Machine Learning, Reinforcement Learning

AI Class 25 (Ch. 21.1, 20.2–20.2.5, 20.3)

Thanks to Tim Finin, Paula Matuszek, Rich Sutton, Andy Barto, and Marie desJardins for the use of their slides

Bookkeeping (Lots)

- No homework 6!
  - Instead, our final "slip" day will review:
    - Homework 6 material
    - If we have time, final exam review
- Grading
  - Phase I: this week
    - But don't wait to start on…
  - Phase II: specifics out by tonight, 11:59pm
- Final Review Time } http://tiny.cc/ExamReviewPoll

Today's Class

- Machine Learning: A quick retrospective
- Reinforcement Learning: What is it?
- Next time:
  - The EM algorithm, EM in RL
  - Monte Carlo and Temporal Difference
- Upcoming classes:
  - EM (more)
  - Applications (Robotics?)
  - Applications (Natural Language?)
  - Review

What Is Machine Learning?

- "Learning denotes changes in a system that ... enable a system to do the same task more efficiently the next time." –Herbert Simon
  - In other words, the end result is a changed model or of some kind; the focus is on the end product
- "Learning is constructing or modifying representations of what is being experienced." –Ryszard Michalski
  - The experiences perceived must be captured or represented in some way; learning modifies that representation. This definition focuses on the process, rather than the result.

4

So what is Machine Learning?

- The "system" is a computer and its programs, or a statistical model with parameters.

-or-

- ML is a way to get a computer to do things without having to explicitly describe what steps to take
  - By giving it **examples** or **feedback**
- It then looks for **patterns** which can explain or predict what happens.
- It is trained through examples.

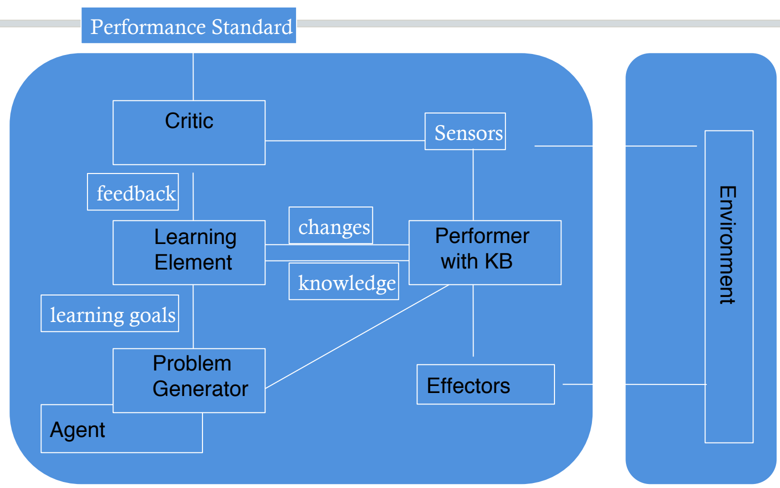
5

The Architecture of a ML System

- Every machine learning system has four parts:
  - A **representation or model** of what is being learned.
  - An **actor**.
    - Uses the representation and actually does something.
  - A **critic.**
    - Provides feedback.
  - A **learner.**
    - Modifies the representation / model, using the feedback.

6

General Model of Learning Agent



Representation

- A learning system must have a representation or model of what is being learned.
- This is the component that changes based on experience.
- In a machine learning system this may be a mathematical model or formula, a set of rules, a decision tree, or some other form of information.
- For classification and clustering we have studied a number of models.

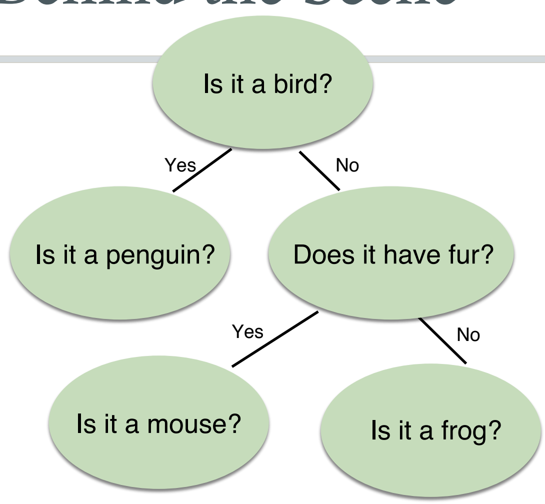
Animals Game

- Human: I'm thinking of an animal.
- Computer: Is it a bird?
- Human: No.
- Computer: Is it a frog?
- Human: No.
- Computer: What is it?
- Human: A mouse.
- Computer: **Tell me a question that distinguishes a frog from a mouse.**
- Human: Does it have fur?
- Computer: What is the answer for a frog?
- Human: No.

9

Animals Behind the Scene

Computer: Is it a bird?
Human: No
Computer: Is it a frog?
Human: No
Computer: What is it?
Human: A mouse
Computer: Tell me a question that distinguishes a frog from a mouse.
Human: Does it have fur?
Computer: What is the answer for a frog?
Human: no



After several rounds…

10

Animals Guessing Game Architecture

- All of the parts of ML Architecture:
  - The Representation is a sequence of questions and pairs of yes/no answers (called a binary decision tree).
  - The Actor "walks" the tree, interacting with a human; at each question it chooses whether to follow the "yes" branch or the "no" branch.
  - The Critic is the human player telling the game whether it has guessed correctly.
  - The Learner elicits new questions and adds questions, guesses and branches to the tree.

11

Reinforcement Learning

- The Animals Game is a simple form of Reinforcement Learning: the feedback is at the end, on a series of actions.
- Very early concept in Artificial Intelligence!
- Arthur Samuels' checker program was a simple reinforcement based learner, initially developed in 1956.
- In 1962 it beat a human checkers master.

www-03.ibm.com/ibm/history/ibm100/us/en/icons/ibm700series/impacts/

Machine Learning So Far

- **Supervised learning**
  - Simplest, most studied type of machine learning
  - But requires training cases
- **Unsupervised learning** uses some measure of similarity as a critic
- Both are **static**
  - All data from which the system will learn already exist
- However!
  - Real-world situations are more complex
  - Rather than a single action or decision, there are a series of decisions to be made
  - Feedback is not available at each step

13

Learning Without a Model

- Last time, we saw how to learn a value function and/or a policy from a transition model
- What if we don't have a transition model??
- Idea #1:
  - Explore the environment for a long time
  - Record all transitions
  - Learn the transition model
  - Apply value iteration/policy iteration
  - Slow and requires a lot of exploration! No intermediate learning!
- Idea #2: Learn a value function (or policy) directly from interactions with the environment, **while exploring**

Reinforcement Learning

- We often have an agent which has a **task** to perform
  - It takes some actions in the world
  - At some later point, gets feedback on how well it did
  - The agent performs the same task repeatedly
- This problem is called **reinforcement learning**:
  - The agent gets positive reinforcement for tasks done well
  - And gets negative reinforcement for tasks done poorly
  - Must somehow figure out which actions to take next time

15

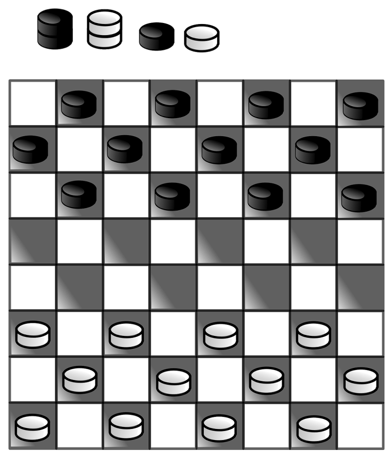
Reinforcement Learning (cont.)

- The goal is to get the agent to act in the world so as to maximize its rewards
- The agent has to figure out **what it did** that made it get that reward/punishment
  - This is known as the credit assignment problem
- Reinforcement learning approaches can be used to train computers to do many tasks
  - Backgammon and chess playing
  - Job shop scheduling
  - Controlling robot limbs

16

Simple Example

- Learn to play checkers
  - Two-person game
  - 8x8 boards, 12 checkers/ side
  - relatively simple set of rules: http://www.darkfish.com/checkers/rules.html
  - Goal is to eliminate all your opponent's pieces



https://pixabay.com/en/checker-board-black-game-pattern-29911

Representing Checkers

- First we need to represent the game
- To completely describe one step in the game you need
  - A representation of the game board.
  - A representation of the current pieces
  - A variable which indicates whose turn it is
  - A variable which tells you which side is "black"
- There is no history needed
- A look at the current board setup gives you a complete picture of the state of the game

which makes it a ____ problem?

Representing Rules

- Second, we need to represent the rules
- Represented as a **set of allowable moves** given board state
  - If a checker is at row x, column y, and row x+1 column y+-1 is empty, it can move there
  - If a checker is at (x,y) , a checker of the opposite color is at (x+1, y +1), and (x+2,y+2) is empty, the checker must move there, and remove the "jumped" checker from play.
- There are additional rules, but all can be expressed in terms of the state of the board and the checkers.
- Each rule includes the outcome of the relevant action in terms of the state.

19

A More Complex Example

- Consider a driving agent, which must learn to drive a car
- State?
- Possible actions?
- Reward value?

20

Formalization for Agent

- Given:
  - A state space S
  - A set of actions $a_1, \ldots, a_k$ including their results
  - Reward value at the end of each trial (series of action) (may be positive or negative)
- Output:
  - A **mapping from states to actions**
  - Which is a…

    policy, π

Reactive Agent

- This kind of agent is a reactive agent
- The general algorithm for a reactive agent is:
  - Observe some state
  - If it is a terminal state, stop
  - Otherwise choose an action from the actions possible in that state
  - Perform the action
  - Recur.

What Do We Want to Learn

- Given
  - A description of some state of the game
  - A list of the moves allowed by the rules
  - **What move should we make?**
- Typically more than one move is possible
  - Need strategies or heuristics or hints about what move to make
  - **This is what we are learning**
- What we have to **learn from** is whether the game was won or lost

23

Simple Checkers Learning

- We can represent a number of heuristics or rules-of-thumb in the same formalism as we have used for the board and the rules
  - If there is a legal move that will create a king, take it.
    - If checkers at (7,y) and (8,y-1) or (8,y+1) is free, move there.
  - If there are two legal moves, choose the one that moves a checker farther toward the top row
    - If checker(x,y) and checker(p,q) can both move, and x>p, move checker(x,y).
  - Each of these heuristics also needs some kind of priority or **weight**

24

Formalization for Agent

- Given:
  - A state space S
  - A set of actions $a_1, \ldots, a_k$ including their results
  - A set of heuristics for resolving conflict among actions
  - Reward value at the end of each trial (series of action) (may be positive or negative)
- Output:
  - A policy (a mapping from states to preferred actions)

25

Learning Agent

- This kind of agent is a simple learning agent
- The general algorithm for a learning agent is:
  - Observe some state
  - If it is a terminal state
    - stop
    - If a win, increase the weight on all heuristics used
    - If a lose, decrease the weight on all heuristics used
  - Otherwise choose an action from the actions possible in that state, using the heuristics to select the preferred action
  - Perform the action
  - Recur.

26

Policy

- A policy is a complete mapping from states to actions
  - There must be an action for each state
  - There may be more than one action
- A policy is not necessarily optimal
- The goal of a learning agent is to tune the policy so that the preferred action is optimal, or at least good.
  - analogous to training a classifier
- Checkers
  - Trained policy includes all legal actions
  - with a weight for preferred actions

27

Approaches

- Learn policy directly– function mapping from states to actions
  - This function could be directly learned values
    - Value of state which removes last opponent checker is +1.
  - Or a heuristic function which has itself been trained
- Learn utility values for states (value function)
  - Estimate the value for each state
  - Checkers:
    - How happy am I with this state that turns a man into a king?

28

Value Function

- The agent knows what state it is in
- The agent has a number of actions it can perform in each state.
- Initially, it doesn't know the value of any of the states
- If the outcome of performing an action at a state is deterministic, then the agent can update the utility value U() of states:
  - U(oldstate) = reward + U(newstate)
- The agent learns the utility values of states as it works its way through the state space

29

Learning States and Actions

- A typical approach is:
- At state S choose some action A
- Taking us to new State S1.
  - If S1 has a positive value, increase value of A at S.
  - If S1 has a negative value, decrease value of A at S.
  - If S1 is new initial value is unknown. Leave value at A unchanged.
- Repeat until? Convergence?
- One complete learning pass or **trial** eventually gets to a terminal, deterministic state. (win or lose)

30

Selecting an Action

- Simply choose action with highest (current) expected utility?
- Problem: each action has two effects
  - Yields a reward (or penalty) on current sequence
  - Information is received and used in learning for future sequences
- Trade-off: immediate good for long-term well-being
  - Like trying a shortcut: might get lost, might learn a quicker route.

31

Exploration vs. Exploitation

- Problem with naive reinforcement learning:
  - What action to take?
  - **Best apparent action, based on learning to date** } Exploitation
    - Greedy strategy
    - Often prematurely converges to a suboptimal policy!
  - **Random (or unknown) action** } Exploration
    - Will cover entire state space
    - Very expensive and slow to learn!
    - When to stop being random?
  - Balance exploration (try random actions) with exploitation (use best action so far)

More on Exploration

- The agent may sometimes choose to explore suboptimal moves in the hopes of finding better outcomes
  - Only by visiting all the states frequently enough can we guarantee learning the true values of all the states
- When the agent is **learning**, ideal would be to get accurate values for all states
  - Even though that may mean getting a negative outcome
- When agent is **performing**, ideal would be to get optimal outcome.
- A learning agent should have an exploration policy

33

Exploration policy

- Wacky approach (exploration): act randomly in hopes of eventually exploring entire environment
  - Choose any legal checkers move
- Greedy approach (exploitation): act to maximize utility using current estimate
  - Choose moves that have in the past led to wins
- Reasonable balance: act more wacky (exploratory) when agent has little idea of environment; more greedy when the model is close to correct
  - Suppose you know no checkers strategy? What's the best way to get better?

34

Example: N-Armed Bandits

- A row of slot machines
- Which to play and how often?
- State Space is a set of machines
  - Each has cost, payout, and percentage values
- Action is pull a lever.
- Each action has a positive or negative result
  - …which then adjusts the utility of that action (pulling that lever)

N-Armed Bandits Example

- Each action starts with a standard payout.
- Result is either some cash (a win) or none (a lose)
- Initially we don't know anything about which
- Exploration:
  - Try things until we get some estimates for the payouts.
- Exploitation:
  - When we have some idea of the values of each action, choose the best.
- Clearly this is heuristic. May not find the best lever to pull!
  - The more exploration we can do the better our model
  - But the higher the cost over multiple trials

Reinforcement Learning

- **Reinforcement learning systems**
  - Learn **series** of actions or decisions, rather than a single decision
  - Based on feedback given at the end of the series
- A reinforcement learner has
  - A goal
  - Carries out trial-and-error search
  - Finds the best paths toward that goal

Reinforcement Learning

- A typical reinforcement learning system is an active agent, interacting with its environment.
- It must balance
  - Exploration: trying different actions and sequences of actions to discover which ones work best
  - Exploitation (achievement): using sequences which have worked well so far
- Must learn **successful sequences of actions** in an uncertain environment

RL Summary

- Active area of research
- There are **many** more sophisticated algorithms
  - Most notably: probabilistic approaches
- Applicable to game-playing, robot controllers, others

39