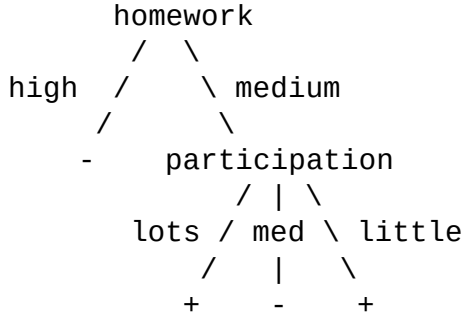
CS472

Foundations of Artificial Intelligence

Prelim II Solutions

Decision Trees and the Version Space Algorithm

1. 

2. G: $<$? ? ? $>$; S: $<$ lots medium novice $>$;

3. G: $<$? medium ? $>$ $<$? ? novice $>$

   ($<$ medium ? ? $>$ and $<$ little ? ? $>$ were generated, but pruned because they do not lead to S.)

   S: $<$ lots medium novice $>$ (unchanged)

Grading:

1. Four points for getting the correct root attribute, three for the next attribute, and three for the leaves. You got at least one point if the tree correctly classified all examples, and may have lost one point if it did not.

2. One point for the correct G set and two points for the correct S set.

3. Five points for the correct G set and two points for the correct S set. Neglecting to prune $<$ medium ? ? $>$ and $<$ little ? ? $>$, a common error, resulted in a loss of two points.

General Machine Learning
Lots of answers are possible for each question, e.g.:

1. ID3 has a more expressive hypothesis space and can deal better with noise.

2. The version space algorithm can be run incrementally, i.e. on problems where not all of the data is available up front.

3. K-nn is faster to train since it just requires storing all training examples.
   Other training examples can be added to k-NN later, but ID3 needs to be retrained.

Grading:
3 pts: good answer
2 pts: right idea, but there are some flaws in the way you phrased it
1 pts: you might have a point, but it is not very clear, or it is not clear that the advantage you found is really an advantage.
0 pts: wrong answer.

Neural Networks

1. - Not shown here.
   - Not shown here.
   - See the lecture notes or, for a slightly different version of the perceptron learning rule, see p. 742 of R&N. Both variations were allowed (although the one in the book is not the original perceptron learning rule).

2. Reinforcement learning algorithms learn from delayed feedback and rewards while supervised inductive learning algorithms learn directly from examples that have been labeled w.r.t. the correct action, category, or class label.

3. The dynamic programming solution calculates the utility values directly from a set of equations derived from the model while the TD learning approach calculates utility values by executing a set of trials or "runs" through the environment. See p. 769-770 of R&N for additional differences.

Grading:

1. Part a was worth 2 points. One point taken out if someone forgot to include the input from $x_0 = -1$ (even if the weight is 0), or for any other small mistake.

2. 2 points removed if slope was incorrect, 2 points if offset of line was incorrect (line was not passing through $0, 0$), 2 points if the classification was done incorrectly (classified the half plane where the output is 0 as +1 and vice versa). Also up to 2 points given if the graph was wrong (slope and offset), but at least some explanation of the equations was given.

3. For an equation that in the correct form, 1 point was given for the following: learning rate, input and previous weight. 2 points for the error (1 for mentioning it and 1 more for describing the equation of how the error is calculated). Finally the last 2 poins were given by default to people who had the equation is the correct form (that is newWeight = oldWeight + product of terms) even if a term was missing or incorrect).

4. 2 points given for answers that had some (relatively small) error or were somewhat vague, 1 point in the case that the answer was mostly wrong but had some useful element.

5. 2 points given for answers that had some (relatively small) error or were somewhat vague, 1 point in the case that the answer was mostly wrong but had some useful element.

First-Order Logic and Resolution Theorem-Proving

- 1. Cats are animals.
  $\forall x \;\; Cat(x) \rightarrow Animal(x)$

2. Tuna is a cat.
   $Cat(Tuna)$.
3. Every dog owner is an animal lover. $\forall x \ \ (\exists y \ \ Dog(y) \wedge Owns(x, y)) \rightarrow AnimalLover(x)$
4. Jack owns a dog.
   $\exists x : Dog(x) \wedge Owns(Jack, x)$
5. No animal lover kills an animal.
   $\forall x \ \ AnimalLover(x) \rightarrow \forall y \ \ Animal(y) \rightarrow \neg Kills(x, y)$
6. Either Jack or Curiosity kills Tuna.
   $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

- 1. $\neg Cat(z) \vee Animal(z)$
  2. $Cat(Tuna)$.
  3. $\neg Dog(S(x)) \vee \neg Owns(x, S(x)) \vee AnimalLover(x)$
  4. $Dog(D)$ (D is the function that finds Jack's dog)
     $Owns(Jack, D)$
  5. $\neg AnimalLover(w) \vee \neg Animal(y) \vee \neg Kills(w, y)$
  6. $Kills(Jack, Tuna) \vee Kills(Curiosity, Tuna)$

- 1. Add $\neg Animal(Tuna)$ to the KB as statement 7.
  2. Resolve statements 7 and 1 (unifier: $z = Tuna$) to get $\neg Cat(Tuna)$ (which becomes statement 8).
  3. Resolve statements 8 and 2 to produce *nil/fail*.

Grading:
1.

- 1 point for getting 1,2,6 correct.
- 2 points for getting 4 correct.
- subtract half of the points for not putting the solution for this part in this part.

2.

- 1 point for getting 1,2,6 correct.
- 3 points for getting 4 correct(2 points for skolemization, 1 point for splitting $Dog(D)$ and $Owns(Jack, D)$ into two sentences.).

- 2 points for getting 5 correct.
- subtract 1 point for using the same name for distinct variables.

3.

- 2 points for adding $\neg Animal(Tuna)$ to the KB as statement 7.
- 2 points for resolving the first sentence pair.
- 1 point for indicating the unifier.
- 2 points for resolving the second sentence pair and getting *nil/fail*.

Most people did the first part of this question (translating the English sentences into first-order logic) correctly. For the sentence "Either Jack or Curiosity kills Tuna" we accepted both inclusive-or and exclusive-or interpretations. The solutions above use an inclusive-or interpretation.

For the conversion to Clausal Normal Form, the most common error was not using a different letter for every distinct variable. For example, if you used z in $\neg Cat(z) \vee Animal(z)$, you can't later have $AnimalLover(z)$ in a clause unless z is represents the same object in both clauses. We were lenient in the grading of this rule and did not penalize you for reusing a variable in your sentences that was used in the provided CNF of sentence 3– Technically, you would not be allowed to use the variable x in your clauses since it was used in $\neg Dog(S(x)) \vee \neg Owns(x, S(x)) \vee AnimalLover(x)$. Please be careful of this in the future.

Another common mistake was not separating sentence four into separate clauses. Conjunction is implied between all clauses in the knowledge base, and no $\wedge$ should appear in your solutions.

Most students handled the resolution problem correctly.