Proposal for a thesis entitled
‘ Learning Dynamic stereotypes for opponent modelling’

In Partial fulfilment of an Interdisciplinary PhD
Submitted by Joe MacInnes
August 24, 2001
abstract - One of the earliest and simplest methods of modelling a user’s behaviour involved clustering each user by one of a collection of pre-defined stereotypes. Although successful to a degree, these attempts were limited since the stereotypes were static while a user’s behaviour could be unpredictable and dynamic over time. This research will attempt to create a collection of dynamic stereotypes using Neural Networks in order to model the user’s behaviour in an adversarial environment. These net/stereotypes will be created offline by clustering the training data (previous user’s behaviour) into groups with similar behaviour. Each of these groups will train a different network/stereotype that will be used to predict on-line behaviour.

Introduction
Applications of user modelling usually falls under one of three categories. User interfaces (esp. Adaptive or intelligent user interfaces) will attempt to model the user in order to improve the ease and/or effectiveness of the (Human/computer) Interaction. Data mining also benefits from modelling since an accurate model of the user’s behaviour and preferences can significantly reduce the scope of searches. Finally, multi-agent systems will benefit by successfully modelling human agents with whom they interact. Collaborative systems could cooperate more effectively by modelling their team-members and competitive environments would benefit by correctly anticipating their opponents actions. It is in this final scenario - Multi-agent, competitive environments - that I propose to develop and test these dynamic stereotypes. Specifically I will test modelling performance using competitive multi-agent games, building on the environment and work from MacInnes etal (2001).

Competitive environments offer an additional difficulty/challenge by not being able to get assistance from the user being modelled. In fact, the user may hide or masquerade their behaviour. Also co-operative modellers may suggest options to the user which may or may not be taken, the competitive modeller has only its own recommendations with which to make its decision. The modeller, in effect, needs to achieve the same (or higher) accuracy with less information.

Literature review
Although there has been much work done recently in related and similar fields, the topic of machine learning for opponent-user modelling remains relatively unexplored. Areas from which literature will be borrowed for this thesis proposal include user modelling, opponent modelling, machine learning, and of course the topic that does contain significant overlap, machine learning for user modelling. I will borrow ideas and methods from these diverse areas as well as point out situations where they differ to such an extent as to render the methods inadvisable. The following is a summary of works which I believe to be very relevant to my proposal.

Stereotypes
One of the pioneer works in the user modelling literature and definitely one of the most cited is Rich’s paper on using stereotypes for user modelling (Rich, 1979). Although stereotypes had been used prior to 1979, the work focussed on stereotyping as it applied to human cognitive understanding. Rich, however,
suggested that the usefulness of stereotypes could be extended to models of computer users. Since the rational behind stereotypes is that the world is far too complex to understand (and remember) without simplification and categorization of the details, the same benefits could be achieved by computers.

Rich implemented ‘Grundy’, a system which attempted to model users in order to predict the user’s preference in books. User models were implemented as a series of ‘facets’ which represented individual facts about the user. Contained within each facet is a rating which shows the bias towards or against this attribute, a rating which reflects the certainty (probability) of this aspect, as well as the justification for the belief in this aspect. Therefore, the grouping “education;5;900;INTELLECTUAL” would represent a facet for a user which suggests an interest in educational material with a value of 5 (high) and a rating of 900 (also high) with the justification being that the user activates the INTELLECTUAL stereotype. Stereotypes get activated in this systems by one of a number of triggers. A trigger can be any event which in which the user interacts with the system. Many triggers will be activated in the beginning of a session in which Grundy asks the user a series of personal questions. Triggers can also be activated later in the session during the systems normal interaction with the user, either confirming, adding or modifying assumptions that were made earlier.

Stereotypes also contained these facets, but of the ‘typical’ member of that group. Whenever a stereotype is triggered, the values of that stereotype are added to the users profile. The value and rating of facets which coexist in the user’s profile are adjusted according to whether they confirm or conflict with those found in the stereotype. All stereotypes were arranged using a directed-Acyclic Graph (DAG) going from the most generic to specific. The DAG was implemented as opposed to a tree to allow for any child stereotype to be specifications of more than one parent (more generic) stereotype. This is important since activating a stereotype also activates all of its generalizations.

Although much has been said recently about the inflexibility of stereotypes in user modelling, Rich did make attempts to implement learning within the stereotypes. As new information is acquired about members of a given stereotype, the values and ratings of the facets may be adjusted to reflect that new information. Although it was not implemented in Grundy, Rich even suggests ways in which new stereotypes may be learned using classification techniques in use at the time.

Although no statistical analysis was done on the effectiveness of this system, Rich did demonstrate that the number of successful book recommendations nearly doubled by using Grundy compared to random selection.

**Machine learning for user modelling**

Recent work in machine learning for user modelling has attempted to learn not only group membership, but the group models themselves. Doppelganger (Orwant, 1996) is an attempt to learn and adapt these group models based on user behaviour, and in turn use these models to predict the behaviour of future users. This system is used to model user preferences across a variety of applications. Doppelganger will be covered in greater detail than the rest since it bears the most resemblance to my proposed research.
Doppelganger uses a pragmatic (bottom-up) user modelling system as opposed to cognitive models used in many previous systems. Although inferences may be made about a user’s cognitive state, they are done without forming a formal cognitive model. Doppelganger is implemented as an on-line distributed agent. A version of the modeller exists on many machines and each may model a user in slightly different ways (e.g., a user’s home and work computer would share data, but not necessarily form the same model). Applications or other agents query Doppelganger using a client/server architecture. Although this opens some security concerns over the privacy and access of a user’s information, the author does take a number of steps to achieve a high level of security.

Information is fed to Doppelganger via a number of software and hardware sensors. Since the information from these sensors may often be incomplete or erroneous, an accuracy estimate is provided with each sensor. Although the majority of Doppelganger’s sensors are unobtrusive, the system does allow for user prompting by sending a query to the user over email and parsing the response, if any.

Doppelganger attempts to borrow from the stereotype literature by using what the author calls ‘communities’. These are typical stereotypes found in many user modelling applications with the exception that membership in these communities is not all or nothing. Users belong to all communities based on a matter of degree. Also these communities are not static; they are computed as the weighted averages of their membership (user models). The communities are therefore dynamic, changing as its user base changes.

Doppelganger uses a variety of learning techniques linked, in part, to the different sensor input streams. Examples of these techniques given in Orwant (1996) include: beta distribution, linear prediction, and Markov models. The beta distribution is used for boolean inputs (sensors); it accepts a string of examples of this boolean variable and uses the mean and variability of the distribution to determine the expected value and its probability. Note that there is no order given to these samples and therefore, equal weight is given to all sensor input regardless of recency.

Linear prediction is a method used by Doppelganger which does contain memory, in particular, cyclical pattern such as log in times. Much work could be done in advance if the system knew when the user was likely to log in next. This learner predicts future patterns (with decreasing accuracy into the future) based on previous repetitive (though not necessarily perfect) patterns.

Finally Markov Models were also used for situations which were dependent on the current state of the system and the user. Weighted probabilities were once again used to determine (predict) which state the user (or system) was likely to enter based on the current state information (busy, idle, etc). Doppelganger uses 8 Hidden Markov Models to describe the types of states that are likely for its users. These states resemble stereotypes very closely and include, for example: hacking, idle, frustrated, writing, learning, playing, concentrating, image processing, and connecting. The system uses its sensors to determine which state a user is likely to be in, and from there, which Markov model/stereotype most closely describes the user’s behaviour pattern.
There are two areas in which the Doppelganger system may be compared to stereotypes; Communities and Hidden Markov models of the user’s state. Although Doppelganger does allow for dynamic models in the former, the latter are static. Even with the dynamic communities, however, there are a fixed number of communities (22) with set labels (artists, children, students, etc). The communities are only dynamic in that their predictions may change based on the changing behaviour of their membership and that their membership is not binary.

**Multi agent systems**

Alexandros Moukas (1997) used user modelling techniques in an evolving multi-agent system. A multi-agent system was developed to model user’s preferences in web browsing and suggest new web pages to improve the user’s performance and experience. The system creates an evolving system of agents which compete and cooperate in a limited resource environment. The two types of agents which inhabit this environment are information filtering agents which are responsible for the user’s profile, and information discovery agents which control the gathering of information. Information is gathered about a user by observation (browser bookmarks, history files) and direct questioning. In order to address ethical and privacy issues, the author also allowed users to view and modify their profile at any time.

Since these agents are adaptive, the filtering agent responsible for a given user may change as that user’s goals and preferences change. Since each user has their own closed system of agents, these changes only apply to the current user. Agents adapt and mutate using genetic learning techniques. Performance of these agents is inferred by observing the user’s interaction with the environment. Filtering agents receive credit for good choices and lose credit for bad choices. Filtering agents also ‘pay’ information retrieval agents that were used for good results.

Although stereotypes were not mentioned in this article, a similar construction called ‘packages’ were used. These were pre-trained collections of agents which reflected an interest in a particular area such as ‘Agents’, or ‘Soccer’. The user could, at any time, add one of these packages to their profile to express a new interest.

**Adversarial Modelling**

Burns & Vollymeyer (1998) look at subject’s models of other human opponents and their utility in competitive games. The authors extended this study to include recursive models up to the 3rd order (What I think my opponent thinks of me). He references Thagard (1992) for proposing the need of recursive modelling. (See below)

Pairs of participants were asked to play a purely adversarial 2-player game in which an ‘avoider’ selects a number from 1-3 and the ‘chooser’ has try select the same number. A sliding payoff matrix assigns positive and negative scores based on the success of each opponent. It is a ‘zero-sum’ game, meaning that random selection should lead to even scoring. Optimal strategy, however, as shown by game theory puts the advantage with the avoider.
Modelling was measured by self-report using descriptive assessment pairs (e.g., Humorous-Serious, negative-positive, hard-soft, rational-intuitive, and risk taking-risk avoiding) and a 7-point scale. Each participant is asked to rate themselves (1st order), their opponent (2nd order) and how they thought their opponent would rate them (3rd order). The authors argue that relative performance is more important than absolute performance which they use as justification for calculating all scores as relative for the two players involved. Only relative 2nd order and 3rd order modelling accuracy was used for analysis. Only 3rd order was positively correlated to performance. (This may have more to do with distance from what I see to be the most error prone assessment- 1st order).

**Thagard- Adversarial problem solving**

Thagard (1992) proposes to demonstrate the cognitive processes necessary for successful opponent modelling. He uses ECHO, a connectionist model of explanatory coherence, to simulate examples of opponent modelling and deception. Thagard looks at a wide variety of examples for opponent modelling from diverse areas such as war strategy, business and game playing.

The author begins the analysis by listing some common ‘rules’ of Adversarial Problem Solving (APS). They are: “

1) Construct a model of the opponent. Include situation, past behaviour, competitiveness and attitude toward risk among others.
2) Include 2nd degree modelling. (Opponents model of you)
3) Use this model to infer Opp’s plan, and include this plan in your model.
4) Use the revised model to devise plans that Opp. May not expect.
5) Take steps to conceal this plan from Opp.

Thagard claims that only the last two steps are unique to adversarial problem solving.

I believe Thagard’s most important contribution with this work is his discussion of deception. It is in this article that the author suggests it is third order modelling that is critical for successful deception. It is critical to understand how the opponent will interpret your actions in order to cause an erroneous interpretation. The author raises a very important point when suggesting that this recursive modelling requires very little extra representation, and that is the assumption that the Opponent has roughly the same cognitive abilities as oneself. In competitive agent environments where human and software agents are supposed to be interchangeable, this is often not the case. Indeed, as business and war become ever more technical, we may find an ever increasing rate of human/machine modelling scenarios.

Laird (2000) uses agents from the popular game ‘Quake’ in an attempt to add anticipation to the autonomous agents (QuakeBots). QuakeBots have been used with increasing frequency for competitive agents due to their complex environments, well defined rule sets as well as the interchangeability of human and computer agents. The Quake environment also has the added difficulty of being a real-time (as opposed to turn based) competition. Any model of a plan must include time information to be effective. The author’s saw a need for anticipation with their intelligent bots after many attempts to improve performance by adding very specific ‘scenarios’ as a model of opponent behaviour. These scenarios were
fixed, pre-programmed and relied completely on the author’s observation of the bot’s past performance on opponents. Anticipation would allow the bot to determine and respond its opponent’s moves dynamically.

The authors don’t explain the architecture of the intelligent Quakebot, but it seems to be a hierarchical state machine with some measure of planning incorporated for the anticipation. Planning may be implicit, however since they state there is no automatic progression from one state to the next, as each action is selected by continually testing the current situation although the algorithm may ‘look ahead’ its decision is based on the current state of the environment.

By adding anticipation to the Quakebot program, the Author first had to add the capability to model the opponent. Assuming the opponent uses a similar state machine to its own (The authors not only assume a similar representation but also similar goals and tactics) the Quakebot assumes the role of the opponent until it gets a useful prediction or determines that there is not enough information to predict. This is determined by how much information the bot knows about its opponent as well as how useful the prediction is likely to be. For example, anticipation is not used when the bot does not know where its opponent is or when the next move is obvious (avoid a shooting opponent). Also, The opponent model does not appear to be a complete duplicate of its own state machine, but only the uppermost level of the behaviour hierarchy.

The authors create 3 new sub-states to take advantage of this anticipation; Hunt, Ambush and deny-powerup. These new states are activated if it is seen that the bot can reach a position prior to the anticipated arrival of its opponent. Some learning is incorporated into the algorithm in the form of ‘chunking’. This allows the system to pre-compile a set of rules that is shown to be effective on repeated occasions. This will avoid the need for the modelling step under repetitions of this situation, possibly saving critical time. The authors also mention possible extensions to their algorithm using; recursive, enemy specific and adaptive anticipation. No results are shown on the effectiveness of adding anticipation.

MacInnes et al (2001) demonstrate the use of recursive modelling of opponent agents in an adversarial environment. In many adversarial environments, agents need to model their opponents and other environmental objects to predict their actions in order to outperform them. In this work, the authors use Deterministic Finite Automata (DFA) for modelling agents. We also assume that all the actions performed by agents are regular. Every agent assumes that other agents use the same model as its own but without recursion. The objective of this work is to investigate if recursive modelling allows an agent to outperform its opponents that are using similar models.

The authors developed a 3-D “Quake-like” engine to test recursive modelling of autonomous “Quake-Bots” or in their case “Maze-Bots”. Each agent had imperfect knowledge of the world, using the same perceptions that a human agent would have. Sight was limited to a 55 degree field of view, perception was blocked by walls, and agents could ‘hear’ the other agent if a gun was fired. Agents had unlimited bullets in their guns, but a second shot could not be fired until the current shot hit an object, making accuracy a
very important variable.

Each bot differed only in the depth of recursion it used in its modelling process. Levels from 0-3 were tested in a Latin-squares design with each bot fighting the others a total of 15 times. Level 0 recursive bots did not model their opponent at all, level 1 modelled their opponent, level two also modelled what their opponent thought they were going to do and so on. (Note this is different language from previous examples where self modelling only is called level 1). During the first experiment, opponent modelling did improve maze-bot performance but the optimal depth of recursion was 1. Depth 2 and 3 recursion, while better than 0, were significantly worse than depth 1.

Considering the importance of shooting accuracy to the prediction function, the authors made an attempt to improve this function to test the unusual first results. (Papers above often cite 2 or 3 as optimal recursive depths). Since any Bot who modelled where their opponent was going to be had to use this location in an attempt to try ‘shoot ahead’ of their target, this calculation was critical. In fact, any slight error in this function would be cumulative (exponential?) for deeper recursion since it would be used at every recursive depth. For experiment 2 the authors improved the accuracy (see the paper for details on the problem and solution) and retested the Maze-bots. With the small increase in accuracy, the recursive bots still performed better than no opponent modelling, but the recursive depth actually improved performance up to depth 2 before a decline in performance was noticed (significant interaction with exp 1). Since Depth two in this study was the equivalent of depth 3 in other studies, we found the optimal depth to be at the level where deception was to occur.

Machine learning
The area of machine learning which relates most closely to the learning of stereotypes comes from the clustering literature (also called concept learning or mixture modelling). Jacobs and Nowlan (1991) proposed the idea that dividing a data set into clusters or subsets may be beneficial for a learning algorithm. Using neural nets to cluster letters in a vowel discrimination task, the authors first divide the data using a single Neural network (‘gating network’) into 4 subclasses of data, then use each of these subsets to train 4 other neural networks as experts (‘expert networks’) for their individual areas. The series of local experts were shown to perform on par with a single multilayer neural net, but the local expert solution reached this performance with ½ of the number of training epochs.

This work demonstrates learning methods which suggest much possibility for my thesis proposal. The clustering stage of my suggested learning process is comparable to the gating network of Jacobs etal, and the local expert networks show a very strong overlap with the stereotype literature. Although my proposal will be using a different learning method for the gating portion and a flexible number of recurrent networks for the expert portion, I believe that my results could be a very important replication and extension of this work.

Although standard neural networks are ideal for recognizing patterns, temporal pattern recognition (patterns which develop over time) has typically been seen as a problem for Neural networks. Mozer etal (1993)
demonstrate a number of possible neural net architectures which are well suited for adapting to this type of problem. All of these models contain the typical features associated with a neural network, but in addition, they also incorporate some form of short term memory from which temporal patterns may be detected. Although none of the algorithms presented in this paper were tested using time sensitive spatial navigation, they should be equal to the task.

**Discussion and proposal**

Stereotyping, as it applies to user modelling, refers to an attempt to model user behaviour by determining a user’s membership in one of a set of pre-defined groups. Early work in stereotypes restricted learning to determine group membership (E. Rich, 1979) from a set of predefined models. Models such as Beginner, Novice, and Expert would be developed off-line and an attempt would be made to match each user to their appropriate stereotype in an on-line matching process.

Static stereotypes, however, lead to a number of problems. Foremost is the problem that the stereotypes themselves need to be known beforehand. Since the groups must be defined during development, there is usually little data from which to base the groupings. It is usually the developers biases alone which decide the stereotypes. These stereotypes tend to be very vague and often poorly representative of a user’s actual performance. Also, an accurate stereotype may degrade as a user’s performance and preferences change over time. A user may gradually become more experienced, they may suddenly shift goals or preferences, or an opponent may change tactics in an attempt to utilize deception. All of these cases reflect the difficult machine learning problem of ‘concept drift’ (Widmer and Kubat, 1996) where the concept to be learned or modelled changes over time.

This proposed research will attempt to extend earlier work on user modelling with stereotypes by learning these groups dynamically, building on the machine learning clustering literature. One of the simplest methods that may be used is dividing the data on the discrete attributes which maximize information gain. This would create the desired tree-like structure and leave the continuous attributes for training the individual stereotypes (see discussion of recurrent Neural nets below). Another alternative is to use a feed-forward, back-propagation neural net as found in Jacobs & Nowlan (1991).

The problem of concept drift will also be handled by only modelling the opponent’s recent history. Once the clustering has produced workable stereotypes in the off-line phase, it will be very efficient to compare an opponents current behaviour to these stereotypes on-line. Three possibilities that are being considered for this matching process are: Finding the stereotype which matches the last X number of actions and predict behaviour based on the closest match; Use all of the opponent’s previous actions in this decision, but put a heavier weight on the recent ones; Use one of the above methods but implement a voting or polling mechanism to get a consensus among the stereotypes.

The stereotypes themselves will be implemented as a series of feed-forward, back-propagation Neural Networks. Since the environment is dynamic and opponent’s patterns may only become clear over time, a time-sensitive (Recurrent) Neural net will probably be used. for this process due to the nature of the
domain. Both on-line trading and real-time games are very time sensitive, and a time sensitive Neural net
should be an ideal way to incorporate planning implicitly in the model. (See Klapper-Rybicka et al, 2000;
Mozer, 1993) Classification will be performed in two stages: Off-line clustering used to determine which
samples will be used to train the different Net/Stereotypes; and on-line classification used in determine to
which Net/Stereotype the current user belongs (see Jacobs & Nowlan, 1991 for discussion on gating
networks and expert networks).

Client server architecture (as in Doppelganger) will not be used for the agents to avoid security problems.
Allowing other agents (human or software) to query the agent allows the possibility of unauthorized sources
access the model of the user. Also, all users will be informed prior to testing that their performance will
be modelled. Finally, since the stereotypes are implemented as Dynamic Neural Net any particular user
is not tied to a stereotype until run time. At no time can a user ever be directly tied to previous behaviour.

Russil & Norvig define supervised, unsupervised and reinforcement learning as follows: Supervised learning
occurs when there is a direct connection between action and the success of that action, Unsupervised
learning must learn when there is no indication of what the correct action should be, and reinforcement
learning occurs when the agent’s action is evaluated but the proper action is not given. It is the third case
of reinforcement learning that most closely resembles the maze environment for the on-line modelling
portion. Although there is direct (supervised) feedback on the final action of any game, the correctness of
all actions leading up to that finale are increasingly less clear the farther back in time one goes.

The off-line clustering problem will be the most difficult, and I’m unsure at this point whether I will be able
to achieve my preference of unsupervised learning. Ideally, the algorithm I am proposing would begin with
the set of all samples (of user behaviour) and train a single, large Neural Net/Stereotype. (See figure 1 for
pseudo-code) This Net/Stereotype will be measured against a test set and be the base mark for future
performance. The algorithm would then divide the samples into groups of similar behaviour, and train a
group of distinct Neural Networks. For the divisions at this point, I propose to cluster the samples so as
to maximize information gain. If tests on these new stereotypes perform worse than the original stereotype,
samples are resorted and tested again, perhaps on the attribute with the next best information gain. If the
new Net/Stereotypes improve predictability, the process gets repeated recursively for each of the new
stereotypes. This algorithm continues until some measure of accuracy, min/max number of Net/Stereotypes
and min/max number of samples per Net/Stereotype is achieved. I may even keep the tree-like structure
as the Neural nets divide themselves, with the most general net at the base of the tree and the most specific
at the leaves.

Figure 1. Clustering pseudo code
1) Create a neural network with available samples.
2) test and record accuracy against a sample set.
3) divide samples by the attribute which maximizes information gain.
4) train new neural networks based on the clusters in 3
5) Test each new Network against the sample set.
6) Does the average accuracy of the new Networks exceed that of the single network?
   Yes: Start at step 1 with each of the new networks (recursively)
   No: Repeat from step 3 with the next best information gain.

:Recursion will stop when: a maximum recursive depth is reached, a minimum number of samples is reached, or division of neural nets no longer brings about a performance increase (Another alternative is to stop when information gain drops to ~0).

The final classification will be a simple process of comparing the users recent performance to each of the stereotypes in an attempt to determine which is the closest fit. In order to allow for a change of strategy by the user (concept drift), only the X most recent samples from that user will be used to in matching with a Net/Stereotype.

**Testing and results**
Testing will be done in a real-time scenario in which agents using various modelling techniques will be compared. Performance of dynamic stereotypes will be compared to that of deterministic finite state automatons (DFA’s from MacInnes, Banyasad, and Upal, 2001) as well as no modelling at all. Other variables that may be tested to determine their effectiveness include recursive modelling and adding eye tracking data as input to the neural networks.

**Anticipated contribution to the field**
The primary contribution for this work will be in the field of user and opponent modelling. A great deal of work has been done in an attempt to understand people’s behaviour so as to be better able to predict their actions. No where is this more noticeable (or difficult) than in the in the area of opponent modelling. Games, Game theory, Combat and on-line trading are all arenas in which opponent modelling has been used to try maximize performance against the adversary. Even a small performance increase in many of these areas can be critical.

Although the idea of stereotypes in this domain is nearly as old as user modelling itself, applying machine learning techniques to develop these stereotypes is relatively new. Excellent examples have surfaced recently (Orwant, 1996) but all have stopped short of total dynamic development of these stereotypes. Usually the categories or the numbers of stereotypes themselves remain fixed while the behaviour of those stereotypes is learned. My algorithm should be able to, not only learn behaviour, but learn the stereotypes themselves.

Secondly, I believe that the clustering scheme I am proposing would be a significant contribution to the field of Artificial Intelligence in its own right. The dynamic classification of training samples into stereotypes could apply to many other similar clustering problems. Indeed, Jacobs and Nowlan (1991) have used a similar scheme (although not completely dynamic) for letter classification. The stereotypes themselves, implemented as Recurrent Neural Networks and used to represent temporal patterns in a spatial environment, could also have a wide variety of uses in pathfinding tasks. Finally, this combined with the
tree-like organization of these stereotypes (Similar to Rich, 1979) would be a novel addition to the AI literature.
**Definitions**

Agent - An agent is an autonomous entity which interacts in its environment through its sensors (input) and effectors (output).

Recurrent Neural Network - A neural network which is sensitive to patterns over time.

Bayesian Network - A probabilistic learning method which combines prior knowledge with observed data.

Information Gain - The amount of knowledge to be gained by knowing the value of a given attribute. The more random the attribute, the higher the information gain.

Overfitting - Creating a model of the training data that fits so closely that it is less capable of modelling other data.

Markov Model - A description of a process as a series of states. The probability of transition from one state to any other depends only on the current state (no memory).

Deterministic Finite State Automaton (DFA) - An state machine (agent) in which transitions are determined only by the current state (no probabilities).
References


Laird, J. It Knows What You’re Going To Do: Adding Anticipation to a Quakebot. Presented at the AAAI Spring Symposium on Artificial Intelligence and Interactive Entertainment, March 2000.

MacInnes, J; Banyasad, O; and Upal, A. Watching You, Watching Me... *Abstracts for the Xth annual conference of Artificial Intelligence*.


