Machine intelligence: Quick reference

vishvAs vAsuki

February 28, 2012

Contents

Based on [1], and a course by Shlomo Zilberstein.

Part I

Introduction

1 Themes

Dealing with NP hard problems and uncertainty.

1.1 Tasks which may befall the intelligent agent

Framing goals. Planning actions to achieve goals; searching. Storing knowledge. Inference. Learning.
Building mechanical robots. Making actuators. Manipulation. Vision and sensing; feature extraction.
Interacting with humans. Safety.

1.2 Characterization of research effort

Experiment on real data: observe flaws of algorithms; improve performance by tweaking them.

2 The intelligent agent

2.1 Interaction model

An abstract agent can be thought of having actuators (to perform actions) and sensors (to gain percepts).
Feedback to the agent may be immediate or deferred.

2.2 Environment traits

The environment an agent operates in may be characterized based on its observability, stochasticity, episodes, static/ dynamic, discrete / continuous, adverseries.

2.3 Rationality

Rationality, bounded rationality, satisficing. Performance measure. [**Incomplete**]

2.3.1 Attitude towards risk

See decision theory in statstics ref, game theory ref to observe risk evaluation. But, an agent may not always go for the option with the least expected risk, but for the option which yields some low risk whp. This is risk averseness, observed in humans.

2.4 Models of rational agents

Simple reflex agents with condition/ action rules. Such an agent is essentially procedural.
Model based agents maintain an internal state (which depends on the actual state of the environment). The agent then does declarative reasoning to determine goals and actions.

2.4.1 Performance measures

Goals, utility function. Learning agents.
Exploration vs exploitation tradeoff. [**Incomplete**]

Part II

Basic procedures

3 Search graph for goal node

See graph theory ref.

4 Constraint satisfaction problems

Discrete vs continuous valued variables. Number of variables involved in constraints; preferences. Problem structure: the constraint graph.

4.1 As an uninformed search problem

The naive backtracking search. Variable ordering: Most constrained variable / minimum remaining values.
Value ordering: least constraining value. Solve independent subproblems separately; collapse nodes in constraint graph. Propogating information through constraints: forward checking: delete unsuitable values in current node; constraint propogation: delete unsuitable values in lower nodes: arc consistency. Intelligent backtracking.

4.2 Local search for CSPs

Eg: n-queens. Effective for overconstrained problems?.

5 Adverserial search / games

The game tree: max and min levels. Utility functions and evaluation functions to approximate utility. Labelling the nodes with utilities.

5.1 Minmax algorithm

$\alpha\beta$ pruning : visit nodes only to decide the best move. Games against nature: Expectiminimax algorithm.
The horizon effect: exploring some branches more deeply.

6 Inference

See Inference survey.

7 Stochastic control processes

7.1 Reinforcement learning setting

7.1.1 Interaction with the environment

The agent can be in a set of states S, and can perform a set of actions A.
The effect of an action is given by transition probability distributions associated with each (sate, action) pair $P_{s,a} : S \to [0, 1]$, and by rewards which may sometimes be provided by the environment.

7.1.2 Policy learning

Agent wants to find the optimum policy $\pi' : S \to A$ by trial and error

7.1.2.1 Sub-problems

Looking at occasional reward, agent must assign credit to past actions: credit assignment problem.
Exploration vs exploitation trade-off: can't find high reward states if you try to greedily known best state.

7.1.3 Reward learning

Aka Inverse reinforcement learning problem.

Sometimes, given the description of the environment (states S, actions A, transition distributions $P_{s,a}$) and an agent's policy π, one wants to learn the rewards which motivated the agent to arrive at the policy.

7.1.3.1 Non triviality criteria

A trivial solution would be a reward function which assigns equal reward to all (a, s_1, s_2) triplets - this would make all policies equally desirable. So, one would seek a sort of maximum entropy solution which favors the observed policy as strongly as possible over other policies.

7.1.3.2 Motivation

In economics and biology, the precise mechanism that motivates an animal is interesting. Eg: Are waiters who tend to seat customers outside, do so in order to attract other customers? [**Incomplete**]

7.1.4 Reinforcement learning in Animals

Stimulus $\equiv$ belief state; response $\equiv$ action. An inspiration for devoloping reinforcement learning algs.

7.1.4.1 Conditioning

Eg: Pavlov's dog; behaviorist Skinner training dog to jump against wall in 20 minutes. 1st order vs higher order conditioning. Cats escaping a box to get to fish.
Acquisition due to reinforcement during various trials; Extinction due to removal/ change in R(q): Visualize with a salivation level vs number of trials graph. Spaced trials better for acquisition than massed trials: more time for animal to make correlations.
Habituation: Plateau in response level.
Extinction burst: When you stop reward, animal temporarily tries much harder.
Avoidance: Animal avoids certain states/ stimuli after -ve reward, thereby looses chance to sample/ explore the state further.
Reinforcement schedules: Fixed reward: performance ratio, fixed interval (not good: animal learns interval) etc..
Conditioning due to different reinforcers (food, water etc..).

7.1.5 Learning by simulation

Suppose you wanted to create an intelligent automaton - eg: pilot program for a drone plane. It is very hard to think of all possible cases - a pilot's knowledge is often intuitive and trained by the environment. On the other hand, it could be simpler to specify and thus simulate the environment. Thus, one can train the automaton in the simulated environment.

7.2 Markov decision process (MDP)

7.2.1 Abstract problem

We consider the reinforcement learning problem, but we consider an immediate reward function $R : A \times S \times S \to \Re$. So the reward is fixed for each (a, s_1, s_2) - this can be interpreted as expected reward (which may be determined by an agent from past experience).

7.2.1.1 Limited dependence on history

State transition shows the Markovian property: dependence only on the previous state and action.

7.2.1.2 Representation, notation

For a Diagrammatic representation and summary of alternative terms/ symbols used, see Wiki.
Essentially, one considers the state transition graph of a Markov chain whose states are given by $S \cup A$, where any path between two state-nodes always needs to pass through an action-node (and similarly for paths between action nodes).
The extension is that edges $(a, s) \in A \times S$ may be labelled also with a reward value, apart from the transition probability. Edges $(s, a) \in S \times A$ are labelled with transition probabilities according to the policy being depicted.

7.2.2 Policy

A policy is a map $p : S \to A$.

7.2.2.1 Long term reward

One can compare policies by somehow aggregating rewards one would expect over k (aka horizon) time-steps. This is better than a simplistic evaluation of a policy which only considers the immediate expected reward. The horizon considered could either be finite or infinite.
It is logical for the weights corresponding to distant times to be smaller.

7.2.2.2 Weighted sum: state values

One common way of aggregating reward over k time-steps while assigning lower weights to future rewards is to take a weighted sum of rewards accrued during this period. Often geometrically decreasing weights - powers g^k of the discounting factor $g \in (0, 1)$ are used.
This expected reward for applying policy p over horizon k starting from state s_0, aka value of the state s_0, is $V_p(s_0) = \sum_{t \in \{0..k\}} g^t E_P[R(p(s_t), s_t, s_{t+1})]$. This is proportional to the geometrically weighted moving average, and hence equivalent to using it where comparing policies is concerned.

7.2.2.3 Best policy

One ideally wants to find a policy which maximizes the value of the current state.
For a finite state MDP with an infinite horizon where rewards are aggregated using geometrically decreasing weights, one only needs to solve the following linear program to find the policy vector p:

$$\arg\max_{p(s)} \sum_{s'} P(p(s), s, s')(R(p(s), s, s') + gV(s'))$$

subject to

$$V(s) = \sum_{s'} P(p(s), s, s')(R(p(s), s, s') + gV(s'))$$

(which is aka the Bellman equation).

7.2.2.4 Solution techniques

Besides linear programming, one can use various iterative dynamic programming techniques.
The policy iteration starts with an arbitrary vectors p. It then does the following repeatedly until convergence: a] compute corresponding value vector V by repeatedly computing $V_p(s_0) = \sum_{t\in\{0..k\}} g^t E_P[R(p(s_t), s_t, s_{t+1})]$ until convergence, b] improve p using current estimate of V. As each step improves the solution, and because there is only one minimum, this is guaranteed to converge.

7.2.3 Dealing with large state spaces

Dealing with large state-spaces. One can deal with large state spaces by collapsing similar state together. Eg: In the case where the state space corresponds to the 3-dimensional coordinates of an aircraft relative to a target, one can use an alternative state space defined instead by distance to the target.
One can do Q-learning: There is no need to specify explicitly the transition probability P or list the states. This is advantageous when the state space is huge.

7.3 Partially observable Markov decision process (POMDP)

7.3.1 Problem setting

As in case of MDP's, we have sets of states S, actions A and transition probabilities P.

7.3.1.1 Observations

But, the states are not necessarily fully observable. So, we have a set of observations O, and observation probabilities P_O.

7.3.1.2 Simplified rewards

The reward function is simpler than in MDP (possibly at the expense of a bigger state space): $R : A \times S \to \Re$.

7.3.2 Belief states

[**Incomplete**]

8 Planning

8.1 Specifying decision algorithm

An agent may have several (sub)objectives it can act towards.
Eg: An navigation agent that wants to know the user's destination may have to decide between asking for a nearby landmark, or for the street name or for a clarification of prior utterence.

8.1.1 Rule based vs utility computation

So, the agent needs some way of prioritizing various sub-objectives/ actions. It may do this using some rigid rules.
An alternative technique computes utilities - various actions are mapped to integers, and the action with the highest utility is executed. This can take into account probabilities. Eg: a navigation agent very sure of the street may want to assign low utility/ weight to the 'ask street name' action.
Some decision algorithms incorporate both techniques in a hierarchy.

9 Computer vision

9.1 SLAM (Simultaneous localization and mapping) problem

The objective is to localize one's position relative to those objects. There are two important subproblems: a] object or feature detection, b] measurement of position (distance and orientation) relative to a given feature. Solution approaches depend on the available tools - which may vary with situation.

9.1.1 Tools and approaches

Localization using bounced light from lasers is essentially a solved problem; but lasers are expensive. Localization using light detected by passive sensors (eg: visible spectrum cameras) is harder.

9.1.2 Features

Features, for the purpose of vision problems, are actual phyisical locations/ regions (not their image representation). Examples: Corners, edges, spots with different contrasts. Feature detection and extraction are major themes in vision research. Important feature detection toolkits include SIFT and SURF, which include feature detectors like edge detectors and contrast detectors.

9.1.3 Feature detection

Finding features whose detectability is robust to illumination and view-point changes is a challenge. An edge between two parallel surfaces is easy to detect only when they contrast each other. Identification of robust features requires gathering images from multiple view points and illumination conditions. Eg: a robotic arm rotating around a scene with a fixed light source.
This is closely related to the object recognition problem considered elsewhere.

9.1.3.1 Scale invariant feature transform (SIFT)

It includes a feature/ point-of-interest detector, and a feature descriptor. An important feature descriptor used is pixel orientation: the gradients in pixel intensity within a certain patch (say of size 4*4) of pixels, another is location. Having extracted a feature, the algorithm stores it at different scales, enabling mapping to corresponding features in different scaled images.
Features/ points of interest are often visually marked by colored dots on images.

9.1.4 Feature matching

Even when the same features are detected in two images, matching identical features is a problem. A visual tool for comparison involves placing two images next to each other and drawing lines connecting features thought to be identical. An important subproblem here is finding good feature descriptors/ representations.

9.2 Visual object recognition

Got set of n photographs (p_i): perhaps of repeated subjects. Want to identify person in photo. Each p_i is a $1 \times m$ pixel vector.

9.2.1 Eigenfaces

One can use the PCA technique to find the the subspace S of greatest variability (p_i). When query image p comes, project it to S, then do $\langle p, p_i \rangle$ to find closest match.

Bibliography

[1] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach (International Edition)*. Pearson US Imports & PHIPEs, November 2002.