CSE 513T Theory of Artificial Intelligence & Machine Learning January 13, 2015

Lecture 1

Lecturer: Brendan Juba *Scribe: James Jackson*

1 Administrative Notes

Course webpage can be accessed at http://www.cse.wustl.edu/ bjuba/cse513t/s15/
It contains:

- Piazza link
- Homework Details
- Project Details
- Calendar of lecture topics (subject to change)

Office hours: Wednesdays 3:30-5:00pm in Bryan 522
(Only 3:30-4:00pm on 1/14)

2 An Introductory Exercise

Class exercise: Randomly ask 10 people why they are interested in taking this class.
This brings up the following questions:

- What does it achieve to ask why 10 people out of 50ish are interested in taking this class?
- How can I infer anything about the 40 or so I don't ask?
- What can/can't I learn by asking a few selected at random?
- What can I actually do with this knowledge?

These questions are representative of the main themes of this course.

3 Knowledge Representations

3.1 Knowledge (Classical)

We consider a "domain" of "space of instances" some set X typically defined by some (finite) list of attributes.

e.g.

$X \equiv$ all possible students

$X_1 \equiv$ "wants to integrate Machine Learning into some larger system"

$X_2 \equiv$ "interested in philosophical side of Artificial Intelligence"

etc.

We can abstractly respresent the set by some number ("n") of Boolean values. Think of $X \equiv \{0,1\}^n$. Components indicate whether or not each attribute is true of a given instance.

Any $C \subseteq X$ specifies a property. The actual space of possibilities, D, is likewise a subset of X. We'll say that C is true (i.e. valid for D) if $D \subseteq C$.

e.g. $C = \{x \in X :$ "thought 513T would help improve my golf game" $= 0\}$
(C is the set of students that were not here for golf game improvement)

Presumably, $C \subseteq X$ is valid here.

Let $C \subset X$ ("nontrivial" property). Then until we examine every last $x \in D$, we can't conclude that C is valid for D.

3.2 PAC Semantics

Suppose that instead of $D \subseteq X$, we had a probability distribution D over X – e.g. each student has probability of 1/50 of being queried.

As before, I can't establish anything about 100% of D without observing 100%. But, I can make a good guess about 90% without having to ask anywhere close to 90%.

Suppose $\geq 10\%$ did enter the class with an improved golf game in mind. How likely was I to miss this 10% when I chose 10 at random (i.e. What's the probability I would finish my poll with no indications that anyone has that motivation)?

If we consider this question as picking without replacement, we'd get the following:

$$\frac{5}{50} \text{ thinking about golf}$$

$$\text{and Pr} = \frac{\binom{45}{10}}{\binom{50}{10}}$$

But what's more interesting to us is actually picking with replacement, giving us the following: In a single query, the probability of missing all the golf students is:

$$\Pr[\text{miss the golf students}] = 1 - \Pr[\text{student interested in golf}]$$

$$\Pr[\text{miss the golf students}] \leq 90\%$$

Since we're asking 10 independently,

$$\Pr[\text{all 10 miss the golf students}] = \Pr[\text{miss the golf students}]^{10}$$

$$\Pr[\text{all 10 miss the golf students}] \leq (0.9)^{10} \approx 0.34$$

What if I asked another 10? Square it...

$$\Pr[\text{all 20 miss the golf students}] \leq (0.9)^{20} \approx 0.11$$

So, the conclusion "no one is interested in improving their golf game" is Probably (0.89) Approximately (0.90) Correct ("PAC").

Note: This does **not** depend on the size of the class. Even if there were 100 or 10,000 in the class, by examining 20% we gain "89% confidence about 90% of the population".

3.3 Representing Properties

Properties can be shown as a database like the following table:

	gives_you_up	lets_you_down
Rick Astley	0	0

Properties (e.g "C") are defined by the sets of attributes that make them true. Instead of writing out the database defining C, we'd prefer some more compact Knowledge Representation.

Conjunctions One example of a compact knowledge representation is the conjunction, shown below for Rick Astley.

$$\text{Rick Astley} = \neg\text{gives_you_up} \wedge \neg\text{lets_you_down} = \varphi(x)$$

where $\neg$ and $\wedge$ represent Boolean operators "NOT" and "AND", respectively.

A Boolean variable or its negation (e.g. $\neg$gives_you_up) is referred to as a literal.

The class of conjunctions consists of all possible sets of literals joined by the AND connector.

Decision Trees Another example of a compact knowledge representation is the decision tree. An example decision tree is shown below.

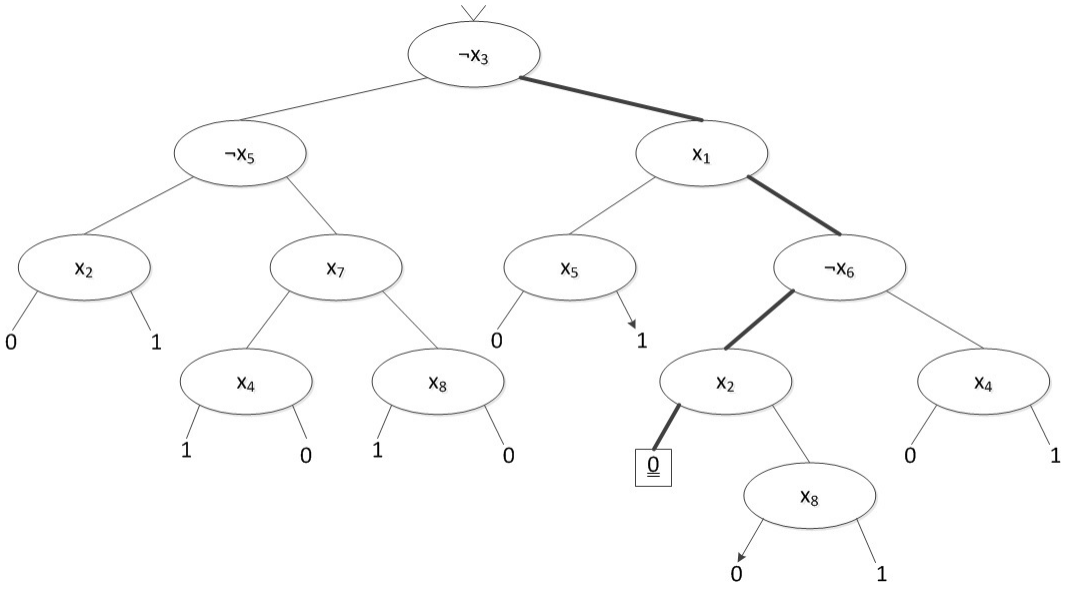

Figure 1: Decision Tree

The node at the top of the tree (containing x_3 in the example above) is referred to as the root.

The TRUE path goes to the right from each node.

The darkened path segments represent the path followed by $x = (10010110)$.

The chosen x does not have the property defined by this decision tree as can be seen since the darkened segments lead to 0 (boxed).

The conjunction $\neg x_3 \wedge x_1 \wedge x_6 \wedge \neg x_2$ defines the "bucket" that collects x.

Formally, a Knowledge Respresentation is just a way of assigning a subset of X to a sequence of symbols over an alphabet (e.g. a binary encoding).

We want, moreover to be able to evaluate whether or not the property holds on a given instance efficiently. (Generally, there should be an efficient algorithm for this evaluation).

Knowledge representations differ in their expressive power. Every conjunction can be written as a decision tree, but there are decision trees that can't be written as conjunctions.

A decision tree derived from an arbitrary conjunction is described as follows:

- Chain of node for all literals
- True path connects to next node
- Deepest node (leaf) true path goes to 1
- Every false path goes to 0

As such, Rick Astley's decision tree is shown below.

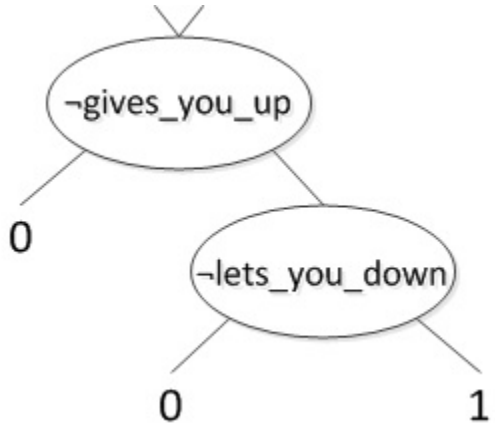

Figure 2: Rick Astley's Decision Tree

If we change the false path of the root node to go to 1, we get the following decision tree, which can **not** be expressed as a conjunction:

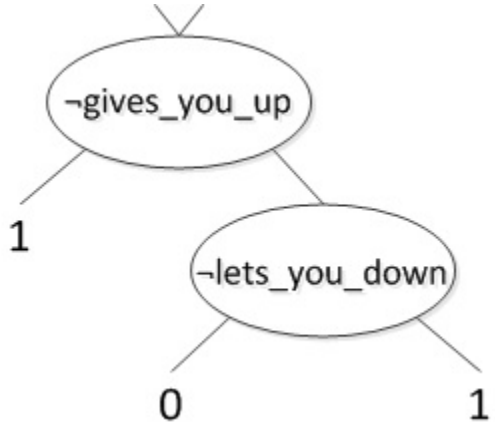

Figure 3: Decision Tree without Equivalent Conjunction

The Boolean expression for this decision tree would be gives_you_up $\vee$ ¬lets_you_down, where $\vee$ represents the Boolean operator "OR". The truth table for this decision tree is shown below.

	gives_you_up	lets_you_down
1	0	0
0	0	1
1	1	0
1	1	1