Finding Promising Exploration Regions by Weighting Expected Navigation Costs

Mark B. Ring
Research Group for Adaptive Systems
GMD - German National Research Center for Information Technology
Schloß Birlinghoven
D-53 754 Sankt Augustin
Germany

Mark.Ring@gmd.de

Abstract

Active learning is composed of two equally important problems: deciding which areas in state space most warrant further learning, and deciding *how to get* to these areas. Very little effort has been devoted to the latter problem. I will address this area in the symposium as described below.

Tasks with Distance Relationships

In many learning tasks, data-query is neither free nor of constant cost. Often the cost of a query depends on the distance from the current location in state space to the desired query point. This is easiest to visualize in robotics environments where a robot must physically move to a location in order to learn something there. The cost of this learning is the time and effort it takes to reach the new location. Furthermore, this cost is characterized by a *distance relationship*: When the robot moves as directly as possible from a source state to a destination state, the states through which it passes are closer (i.e., cheaper to reach) than is the destination state. Distance relationships hold in many real-world non-robotics tasks also — any environment where states are not immediately accessible. Optimizing the performance of a chemical plant, for example, requires the adjustment of analog controls which have a continuum of intermediate states. Querying possibly optimal regions of state space in these environments is inadvisable if the path to the query point intersects a region of known volatility.

In continuous environments, some first-order approximations to such distance-dependent active learning has been done (Cohn 1994; Linden & Weber 1993; Schmidhuber 1991; Thrun & Möller 1992). In these cases, the learning agent follows a gradient towards promising learning areas by taking the action at each step that maximizes a local ignorance measure. These techniques have no explicit way of balancing the exploration of mildly promising local areas with greatly promising distant areas.

Keeping Track of Navigation Costs

In discrete environments with small numbers of states, it's possible to keep track of precisely where and to what degree learning has already been done sufficiently and where it still needs to be done. It is also possible to keep best known estimates of the distances from each state to each other (see Kaelbling, 1993). Kaelbling's DG-learning algorithm is based on Floyd's all-pairs shortest-path algorithm (Aho, Hopcroft, & Ullman 1983) and is just slightly different from that used here. These "all-goals" algorithms (after Kaelbling) can provide a highly satisfying representation of the distance/benefit tradeoff.

Associated with every state x is a value E_x and a set S_x of N pairs (where N is the number of states):

$$S_{xy} = (D_{xy}, A_{xy}),$$

where E_x is the exploration value of state x (the potential benefit of exploring state x), D_{xy} is the distance to state y, and A_{xy} is the action to take in state x to move most cheaply to state y. This information can be learned incrementally and *completely*: That is, it can be guaranteed that if a path from any state x to any state y is deducible from the state transitions seen so far, then (1) the algorithm will have a non-null entry for S_{xy} (i.e., the algorithm will know a path from x to y), and (2) The current value for D_{xy} will be the best deducible value from all data seen so far.

Weighting Actions by Exploration Benefit and Navigation Costs

With this information, decisions about which areas to explore next can be based on not just the amount to be gained from such exploration but also on the cost of reaching each area together with the benefit of incidental exploration done on the way. Though optimal exploration is NP-hard (i.e., it's at least as difficult as TSP) good approximations are easily computable. One such good approximation is to take the action at each state that leads in the direction of greatest accumulated exploration benefit:

`For each action` a `in state` x

$$w_a = \sum_{\{y|a=A_{xy}\}} f(D_{xy}, E_y)$$

Every action gets a weight, w_a, and the one with the greatest weight *will* be taken next (only in the case of a tie is an action chosen stochastically). Different functions f can be used to balance small local benefit with large but distant benefit in different ways. And, of course, even when all nearby regions are fully explored, the agent will automatically take the least-cost path to distant regions of highest potential exploration benefit.

The E values can also be modified to explicitly balance exploitation with exploration. That is, they can be directly manipulated to represent the goodness of being in a particular state or set of states, if, for example, exploration is not the only goal of the system.

Full Exploration Guaranteed

The following property can be proven for any deterministic environment, regardless of the actions available:

> If there are any *reachable* states that have not yet been fully explored, then there is a known path from the current state to a state with unexplored actions.

More formally, define the following sets: R, K, and U. R is the set of all states reachable from the current state. K is the set of all states to which a path from the current state is known ($K \subset R$). U is the set of all underexplored states (i.e., states in which there is at least one action that has not yet been tried).

Theorem 1 $[\exists x : x \in U \cap R] \rightarrow [\exists y : y \in U \cap K]$.

Proof: Since x is reachable, there is a path from the current state to x. The first underexplored state y encountered on this path (i.e., $y \in U$) will be x if no earlier such state is encountered. All states on the path before y are fully explored. Therefore, all actions on the path from the current state to y have been tried. Since all actions from the current state to y have been tried, a path from the current state to y is deducible and therefore (from above) such a path is also known, i.e., $y \in K$.

As a result, the algorithm will always know a path to an underexplored state until all reachable states have been explored.

Some Results

When implemented in a two-dimensional grid world with random obstacles, the benefits of this approach become evident. In these environments there are four possible actions in each state, and each state is accessible from every other. A lower bound on the number of actions needed to fully explore these environments is NA (where A is the number of actions), though this can only be achieved if the topology of the environment is known in advance (or by great luck). Random exploration performs particularly poorly. With about 85 states in a 10x10 grid (i.e., about 15% of the grid is occupied by obstacles), about 4200 actions are required on average before all state/action pairs have been tried, or roughly 1100% above the theoretical optimum (which is 340 in this environment). Better performance is achieved when actions are locally intelligent: untested actions in a state are tried before already-tested actions are retried. This yields about 2200 actions to fully explore 85 states, or about 550% above the theoretical optimum. In contrast, the method described above results in approximately 410 actions in an environment of 85 states when $f(x, y) = y/x^4$. This is about 20% above the optimum. Of course, the performance of the two random approaches scales miserably. The intelligent approach scales extremely well: even with 4500 states, the number of actions taken was still less than 20% above the optimum.

The principle disadvantage of the method is $O(N^2)$ storage. The amount of storage required for stochastic environments is even higher. This limits the number of states to about 5-10K for realistic hardware configurations. Continuous environments having an infinite number of states are, of course, well beyond the capabilities of this method. Techniques that finesse these storage requirements are currently under investigation.

References

Aho, A. V.; Hopcroft, J. E.; and Ullman, J. D. 1983. *Data Structures and Algorithms*. Addison-Wesley Series in Computer Science and Information Processing. Addison-Wesley.

Cohn, D. 1994. Neural network exploration using optimal experiment design. In Cowan, J. D.; Tesauro, G.; and Alspector, J., eds., *Advances in Neural Information Processing Systems 6*, 679–686. San Mateo, California: Morgan Kaufmann Publishers.

Kaelbling, L. P. 1993. Learning to achieve goals. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1094–1098. Chambéry, France: Morgan Kaufmann.

Linden, A., and Weber, F. 1993. Implementing inner drive through competence reflection. In Meyer, J. A.; Roitblat, H.; and Wilson, S., eds., *From Animals to Animats 2: Proceedings of the Second International Conference on Simulation of Adaptive Behavior*, 321–326. MIT Press.

Schmidhuber, J. 1991. Adaptive confidence and adaptive curiosity. Technical Report FKI–149–91 (revised), Technische Universität München, Institut fü Informatik.

Thrun, S. B., and Möller, K. 1992. Active exploration in dynamic environments. In Moody, J. E.; Hanson, S. J.; and Lippman, R. P., eds., *Advances in Neural Information Processing Systems 4*, 531–538. San Mateo, California: Morgan Kaufmann Publishers.