen Beispielen, eventuell auch mehrfach, wiederholt, bis sich die Gewichtungen nicht mehr entscheidend ändern. Auf diese Weise ist eine Hypothese aus dem Hypothesenraum ausgewählt, die

zur Klassifikation von neuen Beispielen dienen kann.

3. *Case-Based-Reasoning (impliziter Hypothesenraum)*

   Diese Klasse von Verfahren bildet überhaupt keine explizite Hypothese, sie speichert vielmehr die Trainingsbeispiele, ohne sie zu verarbeiten. Um ein neues Beispiel zu klassifizieren wird dasjenige Trainingsbeispiel ausgewählt, welches dem zu klassifizierenden am ähnlichsten ist. Das neue Beispiel erhält dann dieselbe Klasse, wie das ihm ähnliche aus der Menge der Trainingsbeispiele. Die meisten Verfahren unterscheiden sich darin, wie die Ähnlichkeit zwischen zwei Beispielen definiert ist. Dies kann ein einfaches Entfernungsmaß sein, wenn die Beispiele als Punkte in einem Vektorraum aufgefasst werden. Es ist aber auch möglich, weitere Informationen, wie z.B. Wahrscheinlichkeiten, einzubauen, was dann zu entsprechend komplexeren Verfahren führt.

4. *Bedingte Wahrscheinlichkeiten*

   Der Hypothesenraum wird bei diesen Verfahren aus bedingten Wahrscheinlichkeiten zwischen den Attributen untereinander und der Klasse gebildet. Ein sehr einfaches Verfahren nimmt dabei an, dass die Wahrscheinlichkeiten der einzelnen Attribute unabhängig sind. Daher nennt sich dieses Verfahren "Naive Bayes".

   Ein komplexeres Verfahren bilden die so genannten Baysian Networks. Diese erlauben beliebige wechselseitige Abhängigkeiten zwischen den Attributen untereinander und der Klasse.

Daneben gibt es noch einige allgemeine Strategien, effizient im Hypothesenraum zu suchen. Zu diesen gehören beispielsweise Evolutionäre Algorithmen. Hypothesen werden als genetischer Code repräsentiert, der über viele Generationen mutiert und rekombiniert wird. Dabei werden in jeder Generation die besten Hypothesen (die mit dem kleinsten Fehler auf den Trainingsbeispielen) ausgewählt.

3.4.5 Konzeptlernen und Wissen

In vielen Fällen sind die möglichen Hypothesen nicht nur durch die gegebenen Trainingsbeispiele beschränkt, sondern auch durch allgemeines a priori Wissen (also Wissen, dass schon bekannt ist, ohne ein einziges Trainingsbeispiel zu betrachten). Beispielsweise könnte ein Agent aus früherer Erfahrung wissen, dass immer wenn Attribut $a_1 = true$, dann $a_3 = false$ (oder als Regel $a_1 \rightarrow \neg a_3$). Damit sind Hypothesen ausgeschlossen, in denen $a_1 \wedge a_3$ vorkommt, denn diese Kombination führt immer zur falschen Aussage. Durch Regeln dieser Art lässt sich also der Hypothesenraum noch vor der Betrachtung von Trainingsbeispielen einschränken. Werden die Hypothesen als logische Formel repräsentiert kann man dies formal schreiben als

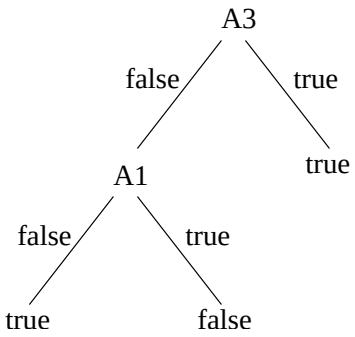


Abbildung 3: Entscheidungsbaum - Beispiel

$HW \wedge h_i$

Wobei HW die Menge des Hintergrundwissens ist und nur solche Hypothesen $h_i \in H$ betrachtet werden, für die diese Formel erfüllbar ist.

3.5 Entscheidungsbäume

3.5.1 Definition

Eine wichtige Klasse von Lernalgorithmen zum Konzeptlernen baut auf so genannten Entscheidungsbäumen auf. Abbildung 3 zeigt einen solchen Entscheidungsbaum.

Um ein Beispiel mit diesem Baum zu klassifizieren, z.B. $((true, false, false), ?)$ geht man von der Wurzel aus. Der Attributname an der Wurzel entscheidet, nach welchem Attribut als erstes unterschieden wird (hier: A_3). Man folgt dann derjenigen Verzweigung, die den tatsächlichen Wert des Beispiels für dieses Attribut hat (also hier: *false*). Am den folgenden inneren Knoten geschieht dasselbe, bis man zu einem Blatt kommt. Dieses Blatt enthält den Wert des Klassenattributs (hier: *false*).

Formal handelt es sich bei einem Entscheidungsbaum um einen Baum $B = (V, E)$.

Sei A die Menge aller Attributnamen $\{A_1, ...A_n\}$ mit ihnen jeweils korrespondierenden Wertemengen $W_1, .., W_n$ sowie K die Menge aller möglichen Klassen $K = \{K_1, ..., K_l\}$. Dann gilt $V = A \cup K$ und $E = \bigcup W_i$. Es gelten allerdings noch weitere Bedingungen, welche die Menge der so möglichen Bäume einschränken:

- Alle inneren Knoten sowie die Wurzel des Baums sind aus A. Alle Blätter aus K.
- Die Kanten sind jeweils aus der Wertemenge W_i, welche durch den Attributnamen am Ausgangsknoten dieser Kante bezeichnet ist.

- Es gilt weiterhin, dass von jedem inneren Knoten mindestens zwei Kanten ausgehen und dass alle Kanten, die von einem inneren Knoten ausgehen, verschieden sind.
- Außerdem darf sich auf keinem Weg von der Wurzel zu einem Blatt ein Attributname wiederholen.

Im folgenden beschränkt sich die Darstellung auf Entscheidungsbäume, die nur aus binären Attributen und einer binären Klasse aufgebaut sind.

3.5.2 Algorithmen zum Aufbau von Entscheidungsbäumen

Lernen mit Entscheidungsbäumen besteht darin, anhand der gegebenen Trainingsbeispiele einen geeigneten Entscheidungsbaum zu generieren. Der Hypothesenraum ist also die Menge aller möglichen Entscheidungsbäume, die sich mit den vorgegebenen Attributen und Klassen aufbauen lassen. Es gibt verschiedene Klassen von Algorithmen zum Aufbau von Entscheidungsbäumen. Dabei sind die so genannten Top-Down Algorithmen bei weitem am verbreitetsten. In einigen Fällen ist es allerdings wünschenswert, Entscheidungsbäume inkrementell aufzubauen, also indem nach und nach Beispiele hinzugefügt werden. Für diesen Zweck wurden inkrementelle Algorithmen zum Aufbau von Entscheidungsbäumen entwickelt. Beide Klassen sollen im folgenden kurz dargestellt werden.

3.5.3 Top-Down-Algorithmen

Bei dieser Klasse von Algorithmen wird mit einem leeren Baum begonnen, der dann von der Wurzel zu den Blättern aufgebaut wird. Die Idee besteht darin, an der Wurzel und an jedem inneren Knoten eines der Attribute auszuwählen, die noch nicht auf einem Pfad zur Wurzel auftauchen, und dann nach diesem Attribut zu verzweigen. Daraus ergeben sich dann rekursiv zwei Unterbäume. Jedem von beiden wird die Untermenge an Beispielen zugewiesen, welche für das Attribut, nach dem aufgeteilt wurde, den Wert der entsprechenden Kante hat. Dann wird auf diesen Beispielmengen rekursiv dasselbe wiederholt. Wird ein gewisses Abbruchkriterium erfüllt, dann wird nicht mehr weiter verzweigt und statt dessen ein Blatt, mit dem entsprechenden Klassenattribut angelegt.

Es ergeben sich drei Fragen:

- Welches Attribut wird jeweils zur weiteren Verzweigung an einem inneren Knoten und an der Wurzel ausgewählt?
- Wie lautet das Abbruchkriterium, d.h. wann wird nicht mehr weiter verzweigt?
- Welchen Wert (welche Klasse) erhält der Terminalknoten?

Einer der populärsten Algorithmen zum Aufbau von Entscheidungsbäumen ist ID3[5]. Bezüglich der drei genannten Fragen wird folgendermaßen vorgegangen:

1. Die Auswahl des Attributs, nach dem geteilt wird, erfolgt auf Grundlage von Überlegungen zur Informationsentropie. Diese Entropie liegt immer zwischen 0.0 und 1.0 und gibt den Grad der Unordnung von Information an. Angewendet auf Trainingsbeispiele, ist diese Unordnung maximal, wenn genau so viele Beispiele der Klasse *false*, wie der Klasse *true* angehören. Die Häufigkeit der Zugehörigkeit zu einer der beiden Klassen sei h_{false}, bzw. h_{true}. Die Gesamtanzahl der Trainingsbeispiele D sei m. Dann ergibt sich für die Informationsentropie:

   $$entropie(D) = -\frac{h_{false}}{m}\log_2(\frac{h_{false}}{m}) - \frac{h_{true}}{m}\log_2(\frac{h_{true}}{m}), 0 < h_{true} < m$$

   $$entropie(D) = 0, (h_{true} = 0) \vee (h_{false} = 0)$$

   Die Idee besteht nun darin, die Gesamtmenge der Beispiele nach jedem noch möglichen Attribut in zwei Mengen aufzuteilen, die Beispiele, die für dieses Attribut den Wert *false* enthalten und diejenigen, die den Wert *true* enthalten, D_{false} und D_{true}, mit $m_{false} = |D_{false}|$ und $m_{true} = |D_{true}|$. Für diese beiden Mengen kann wiederum die Entropie berechnet werden. Die Gesamtentropie nach der Teilung wird dann durch:

   $$entropie(D_{false}, D_{true}) = \frac{m_{false}}{m} entropie(D_{false}) + \frac{m_{true}}{m} entropie(D_{true})$$

   berechnet. Der Gewinn, d.h. die Verringerung von Entropie, die durch die Teilung bewirkt wurde beträgt dann:

   $$entropie(D) - entropie(D_{false}, D_{true})$$

   Ausgewählt wird dasjenige Attribut, für das diese Differenz maximal wird, das also zu einer maximalen Verringerung der Informationsentropie führt.

2. Das Abbruchkriterium ist zum einen erfüllt, wenn alle Beispiele an einem Knoten zu derselben Klasse gehören, zum anderen wenn es keine Attribute mehr gibt, nach denen noch geteilt werden könnte (d.h. wenn schon alle Attributnamen auf dem Pfad zur Wurzel auftauchen).

3. Wenn die Beispielmenge zum Schluss nur noch Beispiele enthält, die zu einer Klasse gehören, dann erhält das Blatt diese Klasse als Bezeichnung. Falls sowohl Beispiele vorkommen, die zu der einen, wie zu der anderen Klasse gehören, dann wird die Klasse ausgewählt, zu der mehr Beispiele gehören als zu der anderen. Gehören gleichviele Beispiele zu der einen und zu der anderen Klasse, wird zufällig eine der Klassen ausgewählt.

[5] Siehe beispielsweise [9] Kapitel 3

Die Zeitkomplexität zum Aufbau eines Entscheidungsbaums mit ID3 lässt sich folgendermaßen abschätzen (wobei m die Gesamtanzahl der Trainingsbeispiele ist und n die Anzahl der binären Attribute). Da auf jeder Ebene des Baumes, jedes Beispiel höchstens n mal betrachtet wird, der Baum aber höchstens n Ebenen hat, ergibt sich $O(n^2 * m)$.

3.5.4 Inkrementelle Algorithmen

In manchen Situationen sind nicht alle Trainingsbeispiele von Anfang an gegeben, sondern werden erst nach und nach hinzugefügt. Wird in solchen Fällen ein Top-Down-Algorithmus angewendet, so muss jedesmal der Baum von neuem aufgebaut werden, was oft zu erheblichem Aufwand führt. Daher ist die Idee, den Entscheidungsbaum nicht jedesmal neu aufzubauen, sondern nur zu modifizieren, wenn ein weiteres Trainingsbeispiel hinzukommt. Ein Algorithmus, der so verfährt wird in [16] vorgestellt.

Dabei werden an jedem Blatt die Trainingsbeispiele, die diesem Blatt zugeordnet sind, abgespeichert. Wenn ein neues Trainingsbeispiel hinzukommt, wird es zunächst anhand des bestehenden Baumes klassifiziert. Ist die Klasse, die der Entscheidungsbaum diesem Beispiel zuordnet, mit der Klasse des Beispiels identisch, dann wird das Beispiel an dem entsprechenden Blatt des Baumes abgespeichert. Sind die Klassen nicht identisch, dann wird aus dem Blatt ein innerer Knoten mit zwei Nachfolgern. Nach welchem Attribut verzweigt wird, wird analog zu ID3 entschieden.

Da es bei dieser Art von Aufbau eines Entscheidungsbaums dazu kommen kann, dass der Baum nur suboptimal ist, werden immer wieder Restrukturierungen des Baumes vorgenommen, bei denen die Position der inneren Knoten des Baums verändert wird.

4 Das Lernszenario

4.1 Allgemeine Aspekte einer Umwelt aus Sicht der KI

Wie bereits in Abschnitt 2 erläutert wurde, nimmt man meist an, dass ein intelligenter Agent mit seiner Umwelt in Interaktion tritt, d.h. dass er mit Hilfe seiner Sensoren seine Umwelt wahrnehmen und mittels seiner Aktoren diese Umwelt auch beeinflussen kann. Wie diese "Umwelt" aussieht hängt meist von der konkreten Anwendung ab. Dennoch kann man einige allgemeine Charakteristika von Umwelten festhalten, welche für die Entwicklung eines Agenten von besonderer Bedeutung sind:[6]

- *verfügbar/nicht verfügbar*
  Wenn die Sensorik dem Agenten den kompletten Zustand der Umwelt liefert, nennt man eine solche Umwelt vollständig verfügbar. Eine nicht vollständig verfügbare Umwelt erfordert komplexere Mechanismen, da meist auf die nicht direkt wahrnehmbaren Eigenschaften der Umwelt geschlossen werden muss.

- *deterministisch/indeterministisch*
  Wenn das Ergebnis der Aktionen eines Agenten nur von der gewählten Aktion und dem Zustand der Umwelt, in welchem die Aktion durchgeführt wird, abhängt, heißt diese Umwelt deterministisch. In einer nichtdeterministischen Umwelt kann dieselbe Aktion in demselben Zustand zu verschiedenen Ergebnissen führen. In manchen Fällen besteht zwischen der Aktion und dem Ergebnis wenigstens ein probabilistisches Verhältnis. Beispielsweise könnte die Aktion A in Zustand Z mit Wahrscheinlichkeit 0.3 zum Zustand Z' und mit Wahrscheinlichkeit 0.7 zu Zustand Z'' führen. Indeterministische Umgebungen erfordern komplexere Planung, da der Agent alle Zustände berücksichtigen muss, die durch seine Aktionen erreicht werden können.

- *statisch/dynamisch*
  Wenn sich die Umwelt verändert, während der Agent "nachdenkt", spricht man von einer dynamischen, sonst von einer statischen Umwelt. In einer dynamischen Umgebung sollte der Agent möglichst schnell Entscheidungen treffen, da sich seine Umwelt ständig verändern kann und damit eventuell seine Aktionen zu spät kommen. Würde beispielsweise ein Robotertorwart beim Roboterfußball mehrere Minuten überlegen, bevor er eine Aktion einleitet, um den Ball abzufangen, würde wohl schon lange ein Tor gefallen sein. Dynamische Umgebungen erfordern also im allgemeinen nicht nur effektive, sondern auch sehr effiziente Planungsmechanismen.

- *diskret/kontinuierlich*
  Wenn die Anzahl von möglichen Wahrnehmungen und Aktionen endlich

[6] Diese Auflistung ist [13] S.46 entnommen.

ist, dann redet man von einer diskreten, sonst von einer kontinuierlichen Umgebung. Beispielsweise befindet sich ein Schach spielender Agent in einer diskreten, ein Fußball spielender in einer kontinuierlichen Umgebung. Sensorische Werte in einer kontinuierlichen Umgebung müssen zunächst geeignet diskretisiert werden. Ebenso müssen Aktionen parametrisiert werden, z.B. könnte ein Stürmer im Roboterfußball statt der Aktion "kicken" auch noch die Stärke für diese Aktion auswählen.

- *episodisch/nicht episodisch*
  Haben die Aktionen eines Agenten nur Einfluss auf den unmittelbaren Zustand, nicht aber auf folgende Zustände, dann spricht man von einer nicht episodischen Umgebung. Ein typisches Beispiel für einen Agenten, der sich in einer nicht episodischen Umgebung aufhält, ist ein Agent, der Gesichter erkennt . Dagegen befindet sich ein Schach spielender Agent in einer episodischen Umgebung. Episodische Umgebungen zeichnen sich dadurch aus, dass sie im allgemeinen Planung erforderlich machen, während für nicht episodische Umgebungen ein einfaches Schema von Wahrnehmung-Aktion ausreicht.

4.2 Das Szenario

Wie bereits in Abschnitt 3 dargelegt, kann man verschiedene Lernverfahren nur dann sinnvoll vergleichen, wenn man vorher festgelegt hat, was gelernt werden soll, wie der Lernerfolg gemessen wird und welche Erfahrung zum Lernen zur Verfügung steht. In diesem Abschnitt soll ein Szenario entworfen werden, welches genau diese Komponenten bestimmt. Neben der Forderung der exakten Bestimmung der einzelnen Komponenten, soll es auch möglich sein, das Szenario durch das Verändern von Parametern zu variieren. Dies ist ein Vorteil gegenüber "real-world" Problemen, da diese oft die Randbedingungen vorgeben.

Das hier gewählte Szenario kann man sich als Spiel vorstellen. Zu Beginn werden die Regeln dieses Spiels festgelegt und dann wird es nacheinander von mehreren Agententeams gespielt[7]. Anhand der Anzahl der Punkte, welche die einzelnen Teams in dem Spiel erreichen, lässt sich dann ihre Leistung vergleichen. Die Spielregeln legen fest, welche Erfahrung den Agenten zur Verfügung steht und welche Aktionen sie ausführen können.

Das für diese Arbeit verwendete Spiel hat folgenden Inhalt. Das Team von Agenten befindet sich auf einer Insel. Auf dieser Insel gibt es mehrere Plätze. An diesen Plätzen können die Agenten graben und finden in manchen Fällen eine Energiezelle. Diese Energiezelle können sie benutzen, um ihre Batterie aufzuladen. Ist die Batterie eines Agenten leer, dann wird er aus dem Spiel genommen. Ziel des Spiels ist es, möglichst viel Energie zu gewinnen. Dazu sollten die Agenten einerseits lernen, wann es sich lohnt nach einer Energiezelle zu graben und

[7]Die Teams treten also nicht direkt gegeneinander an, wie beispielsweise im Roboterfußball, sondern werden nacheinander mit dem Szenario konfrontiert.

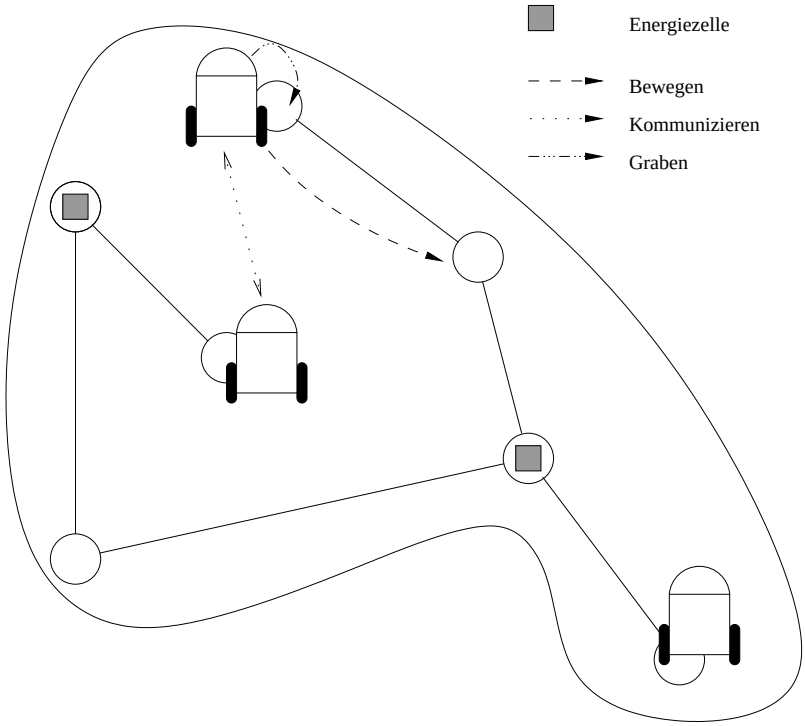


Abbildung 4: Das Szenario

wann nicht, anderseits können sie miteinander kommunizieren, um diesen Lernvorgang zu beschleunigen.

4.2.1 Das Spielfeld (Die Insel)

1. Auf der Insel befinden sich verschiedene Plätze. Diese sind teilweise untereinander verbunden. Agenten können sich nur entlang solcher Verbindungen bewegen.

2. Jedem Platz sind n binäre Attribute $A_1, ..., A_n$ zugeordnet die von den Agenten wahrgenommen werden können. Beispielsweise kann sich an einem Platz Wald befinden oder nicht, der Boden kann weich oder hart sein, etc. Neben diesen Attributen kommt jedem Platz noch eine nicht sichtbare Eigenschaft zu, nämlich ob an diesem Platz eine Energiezelle vergraben ist. Es wird angenommen, dass zwischen dem Vorhandensein einer Energiezelle und den sichtbaren Attributen ein Zusammenhang besteht, und dass dieser auf der gesamten Insel derselbe ist.

3. Die Agenten können an jedem Platz nach Energiezellen graben, vorausgesetzt dass an diesem Platz noch nicht gegraben wurde.

4.2.2 Die Agenten

1. An der Simulation nimmt eine vorher festgelegte Anzahl von k Agenten teil. Jeder Agent hat eine Batterie, welche eine begrenzte Menge an Energie enthält.

2. Jeder Agent hat die Möglichkeit, Anfragen an die Umwelt zu richten. Diese Anfragen sind immer erfolgreich und benötigen keine Energie. Folgende Anfragen stehen einem Agenten zur Verfügung:

   (a) Wieviel Energie befindet sich in meiner Batterie?

   (b) Welche Werte haben die Attribute, die den Platz repräsentieren, an dem ich mich momentan befinde?

   (c) Welche Plätze befinden sich in der Umgebung des Platzes, an dem ich mich momentan befinde?

   (d) War meine letzte Aktion erfolgreich?

   (e) Habe ich bei meinem letzten Graben eine Energiezelle gefunden? (Es ist möglich, dass das Graben zwar erfolgreich war, dass aber keine Energiezelle gefunden wurde)

   (f) Ist der Platz, an dem ich mich momentan befinde, schon ausgebeutet?

   (g) Wenn der Platz bereits ausgebeutet ist, wurde hier eine Energiezelle gefunden?

3. Jeder Agent verfügt darüber hinaus über folgendes Hintergrundwissen:

   (a) Anzahl und Identifikationsnummern aller anderen Agenten des Teams

   (b) Die Werte von E_g, E_z, E_b, E_k, (siehe unten).

4. Jeder Agent hat die Möglichkeit eine Reihe von Aktionen auszuführen. Damit eine Aktion erfolgreich ausgeführt werden kann, müssen die Vorbedingungen für diese Aktion erfüllt sein. Das Ausführen von Aktionen kostet den Agenten normalerweise Energie. In der folgenden Tabelle bezeichnet E_{alt} die Energie des Agenten vor Ausführen der Aktion, E_{neu} nach Ausführen der Aktion:

Aktion	Vorbedingungen	Effekt
Am aktuellen Platz graben	$E_{alt} > E_g$, an diesem Platz wurde noch nicht gegraben	Wenn sich an diesem Platz eine Energiezelle befindet, dann $E_{neu} = E_{alt} + E_z - E_g$ Wenn sich an diesem Platz keine Energiezelle befindet, dann $E_{neu} = E_{alt} - E_g$
Sich zu einem anderen Platz p bewegen	$E_{alt} > E_b$	$E_{neu} = E_{alt} - E_b$ Der Agent befindet sich an einem neuen Platz. Wenn es nicht zum Konflikt mit einem anderen Agenten kommt, ist dies der Platz, den der Agent bei seiner Aktion angegeben hat. Ansonsten kann es eventuell ein anderer, zufällig gewählter Platz sein.
Einem Agenten x eine Anfrage senden	$E_{alt} > E_k$, Agent x ist bereit die Anfrage zu beantworten.	$E_{neu} = E_{alt} - E_k^*$ Der Agent erhält eine Antwort auf seine Anfrage. E_k^* errechnet sich aus E_k und aus der Größe der Nachricht.

4.2.3 Startkonfiguration des Spiels

- Jeder Agent erhält eine festgelegte Menge an Energie. Diese ist für alle Agenten gleich.
- Die Agenten werden zufällig auf die Plätze der Insel verteilt, so dass sich an jedem Platz höchstens ein Agent befindet.

4.2.4 Ablauf des Spiels

- Das Spiel ist in Schritte eingeteilt.
- In jedem Schritt kann der Agent folgende Aktionen, eventuell auch mehrfach, ausführen:
  - Anfragen an den Simulator stellen
  - Nach einer Energiezelle graben
  - Einem anderen Agenten eine Anfrage senden
- Danach muss sich der Agent zu einem anderen Platz bewegen. Er tut dies, indem er den gewünschten Zielplatz angibt. Falls es keine Konflikte gibt, richtet sich der Spielsimulator nach diesem Wunsch. Falls es einen Konflikt gibt (zwei Agenten wollen zu demselben Platz), wird dieser zufällig

aufgelöst, d.h. ein Agent wird an einen anderen Platz transportiert. Verfügt ein Agent nicht über die erforderliche Energie, um sich zu einem anderen Platz zu bewegen, dann wird er aus dem Spiel genommen.

4.2.5 Ende des Spiels

Das Spiel ist beendet, wenn entweder alle Agenten aus dem Spiel genommen wurden (also nicht mehr genug Energie haben, um sich zu bewegen), oder wenn eine maximale Anzahl von Spielschritten erreicht ist.

4.2.6 Auswertung des Spiels

Das Maß für die Leistung eines Agententeams in diesem Szenario ist die Gesamtenergie, über die alle noch aktiven Agenten zusammen am Ende verfügen. D.h. nach Beendigung der Simulation wird die Restenergie aller Agenten, die nicht aus dem Spiel genommen wurden, aufsummiert und ergibt die Punktzahl, welche dieses Team erreicht hat. Wurden alle Agenten aus dem Spiel genommen, dann ist diese Punktzahl 0.

4.2.7 Einstellbare Parameter der Spielsimulation

Bevor eine Spielsimulation gestartet wird, müssen verschiedene Parameter festgelegt werden:

- Struktur der Insel, als Graph (d.h. welche Plätze gibt es, wie sind sie verbunden)
- Anzahl der Attribute, die einen Platz repräsentieren
- Zusammenhang zwischen Attributwerten und dem Vorhandensein einer Energiezelle (das zu lernende Konzept)
- Die Energieparameter E_g, E_z, E_b, E_k
- Die maximale Anzahl an Simulationsschritten
- Die Anfangsenergie der Agenten
- Die Anzahl der Agenten, die ein Team bilden.

5 Nicht-kooperatives Lernen

5.1 Agentenmodell

Zu einem Agentenmodell im Sinne von Abschnitt 2.2 gehört unter anderem eine Menge von Wissensbasen und ein Ablaufzyklus.

Welche Wissensbasen benötigt ein Agent in dem gegebenen Szenario? Zum einen benötigt er eine Wissensbasis, in der er seine bisherige Erfahrung speichert, also die Menge an Trainingsbeispielen, die er durch Erfahrung gewonnen hat. Außerdem benötigt er eine Wissensbasis, in der er die von ihm gebildete Hypothese speichern und auf sie zugreifen kann. Diese beiden Wissensbasen können als Weltwissen aufgefasst werden. Daneben muss er noch über Kenntnis der Regeln des Szenarios verfügen.

Ein einzelner Agentenzyklus ist in Abbildung 5 dargestellt:

1. Zunächst nimmt der Agent die Eigenschaften des Platzes wahr auf dem er sich befindet und prüft, ob dieser schon ausgebeutet ist.
2. Wenn dies der Fall ist, dann erkennt er an den Spuren, ob der Agent, der an dieser Stelle bereits gegraben hat, eine Energiezelle gefunden hat. Er fügt die Attribute des Platzes zusammen mit dem Ergebnis des Grabens seiner Erfahrung hinzu und aktualisiert seine Hypothese.
3. Ist der Platz noch nicht ausgebeutet, dann prüft er auf Grundlage seiner aktuellen Hypothese, ob das Graben an dieser Stelle zum Erfolg führen könnte und entscheidet sich entsprechend, ob er graben soll oder nicht.
4. Wenn er sich für das Graben entscheidet, dann gräbt er und falls er eine Energiezelle findet, nimmt er sie auf. In jedem Fall fügt er die Attribute und das Ergebnis des Grabens als Beispiel zu seiner Erfahrung hinzu und aktualisiert seine Hypothese.
5. Unabhängig davon, ob er gegraben hat oder nicht, bewegt er sich zu einem anderen Platz. Da er die Attribute der umgebenden Plätze nicht wahrnehmen kann, wählt er zufällig einen der Plätze aus, wobei er es aber nach Möglichkeit vermeidet, zu dem Platz zurückzukehren, von dem er unmittelbar gekommen ist.

5.2 Strategie

Die Frage, wann der Agent graben sollte, scheint einfach zu beantworten: immer wenn seine Hypothese anhand der Attribute des aktuellen Platzes ausgibt, dass sich an dieser Stelle eine Energiezelle befindet, sollte er graben. Tatsächlich ist es aber etwas komplexer. Dies zeigt sich, wenn man die Entscheidung des

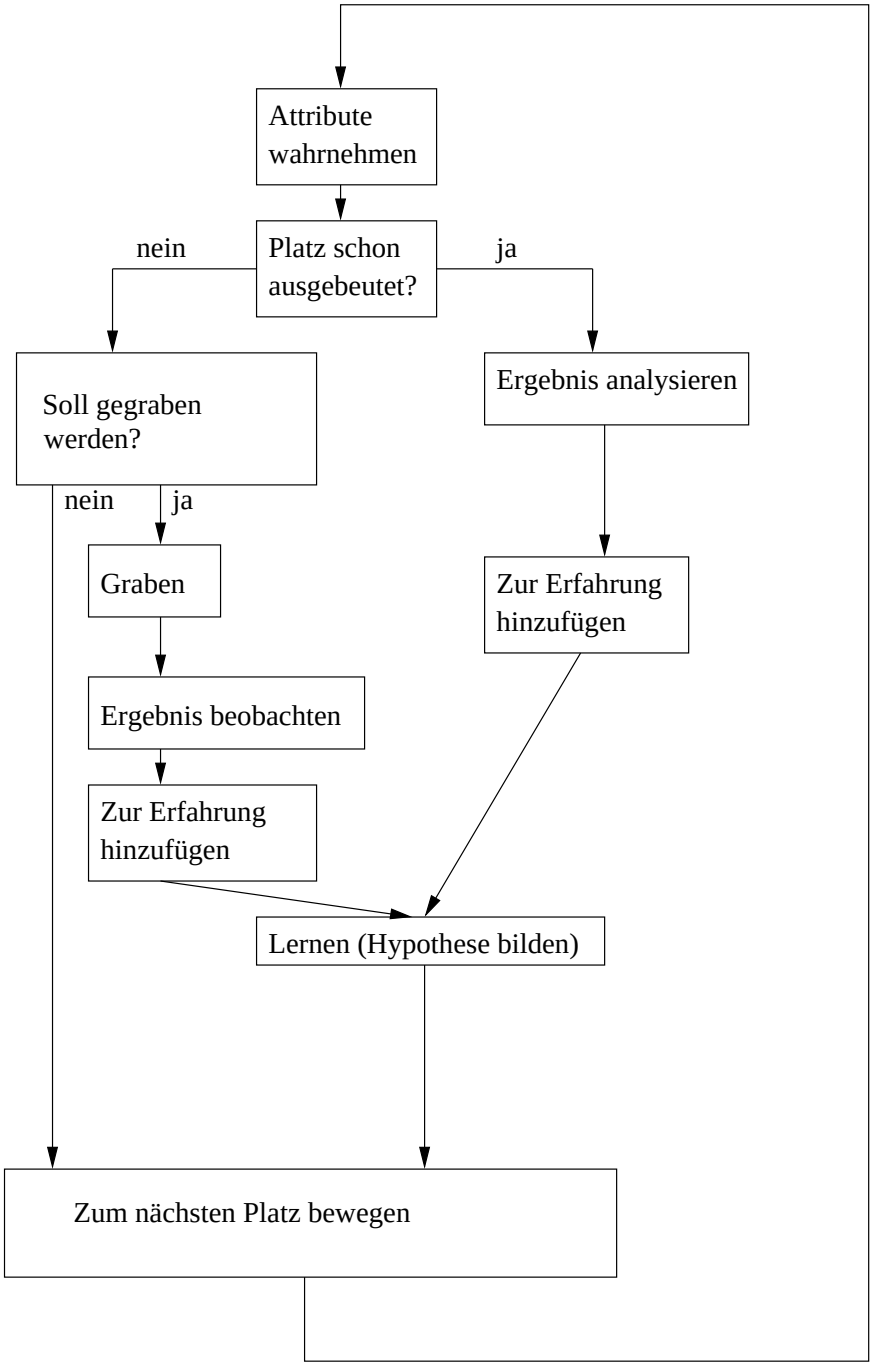


Abbildung 5: Ablauf: Nicht-kooperierender Agent

Agenten als rationale Entscheidung im ökonomischen Sinne modelliert: für jede dem Agenten mögliche Aktion wird der Erwartungswert für die zu gewinnende Energie berechnet und der Agent wählt die Aktion aus, welche den höchsten Erwartungswert hat:

- Wenn der Agent nicht gräbt, dann hat er keine Kosten, aber auch keinen Gewinn. Also ist die Bilanz null.
- Wenn der Agent gräbt und tatsächlich eine Energiezelle findet, so investiert er E_g Energie für das Graben und erhält die Energie der Energiezelle E_z. Also ist die Bilanz $E_z - E_g$.
- Wenn der Agent gräbt, aber keine Energiezelle vorfindet, dann hat er nur die Kosten E_g, aber keinen Gewinn. Die Bilanz ist also $-E_g$.
- Unabhängig vom Erfolg gewinnt aber der Agent, indem er gräbt, neue Erfahrung hinzu. Diese kann er nutzen, um seine Hypothese zu verbessern, und es ist wahrscheinlich, dass er mit dieser Hypothese zukünftig genauer entscheiden wird, da sie auf mehr Information beruht. Dies führt aber dazu, dass er Energie einspart, die er durch falsche Entscheidungen hätte aufbringen müssen. Er hat also, wenn er gräbt, einen Vorteil für zukünftige Entscheidungen. Dieser Vorteil wird durch E_l für *lernen* bezeichnet.

Folgende Tabelle fasst dies noch einmal zusammen:

	graben	nicht graben
keine Energiezelle	$-E_g + E_l$	0
Energiezelle	$E_z - E_g + E_l$	0

Nun trete das Ereignis, dass eine Energiezelle vorhanden ist mit Wahrscheinlichkeit p ein, dass keine solche vorhanden ist mit Wahrscheinlichkeit $(1 - p)$. Dann ergibt sich als Erwartungswert der beiden Aktionen folgendes:

$E(graben) = p * E_z - E_g + E_l$

$E(nicht\ graben) = 0$

Damit es rational ist, zu graben, muss gelten:

$E(graben) > E(nicht\ graben)$, und damit

$p * E_z - E_g + E_l > 0$

Sind E_g, E_z, E_l und p gegeben, dann kann der Agent auf dieser Grundlage eine rationale Entscheidung treffen, ob er graben soll. $E_g$und E_z sind als Konstanten vorgegeben. Einen geschätzten Wert für die Wahrscheinlichkeit p gewinnt der Agent mit Hilfe seiner Hypothese. Denn die Ausgabe der Hypothese ist Ausdruck

der wahrscheinlichsten Klassifizierung unter Annahme der Vorbedingungen und Betrachtung der gegebenen Erfahrung. Das heißt aber, der Schätzwert für die Wahrscheinlichkeit p kann nur die Werte 0.0 und 1.0 annehmen. Die Bestimmung von E_l ist wesentlich komplexer. Im folgenden soll grob untersucht werden, welchen Einfluss E_l auf die Lernstrategie hat. Dabei sollen drei Annahmen benützt werden:

1. Wenn der Agent nur noch sehr wenige Beispiele klassifizieren muss, ist der Nutzen durch eine verbesserte Hypothese gering. Muss er hingegen in Zukunft noch sehr viele Beispiele klassifizieren, dann lohnt sich Lernen eher.
2. Hat der Agent erst sehr wenig Erfahrung, so verbessert sich seine Hypothese durch ein weiteres Trainingsbeispiel sehr stark. Hat er schon sehr viel Erfahrung, dann ändert dies wenig an der Qualität seiner Hypothese.
3. E_l ist nie negativ, d.h. Lernen ist nie schädlich. Dies ist eine Annahme, die man zumindest dann immer sinnvoll treffen kann, wenn die Trainingsbeispiele kein Rauschen enthalten.

Aus diesen Annahmen folgt, dass E_l sowohl mit der Anzahl der Schritte, als auch mit der Anzahl der gewonnenen Trainingsbeispiele monoton abnimmt. Was dies zur Folge hat, zeigt sich, wenn man eine Fallunterscheidung nach der Ausgabe der Hypothese macht.

Nimmt man an, dass $p_{hypothese} = 1.0$ gilt, also dass die Hypothese ausgibt, dass sich an dem gegebenen Platz eine Energiezelle befindet, ergibt sich folgendes

$$1.0 * E_z - E_g + E_l > 0$$

Nun gilt sinnvollerweise $E_z > E_g$, denn sonst würde es sich nie lohnen zu graben. Außerdem ist nach Annahme $E_l > 0$, und somit ist diese Ungleichung immer erfüllt. Also sollte der Agent immer graben, wenn seine Hypothese es ihm nahe legt.

Nimmt man an, dass $p_{hypothese} = 0.0$, also dass auf Grund der Hypothese davon auszugehen ist, keine Energiezelle vorzufinden, dann ergibt sich

$$0.0 * E_z - E_g + E_l > 0$$

$$E_l > E_g$$

Dies bedeutet, dass der Agent unter Umständen graben soll, obwohl seine Hypothese ihm sagt, dass wahrscheinlich keine Energiezelle gefunden wird. Da nun E_l mindestens in der Anzahl der erhaltenen Trainingsbeispiele als monoton fallend angesehen werden kann, gibt es stets ein m_0 bis zu dem der Agent immer graben sollte. Hat der Agent mehr als m_0 Beispiele gesammelt, sollte er sich im folgenden immer an seine Hypothese halten. Da E_l theoretisch nur schwer

bestimmbar ist, muss ein optimales m_0 geschätzt werden, was beispielsweise durch eine große Anzahl von Simulationsläufen mit verschiedenen Parametern und Werten für m_0 geschehen könnte (Dies wird in Abschnitt 7 durchgeführt).

Das etwas überraschende Ergebnis, dass der Agent unter Umständen graben soll, obwohl seine Hypothese ausgibt, dass wahrscheinlich keine Energiezelle vorhanden ist, lässt sich daran etwas plausibler machen, dass der Agent beispielsweise als erstes ein negatives Beispiel erhält. Bildet er aus diesem Beispiel eine Hypothese, so lautet diese: "grabe nie". Wenn er sich nun an diese Hypothese hält und nie wieder gräbt, wird er seine Hypothese nie wieder ändern, denn nur indem er gräbt erhält er neue Trainingsbeispiele und nur mit Hilfe von weiteren Trainingsbeispielen kann er seine Hypothese verändern. Dann verbleibt der Agent aber bei seiner Hypothese und verhält sich langfristig nicht optimal (natürlich angenommen, dass es tatsächlich Energiezellen auf der Insel gibt)

5.3 Lernen mit Entscheidungsbäumen

Wie oben darstellt, sind Entscheidungsbäume eine flexible und effiziente Möglichkeit zum Konzeptlernen. Die Verwendung von Entscheidungsbäumen innerhalb des Ablaufzyklus des Agenten ist sehr einfach. Jedesmal wenn er seine Hypothese aktualisieren soll, wendet er den Algorithmus zum Aufbau von Entscheidungsbäumen auf die Menge seiner Beispiele an. Dabei scheint im gegebenen Szenario die inkrementelle Variante des Aufbaus von Entscheidungsbäumen die geeignetere zu sein, da der Agent seine Hypothese sehr häufig aktualisieren muss. Da aber der Schwerpunkt dieser Arbeit auf dem Vergleich zwischen nicht-kooperativem und kooperativem Lernen liegt, soll die top-down Methode vorgezogen werden. Diese hat den Vorteil, dass sie gut untersucht ist, zum Beispiel in Bezug auf ihre Komplexität. Außerdem gibt es viele Varianten und Erweiterungen, beispielsweise zum Umgang mit fehlenden Attributen.

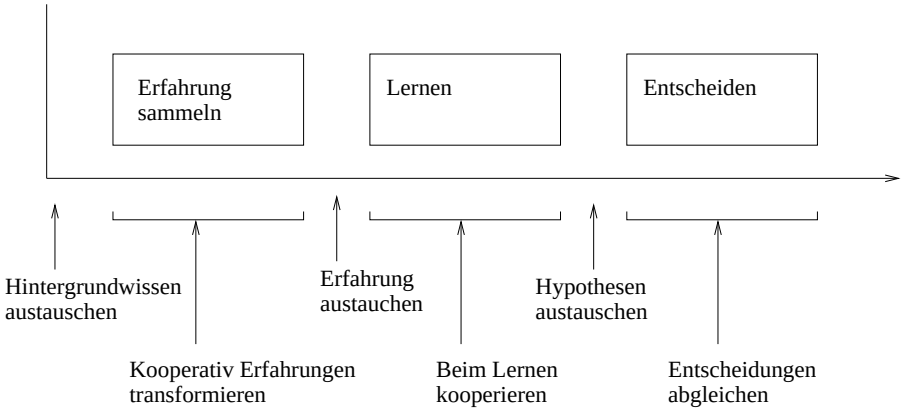
6 Kooperatives Lernen

6.1 Übersicht

Im bisherigen Agentenmodell können die Agenten nur drei lernbezogene Aktionen ausführen:

1. Erfahrung sammeln
2. Hypothesen bilden (Lernen)
3. Hypothesen auf neue Probleme anwenden (Entscheidungen treffen)

Im kooperativen Lernen kommt eine weitere Aktion hinzu, nämlich die Kommunikation. Prinzipiell können in dem hier betrachteten Szenario die Aktionen in beliebiger Reihenfolge auftreten. Um die Möglichkeiten, die sich durch die Kommunikationsfähigkeit ergeben, einfacher klassifizieren zu können, sollen die Aktionen der Agenten zunächst in drei isolierten Phasen betrachtet werden, so dass ein Agent erst Erfahrungen sammelt, dann daraus lernt und dann entscheidet, wobei jede Phase nur einmal durchlaufen wird. Abhängig davon, wohin in diesen Prozess die Kommunikation eingeschoben wird, ergeben sich verschiedene Szenarien für kooperatives Lernen (siehe auch [1]):



1. Die Kommunikation findet vor dem Erfahrung sammeln statt. In dieser Variante könnten die Agenten nur Hintergrundwissen kommunizieren, da sie noch keine Erfahrung gemacht haben. Der Austausch von Wissen ist eine ganz eigene Problematik, die hier aber nicht näher betrachtet wird.
2. Die Kommunikation findet während des Sammelns von Erfahrung statt. Dies hat nur Sinn, wenn die Trainingsbeispiele noch transformiert werden, bevor sie in die Trainingsmenge aufgenommen werden. In einem solchen Fall könnte diese Transformation kooperativ erfolgen. Wenn beispielsweise

jeder Agent nur eine bestimmte Untermenge von Attributen wahrnehmen kann, dann könnten mehrere Agenten zusammen die jeweils fehlenden Attribute der anderen ersetzen. Auch das hat aber nicht direkt mit Lernen zu tun.

3. Die Kommunikation findet statt, nachdem die Agenten Erfahrung gesammelt haben, aber bevor sie lernen und ihr Wissen anwenden. Die Beispiele werden erst isoliert gesammelt, dann per Kommunikation ausgetauscht. Jeder Agenten lernt nun unabhängig aus diesen Beispielen und wendet die gelernte Hypothese unabhängig von den anderen Agenten an.

4. Die Kommunikation findet während der Lernphase statt.

5. Die Kommunikation findet vor der Problemlösungsphase statt, aber nach dem individuellen Lernen. D.h. die Agenten lernen jeder für sich eine Hypothese, dann tauschen sie diese Hypothesen oder Teile der Hypothesen aus, integrieren die fremden Hypothesen in ihre eigenen und wenden die so entstandene Hypothese wiederum individuell an.

6. Die Kommunikation findet während der Entscheidungsphase und nach dem individuellen Lernen statt. Nach diesem Szenario lernen die Agenten unabhängig voneinander und kommunizieren erst, wenn sie ihr Wissen auf ein neues, bislang unbekanntes Problem anwenden. Beispielsweise können sie mehrere andere Agenten um Rat fragen und dann dem Rat folgen, der ihnen am häufigsten erteilt wurde.

Jegliche Kommunikation nach dem individuellen Problemlösen kommt zu spät, da hier angenommen wird, dass die Agenten jede Phase nur einmal durchlaufen.

Im allgemeineren Fall können die einzelnen Phasen mehr als einmal und in beliebiger Reihenfolge durchlaufen werden. Beispielsweise kann ein Agent erst Erfahrung sammeln, dann Lernen, dann wieder Erfahrung sammeln, dann die Erfahrung anwenden, etc. An den grundlegenden Möglichkeiten ändert sich dadurch nichts. Allerdings ergibt sich noch eine weitere Möglichkeit als Kombination von 3. und 4. : Hat ein Agent die Hypothese eines anderen Agenten erhalten, so muss er sie nicht direkt mit seiner eigenen Hypothese verschmelzen, sondern kann sie bis zur nächsten Lernphase aufbewahren und dann während des Lernens als Hintergrundwissen benutzen.

Verfahren, die unter 1. oder 2. fallen, werden nicht weiter untersucht, da sie nicht direkt mit Lernen zu tun haben. Verfahren, die unter 4. fallen, werden nicht weiter untersucht, da Kommunikation als aufwendig und nicht immer möglich angesehen wird und es somit von vornherein besser ist, komplette Hypothesen zu kommunizieren. Damit ergeben sich drei Klassen von Verfahren, die im folgenden untersucht werden: Austausch von Erfahrung, Abgleich von Entscheidungen und Austausch von Hypothesen.

Für die drei Klassen von Verfahren sollen jeweils folgende Punkte untersucht werden:

- Das Agentenmodell, welches dem Verfahren zu Grunde liegt, d.h. vor allem die Struktur der Wissensbasen des Agenten, sowie sein Ablaufzyklus.
- Die Algorithmen, die für das jeweilige Verfahren verwendet werden.
- Die Lern- bzw. Kooperationsstrategie die der Agent bei diesem Verfahren anwenden sollte. Dies bezieht sich auf die Frage, wann der Agent mit wem kooperieren soll.
- Die Aufwendigkeit des Verfahrens. Das betrifft zum einen den zu erwartenden Kommunikationsaufwand, zum anderen den Aufwand an Rechenzeit, im Vergleich zum nicht-kooperativen Lernen.
- Der zu erwartende Nutzen des jeweiligen Verfahrens.
- Die Voraussetzungen, die erfüllt sein müssen, damit das Verfahren angewendet werden kann. Eine solche Voraussetzung kann z.B. sein, dass alle beteiligten Agenten denselben Lernalgorithmus verwenden.

Für alle Verfahren müssen noch zwei Dinge beachtet werden:

1. Ist sicher, dass die einzelnen Agenten nicht lügen? Und kann es sein, dass die Beispiele, welche die einzelnen Agenten erhalten, nicht demselben Zusammenhang zwischen Attributen und Klassen entsprechen? Wenn solche Fälle auftreten können, dann wird das Problem erheblich komplexer, denn indem der Agent fremdes Wissen miteinbezieht, kann sich seine Leistung beim Problemlösen sogar verschlechtern. In diesen Fällen muss der Agent zunächst prüfen, ob er die Information eines bestimmten anderen Agenten einbeziehen will oder nicht, bevor er sie verwendet.
2. Ein weiteres Problem tritt auf, wenn die Agenten zwar Beispiele erhalten, welche alle aus demselben Zusammenhang zwischen Attributen und Klasse stammen, aber verschieden dargestellt sind. Dies könnte beispielsweise der Fall sein, wenn die Agenten verschiedene Sensoren haben. Dann müssen die Beispiele, Hypothesen und Probleme des einen Agenten, in die Sprache der anderen übersetzt werden, um die obigen Methoden anwenden zu können. Dies ist dann meist höchstens annäherungsweise möglich und es zeigt sich vor allem, dass es auch hier durch Einbeziehung fremden Wissens zu einer Verschlechterung der Leistung kommen kann.

Aus diesen Gründen soll hier nur der Fall untersucht werden, in dem die Erfahrung der Agenten demselben Zusammenhang gehorcht, die Agenten nicht lügen und alle dieselben Sensoren haben.

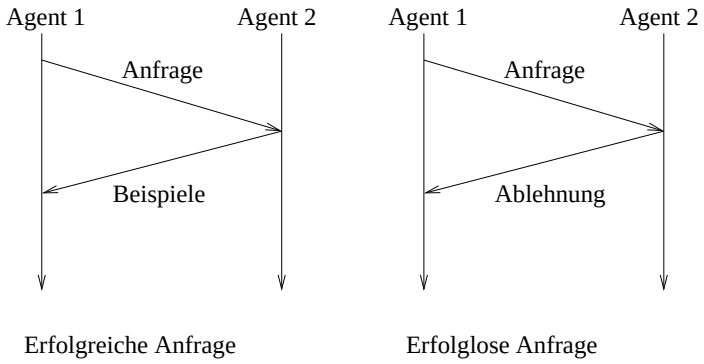


Abbildung 6: Protokoll: Austausch von Erfahrung

6.2 Austausch von Erfahrung

Dies ist die vom Standpunkt des Maschinellen Lernens einfachste Variante, denn die Verfahren zum nicht-kooperativen Lernen können hier direkt zum kooperativen Lernen verwendet werden. Ein Agent erhält in regelmäßigen Abständen von anderen Agenten Trainingsbeispiele. Nach den getroffenen Annahmen hat der Agent keinen Grund, die fremden Trainingsbeispiele als besser oder schlechter zu betrachten als seine eigenen. Daher kann er sie einfach zu der Menge seiner Beispiele hinzufügen und aus dieser Gesamtmenge seine Hypothese bilden.

6.2.1 Agentenmodell

Der Agent erhält zu seinen bisherigen Wissensbasen nun noch eine weitere hinzu, welche sein Wissen über andere Agenten repräsentiert. Dabei werden über jeden anderen Agenten folgende Informationen gespeichert:

- eine Identifikationsnummer, um mit diesem Agenten Kontakt aufnehmen zu können
- den Zeitpunkt, an welchem dem Agenten zum letzten Mal Beispiele geschickt wurden, um zu verhindern, ihm Beispiele doppelt zu schicken
- die Gesamtzahl von Beispielen, die von dem Agenten erhalten wurden, bzw. ihm gesendet wurden, um Strategien zu entwickeln, wann mit diesem Agenten kooperiert werden soll

Zum Ablaufmodell des Agenten wird eine Kommunikationsphase hinzugefügt. Während dieser Phase ist es dem Agenten möglich, Nachrichten mit anderen Agenten auszutauschen. Das Kommunikationsprotokoll sieht nur zwei Nachrichtentypen vor und ist in Abbildung 6 dargestellt.

1. *Anfrage*: Ein Agent schickt einem anderen Agenten die Aufforderung, neue Beispiele zu senden.

2. *Antwort*: Ein Agent reagiert auf eine Anfrage, entweder indem er auf sie eingeht und Beispiele in Form von jeweils einem Attributvektor und dem zugehörigen Klassenattribut sendet, oder dies ablehnt und entsprechend eine Ablehnung sendet.

Abbildung 7 zeigt den Ablaufzyklus des Agenten.

In der Kommunikationsphase analysiert der Agent zunächst, von welchen Agenten er Beispiele anfordern soll. Dann schickt er die entsprechenden Anfragen an die gewählten Agenten und wartet auf Antwort. Parallel dazu bearbeitet er die Anfragen, die von anderen Agenten an ihn gerichtet sind. Er entscheidet jeweils, ob er auf die Anfrage eingehen soll. Ist dies der Fall, dann schickt er Beispiele und zwar nur Beispiele, die seiner eigenen Erfahrung entspringen und nach dem Zeitpunkt gemacht wurden, an dem er zum letzten Mal diesem Agenten Beispiele geschickt hat. Auf diese Weise wird verhindert, dass Beispiele doppelt verschickt werden. Wann und an wen der Agent Anfragen schickt, sowie auf welche Anfragen er eingeht, entscheidet seine Strategie.

Nach der Kommunikationsphase verfährt der Agent wie im Fall des nicht-kooperativen Lernens, nur dass er in jedem Fall seine Hypothese aktualisiert, wenn er neue Beispiele von anderen Agenten erhalten hat.

6.2.2 Algorithmus

Der Agent fügt die von anderen Agenten erhaltenen Beispiele seinen eigenen hinzu und wendet denselben Algorithmus an, wie beim nicht-kooperativen Lernen.

6.2.3 Strategie

Der Algorithmus lässt noch die Frage offen, wann und von wem der Agent Trainingsbeispiele fordern soll, sowie die Frage, wem er seine eigenen Beispiele schicken soll, wenn er eine entsprechende Anfrage erhält.

Was die zweite Frage betrifft, wann der Agent auf eine Anfrage seine Beispiele senden soll, kann man zum einen annehmen, dass die Agenten sich gegenseitig voll unterstützen, so dass jede Nachfrage positiv beantwortet wird. Es ist plausibel dies anzunehmen, da die Agenten ein Team bilden. Will man diese Annahme nicht machen, dann wäre eine plausible Strategie, dass der Agent zu Beginn auf jede Anfrage eingeht, danach aber einem Agenten nur noch Beispiele zuschickt, wenn dieser ihm Beispiele zuschickt. Dazu könnte er die Information nutzen, wie

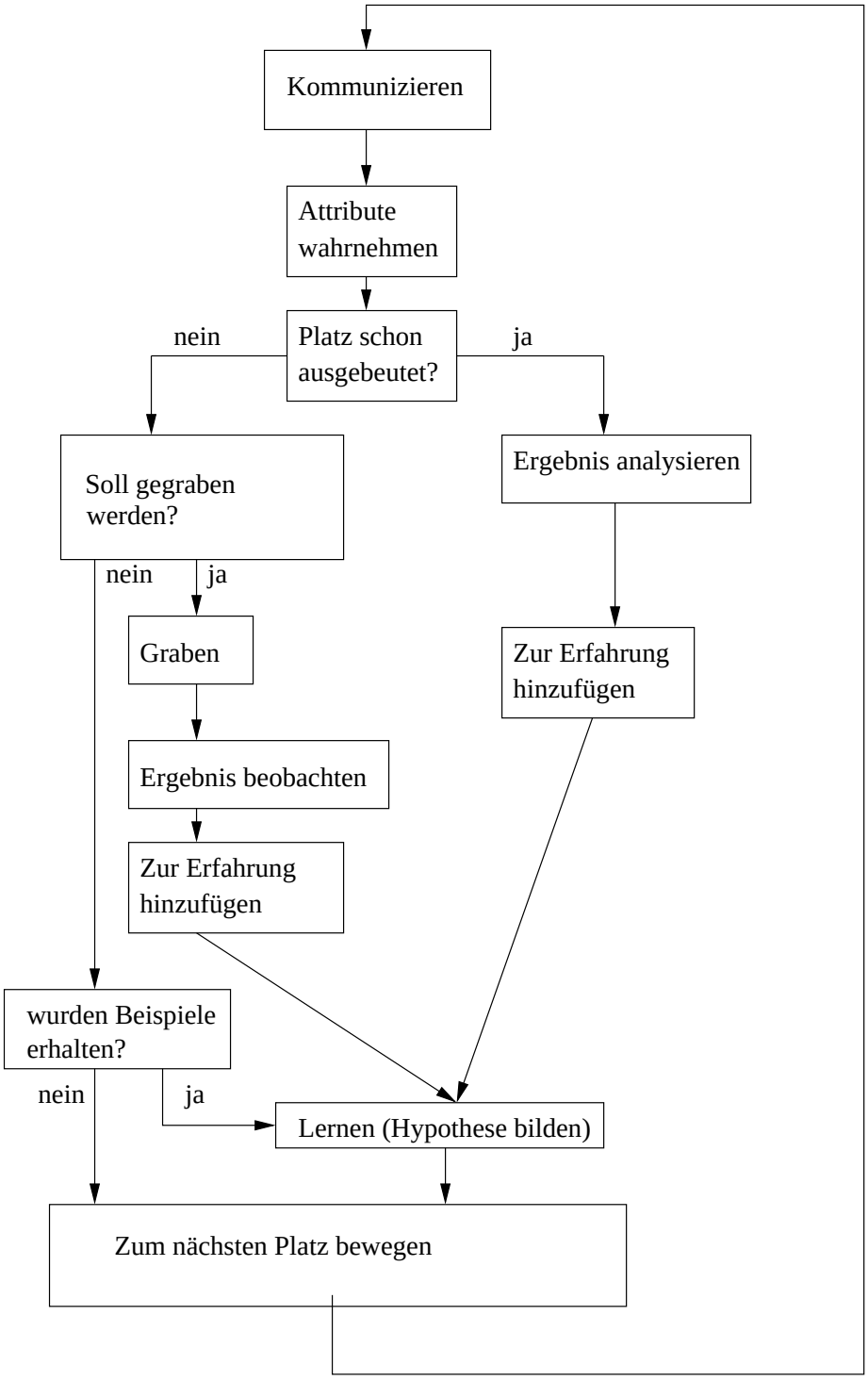


Abbildung 7: Ablauf: Austausch von Erfahrung

viele Beispiele er einem bestimmten Agenten bereits gesendet hat und wie viele er von ihm erhalten hat. Ist die Differenz zu groß, dann verweigert er, weitere Beispiele zu senden.

Die erste Frage, wann der Agent von einem bestimmten anderen Agenten Beispiele fordern soll, ist schwieriger zu beantworten. Denn der Agent muss ermitteln, ob der Nutzen, den er durch die weiteren Beispiele hat, die Kosten der Kommunikation aufwiegt.

Die Kosten für eine Übertragung von Beispielen steigen proportional zur Anzahl der erhaltenen Beispiele. Sei E_l der Nutzen durch ein Beispiel, welches der Agent durch Kommunikation erhält. Über E_l lassen sich nun wiederum zwei plausible Aussagen machen:

1. E_l nimmt mit der Anzahl der Trainingsbeispiele, über die der Agent schon verfügt, ab.
2. E_l nimmt auch in der Anzahl der Schritte ab, denn der Nutzen einer gelernten Hypothese hängt davon ab, wie oft sie noch angewendet werden kann. Die Simulation endet nach endlich vielen Schritten.

Aus diesen beiden Beobachtungen ergibt sich nun ein ähnliches Ergebnis, wie im Fall des nicht-kooperativen Lernens. Da der Nutzen durch ein einzelnes erhaltenes Beispiel immer weiter sinkt, sinkt auch der Nutzen durch m Beispiele. Die Kosten für m Beispiele sind hingegen immer konstant. Da E_l zumindest mit Anzahl der Trainingsbeispiele sinkt, gibt es ein m_1 bis zu dem es sich lohnt, in jedem Schritt Beispiele anzufordern. Nimmt man dies zusammen mit der Strategie für isoliertes Lernen, dann erhält man folgendes:

1. Verfügt der Agent über weniger als m_0 Beispiele, so sollte er immer graben. Da er immer gräbt, braucht er nicht zu kommunizieren, denn eine Verbesserung seiner Hypothese ist nicht relevant, er macht keinen Gebrauch von ihr.
2. Verfügt der Agent über mehr als m_0 Beispiele, aber weniger als m_1, dann sollte er sich zwar nach seiner Hypothese richten, gleichzeitig aber von den anderen Agenten Trainingsbeispiele anfordern, um seine Hypothese zu verbessern.
3. Verfügt der Agent über mehr als m_1 Beispiele, dann ist der Nutzen durch weitere Beispiele nicht hoch genug, um die Kosten für die Kommunikation auszugleichen. Daher fordert der Agent keine weiteren Beispiele mehr an.

6.2.4 Qualitative Analyse

Kosten

Die Kosten für dieses Verfahren lassen sich in Rechen- und Kommunikationskosten aufteilen. Betrachtet man eine einzelnes Paar von Anfrage und Antwort so müssen mindestens $(n+1)*l$ Bits übertragen werden, wobei n die Anzahl der Attribute pro Beispiel ist und l die Anzahl der Beispiele. Es ist natürlich wesentlich, wie viele Beispiele insgesamt übertragen werden müssen. Da diese Anzahl aber von der Strategie des Agenten abhängt, muss sie experimentell ermittelt werden.

Nimmt man an, dass die Agenten ihre Beispiele ständig austauschen und dass alle Agenten immer gleich viele Beispiele haben, dann muss jeder Agent, in einem vollständigen Ablauf des Lernalgorithmus, statt m nun $m*k$ Beispiele verarbeiten, wobei k die Anzahl der Agenten bezeichnet. Denn wenn der Agent selbst m Beispiele hat, dann erhält er von den anderen Agenten noch $(k-1)*m$ Beispiele hinzu, woraus sich insgesamt $m*k$ ergibt. Die Laufzeit liegt daher in $O(m*k*n^2)$.

Nutzen

Der zu erwartende Nutzen dieser Methode lässt sich annähernd durch die Formel für die mindestens benötigten Trainingsbeispiele (siehe 3.4.3) abschätzen. Um mit Wahrscheinlichkeit $(1-\delta)$ eine Hypothese zu erhalten die einen Fehler kleiner ε enthält sind mindestens

$m_0 = \frac{1}{\varepsilon} * (ln|H| + ln(\frac{1}{\delta}))$

Beispiele nötig. Wieder angenommen die Agenten machen ihre Erfahrung mit derselben Geschwindigkeit und tauschen ihre Beispiele ständig aus, dann reduziert sich diese Zahl bei k Agenten auf $\frac{m_0}{k}$, denn der Agent erhält zu seinen eigenen m Bespielen $(k-1)*m$ Beispiele hinzu, hat also die erforderliche Anzahl von m_0 Beispielen, wenn er $\frac{m_0}{k}$ eigene Beispiele hat. Bei fünf Agenten heißt dass beispielsweise, dass er statt hundert eigenen Beispielen nur noch zwanzig benötigt.

Voraussetzungen für den Einsatz

Die Voraussetzung für den Einsatz dieser Methode ist zum einen, dass die Agenten ihre Beispiele explizit speichern (was nicht unbedingt bei jeder Lernmethode der Fall ist). Zum anderen müssen die Beispiele wenigstens ähnlich repräsentiert sein, so dass die Agenten die "fremden" Beispiele verarbeiten können. Diese Voraussetzungen sind in vielen Fällen gegeben.

Vorteile

- sehr einfach zu implementieren
- unabhängig von dem verwendeten Lernverfahren
- alle Information, die in der Erfahrung enthalten ist, wird auch weitergegeben

- funktioniert auch noch, wenn Kommunikation nur von Zeit zu Zeit verfügbar ist

Nachteile

- hoher Kommunikationsaufwand
- sehr hohe Rechenkosten
- Alle Beispiele werden von jedem Agenten verarbeitet, es findet also keine Arbeitsteilung beim Lernen statt

6.3 Austausch von Hypothesen

Statt ihre gesamte Erfahrung auszutauschen, was sehr aufwendig sein kann, können die Agenten auch nur ihre Hypothesen austauschen. Da man nun annehmen kann, dass in vielen Fällen diese Hypothesen kleiner und schneller zu verarbeiten sind als die Trainingsbeispiele, die ihnen zu Grunde liegen, aber dennoch alle relevante Informationen enthalten, werden somit die Vorteile des Austauschs von Trainingsbeispielen genutzt, ohne die Nachteile in Kauf nehmen zu müssen.

6.3.1 Agentenmodell

Der Agent erhält zu seinen bisherigen Wissensbasen nun noch eine weitere hinzu, welche sein Wissen über andere Agenten repräsentiert. Dabei werden über jeden Agenten folgende Informationen gespeichert:

- eine Identifikationsnummer, um mit diesem Agenten Kontakt aufnehmen zu können
- den Zeitpunkt, an dem zum letzten Mal von dem Agenten eine Hypothese erhalten wurde, um zu ermitteln, ob es sich lohnt ihm eine neue Anfrage zu senden
- die Gesamtzahl von Fällen, in denen von diesem Agenten eine Hypothese erhalten wurde, bzw. ihm gesendet wurde, um Strategien zu entwickeln, wann mit diesem Agenten kooperiert werden soll.

Außerdem muss der Agent zwei verschiedene Hypothesen speichern, zum einen diejenige, die er nur aus seinen eigenen Beispielen gewonnen hat, und die er an andere Agenten schickt, zum anderen diejenige Hypothese, welche er aus seinen Beispielen und den fremden Hypothesen gebildet hat, und welche die Grundlage seiner Entscheidungen ist. Damit wird verhindert, dass es zu Rückkopplungseffekten kommt.

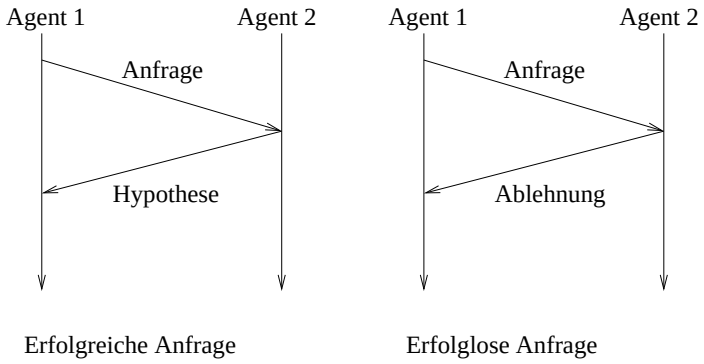


Abbildung 8: Protokoll: Austausch von Hypothesen

Zum Ablaufmodell des Agentens wird eine Kommunikationsphase hinzugefügt. Während dieser Phase ist es dem Agenten möglich, Nachrichten mit anderen Agenten auszutauschen. Das in Abbildung 8 dargestellte Kommunikationsprotokoll sieht nur zwei Nachrichtentypen vor:

1. *Anfrage*: Ein Agent schickt einem anderen Agenten die Aufforderung, seine Hypothese zu senden
2. *Antwort*: Dieser reagiert auf eine Anfrage, entweder indem er auf sie eingeht und seine Hypothese in Form von Regeln sendet, oder dies ablehnt und entsprechend eine Ablehnung sendet.

Das gesamte Ablaufmodell ist in Abbildung 9 dargestellt.

Der Agent entscheidet zunächst auf Grundlage seiner Strategie, wem er eine Anfrage senden soll. Dann versendet er die entsprechenden Anfragen und wartet auf die Antwort. Der weitere Verlauf ist wie im Fall des nicht-kooperativen Lernens, mit der Ausnahme, dass der Agent seine eigene Hypothese in jedem Fall aktualisiert, wenn er mindestens eine fremde Hypothese erhalten hat.

Parallel dazu nimmt er Anfragen anderer Agenten entgegen und antwortet seiner Strategie entsprechend, entweder damit, dass er eine Hypothese sendet (und zwar eine, die er nur auf Grundlage seiner eigenen Beispiele gebildet hat), oder mit einer Ablehnung.

6.3.2 Algorithmus

Nachdem ein Agent die Hypothese eines anderen Agenten erhalten hat, muss er diese mit seiner eigenen Hypothese verschmelzen. Dazu gibt es, wie bereits oben erwähnt, zwei Ansätze. Entweder der Agent bildet zunächst seine eigene Hypothese auf Grundlage seiner eigenen Trainingsbeispiele und verschmilzt

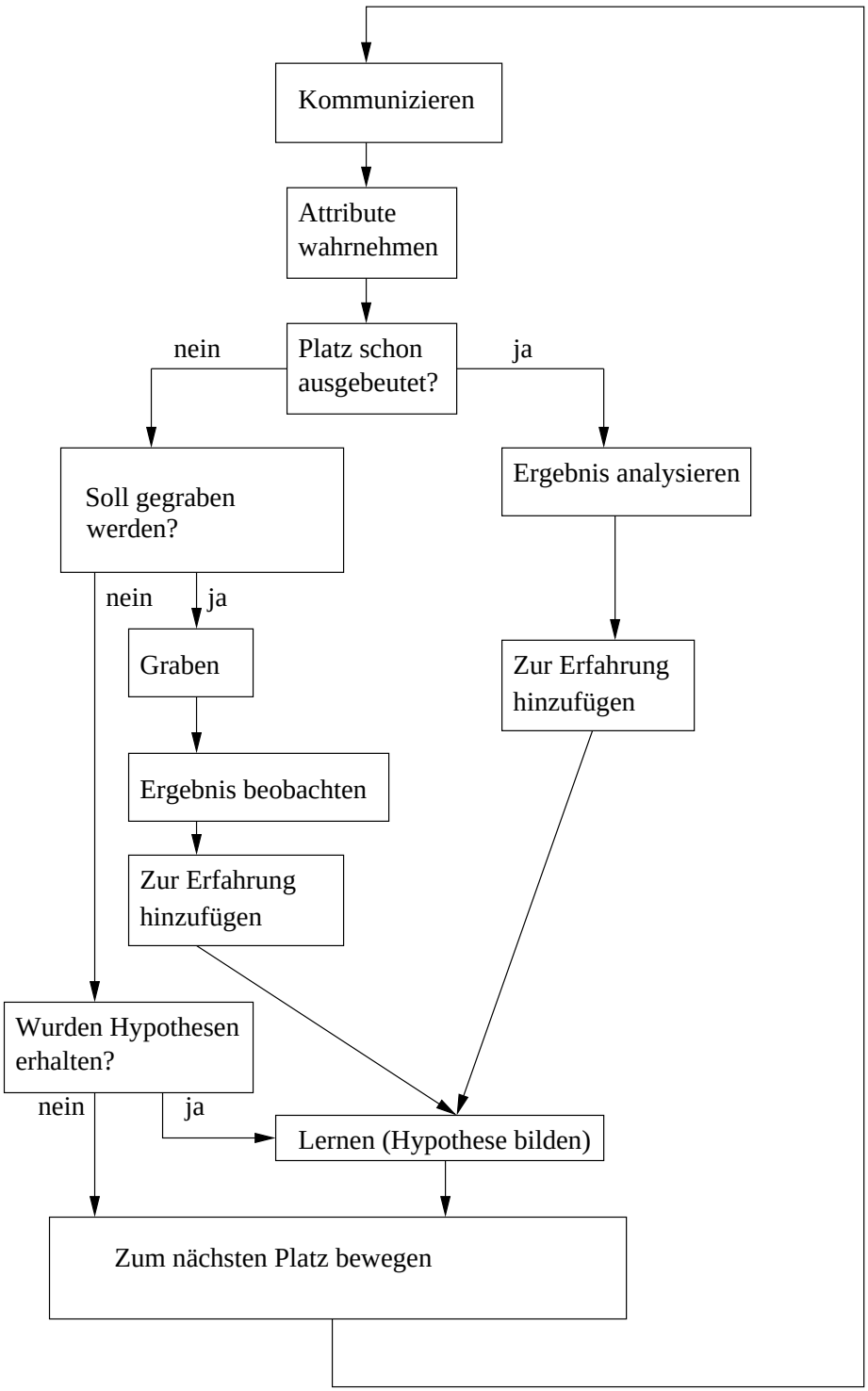


Abbildung 9: Ablauf: Austausch von Hypothesen

dann diese Hypothese mit der des anderen Agenten. Die andere Möglichkeit besteht darin, dass der Agent bereits zum Zeitpunkt, an dem er aus seinen eigenen Trainingsbeispielen lernt, die fremde Hypothese heranzieht.

Beide Ansätze wurden bereits mehrfach untersucht.

In [2] wird untersucht, wie sich Regeln, die auf verschiedene Weise aus einer Menge von Trainingsbeispielen gewonnen wurden, zu einer einzigen Regelmenge zusammenfassen lassen. Dabei wird für jede Regel festgehalten, welche Trainingsbeispiele sie überdeckt. Auf dieser Grundlage wird aus allen Einzelregelmengen eine optimierte Gesamtregelmenge gebildet.

Ein Problem, das sich dabei ergeben kann, ist, dass die Regeln der einzelnen Agenten logisch widersprüchlich sind. Dazu kommt es beispielsweise, wenn die Trainingsbeispiele verschiedene Hypothesen zulassen, die mit ihnen vereinbar sind (wenn der "Version-Space" mehr als eine Hypothese enthält). Dann kann es sein, dass verschiedene Agenten verschiedene Hypothesen auswählen und diese Hypothesen müssen nicht mehr unbedingt logisch vereinbar sein. In [2] wird dieses Problem gelöst, indem auf Ebene der Verschmelzung der Hypothesen (Regeln) eine quantitative Bewertung der einzelnen Regeln erfolgt und zwar auf Grundlage der Trainingsbeispiele. Es wird dann die nach dieser Bewertung beste logisch konsistente Menge von Regeln ausgewählt.

Ein anderer Ansatz zur Lösung dieses Problems wird in [4] vorgestellt. Die Idee besteht darin, die Agenten nicht nur die Hypothese übermitteln zu lassen, sondern den gesamten "version-space", also die Menge aller mit den Trainingsbeispielen konsistenten Hypothesen. Der Agent, welcher diese Information erhält, kann nun zunächst seinen eigenen "version space" mit demjenigen, den er von dem anderen Agenten erhalten hat, schneiden und erhält somit die Menge aller möglichen Hypothesen, auf Grundlage der Erfahrung beider Agenten. Da es in vielen Fällen sehr aufwendig wäre, den "version-space" als solchen zu übertragen, wird eine Methode benutzt, die beispielsweise in [13][8] vorgestellt wird: der "version-space" wird durch zwei Mengen dargestellt, welche ihn in Richtung der spezifischeren und generelleren Hypothesen begrenzen.

Der zweite Ansatz, die Einbindung einer Fremdhypothese als Hintergrundwissen für den Induktionsvorgang, wird in allgemeiner Form in [12] diskutiert. Die Autoren stellen einen allgemeinen Rahmen auf, der es zum einen erlauben soll, bisherige Induktionsalgorithmen, die Gebrauch von Hintergrundwissen machen, zu klassifizieren, und zum anderen Anregungen für die zukünftige Entwicklung von solchen Algorithmen zu geben. Dabei gehen sie zunächst von der Frage aus, wie Algorithmen, die keinen Gebrauch von Hintergrundwissen machen, in solche umgewandelt werden, die dies tun. Zum einen würde dies durch Umformulierung der Algorithmen selbst geschehen, so dass diese andere Informationen verarbeiten könnten als nur Trainingsbeispiele. Zum anderen kann die Eingabe in den Algorithmus zunächst vorverarbeitet werden. Auf diese Weise lässt sich

[8] S.549 ff.

Hintergrundwissen in Pseudobeispiele umformen, welche dann zusammen mit den Trainingsbeispielen von dem Originalalgorithmus verarbeitet werden können. Im letzten Teil der Arbeit beschreiben die Autoren noch ein ehrgeizigeres Projekt, nämlich einen Rahmen zu finden in den alle Algorithmen passen, die Hintergrundwissen zur Induktion nutzen, also nicht nur diejenigen, welche durch Modifikation aus Algorithmen entstanden sind, die kein Hintergrundwissen nutzen.

Ein konkreter Ansatz zum Einsatz von Hintergrundwissen aus verschiedenen Quellen wird in [15] beschrieben. Dabei wird Hintergrundwissen grundsätzlich als Tripel (H, C, P) beschrieben, wobei H der fragliche Hypothesenraum ist, $C \subseteq H$ Randbedingungen (constrains) und zwar als Menge aller Hypothesen des Hypothesenraums, welche den constraints entsprechen. P gibt auf den Hypothesen aus C noch eine Präferenzordnung vor. Hintergrundwissen aus verschiedenen Quellen wird nun jeweils zunächst auf die Form (H, C, P) gebracht, anschließend werden alle diese Tripel zu einem einzigen Tripel verschmolzen. Beim Induktionsvorgang werden dann nur noch Hypothesen aus C betrachtet, für die es keine andere Hypothese gibt, die genauso gut mit den Daten zusammenpasst und in der Präferenzordnung höher steht. Voraussetzung dabei ist, dass durch das Verschmelzen der einzelnen Teile des Hintergrundwissens weder eine zyklische Präferenzordnung, noch $C = \emptyset$ entsteht.

Der in [2] dargestellte Ansatz ist in dem gegebenen Szenario kaum zu verwirklichen, da der Rückgriff auf die ursprüngliche Menge an Trainingsbeispielen erforderlich ist. Der Austausch von "version-spaces", wie er in [4] dargestellt ist, ist ebenfalls problematisch. Denn es sollen auch inkonsistente Daten verarbeitet werden. Sich widersprechende Beispiele führen aber oft zu einem leeren "version space", also dazu, dass der Agent überhaupt keine Hypothese findet.

In dieser Arbeit soll der zweite Ansatz, also das Einbinden der fremden Hypothese als Hintergrundwissen, untersucht werden. Dieser bietet sich besonders an, da die Agenten ihre Hypothesen ständig aktualisieren müssen. Indem sie dabei direkt die fremden Hypothesen zum Einsatz bringen, sparen sie sich das anschließende Verschmelzen von Hypothesen. Als Algorithmus zum Einbinden von Hintergrundwissen in den Induktionsvorgang soll eine Abwandlung des ID3-Algorithmus benutzt werden. Dabei werden die Hypothesen der anderen Agenten als Pseudobeispiele aufgefasst und zur Gesamtmenge der Trainingsbeispiele dieses Agenten hinzugefügt. Dann wird der Algorithmus zum Aufbau von Entscheidungsbäumen auf diese Menge von Beispielen angewendet.

Umwandlung eines Entscheidungsbaums in eine Menge gewichteter Pseudobeispiele

Der Agent, der die Hypothese sendet, muss zunächst den Entscheidungsbaum, der diese Hypothese repräsentiert, in eine Regelmenge umwandeln. Dazu verfolgt er jeden Pfad von der Wurzel zu einem Blatt. Die dabei besuchten Knoten und Kanten ergeben die Bedingung der Regel, das Blatt die Folge. Der in Abbildung 10 dargestellte Entscheidungsbaum ergibt beispielsweise die folgende

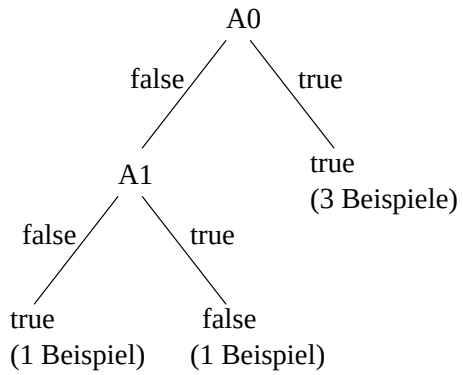


Abbildung 10: Entscheidungsbaum - Umwandlung in Regeln

Regelmenge:

$IF\ A_0 = false \wedge A_1 = false\, THEN\ K = true$

$IF\ A_0 = false \wedge A_1 = true\, THEN\ K = false$

$IF\ A_0 = true\, THEN\ K = true$

Jeder Regel kann ein Gewicht zugeordnet werden, und zwar danach, wie viele der Trainingsbeispiele durch diese Regel abgedeckt werden. Man kann jetzt eine Regel als Pseudobeispiel schreiben, indem man die Bedingungen zu Attributwerten macht und die Folge als Klassenattribut des Beispiels auffasst. Attribute, die nicht im Bedingungsteil auftreten, werden als unbekannt gesetzt. Es ergibt sich:

$((false, false), true, 1.0)$

$((false, true), false, 1.0)$

$((true, ?), true, 3.0)$

Eine Menge von solchen Regeln wird in Form von gewichteten Pseudobeispielen an den anfragenden Agenten übertragen.

Einbindung der Pseudobeispiele in den Induktionsvorgang

Der klassische ID3-Algorithmus muss in drei Punkten erweitert werden, um die Pseudobeispiele effektiv verarbeiten zu können:

1. Er muss mit gewichteten Beispielen umgehen können, denn mit einem Pseudobeispiel wird immer auch gespeichert, wie viele Trainingsbeispiele die ursprüngliche Regel abgedeckt hat.

2. Er muss mit fehlenden Attributwerten umgehen können, die dadurch entstehen, dass in den Pseudobeispielen die Attribute, welche in der Regel nicht vorkommen, als unbekannt gesetzt werden.

3. Es müssen Möglichkeiten geschaffen werden, "overfitting"[9] zu vermeiden, da ID3 dies von sich aus nicht tut. Maßnahmen gegen "overfitting" zu ergreifen ist deshalb wichtig, da logisch inkompatible Hypothesen, die durchaus häufiger auftreten können, ansonsten zu einem sehr komplexen und wenig effektiven Entscheidungsbaum führen würden.

Das erste Problem lässt sich durch eine Modifikation für die Berechnung der Entropie lösen. Dabei wird ein Trainingsbeispiel, welches beispielsweise Gewicht 2.0 hat, genauso behandelt, als wären in der Menge zwei identische Beispiele. Der Zweck davon liegt darin, dass eine Regel, die viele Trainingsbeispiele abdeckt, auch als Pseudobeispiel ein höheres Gewicht erhalten sollte.

Fehlende Attributwerte werden folgendermaßen behandelt. Bei der Berechnung der Informationsentropie wird ein Beispiel mit fehlendem Attributwert beiden Untermengen, also derjenigen, welche die Beispiele enthält, die für dieses Attribut den Wert *true* enthalten und derjenigen, welche die Beispiele enthält, die für dieses Attribute den Wert *false* enthalten, zugerechnet, jeweils mit halbem Gewicht. Dies führt im allgemeinen zu einer Erhöhung der Gesamtentropie, denn beiden Mengen wird ein Beispiel mit gleicher Klasse hinzugefügt (zu einer Verringerung der Informationsentropie kommt es ja, indem Beispiele mit verschiedenen Klassen, separiert werden).

In der selben Art werden nun diese Beispiele auch weiterverarbeitet, d.h. sie werden an beide Unterbäume weitergegeben, wobei wiederum ihr Gewicht jeweils halbiert wird.

Dieses Vorgehen ist deshalb sinnvoll, da ein fehlender Attributwert im vorliegenden Fall darauf hinweist, dass das entsprechende Attribut nicht relevant ist. Indem nun fehlende Attributwerte dazu führen, dass durch eine Aufteilung nach dem entsprechenden Attribut tendenziell ein kleinerer Gewinn an Information erreicht wird, werden Attribute, bei denen weniger fehlende Werte auftreten, bevorzugt. Das ist aber erwünscht, da es darauf hinweist, dass dieses Attribut in mehreren Regeln relevant ist.

Um auch hier den Effekt des "overfittings" zu verhindern, müssen Methoden gefunden werden, wie der Entscheidungsbaum beschnitten werden kann. Während C4.5, als Erweiterung von ID3, post-pruning einsetzt, also den Baum beschneidet nachdem er vollständig aufgestellt wurde, ist dieser Ansatz hier problematisch, da nicht wirkliche Trainingsbeispiele die Grundlage für den Aufbau des Baumes bilden, sondern Pseudobeispiele, die nur in einem indirekten Zusammenhang zu den ursprünglichen Trainingsbeispielen stehen. Es soll daher ein direkteres Verfahren zur Vermeidung von overfitting angewendet werden, nämlich pre-pruning. Wenn die Aufteilung nach keinem der noch möglichen Attribute einen Gewinn bringt, der höher ist als ein gewisser Schwellenwert, dann

[9] "overfitting" bezeichnet den Effekt, dass sich eine Hypothese zwar sehr gut an die Trainingsbeispiele anpasst, aber bei einer Anwendung auf neue Beispiele einen hohen Fehler hat. Dies kann auftreten, wenn die Trainingsbeispiele Rauschen oder Fehler enthalten.

wird dieser Knoten als Blatt aufgefasst, dessen Klasse sich aus einer gewichteten Mehrheitswahl zwischen den verbleibenden Trainingsbeispielen ergibt.

6.3.3 Strategie

Die Strategie lässt sich analog zur Strategie im Fall des Austauschs von Trainingsbeispielen entwickeln. Was die Frage betrifft, wann ein Agent auf eine Anfrage seine Hypothese senden soll, so kann man wiederum annehmen, dass er dies immer tut, da es sich um ein Team von Agenten handelt, in welchem sich die einzelnen Mitglieder voll unterstützen. Andernfalls können die Agenten die Information nutzen, wie oft sie bereits eine Hypothese empfangen bzw. gesendet haben, um zu ermitteln, ob eine Anfrage positiv oder negativ beantwortet wird (positiv, falls die Differenz nicht zu groß ist). Für diese Arbeit wird angenommen, dass die Agenten jede Anfrage positiv beantworten.

Bei der zweiten Frage geht es darum, ob der Nutzen durch die fremde Hypothese die Kosten für die Kommunikation aufwiegt. Worin besteht der Nutzen einer fremden Hypothese? Der Nutzen einer fremden Hypothese besteht für einen Agenten darin, dass er aus dieser zusammen mit seiner Erfahrung eine Hypothese bilden kann, die einen geringeren Fehler hat, als die Hypothese, die er nur auf Grundlage seiner eigenen Erfahrung hätte bilden können. Nun kann man aber wiederum zwei Annahmen machen:

1. Mit steigender Anzahl an Trainingsbeispielen verändern sich die Hypothesen immer weniger.
2. Kann der Agent bereits auf Grundlage seiner eigenen Erfahrung eine Hypothese bilden, die nahe am Optimum ist, dann kann es prinzipiell nicht zu einer größeren Verbesserung durch die fremde Hypothese kommen. Da sich nun die Hypothesen im allgemeinen mit steigender Anzahl an Trainingsbeispielen immer mehr diesem Optimum annähern, sinkt im allgemeinen auch der Nutzen einer fremden Hypothese.

Dabei bietet sich eine ähnliche Strategie an, wie im Fall des Austauschs von Erfahrung. Am Anfang stellen die Agenten recht häufig Anfragen an andere Agenten, da sie davon ausgehen, dass sich die Hypothesen der anderen Agenten häufig ändern und ihnen selbst viel nützen, da sie selbst nicht genug Erfahrung haben, um nur aus ihren Trainingsbeispielen ausreichend gut zu lernen. Mit fortschreitenden Simulationsschritten sinkt der Nutzen durch fremde Hypothesen, da sich diese nur noch wenig verändern und die Agenten andererseits genug eigene Erfahrung gesammelt haben, um daraus eine nur geringfügig verbesserbare Hypothese zu bilden. Zusammen mit der Strategie für nicht-kooperatives Lernen ergibt sich:

1. Verfügt der Agent über weniger als m_0 Beispiele, so sollte er immer graben.

Da er immer gräbt, braucht er nicht zu kommunizieren, denn eine Verbesserung seiner Hypothese ist nicht relevant, er macht keinen Gebrauch von ihr.

2. Verfügt der Agent über mehr als m_0 Beispiele, aber weniger als m_1 Beispiele, dann sollte er sich zwar nach seiner Hypothese richten, gleichzeitig aber von den anderen Agenten Hypothesen anfordern, um seine eigene Hypothese zu verbessern.

3. Verfügt der Agent über mehr als m_1 Beispiele, dann ist der Nutzen durch aktualisierte fremde Hypothesen nicht hoch genug, um die Kosten für die Kommunikation auszugleichen. Daher fordert der Agent keine weiteren Hypothesen mehr an.

Es bleibt noch die Frage, von wem der Agent Hypothesen fordern sollte. Da es keinen Grund gibt einen Agenten einem anderen vorzuziehen, kann man davon ausgehen, dass er immer von den Agenten Hypothesen fordert, von denen er schon länger keine mehr erhalten hat. Hier wird davon ausgegangen, dass ein Agent immer von allen anderen Agenten Hypothesen anfordert.

Eine zweite Frage ist, wie oft der Agent innerhalb der Phase, in der er kommuniziert, Hypothesen anfordern sollte. Diese Frage wird in Abschnitt 7 experimentell untersucht.

6.3.4 Qualitative Analyse

Kosten

Die Kosten, sowohl was Kommunikation, als auch was die Rechenleistung betrifft, lassen sich nur schwer realistisch schätzen, da sie sehr stark davon abhängen, wie groß die Hypothesen sind, welche die Agenten bilden. Das hängt wiederum von dem Zusammenhang ab, der zwischen dem Vorhandensein einer Energiezelle und den Attributen eines Platzes besteht. Ist dieser Zusammenhang sehr komplex, dann ist zu erwarten, dass auch die einzelnen Hypothesen komplexer und damit umfangreicher werden.

Eine obere Abschätzung erhält man aus der Tatsache, dass die Anzahl der Blätter eines Entscheidungsbaums immer kleiner ist als die Anzahl der Beispiele, aus denen dieser Baum aufgebaut wurde. Damit ist auch die Anzahl der Regeln immer kleiner als die Anzahl der Beispiele. Allerdings muss man dabei beachten, dass immer die komplette Hypothese übertragen wird, wohingegen beim Austausch von Trainingsbeispielen immer nur die neuen Beispiele übertragen werden.

Nutzen

Für den Nutzen gilt dasselbe, wie für die Kosten. Geht man davon aus, dass

die Hypothesen alle relevanten Informationen enthalten, dann sollte die Verbesserung durch eine fremde Hypothese meist ähnlich hoch sein, wie durch die entsprechenden Trainingsbeispiele, also wie beim Austausch von Erfahrung.

Voraussetzungen für die Anwendung

Die Voraussetzungen für die Anwendung sind wesentlich höher als bei den anderen beiden Verfahren. Denn alle beteiligten Agenten müssen Lernalgorithmen verwenden, die Regeln generieren. Diese Regeln müssen untereinander austauschbar und jeweils in den Lernprozess integrierbar sein.

Vorteile

- Es muss weniger Information übertragen werden.
- Jeder Agent bearbeitet seine eigenen Beispiele, keine Beispiele werden doppelt ausgewertet.

Nachteile

- Es ist schwer einzuschätzen, wie hoch der Nutzen und wie hoch die Kosten bei diesem Verfahren sind.
- Die Voraussetzungen für den Einsatz sind höher als bei den anderen Verfahren.

6.4 Abgleich von Entscheidungen

Bevor der Agent sich entscheidet, ob er an einer gegebenen Stelle graben soll, fragt er andere Agenten um Rat, indem er ihnen den Attributvektor zusendet, welcher die Stelle repräsentiert, an der er sich befindet. Jeder dieser Agenten entscheidet nun auf Grundlage seiner eigenen Hypothese, ob er an dieser Stelle graben würde und sendet das Ergebnis zurück. Aufgrund dieser Rückmeldungen trifft der Agent seine endgültige Entscheidung.

6.4.1 Agentenmodell

Der Agent erhält zu seinen bisherigen Wissensbasen nun noch eine weitere hinzu, welche sein Wissen über andere Agenten repräsentiert. Dabei werden über jeden anderen Agenten folgende Informationen gespeichert:

- eine Identifikationsnummer, um mit diesem Agenten Kontakt aufnehmen zu können

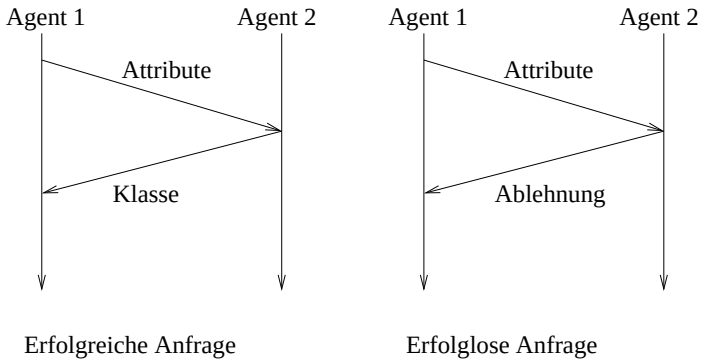


Abbildung 11: Protokoll: Abgleich von Entscheidungen

- die Gesamtzahl von Fällen, in denen diesem Agenten ein Rat gegeben wurde, bzw. die Anzahl der Fälle in welchen dieser Agent einen Rat erteilte, um eine Kooperationsstrategie zu ermöglichen

Für das Kommunikationsprotokoll (Abbildung 11) sind wiederum nur zwei Nachrichtentypen notwendig:

1. eine Anfrage, welche den Attributvektor enthält, der die gegebene Stelle repräsentiert
2. eine Antwort, welche entweder die entsprechende Klasse enthält oder eine Ablehnung

Der Ablaufzyklus des Agenten ist in Abbildung 12 dargestellt.

Zunächst verfährt der Agent wie beim nicht-kooperativen Lernen: er nimmt die Attribute an diesem Platz wahr und prüft, ob der Platz schon ausgebeutet ist. Ist dies der Fall, analysiert er das Ergebnis des Grabens, fügt es seiner Erfahrung hinzu und aktualisiert seine Hypothese. Ist der Platz noch nicht ausgebeutet, muss sich der Agent zunächst überlegen, ob er andere Agenten um Rat fragen oder nur auf Grundlage seiner eigenen Hypothese entscheiden soll. Entscheidet er sich dafür, andere Agenten um Rat zu fragen, dann schickt er ihnen Anfragen mit dem Attributvektor, welcher seine aktuelle Stelle repräsentiert, und wartet auf die Antworten. Aus seiner eigenen Entscheidung und den Antworten bildet er nun seine Entscheidung darüber, ob er an der aktuellen Stelle graben soll oder nicht. Ob er um Rat fragt oder nicht, entscheidet die Strategie des Agenten. Falls er nicht um Rat fragt, entscheidet er direkt auf Grundlage seiner eigenen Hypothese. Der weitere Ablauf ist identisch mit dem Ablauf für nicht-kooperatives Lernen. Parallel zu diesem Ablauf nimmt der Agent noch Anfragen anderer Agenten entgegen. Auf Grundlage seiner Strategie entscheidet er, ob eine Ablehnung zurückgesendet oder das fragliche Beispiel mit Hilfe seiner Hypothese klassifiziert und dann das Ergebnis zurückgesendet wird.

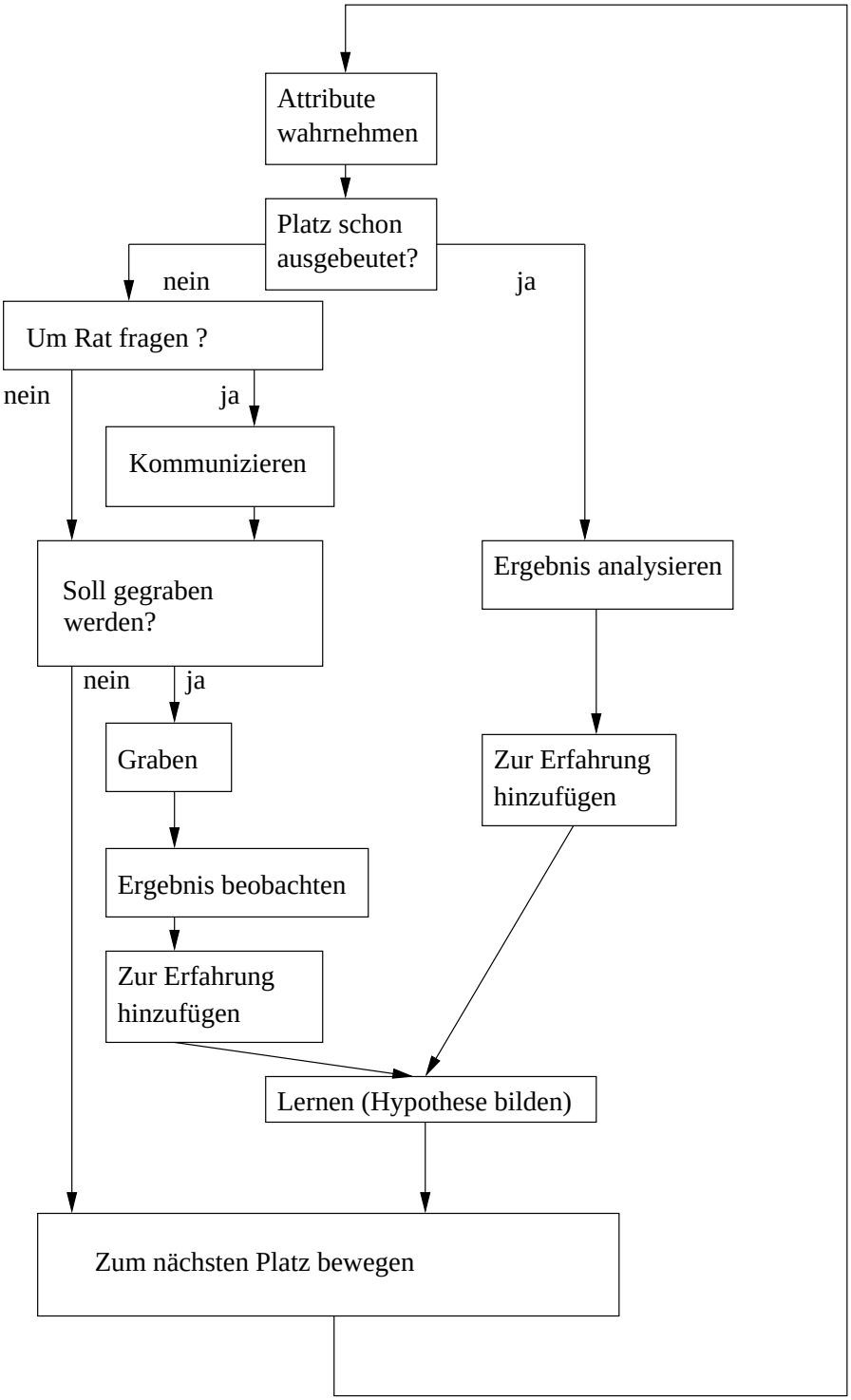


Abbildung 12: Ablauf: Abgleich von Entscheidungen

6.4.2 Algorithmus

In [7] werden verschiedene Verfahren zur Verschmelzung von Entscheidungen untersucht. Dabei wird der allgemeinere Fall betrachtet, in dem jeder Klassifizierer nicht nur die wahrscheinlichste Klasse ausgibt, sondern auch die Wahrscheinlichkeit jeder Klasse. Aus allen Entscheidungen ergibt sich eine Matrix, deren Zeilen jeweils die Ausgabe eines Klassifizierers entspricht, und deren Spalten die Wahrscheinlichkeiten enthalten, welche jeweils dieser Klasse zugeordnet werden. Da hier nur der Fall mit zwei Klassen betrachtet wird, ergibt sich folgende Matrix:

$$DP(x) = \begin{pmatrix} p_{1,0}(x) & p_{1,1}(x) \\ p_{2,0}(x) & p_{2,1}(x) \\ \dots & \dots \\ p_{k,0}(x) & p_{k,1}(x) \end{pmatrix}$$

wobei $p_{i,j}(x)$ die Wahrscheinlichkeit ist, welche der i-te Klassifizierer dafür ausgibt, dass die Instanz x in Klasse j liegt. Die Frage ist nun, wie sich aus $DP(x)$ eine Entscheidung darüber ableiten lässt, welcher Klasse die Instanz x als Gemeinschaftsentscheidung zugeordnet werden sollte. Dazu wird normalerweise zunächst für jede Klasse ein "support" berechnet und dann diejenige Klasse mit dem höchsten "support" ausgewählt. Für die Berechnung dieses "supports" gibt es verschiedene Verfahren, die sich grob nach drei Aspekten einteilen lassen:

- Ist das Verfahren in Fällen anwendbar, in denen die Agenten eine Wahrscheinlichkeitsverteilung über den Klassen ausgeben oder in Fällen, in denen nur die wahrscheinlichste Klasse ausgegeben wird?
- Greift das Verfahren auf der Ebene der Verschmelzung von Einzelentscheidungen auf die Beispiele zurück, welche die Grundlage der einzelnen Entscheidungen bilden? Die einzelnen Hypothesen werden auf Grundlage von den Beispielen getroffen, über die der jeweilige Agent verfügt. Diese Hypothesen entscheiden über die Werte in den einzelnen Zeilen von $DP(x)$. Nun kann man zur Bildung der Gesamtentscheidung die ursprünglichen Beispiele wieder heranziehen, beispielsweise um Informationen über die statistische Verteilung einzelner Attributwerte zu erhalten. Um solche Verfahren einsetzen zu können, muss die Gesamtbeispielmenge zum Zeitpunkt der Verschmelzung zugänglich sein.
- Wird zur Berechnung des "supports" für eine Klasse die gesamte Matrix $DP(x)$ verwendet oder nur die Spalte, welche die Wahrscheinlichkeiten für diese Klasse enthält? Im ersten Fall spricht man von "class indifferent" Verfahren, im zweiten Fall von "class conscious" Verfahren.

Da im gegebenen Szenario ein Rückgriff auf die Gesamtdatenmenge nicht möglich ist, denn die Agenten tauschen ihre Beispiele eben nicht aus, sind Verfahren, welche darauf beruhen von vorne herein nicht anwendbar. Da die Agenten nur

das Klassenattribut ausgeben, welches auf Grundlage ihrer Hypothese das wahrscheinlichste ist, sind auch Verfahren, welche eine Wahrscheinlichkeitsverteilung voraussetzen nicht anwendbar (formal kann man die Schreibweise mit Wahrscheinlichkeiten beibehalten, indem die Klasse, welche der Agent ausgewählt Wahrscheinlichkeit 1.0 erhält, die andere Klasse 0.0). Daher bleibt für den vorliegenden Fall nur ein sinnvolles Verfahren übrig und dies ist die Mehrheitswahl, oder "majority vote".

Der "support" einer Klasse ergibt sich als Anzahl der Agenten, welche diese Klasse für die wahrscheinlichste halten:

$support_j(x) = \sum_{i=0}^{k} p_{i,j}(x)$

Die Klasse mit dem höchsten "support" wird als Gesamtentscheidung ausgegeben. Ist die Anzahl der Agenten, die eine Antwort zurücksenden, ungerade, kommt es immer zu einer eindeutigen Entscheidung (da es nur zwei Klassen gibt). Ist die Anzahl gerade, dann kann es zu einem Unentschieden kommen. In einem solchen Fall wird die Entscheidung zufällig getroffen, wobei jede Klasse gleich wahrscheinlich auftritt.

6.4.3 Strategie

Wiederum geht es bei der Strategie um zwei Fragen:

1. Wann und wen soll der Agent um Rat fragen?
2. Wenn der Agent um Rat gefragt wird, in welchen Fällen soll er antworten?

Was die zweite Frage angeht, kann man wieder von einem hilfsbereiten Agenten ausgehen, der auf alle Anfragen eingeht. Eine andere Möglichkeit ist, den Agenten die Differenz der Anzahl der Fälle, in denen er dem anfragenden Agenten einen Rat gegebenen und der Fälle, in denen er von diesem einen Rat erhalten hat, bilden zu lassen. Ist diese Differenz zu groß, verweigert der Agent die Auskunft. Damit kann man in jedem Fall dafür sorgen, dass kein Agent einen anderen regelmäßig ausnutzt.

Die erste Frage bezieht sich darauf, ob die (vermeintlich) bessere Entscheidung, die von mehreren Agenten getroffen wird, die Kommunikationskosten aufwiegt. Es soll hier davon ausgegangen werden, dass der Agent immer alle $k-1$ Agenten um Rat fragt.

Wann soll der Agent nun die anderen Agenten um Rat fragen? Wenn der Agent rational handelt, immer dann, wenn $E_{maj} > (k-1) * E_k$, wenn der Nutzen der Mehrheitsabstimmung E_{maj} größer ist als die Kosten mit $k-1$ Agenten zu kommunizieren. Wie lässt sich E_{maj} näher bestimmen? Zunächst kann man untersuchen, um wie viel die Entscheidung, die von k Agenten getroffen wird,

besser ist, als die eines einzelnen Agenten. Grundlage bildet die Annahme, dass alle Agenten Hypothesen besitzen, welche einen Fehler von ε haben und dass die Einzelentscheidungen durch "majority vote" kombiniert werden. Dabei wird der Irrtum eines Agenten als Ereignis betrachtet, welches mit Wahrscheinlichkeit ε eintritt. Man kann nun annehmen, dass diese Ereignisse unabhängig sind, dass also das Ereignis, dass sich ein Agent irrt, nicht von dem Ereignis abhängt, dass sich ein anderer Agent irrt. Im folgenden soll diese Annahme gemacht werden. Dann ergibt sich für die Wahrscheinlichkeit des Ereignisses "alle k Agenten zusammen irren sich" folgendes:

- ist k ungerade, dann irren sich die Agenten in ihrer gemeinschaftlichen Entscheidung, wenn sich mindestens $\frac{k+1}{2}$Agenten, also mehr als die Hälfte irren. Aus der Unabhängigkeitsannahme ergibt sich:
  $\varepsilon_{maj} = \sum_{i=\frac{k+1}{2}}^{k} \varepsilon^i (1-\varepsilon)^{(k-i)} * \begin{pmatrix} k \\ i \end{pmatrix}$
- ist k gerade und wird ein Unentschieden zufällig mit derselben Wahrscheinlichkeit für beide Ausgänge gelöst, dann gibt es zwei Fälle in denen sich alle Agenten gemeinsam irren. Zum einen, wenn mehr als $\frac{k}{2}+1$ Agenten sich irren, und zum anderen, wenn sich genau $\frac{k}{2}$ Agenten irren und die zufällige Entscheidung falsch ist. Damit ergibt sich:
  $\varepsilon_{maj} = \frac{1}{2}\varepsilon^{\frac{k}{2}}(1-\varepsilon)^{\frac{k}{2}} \begin{pmatrix} k \\ \frac{k}{2} \end{pmatrix} + \sum_{i=\frac{k}{2}+1}^{k} \varepsilon^i (1-\varepsilon)^{(k-i)} * \begin{pmatrix} k \\ i \end{pmatrix}$

In Abhängigkeit von ε und von der Anzahl der beteiligten Agenten k kann man nun berechnen, wie sich die Fehlerwahrscheinlichkeit verändert, wenn eine Entscheidung nicht von einem, sondern von mehreren Agenten getroffen wird. Abbildung 13 zeigt die Differenz $\varepsilon_{maj} - \varepsilon$ für $k = 3$, $k = 5$ und $k = 21$.

Aus dem Diagramm lassen sich verschiedene Dinge ablesen:

- Wenn sich die einzelnen Agenten mit einer Wahrscheinlichkeit größer als 0.5 irren, dann schadet "majority vote", denn die Gesamtentscheidung ist noch schlechter. (Für $\varepsilon > 0.5$ ist $(\varepsilon_{maj} - \varepsilon) > 0$, d.h. der Fehler mit "majority vote" ist größer als der Fehler ohne "majority vote").
- Ist die Irrtumswahrscheinlichkeit hingegen kleiner als 0.5, dann lohnt sich "majority vote" immer, wenn auch in verschiedenem Ausmaß.
- Ab einem gewissen ε_0 lohnt sich "majority vote" immer weniger. Der Gewinn (d.h die Verringerung des Fehlers) geht gegen null, wenn ε gegen null geht.
- Im Bereich $\varepsilon_0 < \varepsilon < 0.5$ lohnt sich "majority vote" immer mehr, wenn ε sinkt.

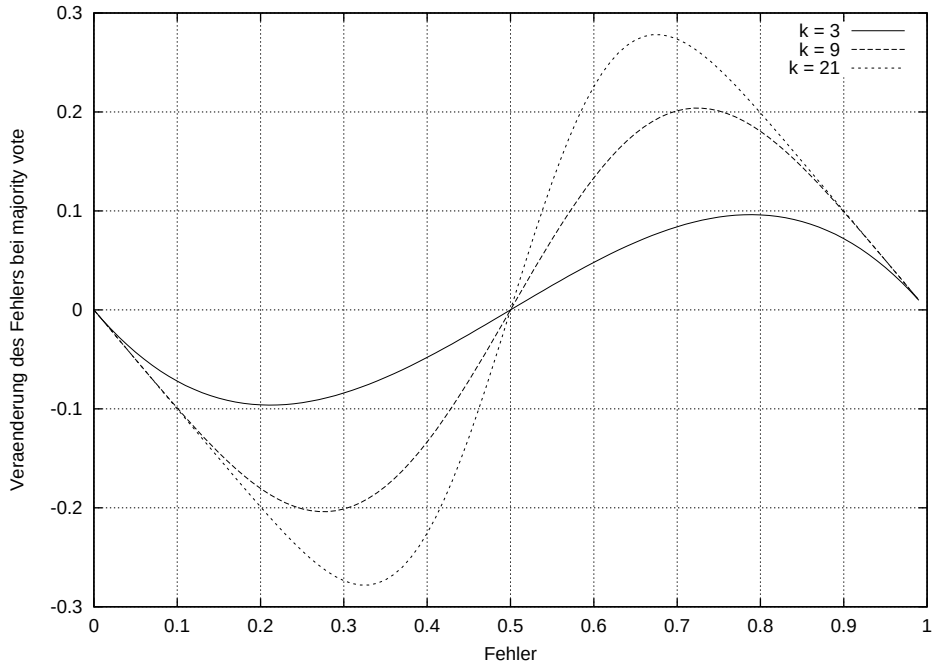


Abbildung 13: Verbesserung durch Mehrheitswahl

- Je größer die Anzahl der Agenten, desto mehr lohnt sich "majority vote". Allerdings reduziert sich der Fehler nicht proportional zu der Anzahl der Agenten, sondern langsamer.

Aus diesen Beobachtungen lassen sich einige Richtlinien für die Strategie des Agenten herleiten. Andere Agenten um Rat zu fragen lohnt sich nur dann, wenn der Nutzen dieser Aktion die Kosten für die Kommunikation überwiegt. Nun sind diese Kosten konstant, wohingegen der Nutzen E_{maj} von verschiedenen Faktoren abhängt. Diese sind zum einen der Gewinn an Präzision der Entscheidung, wie er gerade dargestellt wurde. Zum anderen ist es die Höhe des Schadens, welcher entsteht, wenn sich der Agent irrt. Wenn beispielsweise Graben fast keine Energie kostet, dann wird es sich für den Agenten kaum lohnen einen kleinen Irrtum auszuschließen, da ihn ein Irrtum fast nichts kostet. Die Höhe des Schadens kann für die beiden Entscheidungen graben und nicht graben durchaus verschieden sein und hängt von E_g, E_z aber auch wiederum von E_l ab, also dem Nutzen, den der Agent durch ein weiteres Trainingsbeispiel hat. Vernachlässigt man E_l und geht man davon aus, dass mit zunehmender Anzahl von Trainingsbeispielen die Irrtumswahrscheinlichkeit sinkt, dann ergibt sich zusammen mit der Strategie für nicht-kooperatives Lernen folgende einfache Strategie:

1. Verfügt der Agent über weniger als m_0 Beispiele, so sollte er immer graben. Da er immer gräbt, braucht er nicht zu kommunizieren, denn eine Verbesserung seiner Entscheidung ist nicht relevant, er macht keinen Gebrauch

von ihr.

2. Verfügt der Agent über mehr als m_0 Beispiele, aber weniger als m_1, dann sollte er seine Entscheidungen mit den anderen Agenten abgleichen.

3. Verfügt der Agent über mehr als m_1 Beispiele, dann ist der Nutzen durch das Abgleichen von Entscheidungen geringer als die Kosten und der Agent fragt fortan nicht mehr weiter bei den anderen Agenten um Rat.

Diese Strategie stellt eine gewisse Vereinfachung dar, sollte aber im allgemeinen zu guten Ergebnissen führen.

6.4.4 Qualitative Analyse

Kosten

Die Kosten für jedes Paar von Anfrage und Antwort liegen mindestens bei $n+1$ Bits, wobei n die Anzahl der Attribute ist. Da der Agent immer $(k-1)$ Anfragen auf einmal verschickt, kann man diese Kosten zusammenfassen zu $(k-1)*(n+1)$. Wie oft ein Agent Nachrichten an alle anderen verschickt, hängt von seiner Strategie ab.

Die zusätzliche Rechenzeit für dieses Verfahren ist sehr gering, sie liegt in $O(k)$, wobei k die Anzahl der Agenten ist.

Nutzen

Die Annahme, dass die Entscheidungen aller Agenten als unabhängige Ereignisse zu betrachten sind, wurde im letzten Absatz deshalb gemacht, um einen Anhaltspunkt für mögliche Strategien zu finden. Um zu einer realistischen Einschätzung des Nutzens zu kommen, welcher sich durch den Abgleich von Entscheidungen ergibt, ist diese Annahme zumindest problematisch. Der Grund dafür liegt darin, dass die Agenten aus Beispielen lernen, die aus demselben Zusammenhang generiert sind. Daher ist zu erwarten, dass ihre Entscheidungen in vielen Fällen ähnlich ausfallen werden. Das verringert den Nutzen durch majority vote. Wie hoch der Nutzen tatsächlich ist soll in Abschnitt 7 experimentell untersucht werden.

Voraussetzungen für den Einsatz

Worin liegen die Voraussetzungen für den Einsatz dieser Methode? Wie schon beim Austausch von Beispielen, setzt diese Methode nicht voraus, dass die Agenten denselben Lernalgorithmus verwenden, sondern nur, dass sie die Attribute, welche eine Stelle repräsentieren identisch darstellen. Tatsächlich setzt sie nicht einmal voraus, dass die Beispiele gespeichert werden.

Vorteile des Verfahrens

- Relativ einfach zu implementieren.
- Sehr effizient, geringe Erhöhung des Rechenaufwands.
- Weitgehende Unabhängigkeit von der Struktur des Agenten, vor allem von den eingesetzten Lernalgorithmen.

Nachteile des Verfahrens

- Es wird nicht die vollständige Information, welche in der Erfahrung enthalten ist, übermittelt.
- Bei homogenen Agenten, die alle gleich schnell lernen, ist zu erwarten, dass ihre Entscheidungen oft ähnlich ausfallen.

7 Experimentelle Untersuchung

In der bisherigen Darstellung wurden vier verschiedene Arten von Agententeams vorgestellt, die in der oben dargestellten Spielsimulation antreten können:

1. Nicht-kooperativ lernende Agenten, also ein Agententeam, das vollständig auf Kooperation verzichtet
2. Agenten, die ihre Erfahrung austauschen
3. Agenten, die ihre Hypothesen austauschen
4. Agenten, die ihre Entscheidungen durch Mehrheitswahl abgleichen

Die experimentelle Untersuchung soll nun zwei Zwecken dienen:

1. In den vorhergehenden Abschnitten wurden die Strategien für die jeweiligen Arten von Agententeams grob umrissen. In diesem Abschnitt sollen sie parametrisiert und getestet werden. Dabei geht es vor allem darum, die Abhängigkeit der Strategie von den Simulationsparametern darzustellen (also beispielsweise, wie der Agent seine Strategie verändern muss, wenn die Kosten für Kommunikation steigen). Das Ergebnis soll eine Reihe von Regeln sein, welche der Agent nutzen kann, um sich möglichst gut an eine gegebene Situation anpassen zu können. Das Kriterium für eine optimale Strategie ist die Anzahl an Punkten, welche ein Agententeam, das diese Strategie einsetzt, im Durchschnitt bei der Simulation erreicht.
2. Die verschiedenen Ansätze wurden bereits qualitativ in Bezug auf die jeweils zu erwartende Leistung untersucht. In diesem Abschnitt sollen sie quantitativ verglichen werden. Wiederum geht es darum, zu untersuchen, welcher Ansatz bei welcher Konfiguration am besten geeignet ist, d.h. im Durchschnitt das beste Ergebnis im Sinne des Simulationsspiels erreicht.

Da es sehr aufwendig wäre, den gesamten Parameterraum zu untersuchen, der durch die Konfiguration des Szenarios gegeben ist, sollen hier nur folgende Parameter variiert werden:

- Struktur der Insel, d.h. die Anzahl der Plätze und die Verbindung zwischen den Plätzen. Es werden zwei Inseln untersucht:
  1. *Insel 1*
     Eine Insel mit einer sehr großen Anzahl von Plätzen (50000), die sehr stark miteinander verbunden sind. Auf dieser Insel ist es sehr unwahrscheinlich, auf einen schon ausgebeuteten Platz zu treffen.

2. *Insel 2*
   Eine Insel mit wenigen Plätzen (150), die schwächer verbunden sind. Auf einer solchen Insel trifft der Agent öfter auf einen Platz, der schon ausgebeutet ist.

- Die Beziehung zwischen den Attributen an einem Platz und dem Vorhandensein einer Energiezelle. Es werden vier verschiedene Funktionen verwendet, die alle die Eigenschaft gemeinsam haben, dass in 25% aller Fälle eine Energiezelle vorliegt, in 75% der Fälle keine:

  1. *simple4*
     ein sehr einfacher deterministischer Zusammenhang, der sich durch wenige Regeln darstellen lässt, und vier Attribute enthält (zwei relevante, zwei irrelevante):
     $f \equiv a_1 \wedge a_2$
  2. *simple8*
     ein sehr einfacher deterministischer Zusammenhang, aber eine größere Anzahl von Attributen, d.h. es treten viele irrelevante Attribute auf (*simple8* ensteht aus *simple4*, indem vier irrelavante Attribute hinzugefügt werden).
  3. *complex7*
     ein komplexer Zusammenhang, bei dem alle auftretenden Attribute auch relevant sind:
     $f \equiv (\neg a_1 \wedge \neg a_2 \wedge a_4 \wedge a_6) \vee (a_1 \wedge \neg a_3 \wedge \neg a_5 \wedge \neg a_7) \vee (a_1 \wedge \neg a_3 \wedge a_5)$
  4. *simple8noise20*
     ein einfacher Zusammenhang, der aber nicht deterministisch ist, also bei dem der Wert des Klassenattributs nicht eindeutig durch die Attribute festgelegt ist. Er entsteht aus *simple8* indem bei 20% der Beispiele der Wert der Klasse invertiert wird.

- Die Energie welche für das Graben und Kommunizieren nötig ist, sowie die Energie, welche sich in einer Energiezelle befindet.

- Die Anzahl der Agenten, wobei folgende Werte untersucht werden: $k = 3$, $k = 9$, $k = 21$.

Die Energie, welche für das Bewegen auf der Insel notwendig ist, wird konstant auf Null gesetzt. Die Startenergie der Agenten beträgt konstant 150. Die Anzahl der Simulationsschritte ist auf 50 festgesetzt.

Für jeden der Versuche werden 500 Simulationsdurchläufe ausgeführt und dann jeweils der Durchschnitt der erreichten Energie gebildet.

Für die Agententeams, die kommunizieren, ist noch ein letzter Punkt wichtig, nämlich die Größe der Nachrichten, die ausgetauscht werden. Diese wird einheitlich so festgelegt, dass ein Beispiel genau die Größe 1.0 hat. Werden n Beispiele versendet, dann ergibt sich eine Größe von $n * 1.0$. Eine Regel wird, da sie meist

weit weniger Attributwerte enthält, wie ein halbes Beispiel behandelt. Damit haben n Regeln also die Größe $n * 0.5$. Beim Abgleich von Entscheidungen wird eine Anfrage und die entsprechende Antwort wie ein Beispiel behandelt.

7.1 Strategie

Die beiden wichtigsten Fragen, welche die Strategie eines Agenten beantworten muss, sind:

1. Wann soll der Agent graben?
2. Wann und mit wem soll der Agent kommunizieren?

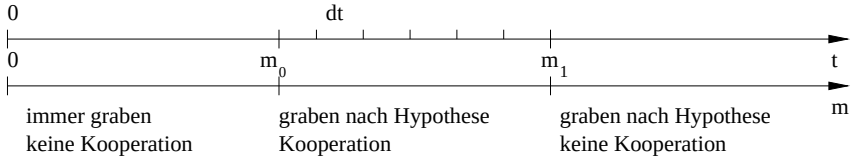
Die prinzipielle Anzahl von Strategien, denen ein Agent dabei folgen kann, ist sehr groß. Es soll daher hier nur eine Klasse von Strategien untersucht werden, welche sich aus den Überlegungen in Abschnitt 5 und 6 ergibt.

Für ein Team von nicht-kooperativ lernenden Agenten ergab sich, dass diese Agenten immer graben sollten, bis sie eine bestimmte Anzahl von Trainingsbeispielen erhalten haben. Danach sollten sie nur noch graben, wenn ihre Hypothese dies nahe legt. Daraus ergibt sich folgendes Diagramm:

Das Problem besteht also darin, den Parameter m_0 in Abhängigkeit von den Simulationsparametern möglichst optimal zu bestimmen.

Ein Team von Agenten, welches Kommunikation einsetzt, benötigt zusätzlich zu dieser Strategie eine Strategie, welche festlegt, wann kooperiert werden soll. Dabei ist zu beachten, dass Kommunikation in jedem Fall unnötig ist, solange der Agent immer gräbt. Weiter zeigte sich, dass ab einem gewissen Zeitpunkt Kommunikation keinen Gewinn mehr einbringt, der die Kosten überwiegt. D.h. es ergeben sich drei Phasen: eine Phase in welcher der Agent immer gräbt und nicht kommuniziert, eine Phase in der der Agent kommuniziert und sich nach seiner Hypothese richtet und eine dritte Phase in welcher der Agent nicht mehr kommuniziert, sich aber weiterhin nach seiner Hypothese richtet. Innerhalb der zweiten Phase wird davon ausgegangen, dass im Falle des Austauschs von Trainingsbeispielen in jedem Schritt Beispiele angefordert werden und im Fall des Abgleichs von Entscheidungen immer dann die anderen Agenten gefragt werden, wenn eine Entscheidung getroffen werden muss. Für den Austausch von Hypothesen wird eine Schrittweite dt festgelegt, die angibt, in welchem Abstand der Agent Hypothesen von den anderen Agenten anfordert. Diese Schrittweite richtet sich nach der Anzahl der Simulationsschritte und nicht nach der Anzahl von

Trainingsbeispielen. Die Abgrenzung der drei Phasen erfolgt jedoch weiterhin mit der Anzahl der Trainingsbeispiele:



Das Problem hier ist also die Werte m_0, m_1 (und im Falle des Hypothesenaustauschs dt) in Abhängigkeit von den Simulationsparametern optimal zu bestimmen.

Dabei kann es sich herausstellen, dass sich bei einer gegebenen Parameterbelegung, Kooperation gar nicht mehr lohnt, da die Kosten den Nutzen übersteigen und das für alle möglichen m_0 bzw. m_0 und m_1 (und dt). Wenn also für keine Kombination aus Strategieparametern ein besseres Ergebnis erzielt werden kann als ohne Kooperation, dann ist das Feld für m_1 frei gelassen. Der Wert für m_0 erhält in diesen Fällen dieselbe Bedeutung wie im Fall des nicht-kooperativen Lernens. Ist nur der Wert für dt ausgelassen, so heißt dies, dass es zu einem einmaligen Austausch von Hypothesen zu dem Zeitpunkt kommt, an dem der Agent m_0 Beispiele hat.

Da davon ausgegangen wird, dass alle Agenten immer mit allen anderen Agenten kooperieren, kann es zu Situationen kommen, in denen sich die Kooperation mit drei Agenten im System lohnt, mit 21 aber nicht. In diesen Fällen ist klar, dass ein Agent in einem System mit 21 Agenten auch nur mit drei Agenten kooperieren könnte, was hier aber nicht mehr extra erwähnt wird.

Um verschiedene Tendenzen zwischen den Parametern des Szenarios und optimalen Werten der Strategieparameter aufzeigen zu können, wird davon ausgegangen, dass die verschiedenen Einflussgrößen unabhängig voneinander in ihrer Wirkung untersucht werden können. Daher sind die folgenden Abschnitte so aufgebaut, dass nacheinander, in verschiedenen Experimenten, der Einfluss der verschiedenen Parameter untersucht wird. Die Werte der Simulationsparameter, die dabei konstant gelassen werden, sind jeweils so gewählt, dass das Ergebnis möglichst aussagekräftig ist.

7.1.1 Strategie für nicht-kooperatives Lernen

Zunächst soll untersucht werden, welchen Einfluss die Parameter E_g und E_z auf ein optimales m_0 haben. Dazu wird die große Insel und der einfache Zusammenhang (*simple8*) zwischen Attributen und dem Vorhandensein einer Energiezelle an einem Platz verwendet. Die Agententeams werden aus drei Agenten gebildet. Es ergibt sich das in Abbildung 14 dargestellte Diagramm. Daran kann man verschiedenes sehen:

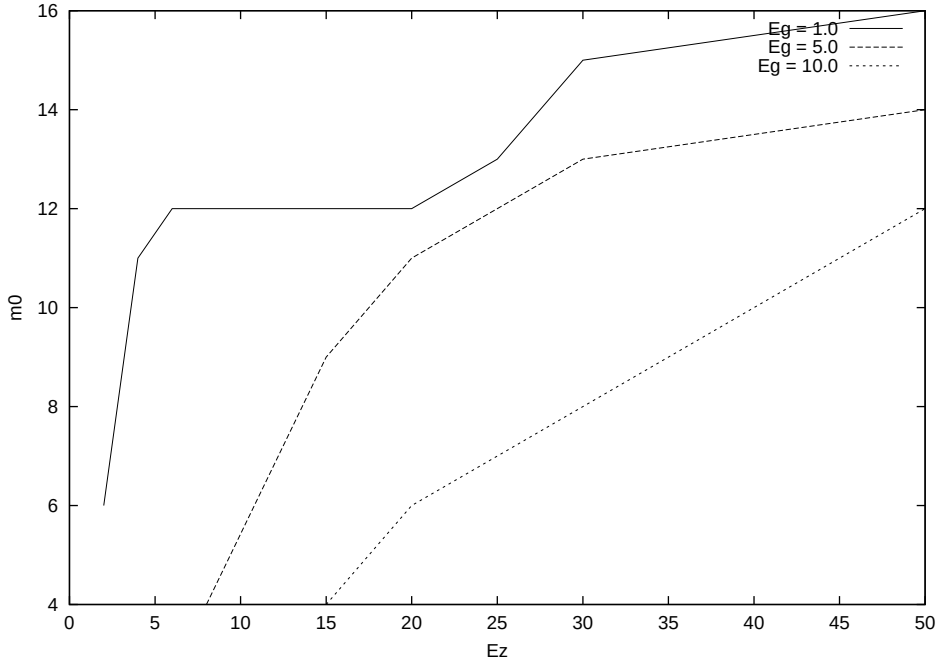


Abbildung 14: Experiment 1.1

1. *Mit steigendem E_z (bei konstantem E_g) steigt auch das optimale m_0.*
   Der Grund dafür liegt darin, dass das Risiko, dort nicht zu graben, wo eine Energiezelle liegt, immer schwerwiegender wird gegenüber dem Risiko, an einem Platz zu graben, an dem keine Energiezelle liegt, denn der Verlust, der durch fälschliches Nicht-Graben entsteht, wird immer größer.

2. *Mit steigendem E_g (bei konstantem E_z) sinkt das optimale m_0.*
   Hier ist der Grund gerade umgekehrt. Das Risiko, dort zu graben, wo keine Energiezelle liegt, wird mit steigendem E_g größer, da mit steigendem E_g auch die Kosten für einen Irrtum in diese Richtung steigen. Geht E_g hingegen auf 0 zurück, dann lohnt es sich immer zu graben, da kein Risiko eines Verlustes besteht.

3. *Liegen E_z und E_g zu nahe beieinander, lohnt es sich gar nicht mehr zu graben.*
   Der Grund liegt darin, dass der Agent zunächst lernen muss, um gezielt graben zu können, was ihn schon Energie kostet. Er hat dann bei jeder weiteren Entscheidung immer noch ein gewisses Risiko eines Irrtums. Wenn die Gewinnspanne, also $E_z - E_g$, zu klein ist, dann kann sie die Lernkosten und das Risiko nicht ausgleichen. Im Extremfall $E_z = E_g$ lohnt sich Graben nie, da überhaupt kein Gewinn erzielt werden kann.

4. *Eine Zunahme von E_z (bei konstantem E_g) führt nicht zu einer linearen Zunahme von m_0, vielmehr nimmt m_0 ab einem gewissen Punkt nur noch wenig zu.*

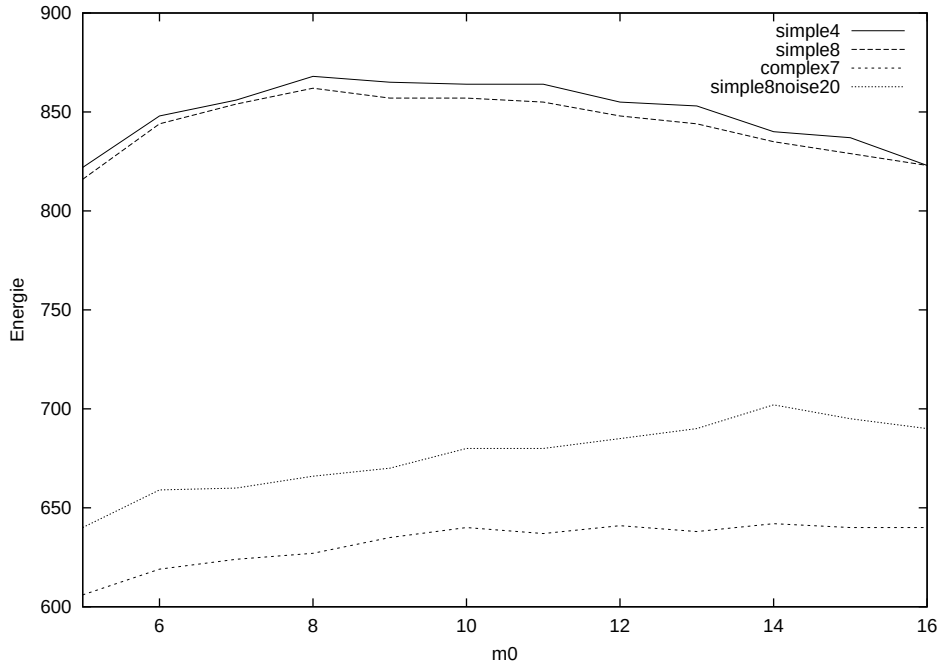


Abbildung 15: Experiment 1.2

Dies liegt daran, dass nachdem der Agent über eine größere Anzahl an Beispielen verfügt, die Wahrscheinlichkeit, dass er eine schlechte Hypothese bildet, sehr gering ist. Hat er aber eine sehr gute Hypothese, dann spart er auf diese Weise Energie ein, die er für unnötiges Graben hätte aufwenden müssen, ohne ein allzu hohes Risiko einzugehen, an einem Platz nicht zu graben, an dem sich tatsächlich eine Energiezelle befindet.

In einem weiteren Experiment soll der Einfluss des Zusammenhangs zwischen Attributen an einem Platz und dem Vorhandensein einer Energiezelle auf ein optimales m_0 untersucht werden. E_z erhält den Wert 15.0, E_g erhält den Wert 5.0. Die Teams werden aus drei Agenten gebildet. Es wird die große Insel verwendet. In Abbildung 15 werden die jeweils von den Agententeams erzielten Energiebeträge dargestellt.

Man kann daraus folgendes interpretieren:

Je einfacher der Zusammenhang, desto kleiner ist ein optimales m_0.
Das ist insofern nicht überraschend, als dass ein komplexer Zusammenhang in der Regel deutlich mehr Beispiele erfordert um gelernt zu werden als ein einfacher. Daher steigt im Durchschnitt die Lernkurve mit steigender Anzahl von Beispielen wesentlich langsamer an. Das führt dazu, dass sich Lernen länger lohnt. Wenn dagegen der Zusammenhang einfach ist, dann ändert sich schon ab einer kleinen Anzahl von Trainingsbeispielen nicht mehr viel an der Leistung der Hypothese, so dass die Kosten durch den Nutzen nicht ausgeglichen werden.

In einem letzten Experiment soll untersucht werden, wie sich die Anzahl der Agenten auswirkt. Da auf der großen Insel der Einfluss der Agenten aufeinander minimal ist, kann dieser Fall vernachlässigt werden. Es wird also nur die kleine Insel untersucht, wobei $E_z = 15$, $E_g = 5$ gesetzt wird. Als Konzept dient die Funktion *simple8*. Es ergeben sich folgende optimale Werte für m_0:

k	3	9	21
m_0	6	3	1

Diese Werte liegen unter den Werten für eine große Insel. Der Grund dafür liegt darin, dass die Agenten aus den Spuren, welche die anderen Agenten beim Graben hinterlassen, schließen können, ob sich an diesem Platz eine Energiezelle befunden hat. Da die Insel klein ist, treffen sie mit relativ hoher Wahrscheinlichkeit auf einen bereits ausgebeuteten Platz und somit auf ein Trainingsbeispiel.

7.1.2 Strategie für den Austausch von Erfahrung

Bei allen Experimenten in diesem Abschnitt, müssen die Parameter m_0 und m_1 gleichzeitig optimiert werden. Dabei sollen für m_1 nur folgende Werte berücksichtigt werden: 0, 10, 15, 30 und 50. Es zeigt sich, dass diese Werte ausreichend sind, um eine grobe Tendenz auszumachen.

Für $E_g = 5.0$, $E_z = 20.0$, Anzahl der Agenten $k = 3$, die große Insel und den Zusammenhang *simple8* ergibt sich beispielsweise das in Abbildung 16 gezeigte Diagramm (hier wurden für m_1 zusätzliche Werte herangezogen). Man sieht, dass das Optimum, also die Kombination von Werten, bei welcher die Agenten den höchsten Energiebetrag erlangen, bei $m_0 = 4$ und $m_1 = 12$ liegt.

Um im einzelnen den Einfluss der verschiedenen Parameter auf optimale Werte von m_0 und m_1 zu ermitteln, müssen mehrere Experimente dieser Art durchgeführt werden.

Zunächst soll wieder die Abhängigkeit optimaler Werte für m_0 und m_1 von den Parametern E_g und E_z bestimmt werden. Die Energie für die Kommunikation wird dabei konstant auf $E_k = 0.1$ gesetzt. Es werden Teams aus drei Agenten gebildet, die große Insel und der Zusammenhang *simple8* verwendet.

E_g	E_z	m_0	m_1
1.0	20.0	10	50
5.0	20.0	4	50
10.0	20.0	3	50
5.0	40.0	7	50
5.0	80.0	10	50

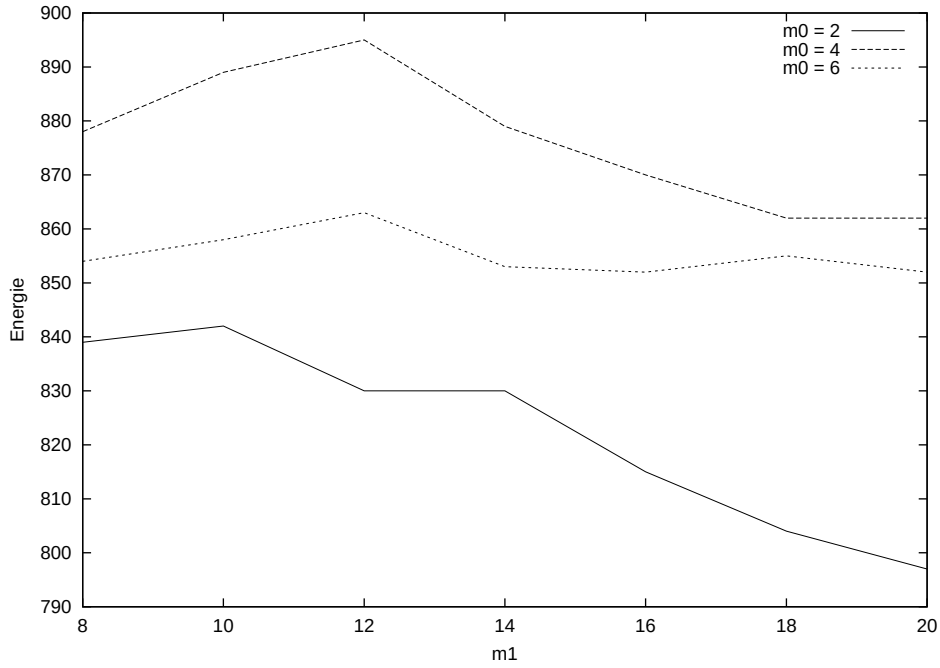


Abbildung 16: Experiment 2.1

Man sieht folgendes:

1. *Mit steigendem E_z (bei konstantem E_g) steigt auch das optimale m_0.*
   Dies ist ein Effekt, der analog zum nicht-kooperativen Lernen auftrifft, allerdings abgeschwächt, da das Risiko, dass alle Agenten Beispiele erhalten, die sie zu schlechten Hypothesen führen, geringer ist, als dass ein Agent solche Beispiele erhält. Da die Agenten ihre Beispiele aber austauschen, verringert sich das Risiko einer schlechten (nicht mehr veränderbaren Hypothese) für jeden einzelnen Agenten.

2. *Mit steigendem E_g (bei konstantem E_z) sinkt das optimale m_0.*
   Auch dieser Effekt tritt beim nicht-kooperativen Lernen auf und ist in analoger Weise abgeschwächt.

Der Parameter m_1 ist deshalb immer am Maximalwert, da die Kosten für die Kommunikation gering sind. In einem weiteren Versuch sollen diese Kosten erhöht werden, wobei wiederum $E_g = 5.0$ und $E_z = 20.0$ ist. Alle anderen Parameter behalten ihre Werte:

E_k	m_0	m_1
0.1	4	50
0.3	4	15
0.5	4	15
1.0	4	15
2.0	4	15
5.0	4	10
10.0	7	-

Das lässt sich folgendermaßen interpretieren:

1. *Mit steigenden Kommunikationskosten sinkt m_1.*
   Indem die Kosten pro Trainingsbeispiel erhöht werden, lohnt es sich für die Agenten immer weniger, Trainingsbeispiele anzufordern, vor allem wenn sie bereits über ausreichend viele Beispiele verfügen, um im Durchschnitt eine gute Hypothese zu bilden (aus experimentellen Lernkurven sieht man, dass der Fehler einer Hypothese mit steigender Anzahl an Trainingsbeispielen immer langsamer sinkt, die Kosten für ein einzelnes Beispiel hingegen sind konstant) . Übersteigen die Kosten für Kommunikation einen bestimmten Wert, dann lohnt sich Kommunikation gar nicht mehr (was in der Tabelle bei $E_k = 10.0$ sichtbar wird).

2. *Der Parameter m_0 ist weitgehend unabhängig von der Energie für die Kommunikation.*
   Solange die Agenten genügend Beispiele austauschen, um das Risiko für eine schlechte (und nicht mehr veränderbare) Hypothese gering zu halten, behält m_0 einen konstanten Wert. Erst wenn die Agenten gar nicht mehr kommunizieren, steigt m_0 auf einen höheren Wert.

In einem dritten Versuch soll die Abhängigkeit optimaler Werte für m_0 und m_1 von der zu lernenden Funktion betrachtet werden. Die Energieparameter werden auf $E_g = 5.0$, $E_z = 20.0$ und $E_k = 0.1$ gesetzt. Wiederum wird ein Team aus drei Agenten gebildet und die große Insel verwendet:

Funktion	m_0	m_1
simple4	4	30
simple8	4	50
complex7	10	50
simple8noise20	12	50

Man kann daran folgendes sehen:

1. *Je komplexer der Zusammenhang, desto größer m_0.*
   Dieser Effekt tritt wiederum analog zum nicht-kooperativen Lernen auf.

2. *Ist der Zusammenhang sehr einfach, dann sinkt m_1 auf einen niedrigeren Wert, obwohl die Kommunikationskosten gering sind.*
   Ist der Zusammenhang so einfach, dass bereits mit 30 Beispielen praktisch immer die optimale Genauigkeit erreicht wird, kann an dieser Stelle weitere Kommunikation eingespart werden. Aber schon die Funktion *simple8* sichert dies nicht mehr, so dass auch hier der maximale Wert für m_1 der beste ist.

Als nächstes soll der Einfluss der Anzahl von Agenten auf optimale Werte von m_0 und $m_1$untersucht werden. Wiederum sind $E_g = 5.0$, $E_z = 20.0$. Diesmal wird $E_k = 2.0$ gesetzt. Alle anderen Parameter behalten ihre Werte. Es ergibt sich folgendes:

k	m_0	m_1
3	4	15
9	1	15
21	1	15

Man sieht, dass die Werte für m_0 sinken, da die Agenten die Beispiele, welche sie sonst selbst hätten aufsuchen müssen, von den anderen Agenten erhalten. Da nun die Kommunikationsenergie höher ist, fordern die Agenten insgesamt weniger Beispiele an.

Nimmt man statt der großen Insel die kleine, dann ergibt sich folgendes:

k	m_0	m_1
3	4	10
9	1	10
21	1	10

Das liegt etwa in einer vergleichbaren Größenordnung wie bei der großen Insel. Die Kommunikation wird allerdings etwas früher abgebrochen, da die Agenten zusätzlich Beispiele von schon ausgebeuteten Plätzen erhalten.

7.1.3 Strategien für den Abgleich von Entscheidungen

Genau wie für das Austauschen von Erfahrung müssen auch beim Abgleich von Entscheidungen zwei Werte gleichzeitig optimiert werden, nämlich m_0 und m_1. Um diese Werte möglichst optimal bestimmen zu können, müssen wiederum Simulationsläufe mit verschiedenen Wertkombinationen für m_0 und m_1 durchführt werden.

Zunächst soll wieder der Einfluss von E_g und E_z untersucht werden. Dabei wird

die große Insel und der Zusammenhang *simple8* verwendet. Die Agententeams werden aus drei Agenten gebildet und die Kommunikationsenergie ist auf $E_k = 0.3$ festgelegt. Es ergeben sich folgende Werte:

E_g	E_z	m_0	m_1
5.0	20.0	8	10
5.0	30.0	10	12
5.0	100.0	12	14
1.0	20.0	13	14
10.0	20.0	6	7

Man sieht folgendes:

1. *Die Werte für m_1 sind immer um ein oder zwei Trainingsbeispiele höher als die Werte für m_0.*
   Kommunikation lohnt sich also immer in einem Band, dass bei m_0 beginnt und etwa zwei Trainingsbeipiele breit ist.

2. *Die Werte für m_0 verhalten sich wie in den vorangegangenen Experimenten.*
   Wiederum ergibt eine Erhöhung von E_z relativ zu E_g eine Erhöhung von m_0, eine Vergrößerung von E_g führt hingegen zu einer Verkleinerung von m_0. Die Erklärung dafür liegt analog wie im Fall des nicht-kooperativen Lernens in der Veränderung des Risikos, entweder zu wenig zu graben oder unnötig zu graben.

Als nächstes wird der Einfluss von E_k auf m_0 und m_1 untersucht. Alle sonstigen Werte bleiben gleich, es wird $E_g = 5.0$ und $E_z = 20.0$ gesetzt.

E_k	m_0	m_1
0.1	9	12
0.3	8	10
1.0	9	10
2.0	9	-

Mit steigender Kommunikationsenergie wird das Band, in welchem kommuniziert wird, immer kleiner.

Bei $E_k = 2.0$ schließlich lohnt sich Kommunikation gar nicht mehr. Die Erklärung dafür liegt darin, dass der Nutzen von Kooperation die Kosten aufwiegen muss, um sich zu lohnen. Da nun aber der Nutzen in diesem Fall immer derselbe ist, die Kosten aber steigen, wird Kommunikation immer mehr eingeschränkt, bis sie sich gar nicht mehr lohnt.

In einem weiteren Experiment soll der Einfluss des zu lernenden Konzepts auf

optimale Werte für m_0 und m_1 untersucht werden. Dabei wird $E_g = 5.0$, $E_z = 20.0$ und $E_k = 0.3$ gesetzt, alle anderen Parameter bleiben unverändert.

Funktion	m_0	m_1
simple4	7	9
simple8	8	10
complex7	12	-
simple8noise20	16	-

1. *Mit steigender Komplexität steigt auch der Wert für ein optimales m_0.*
   Die Werte für m_1 liegen wieder immer etwa zwei Trainingsbeispiele über den Werten von m_0. Dies ist wiederum analog zu den Ergebnissen der vorherigen Experimente.

2. *Ist das Konzept zu komplex, dann lohnt sich der Abgleich von Entscheidungen gar nicht mehr.*
   Der Nutzen durch den Abgleich von Entscheidungen kann die Kosten nicht ausgleichen. Daher lohnt sich der Abgleich von Entscheidungen jeweils bei den beiden komplexen Konzepten nicht mehr.

Die beiden letzten Experimente sollen den Einfluss der Anzahl der Agenten und der Größe der Insel auf m_0 und m_1 zeigen. Dabei wird zunächst die große Insel verwendet und die Anzahl der Agenten gesteigert. Alle anderen Parameter bleiben gleich:

k	m_0	m_1
3	8	10
9	6	7
21	5	-

1. *Mit steigender Anzahl an Agenten sinkt m_0.*
   Dies stimmt mit der theoretischen Vorhersage überein, denn je mehr Agenten an der Mehrheitswahl beteiligt sind, desto größer ist deren Nutzen. Allerdings sinkt m_0 immer langsamer, da viele schlechte Entscheidungen (die auf wenigen Beispielen beruhen) nicht unbedingt zu einer guten Entscheidung führen.

2. *Mit steigender Anzahl an Agenten sinkt $m_1 - m_0$.*
   Das Band, in dem Kommunikation stattfindet, verkleinert sich also mit steigender Anzahl an Agenten. Das liegt daran, dass die Kosten für die Kommunikation mit steigender Anzahl an Agenten linear steigen, der Nutzen hingegen nicht linear steigt (siehe: 6.4.3).

Nimmt man hingegen die kleine Insel, dann zeigt sich folgendes:

k	m_0	m_1
3	7	9
9	4	5
21	3	-

Die Werte für m_0 sinken hier weiter, da die Agenten die Möglichkeit haben, an schon ausgebeuteten Plätzen abzulesen, ob sich dort eine Energiezelle befunden hat. Damit erhalten sie weitere Beispiele, ohne selbst graben zu müssen. Auf einer kleinen Insel ist nun die Wahrscheinlichkeit auf einen schon ausgebeuteten Platz zu stoßen, wesentlich größer als auf einer großen Insel.

7.1.4 Strategien für den Austausch von Hypothesen

Wie oben bereits erwähnt, müssen im Fall des Austauschs von Hypothesen drei Parameter gleichzeitig optimiert werden, nämlich m_0, m_1 und dt. Möglichst optimale Werte für diese Parameter werden wieder in mehreren Experimenten bestimmt.

Auch hier soll zunächst die Abhängigkeit optimaler Werte für m_0, m_1 und dt von den Parametern E_g und E_z ermittelt werden. Dabei wird $E_k = 1.0$, die Anzahl der Agenten auf $k = 3$ und das Konzept auf *simple8* gesetzt. Es wird die große Insel verwendet.

E_g	E_z	m_0	m_1	dt
5.0	20.0	7	10	7
5.0	30.0	8	11	9
5.0	100.0	9	13	7
1.0	20.0	9	12	5
10.0	20.0	4	6	9

Es ergibt sich:

1. *Mit steigendem E_z (und konstantem E_g) steigt auch m_0. Mit steigendem E_g (und konstantem E_z) sinkt m_0.*
   Das ist analog zu den vorhergehenden Experimenten.

2. *Auch hier findet Kommunikation in einem Band statt, das bei m_0 beginnt und etwa 2 bis 4 Trainingsbeispiele breit ist.*
   Dieses Ergebnis tritt wiederum analog zum Abgleich von Entscheidungen auf.

3. *Zwischen dt und den Werten für E_g und E_z besteht kein signifikanter Zusammenhang.*
   D.h., es scheint dass diese Parameter auf dt keinen besonderen Einfluss haben.

Als nächstes soll der Einfluss von E_k untersucht werden. Alle Parameter bleiben gleich, und es wird $E_g = 5.0$ und $E_z = 20.0$ gesetzt.

E_k	m_0	m_1	dt
0.2	7	11	5
1.0	7	10	7
2.0	7	9	9
5.0	7	8	10
13.0	7	-	-

1. *Mit steigendem E_k sinkt m_1 und steigt dt.*
   Der Grund dafür liegt darin, dass der Nutzen durch Kommunikation die Kosten aufwiegen muss. Da nun aber auch hier der Nutzen konstant ist, die Kosten aber steigen, wird Kommunikation immer stärker eingeschränkt, und das einmal was die Länge der Phase betrifft, innerhalb derer kommuniziert wird, zum anderen was die Häufigkeit der Kommunikation betrifft.

2. *Der Parameter m_0 ist weitgehend unabhängig von E_k.*
   Dies zeigt, dass selbst häufige und länger andauernde Kommunikation nicht die Anfangsphase, in der alle Agenten Beispiele sammeln, ausgleichen kann.

In einem weiteren Experiment wird der Einfluss des Konzepts untersucht. Es wird $E_g = 5.0$, $E_z = 20.0$ und $E_k = 1.0$ gesetzt. Alle anderen Werte bleiben gleich.

Funktion	m_0	m_1	dt
simple4	6	9	7
simple8	7	10	7
complex7	12	14	-
simple8noise20	14	16	-

Es zeigt sich folgendes:

Bei einem einfachen Konzept sinkt m_0 etwas. Bei einem komplexen Konzept lohnt sich das Hypothesentauschen nur noch einmal.

Der Effekt, dass m_0 bei einfacherem Konzept sinkt, tritt analog zu den vorhergehenden Experimenten auf. Dass sich Kommunikation nur noch einmal lohnt, wenn ein komplexeres Konzept gelernt werden soll, liegt daran, dass die Kosten für die Kommunikation steigen je mehr Regeln versendet werden. Der Nutzen durch diese Regeln steigt aber nicht in demselben Maß.

In den beiden letzten Experimenten soll der Einfluss der Anzahl von Agenten untersucht werden. Dabei wird das Konzept *simple8* verwendet, alle anderen

Parameter bleiben gleich. Für die große Insel ergibt sich:

k	m_0	m_1	dt
3	7	10	7
9	7	8	-
21	6	-	-

1. *Durch eine größere Anzahl an Agenten sinkt m_0 nur geringfügig.*
   Eine schlechte (d.h. auf wenigen Beispielen beruhende) Hypothese auszutauschen bringt keinen großen Vorteil, da sie als Hintergrundwissen eher zu einer Verschlechterung der Hypothese führt als zu einer Verbesserung. Vor allem werden die trivialen Hypothesen *true* und *false* gar nicht ausgetauscht, so dass es in diesen Fällen zu gar keiner Kooperation kommt. Solche Hypothesen treten aber sehr häufig auf, wenn die Agenten nur sehr wenige Trainingsbeispiele zum Aufbau des Entscheidungsbaums verwenden.

2. *Die Kommunikation beschränkt sich bei einer größeren Anzahl von Agenten auf ein einmaliges Austauschen von Hypothesen.*
   Da die vorliegenden Agenten grundsätzlich von allen anderen Agenten Hypothesen fordern, steigen die Kosten mit steigender Anzahl von Agenten auch stark an. Da nun zu erwarten ist, dass die Hypothesen der einzelnen Agenten oft ähnlich sind, steigt der Nutzen nicht in demselben Maß. Daher wird nur ein einziges Mal kommuniziert.

Wird statt dessen die kleine Insel verwendet, ergibt sich folgendes:

k	m_0	m_1	dt
3	4	6	7
9	3	4	-
21	1	-	-

1. *Der Parameter m_0 sinkt mit steigender Anzahl an Agenten.*
   Hier tritt nun der Effekt auf, dass die Agenten durch schon ausgebeutete Plätze Trainingsbeispiele erhalten. Dies führt, analog zum Ergebnis der vorhergehenden Experimente, dazu, dass sich m_0 reduziert.

2. *Wird die Anzahl der Agenten zu groß, dann lohnt sich Kommunikation gar nicht mehr.*
   Zum einen liegt das daran, dass die Agenten immer von allen anderen Agenten Hypothesen fordern, der Nutzen aber nicht mit den Kosten steigt (da viele der Hypothesen ähnlich sind). Zum anderen erhalten die Agenten völlig ohne eigene Kosten Trainingsbeispiele an schon ausgebeuteten Plätzen. Diese allein erzeugen schon eine gute Hypothese, so dass sich durch die

fremden Hypothesen keine ausreichend große Verbesserung ergibt, welche die Kosten ausgleichen würde.

7.1.5 Zusammenfassung

Dieser Abschnitt soll die Ergebnisse zu den einzelnen Strategien kurz zusammenfassen, wobei sich die Gliederung nach den einzelnen Parametern des Szenarios richtet.

Die Parameter E_g und E_z beeinflussen hauptsächlich den Parameter m_0. Dabei zeigt sich bei allen Verfahren, dass m_0 mit steigendem E_z (und konstantem E_g) auch steigt, mit steigendem E_g (und konstantem E_z) aber sinkt. Ist die Differenz zwischen E_z und E_g zu gering, dann lohnt sich Graben gar nicht mehr, d.h. die beste Strategie ist es nie zu graben.

Während der Wert für m_0 im Fall des Austauschs von Hypothesen und Abgleichs von Entscheidungen meist ungefähr gleich dem Wert für nicht-kooperatives Lernen ist, sinkt dieser beim Austausch von Trainingsbeispielen weiter ab.

Für den Parameter m_1 ergibt sich folgendes: bei den Strategien für den Austausch von Hypothesen und den Abgleich von Entscheidungen ist ein möglichst optimaler Wert für m_1 immer in einem schmalen Band oberhalb des entsprechenden Wertes für m_0. Dahin gegen zeigt sich für die Werte von m_1 beim Austausch von Erfahrung eher eine Unabhängigkeit von E_g und E_z. Der Parameter dt, der nur beim Austauschen von Hypothesen relevant ist, zeigt ebenso keine signifikante Abhängigkeit von E_g und E_z.

Während also E_g und E_z hauptsächlich Einfluss auf m_0 haben, hat E_k starken Einfluss auf den Umfang der Kommunikation. Im Fall des Austauschs von Erfahrung sinkt m_1 mit steigendem E_k. Für die Strategien zum Abgleich von Entscheidungen und dem Austausch von Hypothesen ergibt sich, dass die Breite des Bandes, innerhalb dessen kommuniziert wird, mit steigendem E_k sinkt. Im Fall des Austauschs von Hypothesen steigt zusätzlich dt. In allen Fällen gilt: übersteigt E_k einen bestimmten Wert, dann lohnt sich Kooperation gar nicht mehr, d.h. es kann durch Kooperation nicht mehr Energie erreicht werden, als ohne Kooperation.

Der Parameter m_0 wird von E_k nur wenig beeinflusst, jedenfalls solange es noch zur Kooperation kommt. Steigt E_k soweit an, dass nicht mehr kooperiert wird, dann ergeben sich dieselben optimalen Werte für m_0 wie im Fall des nicht-kooperativen Lernens.

Was den Einfluss des zu lernenden Konzepts betrifft, zeigt sich eine einheitliche Tendenz für alle Verfahren, was den Parameter m_0 betrifft. Dieser steigt mit zunehmender Komplexität des Konzepts an.

Im Fall des Austauschs von Erfahrung sinkt m_1 bei einem einfacheren Kon-

zept. Bei dem Austausch von Hypothesen führt eine höhere Komplexität des zu lernenden Konzepts zu einer Einschränkung der Kommunikation, d.h. das Band, in welchem Kommunikation stattfindet, verkleinert sich und der Abstand zwischen zwei Anfragen steigt. Der Abgleich von Entscheidungen lohnt sich bei komplexeren Konzepten gar nicht mehr.

Bei einer Erhöhung der Anzahl der Agenten unter Verwendung einer großen Insel, ergibt sich folgendes: beim Austausch von Erfahrung sinkt m_0, m_1 bleibt konstant. Beim Austausch von Hypothesen bleibt m_0 konstant, m_1 sinkt und dt steigt, bis zu dem Punkt, an dem sich Kooperation gar nicht mehr lohnt. Beim Abgleichen von Entscheidungen sinkt m_0 etwas, das Band, innerhalb dessen sich Kommunikation lohnt, verkleinert sich ebenfalls bis zu dem Punkt, an dem sich Kommunikation gar nicht mehr lohnt.

Wird die Anzahl der Agenten unter Verwendung einer kleinen Insel erhöht, ergeben sich entsprechende Ergebnisse, allerdings liegen die Werte für m_0 unter denjenigen, welche sich für die große Insel ergeben. Die Werte für m_1 sinken ebenfalls und im Fall des Austauschs von Hypothesen steigt dt.

7.2 Gesamtvergleich der Verfahren

In diesem Abschnitt sollen die verschiedenen Verfahren verglichen werden. Hauptkriterium dabei ist die im Szenario festgelegte Bewertung der Agententeams danach, über wieviel Energie sie am Ende verfügen. Zusätzlich soll allerdings noch die Lernkurve, die sich bei Verwendung der Verfahren jeweils ergibt, untersucht werden, sowie die Rechenzeit, welche ein Agententeam benötigt. Wiederum erfolgt der Vergleich unter Variation der Parameter des Szenarios. Die Tabellen dieses Abschnitts enthalten jeweils die von den Agenten erreichten Energiebeträge, entsprechend der Spezifikation des Szenarios in Abschnitt 4. In den Klammern hinter den eigentlichen Werten ist der relative Gewinn durch Kooperation angegeben, also um wieviel Prozent das Team besser ist als das Team, welches keine Kooperation einsetzt.

7.2.1 Die erreichte Leistung

Zunächst soll auch hier der Einfluss von E_g und E_z untersucht werden. Dabei wird $E_k = 0.5$ gesetzt, das Konzept ist *simple8*, Agententeams werden aus drei Agenten gebildet und es wird die große Insel verwendet. Die Agenten folgen jeweils einer Strategie, wie sie im ersten Teil dieses Abschnitts erörtert wurde.

E_g	E_z	keine Kooperation	Erfahrungen tauschen	Hypothesen tauschen	Entscheidungen abgleichen
5.0	20.0	854	916 (7,3%)	901 (5,5%)	865 (1,2%)
5.0	30.0	1210	1269 (4,8%)	1260 (4,1%)	1230 (1,6%)
5.0	100.0	3750	3860 (2,9%)	3833 (2,2%)	3806 (1,5%)
1.0	20.0	1104	1108 (0,3%)	1106 (0,2%)	1113 (0,8%)
10.0	20.0	604	699 (15,7%)	639 (5,7%)	606 (0,3%)

Man sieht klar, dass sich das Austauschen von Erfahrung, unabhängig von E_g und E_z, am meisten lohnt. Danach folgt das Austauschen von Hypothesen, welches dem Abgleichen von Entscheidungen in fast jedem Fall überlegen ist. Interessant dabei ist die Tatsache, dass sich im Fall des Austauschs von Hypothesen und Erfahrung, der relative Nutzen stark verändert. Kostet Graben sehr wenig Energie, dann ist dieser Nutzen jeweils am geringsten, da die Kosten durch fälschliches Graben minimal sind im Vergleich zu den Kommunikationskosten. Steigt E_g an, dann erhöht sich auch der Nutzen von Kooperation in diesen Fällen. Steigt hingegen E_z, dann sinkt der relative Nutzen. Der Grund dafür ist umgekehrt, dass der Schaden durch irrtümliches Graben sinkt, denn er wird durch die größeren Gewinne im Fall eines Erfolgs ausgeglichen. Beim Abgleich von Entscheidungen verhält es sich deutlich anders. Allerdings kommt es hier weder zu größeren Schwankungen, noch ließe sich aus der obigen Tabelle eine klare Tendenz für eine Abhängigkeit des relativen Nutzens von E_g und E_z ausmachen.

Wie verändert sich dieses Ergebnis, wenn E_k verändert wird? Dazu werden $E_g = 5.0$ und $E_z = 20.0$ gesetzt, alle anderen Parameter bleiben gleich.

E_k	keine Kooperation	Erfahrungen tauschen	Hypothesen tauschen	Entscheidungen abgleichen
0.1	854	924 (8,2%)	911 (6,7%)	886 (3,7%)
0.3	854	917 (7,3%)	905 (5,6%)	879 (2,9%)
0.5	854	916 (7,3%)	901 (5,5%)	865 (1,2%)
1.0	854	905 (5,6%)	875 (2,4%)	857 (0,4%)
5.0	854	-	858 (0,5%)	-

Wie zu erwarten, sinken die Gewinne durch kooperatives Lernen mit steigenden Kommunikationskosten. Bei $E_k = 5.0$ lohnt sich das Austauschen von Erfahrung und das Abgleichen von Entscheidungen gar nicht mehr.

Als nächstes soll der Einfluss des zu lernenden Konzepts untersucht werden, dazu wird wiederum $E_k = 0.5$ gesetzt; alle anderen Parameter bleiben gleich.

Funktion	keine Kooperation	Erfahrungen tauschen	Hypothesen tauschen	Entscheidungen abgleichen
simple4	859	918 (6,7%)	907 (5,6%)	869 (1,2%)
simple8	854	916 (7,3%)	901 (5,5%)	865 (1,3%)
complex7	642	722 (12,5%)	655 (2,0%)	-
simple8noise20	686	705 (2,8%)	693 (1,0%)	-

Für das Austauschen von Erfahrung ergibt sich hier folgendes: bei deterministischen Konzepten erhöht sich der Nutzen durch Kooperation mit steigender Komplexität. Der Grund dafür liegt darin, dass bei den einfachen Konzepten oft auch ohne Kooperation schnell eine Hypothese mit echtem Fehler Null erreicht wird, dies kann dann durch Kooperation kaum verbessert werden. Mit dem indeterministischen Konzept hingegen ergibt sich nur ein geringer relativer Nutzen. Dies liegt an dem hier verwendeten Algorithmus zum Aufbau von Entscheidungsbäumen (ID3), der nicht adäquat mit Rauschen umgehen kann, so dass auch eine höhere Anzahl von Trainingsbeispielen keinen wesentlich größeren Nutzen bringt.

Der Nutzen durch den Austausch von Hypothesen sinkt mit steigender Komplexität. Auch der Grund hierfür ist leicht zu sehen: je komplexer das Konzept, desto größer sind im Mittel die Regelmengen, die ausgetauscht werden müssen, um so höher sind die Kommunikationskosten und um so geringer ist der relative Nutzen.

Der Abgleich von Entscheidungen lohnt sich für die komplexeren Konzepte gar nicht mehr.

Zum Schluss soll wieder der Einfluss der Anzahl der Agenten, und der Größe der Insel ermittelt werden.

Für die große Insel ergibt sich folgendes:

k	keine Kooperation	Erfahrungen tauschen	Hypothesen tauschen	Entscheidungen abgleichen
3	854	916 (7,3%)	901 (5,5%)	865 (1,2%)
9	2547	2769 (8,7%)	2559 (0,5%)	2550 (0,1%)
21	5960	6648 (11,5%)	-	-

Die Agententeams, welche Hypothesen tauschen und Entscheidungen abgleichen, schneiden bei einer höheren Anzahl von Agenten schlechter ab. Der Grund dafür wurde bereits oben erwähnt: sie kooperieren immer mit allen anderen Agenten. Dies ist meist nicht sinnvoll, da der Nutzen nicht proportional zu den Kosten steigt. Das Team, das Erfahrungen austauscht, bleibt von diesem Effekt unberührt: tatsächlich steigt hier der relative Nutzen sogar mit steigender Anzahl von Agenten.

Für die kleine Insel ergibt sich folgendes:

k	keine Kooperation	Erfahrungen tauschen	Hypothesen tauschen	Entscheidungen abgleichen
3	642	682 (6,2%)	654 (1,9%)	648 (0,9%)
9	1709	1762 (3,1%)	1722 (0,8%)	1712 (0,2%)
21	3542	3560 (0,5%)	-	-

Für den Austausch von Hypothesen und das Abgleichen von Entscheidungen ist das Ergebnis in etwa dasselbe. Dagegen ergibt sich beim Austausch von Trainingsbeispielen eine Änderung: der Nutzen sinkt stark mit steigender Anzahl von Agenten. Der Grund dafür liegt darin, dass die Agenten an ausgebeuteten Plätzen Trainingsbeispiele erhalten, ohne das dabei Kosten für sie entstehen. Wenn sie hingegen Trainingsbeispiele von anderen Agenten anfordern, so müssen sie die Kommunikationskosten tragen. Daher sinkt der relative Nutzen von Kooperation.

Um das schlechte Abschneiden des Ansatzes zum Abgleich von Entscheidungen zu klären, sollen weitere Experimente durchgeführt werden. Die Frage ist zunächst, wie oft sich die Entscheidung eines Agenten überhaupt ändert, indem er die anderen Agenten um Rat fragt. Bei $E_z = 20.0$, $E_g = 5.0$, $E_k = 0.5$, dem Zusammenhang *simple8* und Teams aus drei Agenten ergibt sich folgendes:

positiv nach negativ	6,64%
negativ nach positiv	1,2%
gesamt	7,84%

Man sieht zwei interessante Dinge. Zum einen ist der Gesamtanteil der Fälle, in denen es zu einer Änderung der Entscheidung kommt, sehr gering. Das scheint ein Grund dafür zu sein, dass das Abgleichen von Entscheidungen nur zu einem kleinen Gewinn führt. Zum anderen herrscht eine Asymmetrie bezüglich der Entscheidungsänderung. Der Agent wird in den meisten Fällen vom Graben abgehalten. Dies hat Einfluss auf den Lernerfolg, und damit wiederum auf die Gesamtleistung (da der Agent nur lernt, wenn er gräbt).

Teilweise scheint dies daran zu liegen, dass alle untersuchten Konzepte mit a priori Wahrscheinlichkeit 75% "keine Energiezelle" als Klasse haben. Das hat Einfluss auf die Mehrheitswahl. Daher soll zusätzlich ein Konzept eingeführt werden, bei dem beide Klassen mit derselben Wahrscheinlichkeit auftreten. Es wird im folgenden mit *extra8* bezeichnet. Für dieses Konzept ergibt sich folgendes:

positiv nach negativ	12,78%
negativ nach positiv	8,52%
gesamt	21,3%

Man sieht, dass es zum einen in dreimal mehr Fällen zu einer Änderung der ursprünglichen Entscheidung kommt, zum anderen die Asymmetrie stark abgeschwächt ist. Das hat auch einen Einfluss auf den Gesamterfolg. Die folgende Tabelle zeigt nicht-kooperatives Lernen und Abgleichen von Entscheidungen im Vergleich, wobei in diesem Fall der Zusammenhang *extra8* verwendet wird:

	nicht-kooperativ	Entscheidungen abgleichen
$E_k = 0.0$	1133	1311 (15,7%)
$E_k = 0.5$	1133	1178 (4,0%)

Der relative Gewinn durch das Abgleichen von Entscheidungen ist also mehr als dreifach größer als bei dem Konzept *simple8.*

Für das schlechte Ergebnis lassen sich also zumindest zwei Gründe anführen:

1. Die Homogenität der Agenten: alle Agenten lernen etwa gleich schnell, verwenden denselben Lernalgorithmus, nehmen dieselben Attribute wahr, etc. Dies führt dazu, dass ihre Entscheidungen oft gleich sind und das Abgleichen von Entscheidungen dann keinen Effekt hat.

2. Die a priori Wahrscheinlichkeit des zu Grunde liegenden Konzepts: Mehrheitswahl scheint mehr Effekt zu haben, wenn beide Klassen mit derselben Wahrscheinlichkeit auftreten.

7.2.2 Das Lernverhalten

In diesem Abschnitt sollen die Verläufe der Lernkurven untersucht werden, die sich für die jeweiligen Agententeams ergeben. Das Leistungsmaß ist also der zeitliche Verlauf des Anteils an richtig klassifizierten Beispielen der Hypothesen, welche die Agenten bei jeweils optimaler Strategie bilden (also eins minus dem echten Fehler der Hypothesen). Abbildung 17 zeigt das entsprechende Diagramm. Für dieses Experiment wurde die große Insel verwendet, Agententeams wurden aus drei Agenten gebildet. Es wurde $E_g = 5.0$, $E_z = 20.0$ und $E_k = 0.5$ gesetzt. Das zu lernende Konzept war *simple8.*

Die einzelnen Lernkurven zeigen einen für das jeweilige Verfahren charakteristischen Verlauf.

Im Fall des Austauschs von Trainingsbeispielen kommt es nach der ersten Phase, in der alle Agenten graben, zu einem ersten Austausch von Trainingsbeispielen. Dabei erhält jeder Agent von jedem anderen m_0 Beispiele. Dies führt zu einem sprunghaften Anstieg der Lernleistung. Danach steigt diese nur noch langsam.

Beim Austauschen von Hypothesen kommt es zu mehreren Sprüngen, und zwar immer dann, wenn die Hypothesen ausgetauscht werden. Dazwischen steigt die Lernleistung auch nur langsam.

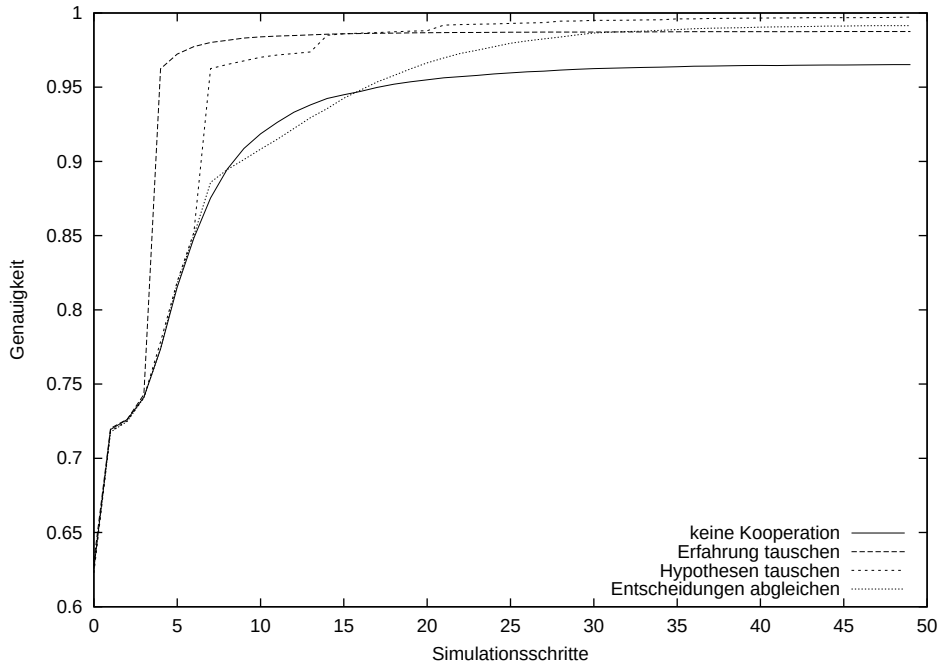


Abbildung 17: Lernkurven

Das Abgleichen von Entscheidungen ist in diesem Zusammenhang etwas außer Konkurrenz, denn dieses Verfahren basiert nicht nur auf dem Durchschnitt aller Hypothesen, sondern auf gemeinsamen Entscheidungen dieser Hypothesen, welche in dem Diagramm nicht sichtbar werden. Man sieht aber, dass dieses Verfahren kurzfristig dem Lernerfolg hinderlich sein kann, denn die entsprechende Kurve verläuft eine Zeit lang unter der Kurve für nicht-kooperatives Lernen.

Im Gesamtvergleich zeigt die Kurve, die sich beim Austausch von Erfahrung ergibt, den stärksten Anstieg. Dann wird allerdings die Kooperation schnell eingestellt, so dass der restliche Verlauf der Kurve sehr flach ist. Beim Austausch von Hypothesen ergibt sich zwar auch ein steiler Anstieg, der allerdings später erfolgt. Dies liegt daran, dass der optimale Punkt, um mit Kooperation zu beginnen, im Fall des Hypothesentauschens später liegt. Die Kurve, welche durch Abgleich von Entscheidungen entsteht, lässt sich wie bereits erwähnt nur schlecht mit den anderen vergleichen, da hier nicht eigentlich eine Hypothese selbst, sondern mehrere Hypothesen auf einmal angewendet werden.

7.2.3 Die benötigte Rechenzeit

Die folgende Tabelle zeigt die Rechenzeit, welche für 500 Simulationsläufe jedes der Agententeams benötigt wurde. Da diese von der Rechneraustattung abhängt, sind die Werte nur relativ zueinander interessant.

	keine Kooperation	Erfahrung tauschen	Entscheidungen abgleichen	Hypothesen tauschen
simple4	0:22	1:00	0:24	0:35
simple8	2:22	5:12	2:26	4:40
complex7	2:23	5:41	2:25	6:17
simple8noise20	3:48	13:02	3:50	11:43

Zunächst sieht man, dass die Anzahl der Attribute einen sehr starken Einfluss auf die Rechenzeit hat. Des weiteren ist die Rechenzeit beim Abgleich von Entscheidungen praktisch dieselbe, wie beim nicht-kooperativen Lernen. Hypothesen tauschen ist bei den beiden einfachen Konzepten *simple4* und *simple8*, schneller als das Austauschen von Erfahrung. Bei *complex7* hingegen ist der Austausch von Erfahrung schneller. Der Grund dafür liegt darin, dass je komplexer das Konzept, desto größer die Hypothesen, und desto aufwendiger ist das Einbinden der Regeln in den Aufbau des Entscheidungsbaums. Der Zusammenhang, der Rauschen enthält, macht allen Verfahren Probleme, besonders aber den Verfahren zum Austausch von Erfahrung und von Hypothesen, deren Rechenzeit überproportional ansteigt.

8 Zusammenfassung und Ausblick

8.1 Zusammenfassung

Die vorliegende Arbeit hatte es zum Ziel, sich systematisch mit dem Problem des Multi-Agenten-Lernens auseinanderzusetzen. Dazu wurde zunächst ein kurzer Überblick zu den Themen Multi-Agenten Systeme, Maschinelles Lernen sowie Multi-Agenten-Lernen gegeben. Daraufhin wurde ein Szenario entwickelt, welches einen Vergleich verschiedener Methoden zum kooperativen Lernen ermöglicht. Auf Grundlage dieses Szenarios wurden dann drei Ansätze zum kooperativen Lernen untersucht: der Austausch von Erfahrung, der Austausch von Hypothesen und das Abgleichen von Entscheidungen. Diese wurden zunächst dargestellt und analysiert, danach mit Hilfe eines Simulators experimentell untersucht und verglichen.

Die beiden folgenden Abschnitte sollen die wichtigsten Ergebnisse der vorliegenden Arbeit kurz darstellen.

8.1.1 Das Szenario

Das in Abschnitt 4 vorgestellte Szenario dient dazu, verschiedene Lernverfahren überhaupt erst vergleichbar zu machen. Es soll dabei verschiedenen Bedingungen genügen. Zum einen soll genau definiert sein, was die Agenten lernen, welche Erfahrung und welche Aktionen ihnen dabei zur Verfügung stehen und wie der Erfolg des Lernens gemessen wird. Zum anderen sollte das Szenario möglichst einfach und überschaubar sein, dabei aber genügend Möglichkeiten für die Variation von Parametern lassen.

Im Rahmen des Szenarios befindet sich eine Gruppe von Agenten auf einer Insel, welche eine Menge von Plätzen enthält. Diese Plätze sind durch Kanten verbunden, entlang derer sich die Agenten bewegen können. Die Plätze selbst sind durch eine endliche Anzahl von binären Attributen repräsentiert. Außerdem ist an manchen der Plätze eine Energiezelle vergraben, welche die Agenten ausgraben können. In diesem Fall erhalten sie die Energie, welche sich in der Energiezelle befindet. Das Vorhandensein einer Energiezelle an einem Platz steht in Zusammenhang mit den Attributen, welche diesen Platz repräsentieren. Da Graben die Agenten Energie kostet, sollten sie versuchen, diesen Zusammenhang zu erlernen. Dabei ist es den einzelnen Agenten erlaubt, beliebige Informationen auszutauschen, was sie allerdings auch Energie kostet. Die Energie, welche alle Agenten eines Teams nach einer festgelegten Anzahl von Schritten zusammen erreicht haben, wird als Leistung des Agententeams gewertet.

Es zeigt sich, dass bereits ein solch einfaches Szenario interessante Probleme aufwirft. Diese Probleme betreffen nicht nur die Lern- und Kooperationsalgorithmen selbst, sondern vor allem auch die Fragen, wann ein Agent graben und

wann er kooperieren sollte. Die einstellbaren Parameter des Szenarios erlauben eine Untersuchung dieser Fragen unter Veränderung der externen Bedingungen, wie beispielsweise der Kommunikationskosten.

8.1.2 Die verschiedenen Ansätze zum kooperativen Lernen

Grundsätzlich beruhen alle hier vorgestellten Ansätze auf demselben Grundprinzip: wenn ein Agent an einem Platz gräbt, zeichnet er auf, ob er dort eine Energiezelle gefunden hat. Dann fügt er dieses Ergebnis zusammen mit den Attributen, welche den Platz repräsentieren, als Trainingsbeispiel zu seiner Erfahrung hinzu. Auf Grundlage solcher Trainingsbeispiele kann er Hypothesen bilden, welchen den Zusammenhang zwischen den Attributen an einem Platz und dem Vorhandensein einer Energiezelle an diesem Platz beschreiben. Diese Hypothesen kann er nun benutzen, um zu ermitteln, ob an einem Platz, an dem noch nicht gegraben wurde, eine Energiezelle zu erwarten ist. Dies hilft ihm bei seiner Entscheidung, ob er an diesem Platz graben soll, oder nicht.

Die Hypothesen bilden dabei aber nur einen Teil der Entscheidungsfindung. Den anderen Teil bildet die Strategie der Agenten.

Die Strategie der Agenten bezieht sich auf die Fragen, wann die Agenten graben sollten und wann sie kooperieren sollten. Dabei ergibt sich für alle Verfahren eine einheitliche Grundstruktur, welche den zeitlichen Ablauf der Agenten in drei Phasen einteilt. In der ersten Phase sollten die Agenten immer graben, wenn sie an einen Platz kommen, der noch nicht ausgebeutet ist. Da sie immer graben, brauchen sie in dieser Phase nichts entscheiden und benötigen daher weder eine Hypothese noch Kommunikation. Verfügt nun ein Agent über eine gewisse Mindestanzahl von Trainingsbeispielen, dann geht er in die zweite Phase über. Nun kommuniziert er mit den anderen Agenten und richtet sich bei der Entscheidung, ob er graben soll oder nicht, nach seiner Hypothese bzw. dem Ergebnis der Kommunikation. Überschreitet nun die Anzahl der Trainingsbeispiele, über die ein Agent verfügt, wiederum eine bestimmte Anzahl, dann kommuniziert der Agent nicht mehr mit den anderen Agenten und richtet sich bei seinen Entscheidungen nur noch nach seiner eigenen Hypothese.

Die Begründung für dieses Vorgehen ist grob folgende: um überhaupt sinnvoll entscheiden und kooperieren zu können, müssen die Agenten über eine gewisse Erfahrung (in Form von Trainingsbeispielen) verfügen. Daher graben sie in der ersten Phase immer. In der zweiten Phase können sie durch Kooperation die Präzision ihrer Hypothese und ihrer Entscheidungen steigern. In der dritten Phase schließlich würde Kooperation zu keiner Verbesserung führen, welche die Kosten für die Kommunikation aufwiegt, daher kooperieren sie nicht mehr.

Dieses Modell erleichtert die weitere Untersuchung der Strategien erheblich, da nun hauptsächlich nur noch die Grenzen zwischen den Phasen experimentell bestimmt werden müssen. Dabei ergeben sich übergreifend einige signifikante Abhängigkeiten. So hängt die Länge der ersten Phase sehr stark von der Ener-

gie für das Graben und der Energie in den Energiezellen ab, sowie von der Komplexität des zu lernenden Konzepts. Die Länge der zweiten Phase hingegen ist eher abhängig von der Energie, welche für Kommunikation aufgebracht werden muss. Die Länge der dritten Phase ist schließlich durch die beiden ersten Phasen und die Gesamtanzahl von Simulationsschritten festgelegt.

In welcher Form die Agenten während der zweiten Phasen kooperieren, ist Gegenstand der einzelnen Ansätze zum kooperativen Lernen, welche im folgenden kurz dargestellt werden.

Austausch von Erfahrung

Die Agenten tauschen Trainingsbeispiele aus, die ihrer Erfahrung entspringen. Sie müssen somit weniger Trainingsbeispiele durch Versuch und Irrtum gewinnen und sparen Energie im Vergleich zum nicht-kooperativen Lernen. Außerdem ist es ihnen möglich, ihre Hypothese auf Grundlage einer größeren Menge von Trainingsbeispielen zu generieren, was im allgemeinen dazu führt, dass diese Hypothese präziser wird. Eine präzisere Hypothese führt aber wiederum zu einer Ersparnis von Energie, da falsche Entscheidungen vermieden werden.

Experimentell ergibt sich, dass der Austausch von Erfahrung in den meisten Fällen den beiden anderen Ansätzen überlegen ist. Die Erklärung dafür liegt darin, dass bei Verwendung dieser Methode ein Maximum an Information, welche in der Erfahrung enthalten ist, übertragen wird. Bei den anderen Verfahren wird diese Information immer nur transformiert weitergegeben. Diese Transformation führt im allgemeinen zu keiner Verbesserung, da alle Agenten denselben Lernalgorithmus verwenden und kein Hintergrundwissen einsetzen.

Weiter zeigt sich, dass der relative Nutzen (d.h. die prozentuale Verbesserung gegenüber nicht-kooperativem Lernen) abhängig ist von den Parametern des Szenarios. Der Nutzen steigt, je geringer die Energie für Kommunikation und je komplexer das Konzept ist, welches gelernt werden soll. Ist die Energie für das Graben gering im Vergleich zu der Energie in einer Energiezelle, dann sinkt der Nutzen durch Kooperation.

Eine Betrachtung der Lernkurve zeigt, dass es im Fall des Austauschs von Trainingsbeispielen früher als bei den anderen Verfahren zu einem sehr steilen Anstieg der Präzision der gebildeten Hypothese kommt, was der Grund für das gute Ergebnis dieses Ansatzes ist.

Nimmt man eine Messung der Rechenzeit vor, so zeigt sich, dass der Austausch von Erfahrung deutlich aufwendiger ist, als das Abgleichen von Entscheidungen und in vielen Fällen auch aufwendiger als der Austausch von Hypothesen.

Die Vorteile der Methode liegen einmal darin, dass sie einfach zu implementieren ist und unabhängig von den verwendeten Lernalgorithmen angewendet werden kann. Vor allem aber ist das Ergebnis, welches durch diese Methode erzielt wird, in vielen Fällen besser als bei Verwendung der anderen beiden Ansätze.

Der Hauptnachteil dieser Methode ist der Rechenaufwand, der entsteht, wenn die Agenten eine größere Menge von Trainingsbeispielen verarbeiten müssen. Dabei fällt auch die Tatsache negativ ins Gewicht, dass ein einzelnes Trainingsbeispiel von allen Agenten verarbeitet werden muss.

Austausch von Hypothesen

Die Agenten tauschen von Zeit zu Zeit ihre Hypothesen in Form von Regelmengen aus. Dann binden sie die fremden Hypothesen als Hintergrundwissen beim Aufbau ihrer eigenen Hypothese ein. Damit verringern sie im allgemeinen den Fehler ihrer eigenen Hypothese und sparen somit Energie.

Experimentell zeigt diese Methode etwas schlechtere Ergebnisse als der Austausch von Erfahrung. Der relative Nutzen durch den Einsatz dieser Methode hängt dabei wiederum von den Simulationsparametern ab. Wie schon beim Austausch von Erfahrung steigt der Nutzen, wenn die Energie für Kommunikation sinkt. Dagegen wirkt sich eine Steigerung der Komplexität des zu lernenden Konzepts negativ auf den Nutzen aus, da im allgemeinen größere Hypothesen übertragen werden müssen, wodurch die Kosten für Kommunikation steigen.

Die Lernkurve, die sich beim Austauschen von Hypothesen ergibt, steigt später an als im Fall des Austauschs von Erfahrung. Der Anstieg erfolgt stufenweise, da die Agenten in regelmäßigen Abständen ihre Hypothesen tauschen, was jeweils zu einer sprunghaften Verbesserung der Hypothesen führt.

Eine Betrachtung der Rechenzeit zeigt folgendes: ist das zu Grunde liegende Konzept einfach, dann liegt diese deutlich unter der Rechenzeit für den Austausch von Erfahrung, ist das Konzept komplexer, dann kann es gerade umgekehrt sein. Wiederum liegt dies daran, dass sich bei einem komplexeren Konzept im allgemeinen eine größere Menge an Regeln ergibt, welche beim Aufbau des Entscheidungsbaums eingebunden werden müssen.

Die Vorteile dieses Verfahrens liegen darin, dass bei einem relativ einfachen Konzept weniger Information übertragen werden muss als bei den anderen beiden Verfahren und dass der Aufwand für die Verarbeitung in diesem Fall geringer ist als beim Austausch von Erfahrung. Bei einem komplexeren Konzept kann dies allerdings umgekehrt sein.

Ein Nachteil dieses Verfahrens ist, dass weder der Nutzen, noch die Kosten, die sich ergeben, allgemein eingeschätzt werden können. Beides hängt stark von den Parametern des Szenarios ab. Außerdem sind die Voraussetzungen für den Einsatz dieser Methode wesentlich höher als bei den anderen Verfahren: alle beteiligen Agenten müssen Hypothesen in Form von Regeln generieren und solche Regeln als Hintergrundwissen in den Lernvorgang einbinden können.

Abgleich von Entscheidungen

Die Agenten lernen alle individuell aus ihrer eigenen Erfahrung. Wenn sie entscheiden müssen, ob sie an einem gegebenen Platz graben sollen, fragen sie

zunächst die anderen Agenten um Rat, indem sie ihnen die Attribute des Platzes zusenden. Empfängt ein Agent eine solche Anfrage, dann klassifiziert er den Platz mit seiner eigenen Hypothese und sendet das Ergebnis zurück. Aus allen Antworten und dem Ergebnis der eigenen Hypothese ermittelt der anfragende Agent durch Mehrheitswahl eine endgültige Entscheidung. Die Idee dabei ist, dass die Wahrscheinlichkeit dafür, dass sich mehrere Agenten zusammen irren, geringer ist, als dass sich ein einzelner Agent irrt.

Experimentell zeigt diese Methode aber wenig Nutzen. Sie ist in fast allen Fällen den beiden anderen Verfahren unterlegen. Dafür ergeben sich zwei Gründe. Zum einen zeigt sich, dass homogene Agenten sehr häufig dieselben Entscheidungen treffen, so dass es nur in wenigen Fällen zu einer Änderung der Entscheidung durch Mehrheitswahl kommt. Zum anderen hat die a priori Wahrscheinlichkeit des zu lernenden Konzepts einen sehr starken Einfluss auf den Erfolg der Methode. Treten beide Klassen mit derselben Wahrscheinlichkeit auf, dann ist der Nutzen durch Mehrheitswahl deutlich größer, als wenn eine Klasse wahrscheinlicher ist als die andere.

Eine interessante Beobachtung ergibt sich im Zusammenhang mit der Lernkurve. Während die Agenten kooperieren verläuft diese nämlich stückweise unter der Lernkurve für nicht-kooperatives Lernen. Der Grund dafür liegt darin, dass Mehrheitswahl bei dem gegebenen Konzept sehr viel häufiger dazu führt, dass ein Agent davon abgehalten wird zu graben, als umgekehrt. Wenn die Agenten aber weniger graben, so lernen sie auch weniger.

Was die Rechenzeit betrifft, so ist diese praktisch identisch mit der Rechenzeit für nicht-kooperatives Lernen, es entsteht also kaum zusätzlicher Aufwand.

Die Vorteile des Verfahrens liegen darin, dass es einfach zu implementieren ist, keinen hohen Rechenaufwand verursacht und völlig unabhängig von den eingesetzten Lernalgorithmen ist.

Der Hauptnachteil dieses Verfahrens ist der geringe Nutzen, vor allem wenn die Agenten relativ homogen aufgebaut sind.

8.2 Ausblick

Die Untersuchungen der vorliegenden Arbeit mussten an verschiedenen Stellen beschränkt bleiben. In diesem Abschnitt soll diskutiert werden, welche weiteren Untersuchungen als Fortführung möglich wären.

Ein Teil der Einschränkungen wurde bei dem gewählten Szenario selbst vorgenommen. Hier könnte man sich folgende Erweiterungen denken:

- Die Plätze könnten nicht nur durch binäre Attribute repräsentiert sein, sondern durch allgemeine, also beispielsweise auch kontinuierliche Attribute.

- Während hier nur Agenten untersucht wurden, die alle Attribute eines Platzes wahrnehmen können, wäre es denkbar, Teams von Agenten zu untersuchen, deren einzelne Mitglieder nur über eingeschränkte Sensoren verfügen.
- Der Zusammenhang zwischen den Attributen, die einen Platz repräsentieren, und dem Vorhandensein einer Energiezelle an diesem Platz könnte in verschiedenen Regionen der Insel verschieden sein.
- Dieser Zusammenhang könnte sich auch mit der Zeit leicht verändern, was die Agenten zwingen würde, gelerntes Wissen wieder zu vergessen.
- Es wäre möglich eine Wahrscheinlichkeit dafür einzuführen, dass die Agenten an einem ausgebeuteten Platz feststellen können, ob sich an diesem Platz eine Energiezelle befunden hat.
- Die Kommunikationsmöglichkeiten auf der Insel könnten so eingeschränkt sein, dass es den Agenten nur von Zeit zu Zeit oder an bestimmten Plätzen möglich ist, zu kommunizieren.

Ein weiterer Teil der Einschränkungen betrifft die untersuchten Ansätze zum kooperativen Lernen. Es wären unter anderem folgende weiterführende Untersuchungen möglich:

- Da es in dieser Arbeit eher darum gehen sollte, die charakteristischen Eigenschaften einzelner Verfahren zu analysieren und zu vergleichen, sind diese Ansätze noch nicht erschöpfend optimiert worden. Dies gilt insbesondere für den Austausch von Hypothesen. Hier wäre eine wesentlich eingehendere Untersuchung möglich.
- Während für diese Arbeit nur Entscheidungsbäume eingesetzt wurden, wäre es möglich auch andere Lernverfahren anzuwenden, wie z.B. Künstliche Neuronale Netze.
- Die Lernverfahren könnten zusätzlich angepasst werden, so dass sie beim Bilden der Hypothese die Kostenparameter berücksichtigen. Dies wäre sinnvoll, wenn ein Irrtum in eine Richtung wesentlich schwerwiegender ist, als in die andere Richtung.
- Die Agenten in dieser Arbeit sind homogen, d.h. alle Agenten sind prinzipiell gleich. Es wäre aber durchaus auch ein Team von Agenten denkbar, in dem die einzelnen Agenten bestimmte Rollen einnehmen. Beispielsweise könnte nur ein Agent lernen, der dann die Beispiele von allen anderen Agenten erhält und die Hypothese an sie zurücksendet.

Schließlich wäre es möglich, die Simulation des Szenarios so zu gestalten, dass verschiedene Forschungsgruppen ihre Agententeams an dieser Simulation testen

könnten. Dies könnte auch in Form eines Wettbewerbs für kooperatives maschinelles Lernen erfolgen, analog zu den Wettbewerben, die im Roboterfußball durchgeführt werden.

Literatur

[1] P. Brazdil, M. Gams, S. Sian, L. Torgo, and W. van de Velde. Learning in distributed systems and multi-agent environments. In Y. Kodratoff, editor, *Lecture Notes in Artificial Intelligence Vol.482: Machine Learning-EWSL-91*, pages 412–423. 1991.

[2] P. Brazdil and L. Torgo. Knowledge acquisition via knowledge integration, 1990.

[3] David Carmel and Shaul Markovitch. Opponent modeling in multi-agent systems. In *Adaption and Learning in Multi-Agent Systems*, 1995.

[4] Winton Davies and Peter Edwards. The communication of inductive inferences. In Gerhard Weiß, editor, *Selected Papers of the ECAI'96 Workshop LDAIS <Budapest, Hungary, August 13, 1996> and the ICMAS'96 Workshop LIOME <Kyoto, Japan, December 10, 1996> on Distributed Artificial Intelligence Meets Machine Learning: Learning in Multi-Agent Environments*, volume 1221, pages 223–241. Springer-Verlag: Heidelberg, Germany, 1997.

[5] Aldo Franco Dragoni and Paolo Giorgini. Learning agents' reliability through baysian conditioning. In *Distributed Artificial Intelligence Meets Machine Learning*, 1997.

[6] Holger Friedrich, Michael Kaiser, Oliver Rogalla, and Rüdiger Dillmann. Learning and communication in multi-agent systems. In *Distributed Artificial Intelligence Meets Machine Learning*, 1997.

[7] L. Kuncheva, J. Bezdek, and R. Duin. Decision templates for multiple classifier fusion: an experimental comparison, 1999.

[8] Paul Levi. Architectures of individual and distributed autonomous agents. In *Proceedings of the 2nd International Conference on Intelligent Autonomous Systems*, 1989.

[9] Tom Mitchell. *Machine Learning.* McGraw Hill, 1997.

[10] Rajani Nadella and Sandip Sen. Correlating internal parameters and external performance: Learning soccer agents. In *Distributed Artificial Intelligence Meets Machine Learning*, 1997.

[11] Takuya Ohko, Kazuo Hiraki, and Yuichiro Anzai. Addressee learning and message interception for communication load reduction in multiple robot environments. In *Distributed Artificial Intelligence Meets Machine Learning*, 1997.

[12] P. Rosenbloom, H. Hirsh, W. Cohen, and B. Smith. Two frameworks for integrating knowledge in induction, 1993.

[13] Stuart Jonathan Russell and Peter Norvig. *Artificial intelligence : a modern approach*. Prentice-Hall, 1995.

[14] Thomas W. Sandholm and Robert H. Crites. On multiagent q-learning in a semi-competitive domain. In *Adaption and Learning in Multi-Agent Systems*, 1995.

[15] B. Smith. Induction as knowledge integration, 1995.

[16] Paul E. Utgoff. An improved algorithm for incremental induction of decision trees. In *Proc. 11th International Conference on Machine Learning*, pages 318–325. Morgan Kaufmann, 1994.

[17] Cristina Versino and Luca Maria Gambardella. Learning real team solutions. In *Distributed Artificial Intelligence Meets Machine Learning*, 1997.

[18] Gerhard Weiss, editor. *Adaption and Learning in Multi-Agent Systems*. Springer, 1996.

[19] Gerhard Weiss, editor. *Distributed Artificial Intelligence Meets Machine Learning*. Springer, 1997.

[20] Gerhard Weiss. *Multiagent Systems*. MIT Press, 2000.