CSE 446: Week 2: Decision Trees (Mar 30)

Instructor: Sergey Levine

I. Making a single decision

Recall the binary classification problems we discussed in the previous lecture. First, let's come up with a binary classification problem that is a bit more concrete. Let's imagine we want to determine whether a given a patient has a particular disease. The patients will describe to us their symptoms, and we will try to predict whether the patient has the disease. All symptoms are binary: the patient either has them or doesn't. Let's say we've seen the following patients:

	fever	cough	strange dreams	has disease?
Patient 1		x	x	
Patient 2	x	x	x	x
Patient 3	x		x	x

Now, as we saw on Monday, if we choose a very expressive hypothesis class, like all possible boolean functions, we can perfectly fit this dataset, but we will not be able to say anything meaningful about new patients. So we need a hypothesis class that is simpler, that has some inductive bias (we will discuss simplicity of hypotheses more later in this lecture).

What if we are allowed to ask the patient only one question -- find out about one symptom, and then we have to predict whether the patient has the disease? That's a very simple hypothesis class, so we might expect it to model this dataset very well.

Exercise I. 1: Which question should we ask?

Exercise I. 2: Let's fit this into the framework of machine learning. Recall from lecture on Monday that a machine learning algorithm consists of the following parts:

1. Data (input **x** and output **y**)
2. Hypothesis space
3. Objective function
4. Algorithm

Let's figure out what each of these is for this problem.

I. 2. a. What is the data? The input **x** corresponds to a binary vector, with a 1 if the patient exhibits that symptom, and 0 otherwise. The output **y** is a binary value.

I. 2. b. What is the hypothesis space? We need to define the hypothesis space in terms of some parameterization -- some set of values that we choose as part of our learning algorithm. Since we only get to ask the patient one question and then produce the answer, we might define our hypothesis space in terms of three values: the index of the attribute, the answer if the attribute is false, and the answer if the attribute is true.

I. 2. c. What is the objective function? This one requires some thought, and we will discuss a few different options for the objective function. For now, let's just choose an easy one: the fewer patients we misdiagnose, the better.

I. 2. d. Now let's design an algorithm. This will be an easy one. Once we have a hypothesis space and an objective function, the algorithm is simply something that will find the hypothesis that optimizes the objective. As we'll see later in the lecture, some objectives cannot be optimized perfectly, in which case we need algorithms that optimize approximately or heuristically. But in this case, we can construct an exact optimization algorithm: we simply iterate over all possible attributes, and for each attribute, we choose the best answer for "false" and the best answer for "true." Note that, once the attribute is selected, there is only one obvious choice for each answer that minimizes our objective, so we always choose the label associated with most patients with that value of that attribute. This means that our hypothesis space really only has one parameter to optimize. We've now designed our first machine learning algorithm.

II. Making multiple decisions

Now let's say that we have one more patient in our dataset, who exhibits symptoms as following:

	fever	cough	strange dreams	has disease?
Patient 1		x	x	
Patient 2	x	x	x	x
Patient 3	x		x	x
Patient 4	x	x		

If we are only allowed to make ask one question, we can't possibly get all of these right. So we have to query at least two attributes from the patients. For example, we could first ask if they have a fever, and if they do, then we have to ask if they also have strange dreams. This is a rudimentary decision tree. A decision tree consists of nodes, where each node tests a single predicate, such as "is attribute x true," and recurses into one of its children. Each leaf node assigns a class label y. Each input can be classified by traversing the tree.

III. Approximate fitting

Now let's see what happens if we have another patient:

	fever	cough	strange dreams	has disease?
Patient 1		x	x	
Patient 2	x	x	x	x
Patient 3	x		x	x
Patient 4	x	x		
Patient 5	x			x

Now things are getting a little complicated. We could fit a tree with three decisions to this data and get perfect accuracy, but considering we only have three attributes, having three decision points seems like a bit much. What if we have another patient like this:

	fever	cough	strange dreams	has disease?
Patient 1		x	x	
Patient 2	x	x	x	x
Patient 3	x		x	x
Patient 4	x	x		
Patient 5	x			x
Patient 6	x			

The trouble now is that there is no decision tree we could fit to this data and still get the right answer. But how can this be? How can we have two patients with identical symptoms but different outcomes? Well, this kind of thing will happen all the time in machine learning. Perhaps Patient 5 lied to us (maybe he didn't want to admit to having strange dreams), or maybe the disease causes high fever, but there are also other factors that cause high fever that are unrelated to the disease, and therefore act as confounding factors. We will often encounter cases where the data is not internally consistent, and only an approximate fit is possible. We therefore almost always want an approximate fit: one that minimizing some objective.

This is why the objective is so important: it allows us to define which approximations are preferable to others. The choice of objective is generally regarded as a design choice, though as we will see later in the course, probability theory can give us useful hints about what kinds of objectives we should be using. For now, let's go with a simple objective: minimize the total number of mistakes we make on the training data. This objective is simple, but we will see that it has some problems.

IV. Define the hypothesis space

Now we need to define a new hypothesis space and algorithm. We can keep the same objective for now -- number of misclassified examples. The hypothesis space is now the space of all decision trees. A decision tree consists of some set of nodes, where each node has a parent, each node has some number of children (typically but not always 2, as we will see later), and each node is associated with a test, which for now we'll take to mean querying the value of some attribute (like "fever") and splitting on it.

Exercise IV. 1. How many possible decision trees are there? Well, imagine that we have N attributes. We must choose one attribute to split on at each level of the tree. We can then have any tree on the remaining attributes in each subtree. So if we have M subtrees on the left and K subtrees on the right, then the total number of trees is M * K for *each* choice of the current split. So let's proceed by induction.

Exercise IV. 1. a. How many decision trees can we construct if we have no attributes? Answer: we can construct two (T or F).

Exercise IV. 1. b. How many decision trees can we construct if we have one attribute? Answer: either T or F, or split, since each subtree has two options, we have 2*2 options in this case, so 2 + 2*2 = 6

Exercise IV. 1. c. How many decision trees can we construct if we have two attributes? Answer: we can choose two possible splits, each of which will result in a subtree with one attribute, so we have M = 2, K = 2, 2 possible splits, plus the choice to not split (T or F), so 2 + 2 * (2 + 2 * 2) * (2 + 2 * 2)

Exercise IV. 1. d. Three attributes? Answer: 3 splits, 2 option to not split, so 2 + 3* (2 + 2 * (2 + 2 * 2)) * (2 + 2 * (2 + 2 * 2))

Exercise IV. 1. e. In general? Worse than exponential

So we see that if we use the simple brute force algorithm that we used for the single decision case, where we simply enumerate all possible hypotheses from our hypothesis space, we will have an exponentially large search space. Indeed, it can be shown that finding an optimal

decision tree with respect to cost functions like classification error is NP-Hard [Hyafil & Rivest ‘76].

V. Greedy vs exact fitting

So now we have an data, a hypothesis space, and an objective, but we don’t have a good algorithm that can tractable optimize for a good hypothesis from our hypothesis space.
In practice, a greedy heuristic method often works very well, though it will not produce an optimal decision tree. This heuristic simply picks a split that produces the tree with the lowest cost after that split.

We first need to define some cost. For now, let’s just say that the cost is the number of incorrect predictions on the training set. The way this greedy heuristic works is that we recursively apply the following operations:

1. Iterate through each leaf of the tree, and for each leaf, try to see what happens if we split that leaf on each attribute that we can use at that leaf
2. After each such split, evaluate the cost of the resulting tree
3. Choose the tree with the lowest cost
4. Repeat

This greedy procedure is not perfect, and will not always find the optimal tree. But in practice it works quite well.