Artificial Intelligence Programming

Agents

Cindi Thompson

Department of Computer Science
University of San Francisco

Overview

- Brief Python recap
- Domains
- Agents
- Types of Agent Programs
- Environments

Python recap

- Questions from last week?
- . . .
- For HW, do not "import" anything other than: sys, urllib, urllib2, re, pickle, cPickle, time, datetime, heapq, and argparse/getopt.
- Applies to all assignments unless I later announce otherwise
- Cmd line arguments can be handled via sys, getopt, or argparse

Command line args

From HW:

```
### usage:buildGraph --pfile=outfile -d=startNode infile
### if --pfile=outfile is provided, write a pickled version of the graph
### to outfile. Otherwise, print it to standard output.
### if --d=startNode is provided, compute dijkstra w
given starting node
```

Simplest: Use "import sys" and "for arg in sys.argv:"

- First arg is name of the module/script!
- Section 10.6 of Dive Into Python will help if you're stuck

Overview

- Brief Python recap
- Domains
- Agents
- Types of Agent Programs
- Environments

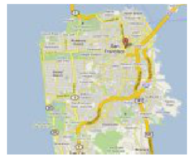
Domains

- A *domain* is a description of a particular problem.
- We'll pay special attention to three domains in this class:
  - Route and logistics planning
  - Text and natural language processing
  - Data mining

Route and Logistics Planning

- Route and logistics planning deals with the problems of moving agents and resources to one or more physical locations.
- Often there are constraints and limits on the problem
  - As fast as possible
  - Only one vehicle on a bridge at a time
  - Package A must be delivered before package B
- A fundamental component of all of these problems is a *map*.

Route and Logistics Planning



- Consider an agent that needs to deliver packages in San Francisco.
- How will we represent this?

Route and Logistics Planning

- We can select prominent or important locations on the map.
- We'll call these *waypoints*
- We can then connect these waypoints with edges indicating distance or time.
- This allows us to treat the map as a graph.

Route and Logistics Planning

- Goal: get rid of all the irrelevant details and nothing more.
  - Too many waypoints and the graph will be unmanageable.
  - Too few waypoints and we will lose important parts of the problem.
- Key: transform a problem we don't know how to work with (a map) into one we *do* know how to work with (a graph).
- We also abstract elements of the environment that we don't want to contend with.

Text & Natural Language Processing

- Intelligently dealing with Web data is a huge AI challenge.
  - Better search
  - Document filtering
  - Classification
  - Machine Translation
  - Recommender systems
  - Sentiment analysis
  - and many more ...

Text and NLP

The 2013 Annual Computer Poker Competition will be held again in 2013, during the month of June. Eric Jackson will be returning as one of the competition chairs, with Neil Burch replacing Jonathan Rubin as the second chair. As in previous years there will be heads-up (two player) limit, three player limit, and heads-up no-limit Texas Hold'em competitions. Rules can be found here.

- How can we abstract this text in a way that lets us work with it programmatically?
- Depends on what we want to do with it.

Text and NLP

The 2013 Annual Computer Poker Competition will be held again in 2013, during the month of June. Eric Jackson will be returning as one of the competition chairs, with Neil Burch replacing Jonathan Rubin as the second chair. As in previous years there will be heads-up (two player) limit, three player limit, and heads-up no-limit Texas Hold'em competitions. Rules can be found here.

- Initial approach: bag of words. Count the number of occurrences of "interesting" words.
- poker: 1, player: 2, competition: 2, rules: 1, ...
- Abstraction: don't worry about ordering, or placement in a document.

Text and NLP

- The bag of words is simple, but highly abstract.
- We can also try to remove "chunks" of text, such as noun phrases.
  - "competition chairs", "poker competition", "Eric Jackson"
- We can also try to deal with *semantics* - what is the meaning of the word "limit"?

Text and NLP

- In processing text, we will think carefully about how much to consider the meaning of words.
- Should we consider documents just as strings of letters, or try to discover abstract meaning?
- Strings are easy to work with and scale well, but have limitations.
- Dealing with meaning is a challenging problem, even for people.

Data Mining

- Suppose that we have lots of medical records from people who've had cancer screening.
- We'd like to know what factors are correlated with breast cancer.
- We'd like to predict whether new patients are at elevated risk of cancer recurrence.
- How can we do this?

Data Mining

- We begin by selecting a number of *attributes* of interest.
  - Age, menopausal, tumor size, degree of severity, etc
- For each attribute, we decide on a domain
  - Should age be a number, or should we group ages into clusters (18-25, 26-30, etc)
  - Should size be a category? How many?
- In making these choices, we create an abstraction of the problem that (hopefully) leaves out irrelevant details.

Data Mining

- We wind up with a data set like:

'40-49','premeno','15-19','0-2','yes','3','right','left_up','no','recurrence-events'
'50-59','ge40','15-19','0-2','no','1','right','central','no','no-recurrence-events'
'50-59','ge40','35-39','0-2','no','2','left','left_low','no','recurrence-events'
'40-49','premeno','35-39','0-2','yes','3','right','left_low','yes','no-recurrence-events'
'40-49','premeno','30-34','3-5','yes','2','left','right_up','no','recurrence-events'

Data Mining

- Once we have this, what next?

Data Mining

- Once we have this, what next?
- We can try to estimate the probabilities of events.
- We can try to discover rules that explain events.
- We can try to group attributes together.
- We can build agents that classify new patients.

Overview

- Brief Python recap
- Domains
- Agents
- Types of Agent Programs
- Environments

Intelligent Agents

- The idea of developing *intelligent agents* has been a unifying one for AI.
  - Previously, subdisciplines were fairly balkanized.
  - Learning, knowledge representation, search, vision, etc.
- Agent may require several of these abilities.

What is an agent?

- There are lots of potential definitions ...
- R & N: An agent is anything that can be viewed as
  - perceiving its environment through sensors and
  - acting on that environment through actuators.
- Mayes: Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so realize a set of goals or tasks for which they are designed.

Qualities of an agent

- Autonomy
- Adaptation
- Goal-directed behavior
- Has “beliefs” and “intentions”
- Proactive
- Situated within an environment
  - Potentially simulated
  - Web is a perfectly fine environment

Autonomy

- Autonomy is often attributed to an agent
- Able to rely on its percepts and past experience to make decisions,
- Avoid asking a "human" for help
- Thorny - might not want complete autonomy
- Challenge is balancing the two extremes (Proactivity)

Beliefs and Intentions

- Belief: representation of "state" of itself and the world
- Intention: what the agent has chosen to do (for the moment)
- Will also have more to say about this later in semester

Agent-oriented Programming

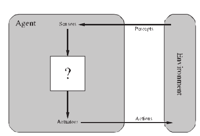
- OOP - Objects are passive
  - Receive messages, return data
  - No agenda of their own
- AOP - Agents are active
  - Perceive (some of) the world
  - Set goals
  - Act in the world

The Agent Metaphor

- Why bother with all of this?
- We already know how to write programs
- Agents are typically more *open-ended*
  - Difficult to specify all cases in advance
  - Instead, write programs that can perform in a wide set of situations
  - Separate the knowledge from the reasoning mechanism

Agents and Environments

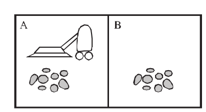
- Shift from "standard" CS: think explicitly about an agent's *environment* and how that affects execution
- Percepts: Information from environment (light, sound)
- Sensors: Mechanisms for gathering data about the environment (eyes, ears, switches,. . .)



- Actuators: Mechanisms for affecting the environment (arms, radios, lights, . . .)
- Actions: Actual changes in behavior or to environment (lifting, flashing lights,. . .)

Example: Vacuum-cleaner World

- Let's start with a very simple approximation.
- Two rooms, A and B. Each room can be either clean or dirty.
- This is the agent's *environment*.



- Percepts: Clean, Dirty
- Sensors: Dirt sensor, location.
- Actuators: Vacuum, wheels.
- Actions: Move left, move right, suck, do nothing.

Agent programs

- How complex do agent programs need to be? Depends on
  - Complexity of the tasks being performed
  - The environment the agent interacts with.
  - How the agent is being evaluated (next slide)
- More complex environments require more care.
- Functional programming paradigm: Map percept (sequences) to actions
- Action = F(current-percept, percept-history)
- The *agent program* implements this function (or some approximation)

Rationality

- Roughly, rationality means “doing the right thing”
- More precision is needed - what is “the right thing”?
- We need a definition of success.
- Begin with a *performance measure*
  - This is a condition or state of the world we’d like the agent to achieve.
  - “Both rooms are clean.” (perhaps more criteria, such as minimizing time, power consumed, or number of actions taken)
  - We might prefer a scalar measure or a boolean condition.

Rationality

- Notice that this is a specification of an *outcome*, rather than how an agent should behave.
- R & N: For each possible percept sequence, a rational agent should select an action that is expected to maximize its performance measure, given the evidence provided by the percept sequence and whatever built-in knowledge the agent has.

Rationality

- “expected” vs. actual. We don’t require that our agent be able to predict the future, or predict unlikely events.
- Information gathering might also be a rational action.
  - Crossing the street without looking is irrational
- Rational agents must be able to *learn* (except in very simple, well-understood environments).
  - Learning is defined as improving an agent’s performance.
  - This could mean reducing uncertainty, or taking observations into account.

Environments

- The simplest possible environments are:
  - Static (nothing changes while the agent is thinking)
  - Deterministic (actions produce a unique result)
  - Discrete (there is a discernable transition between states or inputs)
  - Single-agent (no one else needs to be accounted for)
  - Episodic (Agents do not need to consider the past)
  - Fully observable (everything needed to make a decision can be perceived)

Overview

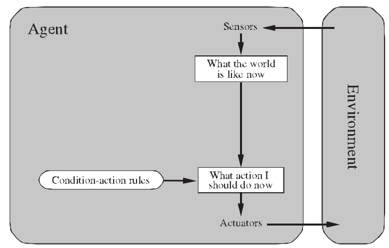
- Brief Python recap
- Domains
- Agents
- Types of Agent Programs
- Environments

Reflex Agents

- In such a simple world, the agent merely needs to observe its environment and then select an action.
  - It might use a look-up table or a set of if-then rules.
- A thermostat is an example of this sort of agent, as is a program that sorts mail.
- We call this sort of agent a *reflex* agent

Reflex agent

- Given the current percept, select an action.
- Ignore history



Reflex agent

- Given the current percept, select an action.

```
class ReflexVacuumAgent(Agent):
    "A reflex agent for the two-state vacuum environment. [Fig. 2.8]

    def program(self,location, status):
        if status == 'Dirty': return 'Suck'
        elif location == loc_A: return 'Right'
        elif location == loc_B: return 'Left'
```

- This agent will only be rational if the best action can be chosen based only on the current percepts.

Reflex agents

Examples?

Reflex agents

Examples

- Thermostat
- Spam-filtering (some)

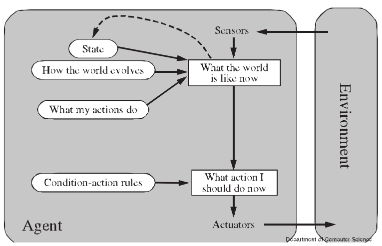
Can represent mapping from percepts to actions as a set of *condition-action* rules

Remembering the past

- What if we need to take a series of actions?
  - Clean all rooms, deliver multiple packages, etc
- We now say that the environment is *sequential* rather than episodic.
- Our agent cannot make a decision based solely on input percepts.
- it must instead construct a *model* of the environment

Model-based agent

- A model-based agent maintains an internal representation of the world.
- Selects actions based on model and current percepts



Model-based agent

- A model-based agent maintains an internal representation of the world.
- We'll refer to this as keeping *state* about the world.
- Actions are selected based on the model and the current percepts.

```
class ModelBasedVacuumAgent(Agent):
    "An agent that keeps track of what locations are clean or dirty."
    def __init__(self):
        model = {loc_A: None, loc_B: None}
    def program(self, location, status):
        "Same as ReflexVacuumAgent, except if everything is clean, do NoOp"
        self.model[location] = status ## Update the model here
        if self.model[loc_A] == self.model[loc_B] == 'Clean': return 'NoOp'
        elif status == 'Dirty': return 'Suck'
        elif location == loc_A: return 'Right'
        elif location == loc_B: return 'Left'
```


Model-based agent

- Maintaining a representation of the environment is extremely useful.
  - Allows the agent to remember things.
  - Can anticipate future events.
  - Can make predictions about unseen parts of the environment.
- Can still uses rules, conditioned on the model and the sensors.
- Much of our time will be spent constructing and manipulating models.

Using models

- A model can also be used to address environments in which an action may have multiple outcomes - this is known as a stochastic environment (as opposed to deterministic)
- Or an environment containing other agents (this is known as multi-agent)
- Or an environment in which not all information can be seen (this is called partially observable)
- Or even a world that changes while the agent is thinking (this is called dynamic)

Types of models

- Attributes and values
- Probability distributions over variables
- Data structures
  - Maps
  - Graphs
  - Finite State Machines
- Facts about the world

Model-based reflex agents

Examples?

Model-based reflex agents

Examples

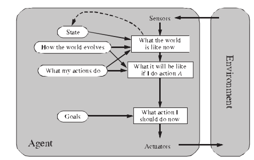
- Vacuum-cleaner agent (with a map)
- Spam-filtering agent (maybe)
- Factory robot

Selecting actions

- In many cases, acting rationally requires more than the current percept sequence and a model.
- Our agent also needs a conception of what it is trying to accomplish.
- This is referred to as a *goal*. Agents that reason specifically about goals are referred to as *goal-based* agents.

Goal-based agent

- A goal-based agent chooses actions that explicitly lead to one or more of its goals.
- The same percept sequence can therefore lead to different actions.
- Search and planning are used to solve this problem.



Goal-based agent

Examples of environments where this would be useful?

Goal-based agent

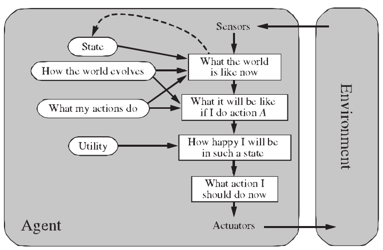
- Goal-based reasoning is very useful for sequential environments.
  - Chess-playing
  - Taxi driving
  - Spaceship piloting
- The right action for a given percept sequence depends upon the agent's knowledge (its model), its current state (percepts) *and what it is trying to achieve currently.*

Next class, we will look at using *search* to accomplish goals

Utility-based agent

- Goals may not be enough in high-complexity environments.
- There may be many action sequences that achieve a goal.
- *Utility* is used to compare the relative desirability of action sequences.
- Maps states to real numbers.
- Can be an estimate of cost, time, or relative value of different outcomes.
- Utilities are very useful in dealing with partially observable or stochastic environments.

Utility-based agent



Utility-based agent

- Utilities assume that a linear ordering can be made over all outcomes.
- Sometimes this is true, sometimes not.
  - Example: Sophie's Choice: which of your children will be killed?
- Utilities are very useful in domains with probability and/or money.
  - Online trading, exploration in uncertain environments, gambling.

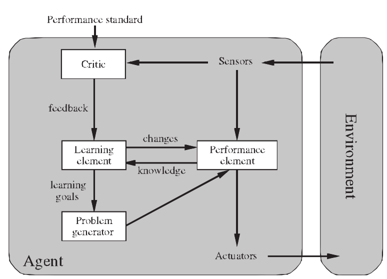
Learning Agent

- Often, an agent may need to update its model or its agent program.
- Programmers may not completely understand the environment.
- The environment may change over time.
- Coding by hand may be tedious.
- A learning agent is one that improves its performance wrt a set of tasks over time.
- Learning (or adaptation) is essential in complex environments.

Learning Agent

- A learning agent needs both a *performance element* and a *learning element*
  - Performance element: Select current action(s)
  - Learning element: evaluate the correctness of the performance element.

Learning Agent



Learning Agent

- Learning can happen *offline* or *online*.
- Learning can be *passive* or *active*.
- Learning can be *supervised* or *unsupervised*.
- *Credit assignment* is a big problem when learning in sequential environments.
- Often, learning is treated as a separate topic in AI; we'll try to integrate it in with other topics.

Summary: Types of Agents

- Table-driven
- Reflex
- Model-based reflex
- Goal-based
- Utility-based
- Learning

Overview

- Brief Python recap
- Domains
- Agents
- Types of Agent Programs
- Environments

Environment Characteristics

- Observability
- Deterministic vs stochastic
- Episodic vs sequential
- Static vs Dynamic
- Discrete vs continuous
- Single versus multi-agent

Representation and Abstraction

- Will need to use care: difference between the world *as it is* versus the world *as it is represented*
- Again, we might need to use *abstraction* to
  - Simplify complex elements of a problem
  - Remove irrelevant details
  - Reduce the size of the problem

Observability

- Observable: if sensors always give complete information about the relevant parts of the environment
- If there is anything the agent needs to know that it cant' sense, then it's only *partially observable*
- Which are these?
  - Chess playing
  - Vacuum cleaner

Deterministic / Stochastic

- Can think of world as going through state transitions
- $CurrentState \times agentActions \rightarrow newState$
- If the transition is unique, environment is deterministic
- Does an action always produce the same result?
- Which are these?
  - Chess playing
  - Vacuum cleaner
  - Driving a car

Episodic vs Sequential

Episodic: each action is independent

- Perceive, decide, act. Repeat
- Next decision does not depend on previous state
- No need to think ahead

Sequential: More than one action to achieve goal

- Past and/or future impact Current decision
- Need to plan

Which are these?

- Chess playing
- Vacuum cleaner
- Driving a car
- Spam filter

Static vs Dynamic

- Static environment "holds still" while agent deliberates
- Static: no time pressure
- Dynamic: environment changes while you think
- Which are these?
  - Spam-filtering
  - Chess playing
  - Driving a car

Semidynamic

There are some "in between" environments

- Environment not changing, but performance measure changes over time
- Examples:
  - Chess playing with a clock
  - Taking a timed test

So pressure to act quickly

Discrete vs Continuous

- From point-of-view of agent's percepts or actions, or in terms of possible states of the environment
- If possible values are discrete, environment is discrete wrt that characteristic
- Otherwise, continuous (continually varying)
- Not same as finite
- Example: spam-filtering is discrete, but (countably) infinite number of possible emails

Discrete vs Continuous

What about?

- Steering angles in a car-driving agent
- Real-valued sensor readings

Time is an element that will come up again

Single vs Multi-agent

Single-agent: agent acting on its own

- World may still be stochastic

Multi-agent: Actions/goals/strategies of *other agents* must be taken into account

Which one?

- Spam-filtering
- Bidding in an auction

Single vs. Multi-agent

Issues

- Recall agents have goals - other agent's goals may conflict
- Even though there are other agents, we may choose to treat it as single agent for simplicity abstraction
- Cooperative vs. Competitive

More examples

- Slot-machine playing
- Robot fetching me coffee
- Mars rover
- Web-crawler
- Siri
- Medical diagnosis

Summary

- An agent is an *autonomous* program situated in an *environment*.
- An agent behaves *rationally* if it acts to optimize its expected *performance measure*.
- Characterizing the environment can help us decide how to build an agent.
- More complex environments often require more sophisticated agent programs.
- A domain is an abstraction of a particular problem. It retains only those aspects essential to solving the problem and discards the rest.

Coming up…

- Get into our first "meaty" topic: search
- How can agents perform goal-directed behavior by searching through a space of possible outcomes
- Uninformed search
- Applying domain knowledge - heuristic search
- Searching enormous spaces: genetic algorithms and simulated annealing