Artificial Intelligence Programming

Intro to Machine Learning

Chris Brooks

Department of Computer Science
University of San Francisco

Introduction

- We've talked about learning previously in the context of specific algorithms.
- Purpose: discuss learning more generally.
- Give a flavor of other approaches to learning
- Talk more carefully about how to evaluate the performance of a learning algorithm.

Defining Learning

- So far, we've defined a learning agent as one that can improve its performance over time.
- We've seen two learning algorithms:
  - Decision tree
  - Bayesian Learning
- Let's define the problem a bit more precisely.

Defining Learning

- A program is said to learn from experiences E with respect to a set of tasks T and a performance measure P if its performance on T, as measured by P, improves with experience E.
- This means that, for a well-formulated learning problem, we need:
  - A set of tasks the agent must perform
  - A way to measure its performance
  - A way to quantify the experience the agent receives

Examples

- Speech recognition
  - Task: successfully recognize spoken words
  - Performance measure: fraction of words correctly recognized
  - Experience: A database of labeled, spoken words
- Learning to drive a car
  - Task: Drive on a public road using vision sensors.
  - Performance: average distance driven without error
  - Experience: sequence of images and reactions from a human driver.
- Learning to play backgammon
  - Task: play backgammon
  - Performance measure: number of games won against humans of the appropriate caliber.
  - Experience: Playing games against itself.

Discussion

- Notice that not all performance measures are the same.
  - In some cases, we want to minimize all errors. In other cases, some sorts of errors can be more easily tolerated than others.
- Also, not all experience is the same.
  - Are examples labeled?
  - Does a learning agent immediately receive a reward after selecting an action?
  - How is experiental data represented? Symbolic? Continuous?
- Also: What is the final product?
  - Do we simply need an agent that performs correctly?
  - Or is it important that we understand *why* the agent performs correctly?

Types of learning problems

- One way to characterize learning problems is by the sorts of data and feedback our agent has access to.
  - batch vs incremental
  - supervised vs unsupervised
  - active vs passive
  - Online vs Offline

Batch vs Incremental

- We can think about problems or algorithms being batch or incremental.
- A *batch* learning algorithm is one in which all of the data is available at once to the agent.
  - Decision trees are a batch learning algorithm.
- An incremental learning algorithm is one that can continue to incorporate new data over time as it becomes available.
  - Naive Bayes can be used incrementally.
- In principle, batch learning is more effective, but it tmay not fit the characteristics of all problems.

Supervised vs Unsupervised

- A *supervised* learning algorithm/problem is one in which the learner has access to labeled training data.
  - Decision trees are an example of this.
- *Unsupervised* algorithms/problems are ones in which no labeled training data is available.
  - The recommender systems used by Amazon and Netflix are examples of this.
- Supervised learning is easier, but it assumes that you have access to labeled training data.

Active vs Passive

- In *active learning*, the learning agent is able to construct examples and find out thier classification.
  - For example, if our spam classifier could create emails and ask a teacher whether it was spam or not.
- In *passive learning*, the learning agent must work with the examples that are presented.
  - Our decision tree problem was a passive learning problem.
- Active learning is more effective, as the agent can choose examples that lets it better "hone" its hypothesis, but may not fit with a particular problem.

Online vs Offline

- An *offline* learing algorithm is able to separate learning from execution.
  - Learning and performance are separate
  - Batch learning is easier, computational complexity is less of a factor.
- An *online* learning algorithm allows an agent to mix learning and execution.
  - Agent takes actions, receives feedback, and updates its performance component.
  - Incremental learning makes more sense, fast algorithms a requirement.
- We will worry about both training time (time needed to construct a hypothesis) and classification time (time needed to classify a new instance).

Other types of learning

- In this class, we'll focus primarily on inductive supervised learning
  - Well-understood, mature, many applications.
- There are other types of learning
  - Deductive learning
  - Unsupervised learning
  - Reinforcement learning

Induction

- Induction is the process of concluding general knowledge from specific facts.
  - On the last ten days that were sunny, we played tennis. Therefore, when it is sunny, we play tennis.
- Allows us to draw general conclusions from data.
- Most machine learning algorithms use induction.

Deductive Learning

- Deductive learning develops rules about specific situations from general principles.
  - "Knowledge-based" learning might be a better name - some induction may take place.
- For example, a deductive learning agent might cache the solution to a previous search problem so that it doesn't need to re-solve the problem.
- It might even try to generalize some of the specifics of the solution to apply to other instances.
  - Case-based reasoning is another example of this style of learning.
  - Start with a number of "cases" or "recipes", and try to fit specific situations to one of these.

Unsupervised Learning

- In *unsupervised learning*, there is no teacher who has presented the learner with labeled examples.
- Instead, all the learner has is data.
- Problem: find a hypotheis (or pattern) that explains the data.

Clustering

- One example of unsupervised learning is ***clustering***
- Given a collection of data, group the data into k clusters, such that similar items are in the same cluster.
- Challenge: don't know the class definitions in advance.
- We will look at techniques for doing this on Tuesday.

Reinforcement Learning

- In some cases, an agent must learn through interaction with the environment.
- Agent selects and executes an action and receives a reward as a result.
- Learning problem: What is the best action to take in a given state?
- Issue - since learning is integrated with execution, we can't just explore every possibility.
- Approach (in a nutshell) - try different actions to see how they do.
- The more confidence we have in our estimate of action values, the more likely we are to take the best-looking action.

Q-learning

- We want to learn a *policy*
  - This is a function that maps states to actions.
- What we get from the environment are state: reward pairs.
- We'll use this to learn a $Q(s,a)$ function that estimates the reward for taking action a in state s.
- This is a form of ***model-free*** learning
  - We do no reasoning about how the world works - we just map states to rewards.
  - This means we can apply the same algorithm to a wide variety of environments.