Recall: The Agent Function

We can think of the entire agent, or some portion of it as implementing a function.

- **inputs**: the agent’s internal state and what it perceives
- **outputs**: the agent’s actions

We have been thinking of this as a function in the programming sense.

Let’s now think of it instead as a function in the mathematical sense.

\[ f \text{ (percept, state)} = \text{command} \]
Agent Function Examples

• state space search (example: traffic jam)
  • input = complete model of the state space
  • output = complete plan of action

• game playing, online planning (example: hex)
  • input = current state
  • output = current action

• Offline planning/learning (example: Pacman)
  • input = current state, history
  • output = current action
Machine Learning Approach

Rather than program a function directly, generalize from data.

- Gather example inputs & outputs.
- Find a function that maps between inputs and outputs effectively.
- Test how well that function generalizes to new examples.
Why learning?

Can’t we just program the solution? Or at least just search for it?

• We can’t anticipate all of the possible situations an agent may face.
• We want the agent to adapt to changes in the environment over time.
• We may not know how to solve the problem.
• We may want to model how humans learn.
What function should be learned?

Our first topic will be reinforcement learning where we try to learn the whole agent function.

- Q-learning updates generate a value function.
- The value function implies an optimal policy.
- Once learning is done, the agent function is trivial: for the current state, look up the best action.

AlphaGo learned multiple helper functions:

- An accurate move-probability distribution for use in the tree policy.
- A fast-to-evaluate move-probability distribution for use in the default policy.
- A board-evaluation heuristic.
Smaller units that we could learn.

Instead of learning the whole agent function, we could learn...

• State space representation
  • What features of the world are important for the task?
• Utility function
  • What outcomes are better for the agent?
• State evaluation heuristics
  • What direction seems more promising?

• Other ideas?
What does the data set look like?

• Discrete or continuous?
  • We mostly care about whether the output is continuous.

• Do we know the right answer?
  • supervised
  • semi-supervised
  • Unsupervised

• Do we have all the data in advance?
  • online learning

• How noisy is the data?
Machine Learning Taxonomy

The first step in machine learning is always to identify what kind of data set we have.

Supervised Learning
- For each input, we know the right output.
  - Regression
    - Outputs are continuous.
  - Classification
    - Outputs come from a small discrete set.

Unsupervised Learning
- We just have a bunch of inputs.

Semi-Supervised Learning
- We have inputs, and occasional feedback.
Hypothesis Space

The second step in machine learning is always to define the hypothesis space.

What kind of mapping are we learning?

• What are the inputs?
• What are the outputs?

• What restrictions are there on the input-to-output mapping?
Supervised Learning: Regression

- Input: x values, continuous y values
- Output: simple function from x to y
Supervised Learning: Classification

- Input: x values, discrete labels
- Output: function to label new points
Unsupervised Learning: Clustering

- Input: unlabeled x values
- Output: breakdown into clusters
Unsupervised Learning: Dimensionality Reduction

• Input: unlabeled x values
• Output: lower-dimensional representation of the data
Semi-Supervised Learning: Reinforcement Learning

• Input: states, occasional utilities
• Output: values/policy
Online vs. Offline Learning

In planning, we contrasted online vs. offline.
• Do we do all of our planning in advance?
• Do we interleaver planning and acting?

There is a similar dichotomy in learning.
• Do we have all of the data in advance?
• Do we interleaving learning and predicting?
Evaluating Hypotheses

• To measure the accuracy of a learned function, we use a **test set** of examples that are distinct from the **training set**.

• A hypothesis generalizes well if it correctly predicts the output for the novel examples in the test set.

• We prefer hypotheses that generalize well over ones that perform optimally on the training set.
Which function models the data better?

There is often a tradeoff between complex hypotheses that fit the data better and simpler hypotheses that generalize better.
Generalization

We want our models to make good predictions on new inputs. But we don’t know what those inputs will be.

Key idea: hold out some of the data.

• Prefer models that make good predictions on the held-out data.

• Prefer models that are robust to changing which data they train on.
Data Representation

There are often many ways to represent the same data.

Choosing good representations can dramatically simplify the learning problem.

• Feature selection
  • We don’t always want to use all the attributes we know.

• Change of basis
  • We may want to compute new features that combine
Suppose we want to learn the value function.

The natural input representation for GridWorld would be \((x,y)\) coordinates.

Feature selection might drop the \(y\) coordinate.

A change of basis might compute Euclidean distances to the goal state.
Why would we want to do this?

For the GridWorld example, neither of the new representations I proposed makes much sense.

What about the other examples?