Prelim II: November 22, 2003

1 Reinforcement Learning

1. (11 pts) Show the state utility update equation, U, for method (1), **direct utility estimation**. Be sure to define all symbols.

   U table of current utilities

   s a unique state in the environment

   M model of environment (for #2)

   N table of visit frequencies

   U(s) = running-average (U(s), reward-to-go, N(s)) where reward-to-go is the sum of the rewards from this state forward for the current trial.

   For this question, 8 points were awarded for a proper equation and 3 for defining the symbols. Many variants of this equation were considered correct. Usually, you would lose 2 to 3 points for leaving out sum of rewards, and without at least mentioning a running average you would have another 5 or so points off. Other than this depending on how close your equation was, you got between 0 and 8 points plus the 3 from defining symbols.

2. (8 pts) Show the state utility update equation(s), U, for method (2), **dynamic programming**. Be sure to define all symbols.

   Utilities follow from:
   $U(i) = R(i) + \sum_j M_{i,j} U(j)$ (note: i, j over states.)

   $R(i)$ is the reward associated with being in state i.

   $M_{i,j}$ is the probability of transition from state i to j.

   You received 6 points on this question for the proper equation and 2 for defining your symbols. 3 points were taken off for leaving out the reward from the equation, which was the most common mistake.

3. (6 pts) Which of the **three** methods above do **not** require a model of the environment?

   TD-learning and the direct utility estimation methods.

   4 points for 1 correct and 6 for two correct. One correct and one wrong was worth 2 points, and none correct was worth 0.

2 Version Spaces

1. After example 1:
   G = (?, ?, ?)
   S = (schneider, cheerios, 8-hours)

   2 points each for correct S, G. You got 1 point off for a small error, a full 2 points off for a large error.

2. After example 2:
   G = (schneider, ?, ?) (?, cheerios, ?) (?, ?, 8-hours)
   S = (schneider, cheerios, 8-hours)

3 points for a correct G, 2 points for a correct S.
If you put hypotheses like: (Pingali, ?, ?) (?, bagel, ?) in addition to the ones already in G, you got 2 points taken off because they can't specialize to S.

3. After example 3:
   G = (?, ?, 8-hours)
   S = (?, ?, 8-hours)

   3 points for a correct S, 2 points for a correct G.
   Something like S = (pingali, bagel, 8 hours) (schneider, cheerios, 8 hours) is not a sufficient generalization from the previous S. You got 1 point off for this.

4. Get 8 hours of sleep.

   2 points for the correct answer. If the previous work was incorrect but you put something reasonable here based on it, you got full credit.

5. Some possible answers are:

   - The algorithm can't handle noise in the training examples. The version space would collapse.
   - There is a very restricted representation of hypotheses. Targets can only be represented as a boolean. Can only use positive literals.
   - The G set can potentially get exponentially large by seeing many negative and no positive examples.

3 First-Order Logic and the Resolution Proof Procedure

1.

$$\forall x, \text{easy(x)} \Rightarrow \exists y.(\text{student}(y) \wedge \text{happy}(y)) \quad (1)$$

$$\forall x, \text{has-prelim}(x) \Rightarrow \neg\exists y.(\text{student}(y) \wedge \text{happy}(y)) \quad (2)$$

Grading: Three points were assigned to each of the two statements. One point was deducted for small logical mistakes (putting an existential quantifier instead of a universal, or an and instead of an implication). Two points were deducted for more serious errors.

2. Let us consider (1) first

$\forall x, \text{easy(x)} \Rightarrow \exists y.(\text{student}(y) \wedge \text{happy}(y))$	
$\forall x, \neg\text{easy(x)} \vee (\exists y.(\text{student}(y) \wedge \text{happy}(y)))$	definition of $\Rightarrow$
$\forall x, \neg\text{easy(x)} \vee (\text{student}(S1(x)) \wedge \text{happy}(S1(x)))$	Skolemization
$\forall x, (\neg\text{easy(x)} \vee \text{student}(S1(x))) \wedge (\neg\text{easy(x)} \vee \text{happy}(S1(x)))$	distribution

   Finally, we rename variables to get two statements for our knowledge base:

$$\neg\text{easy(x)} \vee \text{student}(S1(x)) \quad (3)$$

$$\neg\text{easy(y)} \vee \text{happy}(S1(y)) \quad (4)$$

   Now, we consider (2).

$\forall x, \text{has-prelim}(x) \Rightarrow \neg\exists y.(\text{student}(y) \wedge \text{happy}(y))$	
$\forall x, \neg\text{has-prelim}(x) \vee \neg\exists y.(\text{student}(y) \wedge \text{happy}(y))$	definition of $\Rightarrow$
$\forall x, \neg\text{has-prelim}(x) \vee \forall y.\neg(\text{student}(y) \wedge \text{happy}(y))$	move $\neg$ in
$\forall x \forall y, \neg\text{has-prelim}(x) \vee \neg(\text{student}(y) \wedge \text{happy}(y))$	move $\forall$ in
$\forall x \forall y, \neg\text{has-prelim}(x) \vee \neg\text{student}(y) \vee \neg\text{happy}(y)$	move $\neg$ in

After variable replacement, we have the last statement for our knowledge base

$$\neg\text{has-prelim}(z) \vee \neg\text{student}(s) \vee \neg\text{happy}(s) \tag{5}$$

Grading: This problem was graded given your part 1. Two points were deducted if the Skolemization function (or the fact that it needs a parameter) was forgotten. Three points were deducted if the resulting knowledge base made the proof in part 3 trivial. Points were also deducted for making mistakes in the translation to the knowledge base.

3. We have these additional statements available as well

$$\text{has-prelim}(S2) \tag{6}$$
$$\text{easy(S2)} \tag{7}$$

We resolve 5 with 6, unifying $S2$ and z.

$$\neg\text{student}(s) \vee \neg\text{happy}(s) \tag{8}$$

Next, we resolve 7 and 4, unifying y and $S2$

$$\text{happy}(S1(S2)) \tag{9}$$

Resolve 7 and 3, unifying x and $S2$

$$\text{student}(S1(S2)) \tag{10}$$

We resolve 10 with 8, unifying S and $S1(S2)$

$$\neg\text{happy}(S1(S2)) \tag{11}$$

Finally, we resolve 9 with 11 to get nil, meaning we have proven that *if a course has a prelim, the course isn't easy*.

Grading: This problem was graded with the assumption that your knowledge base was correct. A point was deducted for adding an assumption (such as $\text{course}(S1)$). Not listing substitutions and unifications resulted in a two-point deduction. Having a step in the proof that did not follow resulted in a 2-4 point deduction, depending on the severity of the error.

4 Decision Trees and Neural Networks

1. (5 pts)

   Explain briefly the decision tree that would result if these examples were presented to a decision tree induction system like ID3. (No equations should be necessary.)

   Both the *professor* and the *sleep* features would produce a tree with 3 leaves, each of which contains a single example (and therefore maximum information gain).

   Grading: Three points deducted if only one of the two possible root features was mentioned. One or two points off for smaller errors or confusions in the answer.

2. (5 pts) (True / False) Decision trees produced by the ID3 algorithm never test the same attribute twice along one path. (Briefly explain your answer.)

   True. Once an attribute is used to divide instances into subsets based on the value of that attribute it will either produce a homogeneous set of instances w.r.t. class value (and so no additional tests will be needed) or it will produce a set of instances that have mixed class

values. In this latter case, selecting the same attribute lower in the tree will not result in lowering the IG since it will be unable to provide an additional split among the remaining instances.

Grading: For this question, students generally got full credit or no credit. One or two or three points were deducted for answers that were too confusing or sketchy for us to trust that you really understood what you were talking about.

3. neural net problem

Let $f(x)$ denote the sigmoid function $f(x) = \frac{1}{1+e^{-x}}$. The input for the two biases is -1.

- dot product of input to c and first level weights is $< 1, 0, -1 > \cdot < 0.1, 0.1, 0.1 >= 0$
- since $f(0) = \frac{1}{2}$, output $o_c = \frac{1}{2}$
- dot product of input to d and second level weights is $< \frac{1}{2}, -1 > \cdot < 0.1, 0.1 >= -0.05$
- since $f(-0.05) \approx 0.49$, output $o_d \approx 0.49$
- desired output 1, so error at output node d is $\beta_d = 0.51$
- error at hidden node c is $w_{cd} o_d (1 - o_d) \beta_d = (0.1)(0.49)(0.51)(0.51)$.
- if you used Mitchell, then
  - $\delta_d = o_d(1 - o_d)\beta_d = (0.49)(0.51)(0.51) = 0.128$
  - $\delta_c = o_c(1 - o_c) w_{cd} \delta_d = \frac{1}{2}\frac{1}{2}(0.1)(0.128) = 0.003$

Grading guidelines: For part (a), if you didn't use the sigmoid function properly you lost 3 points. If you forgot to use the biases you lost 1 point. For part (b), if you used the wrong formulas you lost somewhere between 5 and 7 points. If you didn't plug in any values then you lost 3 points, and you lost 1 to 2 points for small numerical or formulaic errors. If you didn't calculate the error for the hidden node, you lost 7 points. For part (c), if you didn't mention that the algorithm learns weights, you lost 2 or 3 points.