Learning From Examples

CS156

Chris Pollett

Nov 24, 2014

Outline

- Forms of Learning
- Quiz
- Supervised Learning
- Learning Decision Trees

Introduction

- An agent is **learning** if it improves its performance on future tasks after making observations about the world.
- Agents that can learn are useful for several reasons:
  - *A programmer cannot necessarily anticipate all possible situations that an agent might encounter.*
  - *A programmer cannot necessarily predict all changes which may occur over time.*
  - *A programmer might not have any idea how to program a solution to some problem themselves.*
- To begin our study of learning we look at some different kinds of learning, then we focus on supervised learning, and, in particular, learning for decision trees.

Forms of Learning

- Improvements to components of an agent, and techniques used to make them, depend on:
  - *Which component is improved*
  - *What prior knowledge the agent already has*
  - *What representation is used for the data and the component.*
  - *What feedback is available to learn from*

Components to be Learned

We have described several agents so far this semester. The components of these agents which might be improved include:

- The mapping from conditions on the current state to actions
- The means to infer relevant properties of the world from percept sequence
- Information about the way the world evolves and about the results of possible actions the agent can take.
- Utility information indicating the desirability of world states
- Action-value information indicating the desirability of actions
- Goals that describe classes of states whose achievement maximizes the agent's utility.

The book gives examples of the above in terms of a taxi driver agent.

Representation and Prior Knowledge

- There are many possible representations for agent components: propositional or first-order sentences; we could develop our probabilistic model into somethings called a Bayesian network, and so on.
- Effective learning algorithms are known when knowledge is represented in each of these systems,
- We will be interested in the case where knowledge is in a factored representation: A vector of attribute values.
- When we go from input-output pairs and attempt to learn a general function that governs these pairs, we say we are doing **inductive learning**.
- If we instead start from general rules, and derive things logically entailed from them, we say we are doing **deductive** or **analytic** learning.

Quiz

Which of the following is true?

1. The principle of Maximum expected utility says that a decision theory agent should choose an action which yields the lowest expected payoff among the available choices.
2. In probabibility theory marginalization and conditioning are the same thing.
3. The following calculation is an example of applying Baye's rule: If a doctor knows that 70% of people with meningitis have stiff necks and the odds of meningitis are 1/50000 and the odds of a stiff neck are 1/100. Then the odds of meningitis given a stiff neck are (.7 * 1/50000)/0.01 = 0.0014.

Feedback to learn from

There are three main types of feedback that correspond to the three main types of learning:

- In **unsupervised learning** the agent learns patterns in the input even though no explicit feedback is supplied. A common task in this area is **clustering**: detecting potentially useful clusters of input examples.
- In **reinforcement learning** the agent learns from a series of reinforcements -- rewards or punishments. For example, the lack of a tip at the end of the journey gives the taxi agent an indication that it did something wrong. Winning a chess game give an agent an indication is did something right.
- In **supervised learning** the agent observes some examples input-output pairs and learns a function that maps from input to output.

In addition, to the above there is also things such as **semi-supervised learning** where we are given a few labeled examples and must make what we can of a large collection of unlabeled examples.

Unsupervised Learning

- For the rest of this class we will mainly be interested in supervised learning algorithms.
- Still, it is not hard to describe a simple mean-based hierarchal clustering algorithm, so let me do that one this slide.
- Suppose the data we want to learn clusters from consists of example vectors $E_i = (e_{i1}, ..., e_{in})$ in $\mathbb{R}^n$.
- There several possible distance functions we might choose between such vectors. For example, Euclidean distance square: $||E_i - E_j||^2 := \sum_{k=0}^{n} (e_{ik} - e_{jk})^2$.
- Suppose we have a data set of N examples. To cluster we use a data structure that maintains a forest of labelled trees. This forest is initialized to a list of single node trees labeled with $E_1, ...E_N$.
- Our algorithm has a while loop that cycles while our forest has more than one tree.
- In the body of this while we cycle over pairs of trees and compute the square euclidean distance of the root labels of these trees to find the two trees which are nearest each other.
- Once we have found these, we delete these two trees T_i, T_j from our forest. Let E_i, E_j denote the vectors labeling the roots of these trees. Let $E' = \left(\frac{e_{i1} + e_{j1}}{2}, ..., \frac{e_{in} + e_{jn}}{2} \right)$. We make a new tree T' with root label E' and subtree T_i and T_j and insert this tree into our forest.
- When the algorithm stops we have a tree. If we want the k most important clusters of our examples, we stop the algorithm when the forest has exactly k trees in it, and look at the examples in the leaves of these trees.

Supervised Learning

- The task of supervised learning is:

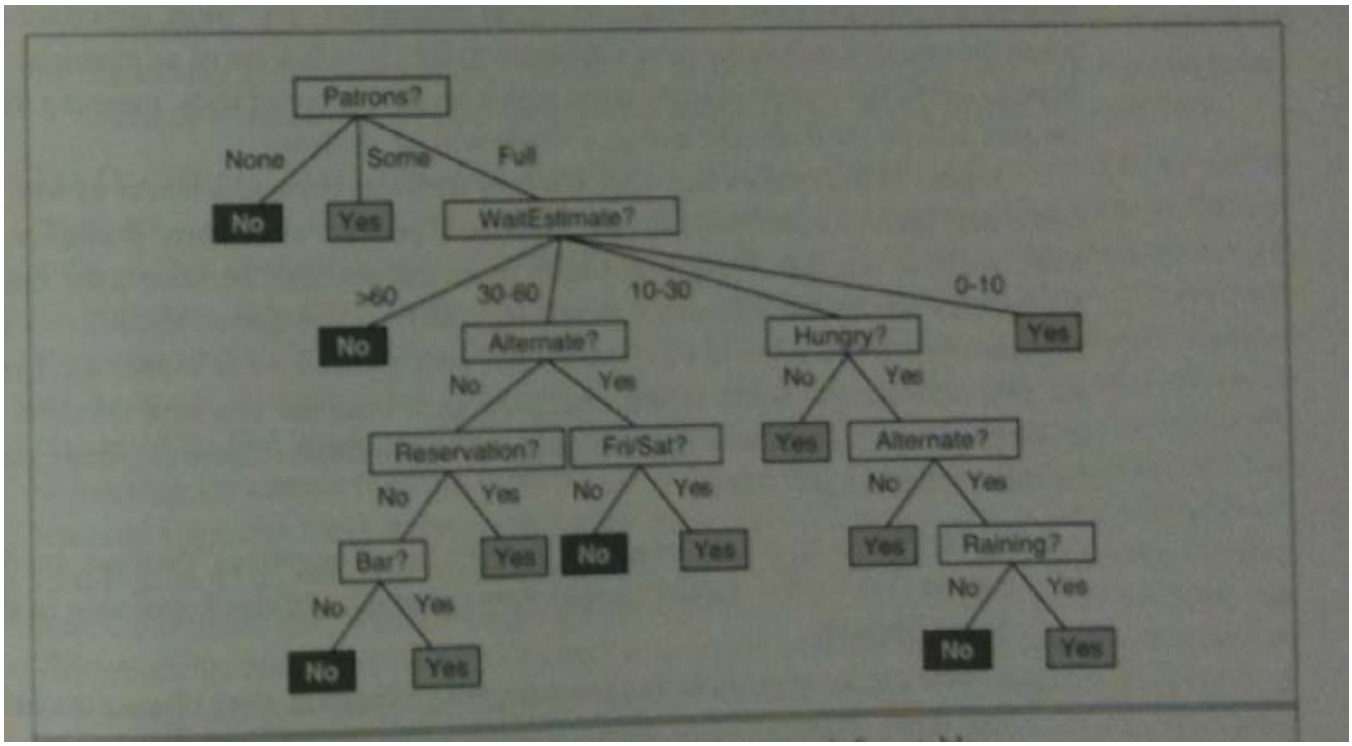
  > *Given a **training set** of N example input-output pairs*
  > *$(x_1, y_1), ..., (x_n, y_n)$,*
  > *where each y_j was generated by an unknown function $y = f(x)$,*
  > *discover a function h that approximates the true function f.*

- Here x and y can be any values; they need not be numbers.
- The function h is a **hypothesis**.
- Learning is a search through the space of possible hypotheses for one that performs well.
- What we want is that it continues to perform well on examples beyond the training set.
- To measure this we give it a **test set** of examples that are distinct from the training set and see how well it predicts their values.
- Sometimes the the function f is stochastic -- we want to learn a conditional probability distribution $\vec{P}(Y|x)$.
- When the output y is one of a finite collection of values, the learning problem is called **classification**.
- When y is always a number the learning problem is called **regression**.

Ockham's Razor

- Consider a collection of ten (x,y) pairs all with distinct x value.
- Regardless of where they are on the plane there will be many functions that go through them.
- We will approximate the real function with some hypothesis h from the hypothesis space.
- Any h that agrees with all the data is called a **consistent hypothesis**.
- Again, there may be many consistent hypotheses, so which one should we choose?
- One answer is to prefer the **simplest** hypothesis consistent with the data.
- This is called **Ockham's Razor**.
- The question then becomes how to decide what simplicity means.
- For example, we might prefer solutions which are degree 1 polynomials over degree 7 polynomials.
- One might also look at how many bits it takes to write down the function together with how many bits it takes to write down any deviations from that the data has from the function.

Learning Decision Trees



- We are now going to look at one of the simplest models in which we can do supervised inductive learning: decision trees.
- A **decision tree** represents a function that takes as input a vector of attribute values and returns a decision -- a single output value.
- The input and output can be discrete (for example, Patrons? in above images) or continuous (WaitEstimate? above).
- The above diagram shows an example decision tree for whether to wait for a table at a restaurant.
- We are going to focus on the case where the output is Boolean: true (a **positive example**) or false (a **negative example**). These will be the values of the leaves of our trees.
- A decision tree reaches its decision by performing a sequence of tests.
- Each internal node of our tree will correspond to querying the value of one of the attributes of the problem. For example, depending on the value of Patrons? we go down different branches of the above tree until we reach a leaf which tells us what to do.

Inducing decision trees from examples.

- An example for a Boolean decision tree consists of a pair $(\vec{x}, y)$. Here $\vec{x}$ is a value for each attribute and y is either true or false.
- Such an example can be viewed as a path in a decision tree.
- For example, if the problem was whether to wait for a seat at a restaurant, the first coordinate of $\vec{x}$ might correspond to whether there are already patrons at the restaurant. There might be three possibilities: None, Some, and Full. But $\vec{x}$ says one specific choice. This corresponds to the decision at the top node of the tree.
- Similarly, the next attribute might be something like Wait Estimate? The particular value for this coordinate might be 30-60. It corresponds to a next level in the tree.
- Here we are envisioning that the top of the tree corresponds to the decision made by the first element of $\vec{x}$. Typically, we have a set E of training pairs. A tree might actually query the variable coordinates in $\vec{x}$ in whatever order it wants. Not all branches in the tree need to be the same length. A tree is consistent with the examples in E if for every pair $(\vec{x}, y)$ in E if we follow the path down the tree according to the settings in $\vec{x}$ when we hit a leaf it has value y.

Decision Tree Learning Algorithm

Here is an algorithm for coming up with a decision for a set of examples.

```
function Decision-Tree-Learning(examples, attributes, parent_examples) returns a tree
    if examples is empty then return Plurality-Value(parent_examples)
    else if all examples have the same classification then return the classification
    else if attributes is empty then return Plurality-Value(examples)
    else
        A := argmax_(a in attributes)Importance(a, examples)
        tree :=  a new decision tree with root test A
        for each value v_k of A do
            exs := {e : e in examples and e.A = v_k}
            subtree := Decision-Tree-Learning(exs, attributes -  A, examples)
            add a branch to tree with label (A=v_k) and subtree subtree
        return tree
```