Today's Lecture

Not in text

-
- Learning (wrap-up)
  - Reinforcement learning
  - Unsupervised learning
    - Kohonen nets
- Planning

DAA pp. 231

Reinforcement Learning

- So far, we had a well-defined set of training examples.

What if feedback is not so clear?
E.g., when playing a game, only after many actions final result: win, loss, or draw.

- In general, agent exploring environment, delayed feedback: survival or not ... (evolution)
- Recall the hexapawn game/computer?

- Issue: **delayed rewards / feedback**.
  - Exacerbates the credit assignment problem.

    ASK NOW if you don't recall what this is!

- Field: **reinforcement learning**

- Main success: Tesauro's backgammon player (TD Gammon).
  start from random play; thousands of games
  world-level performance (changed nature of game itself).

Illustration

Imagine agent wandering around in environment.

- How does it learn **utility** values of each state?
- (i.e., what are good / bad states? avoid bad ones...)

Reinforcement learning
attempts to answer this question.

- Compare: in backgammon game: states = boards. only clear feedback in final states (win/loss).
  (We will assume, for now, that we haven't cooked up a good heuristic evaluation function.)

We want to know utility of the other states

- Intuitively: utility = chance of winning.
- At first, we only know this for the end states.
  - Reinforcement learning: computes for intermediate states.
  - Play by moving to maximum utility states!
- back to simplified world ...

Policy

- Key things we want to learn is a **policy**
- Jargon for the "table" that associates actions with states.
  - It can be non-deterministic.
- Learning the expected utility of a specific action in a specific state is known as ***Q-learning*****.**

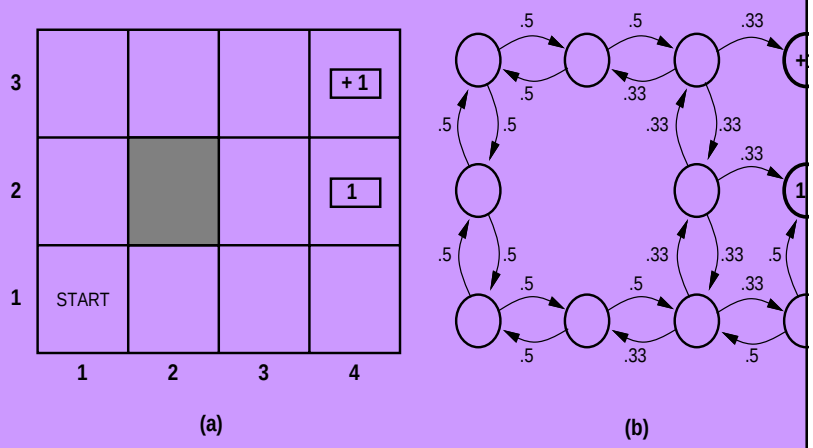
How: strategies

- Three strategies:
- (a) ``Sampling'' (Naive updating)
- (b) ``Calculation'' / ``Equation solving''
  - (Adaptive dynamic programming)
- (c) ``in between (a) and (b)''
  (Temporal Difference Learning --- TD learning)
  used for backgammon

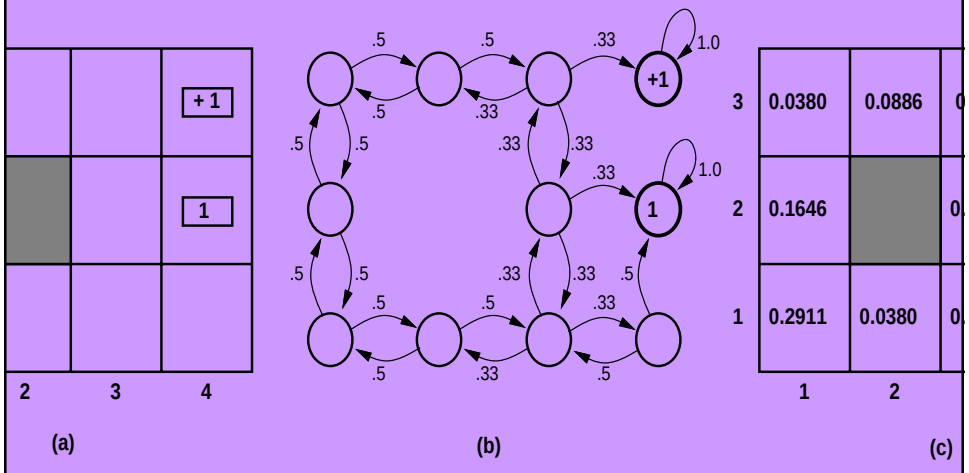
Naive updating

- Comes from **adaptive control theory.**
- (a) ``Sampling" --- agent makes random runs through environment; collect statistics on final payoff for each state (e.g. when at (2,3), how often do you reach +1 vs. -1?)
- Learning algorithm keeps a running average for each state. Provably converges to true expected values (utilities).

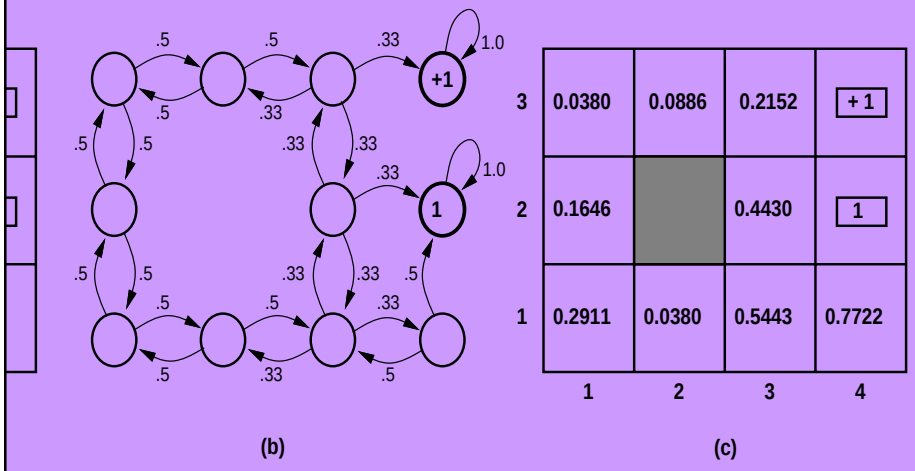
Example: Reinforcement



Stochastic transition network



Stochastic transition network



Issues

- Main drawback: slow convergence.

  See next figure.
- In relatively small world takes agent
- over 1000 sequences to get a reasonably small (< 0.1) root-mean-square error compared with true expected values.

- Question: Is sampling necessary?
- Can we do something completely different? Note: agent knows the environment (i.e. probability of state transitions) and final rewards.

Temporal difference learning

- Combine ``sampling" with ``calculation'' or stated differently: it's using a sampling approach to solve the set of equations.

- Consider the transitions, observed by a wandering agent.

- Use an observed transition to adjust the utilities of the obeserved states to bring them closer to the constraint equations.

Temporal difference learning

- U(i) : utility in state i
- R(i): Reward
- When observing a transition from i to j bring U(i) value closer to that of U(j)
- Use update rule:
- U(i) = U(i) + k(R(i) + U(j) - U(i))**

- k is the learning rate parameter.
- Rule is called the temporal-difference or TD equation. (note we take the difference between successive states.)

Learning by exploration

- Issue raised in text: What if we need to (somehow) move in state space to acquire data.
- *Only then*, can we learn.

We have 3 coupled problems:

- 1. Where are we (in state space)
- 2. Where should we go next
- 3. How do we get there

  Skipping #2 & #3 gives "passive learning" as opposed to "active learning"

- The learning itself "follows" all this.
  - Implict in this is, how do we represent the states, and how to we learn **robustly.**

Mobile Robotics Research

- Mobile Robotics exemplifies these issues.
- The canonical problems:

Where am I
(position estimation)

How do I get there
(path planning)

Mapping and exploration
How do I explain what I saw, and
describe what I need to remember

Mobile Robotics Research

Where am I → `Pose estimation: vision, sonar, GPS`
(position estimation)

How do I get there → `Dynamic programming, A*`
(path planning)

Mapping and exploration → `Geometry, occupancy grids, graphs, uncertainty`
(occupancy and uncertainty)

Unsupervised learning

- Given a set of input examples, can we categorize them into meaningful groups?
- This appears to be a common aspect of human intelligence.
  - A precursor to recognition, hypothesis formation, etc.
- Several mechanisms can be used.
  - The tapper illustrated an essentially statistical approach
    - Supervised?
  - Nice methods based on Bayes rule

Unsupervised learning by cluster

Not in text

Two important ideas:

1. Self-organization: let the learning system re-arrange it's topology or coverage of the function space.
   We saw some of this in the tapper
   We some some of this with backprop.
2. Competitive learning: aspects of the representation **compete** to see which gives the best explanation.
   This is related to **credit assignment** which we saw with GA's.

These are exemplified in an unsupervised context in Kohonen nets.

Kohonen nets

Not in text

- AKA Kohonen feature maps

- A graph **G** that encodes learning
  - Think of vertices as embedded in a metric space, eg. The plane.
  - Inputs represented as points in an N-dimensional space
    - Based on number of degrees of freedom (number of parameters)
- Every node connected to all inputs
- One weight on every edge

Objective

- Want the nodes of our graph to distribute themselves to "explain" the data.
  - Nodes that are close to each other are associated with related examples.
  - Nodes that are distant from one another are related to unrelated examples.

- This leads to an **associative memory**
  - A storage device that allows us to find items that are related to one another
    (What does "related" mean??)

  This is the supreme attribute (and advantage) or human memory.

Kohonen: intuition

- Two key ideas:

1. Let nodes represent points in "example space".
   - Let them learn away of covering/explaining data.

2. Let nodes cover the space "suitably".
   Using both competition with each other and cooperation (recruitment).

Kohonen learning

We are given a set of nodes $\{n_i\}$ and a set of examples $\{s_i\}$.

For each training examples s_i

Compute distance d between s_i and each node n_j descrbied by weights $w_{j,i}$.

$d = \mathrm{sqrt}\ (\ (\ s_i w_{j,i}^T\ )^2\ )$

Find node with min distance

For each node close to n

where close means $D(n,n') < K$

update weights:

$w_{j,i} = w_{j,i} + c(s_i - w_{j,i})$

Make c and K smaller.

Kohonen parameters

Need two functions:

- (Weighted) Distance from nodes to inputs
- Distance between nodes

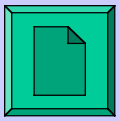
Two constants:

- A threshold *K* on inter-node distance
- A learning rate *c*

- *Also, how fast do c and K change??*

Kohonen example

- Here's the demo you'll use for assignment 4



Kohonen: issues?

- Exploits both distribution of examples **and their frequency** to establish the mapping.
- Accomplishes clustering and recognition.
- May serve as a model of biological

associative memory.

- Can project a high-dimensional space to a lower dimensional comparison space:

dimensionality reduction.

- We had to choose the features in advance.
- *A priori* size and structure of the network.
- What guarantees do we have?

Planning: the idea [DAA:ch 7]

A systematic approach to achieving goals.

Examples:

- Getting home
- Surviving the term
- Fixing your car

Optimizing vs. **Satisficing**.

Planning & search

- Many planning problems can be viewed as search:
  - 1.
    - States are world configurations.
    - Operations are actions that can be taken
    - Find a set of intermediate states from current to goal world configuration.
  - 2.
    - States are plans
    - Operations are changes to the sequence of possible actions.
    - Find a path from the null plan (do nothing) to a plan that gets to the right goal.

Planning differs from search

- Planning has several attributes that distinguish it from basic search.

- **Conditional plans**: plans that explicitly indicate observations that can be made, and actions that depend on those observations.
  - If the front of MC is locked, go to the Chem building.
  - If she takes my rook, then capture the Queen.
- Situated activity: respond to things as they develop.
- Planning with uncertainty: coping with both unknowns and noise.
  - Sometimes this involves specific actions to acquire knowledge as needed.

Planning

Not in text

Like problem solving (search) but with key differences:

- Can decompose a big problem into potentially independent sub-parts that can be solved separately.
- May have incomplete state descriptions at times.
  - E.g. when to tie ones shoes while walking.
  - **Tie-shoes** should be done whenever they are **untied**, without regard to other aspects of the state description.
- Direct connection between actions and components of the current state description
  - State description can be examined and modified
- In robotics specifically (situated problems), there is also the issue **reversibility.** [end of lecture Nov 12]

Planning: general approach

- Use a (restrictive) formal language to describe problems and goals.
  - Why restrictive? More precision and fewer states to search
- Have a goal state specification and an initial state.
- Use a special-purpose **planner** to search for a solution.

Basic formalism

- Basic logical formalism derived from STRIPS.
- State variables determine what actions can or should be taken: in this context they are **conditions**
  - Shoe_untied()
  - Door_open(MC)
- An operator (remember those?) is now a **triple**

  Preconditions

  Additions

  Deletions