Final Exam CS472
SOLUTION

12th of December, 2005

<table>
<thead>
<tr>
<th>Topic</th>
<th>Points / Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>State-Space Search</td>
<td>_____ / 30</td>
</tr>
<tr>
<td>Constraint Satisfaction</td>
<td>_____ / 16</td>
</tr>
<tr>
<td>Decision Tree Learning</td>
<td>_____ / 8</td>
</tr>
<tr>
<td>Linear Classifiers and SVMs</td>
<td>_____ / 11</td>
</tr>
<tr>
<td>Neural Networks</td>
<td>_____ / 10</td>
</tr>
<tr>
<td>Statistical Learning Theory</td>
<td>_____ / 10</td>
</tr>
<tr>
<td>Markov Decision Processes</td>
<td>_____ / 17</td>
</tr>
<tr>
<td>Knowledge-Based Systems</td>
<td>_____ / 16</td>
</tr>
<tr>
<td>Planning</td>
<td>_____ / 10</td>
</tr>
<tr>
<td><strong>Total</strong></td>
<td>_____ / 128</td>
</tr>
</tbody>
</table>
1 **State-Space Search**

1. In what respect is Iterative Deepening A* (IDA*) search preferable over regular A* search?  

   **SOLUTION:**  
   IDA* has lower memory requirements.  
   END SOLUTION

2. Name a class of problems (with more than one state) for which Depth First Search (DFS) is complete. Is DFS also optimal for this class of problems? Be sure to explicitly give the definitions of completeness and optimality in our response.

   **SOLUTION:**  
   Finite and acyclic state space OR search space is finite OR state space is finite tree. Still not optimal.  
   END SOLUTION

3. Name a condition under which Hill Climbing Search is guaranteed to find the maximum of the objective function. Assume that your state space has more than one state.

   **SOLUTION:**  
   Any of the following:  
   - No local optima and plateaus in the objective function  
   - (strictly) convex objective function  
   - all paths to the maximum have strictly increasing steps  

   END SOLUTION
4. The states in the graph below are labeled with their name and with the value of a heuristic function $h$. Edges indicate the successor function and all actions are labeled with their cost. There are two goal states, $G_1$ and $G_2$. Run $A^*$ on the graph from above using the heuristic function $h$. In what order will $A^*$ expand (i.e. compute the successors for) the (partial) paths starting from state $A$ before finding the optimal path (i.e. the path with the lowest cost)? For each (partial) path that is explored, write down

- a number indicating the order of exploration
- the value of $f = g + h$, where $g$ is the cost of the (partial) path.

To save you time writing down the answer to this question, the table below includes the (partial) paths that are relevant for solving the problem (plus some other paths, for which you do not need to fill in the value of $f$).

9 pts.

<table>
<thead>
<tr>
<th>Order</th>
<th>Path</th>
<th>Value of $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>0+3</td>
</tr>
<tr>
<td></td>
<td>AB</td>
<td></td>
</tr>
<tr>
<td></td>
<td>AC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACD</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACE</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACDG_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACDG_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACDC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ACEG_2</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABDG_1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABDC</td>
<td></td>
</tr>
<tr>
<td></td>
<td>ABDG_2</td>
<td></td>
</tr>
</tbody>
</table>

SOLUTION:

<table>
<thead>
<tr>
<th>Order</th>
<th>Path</th>
<th>Value of $f$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>A</td>
<td>0+3</td>
</tr>
<tr>
<td>3</td>
<td>AB</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>AC</td>
<td>2.5</td>
</tr>
<tr>
<td>4</td>
<td>ABD</td>
<td>4</td>
</tr>
<tr>
<td></td>
<td>ACD</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>ACE</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>ACDG_1</td>
<td>8.5</td>
</tr>
<tr>
<td></td>
<td>ACDG_2</td>
<td>6.5</td>
</tr>
<tr>
<td></td>
<td>ACDC</td>
<td>8.5</td>
</tr>
<tr>
<td></td>
<td>ACEG_2</td>
<td>4.5</td>
</tr>
<tr>
<td></td>
<td>ABDG_1</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td>ABDC</td>
<td>6</td>
</tr>
<tr>
<td>5</td>
<td>ABDG_2</td>
<td>4</td>
</tr>
</tbody>
</table>

END SOLUTION
5. For the search problem from above, is there an admissible heuristic function that is better than the $h$ that is depicted? Briefly explain your answer.

SOLUTION:
Yes. The heuristic at A, B and C is an underestimate that is smaller than it needs to be. If it matched the exact costs, we wouldn’t need to explore AC at all.
END SOLUTION

6. In the best case, how much does $\alpha\beta$-pruning improve the efficiency of minimax search?

SOLUTION:
Possible answers:
- Can search twice as deep.
- Goes from $O(b^d)$ to $O(b^{d/2}) = O(\sqrt{b^d})$

END SOLUTION

7. What is this “best case”?

SOLUTION:
The first expanded node is the move with the optimal utility.
END SOLUTION
2 Constraint Satisfaction

1. You are going on a backpacking trip and you are deciding on what food to bring. You bought $n$ pieces of food. Each piece of food $f_i$ has a certain volume $v_i$, a certain weight $w_i$, and a certain number of calories $c_i$. You can only carry a weight of at most $W$ and your backpack holds at most a volume $V$. To not go hungry, you need at least $C$ calories. You want to find out whether there is a selection of food that you can carry, that fits your backpack, and that has enough calories. Write this as a constraint satisfaction problem. The constraint language contains $+, -, *, \leq, \geq, =, \neq$. Be sure to explain the variables and their domains.

SOLUTION:

Variables: $x_i \in \{0, 1\}$ indicating which items are packed.

\[
\begin{align*}
C1 & \sum_{i=1}^{n} x_i v_i \leq V \quad (1) \\
C2 & \sum_{i=1}^{n} x_i w_i \leq W \quad (2) \\
C3 & \sum_{i=1}^{n} x_i c_i \geq C \quad (3) \\
C4 & \sum_{i=1}^{n} x_i = n \quad (4)
\end{align*}
\]

END SOLUTION
2. You have a constraint satisfaction problem with the variables \{A, B, C, D, E\}, each having domain \{1, 2, 3\}. The problem has the following constraints.

(a) \(A \neq B\)
(b) \(B \neq C\)
(c) \(D \geq E\)
(d) \(C \leq E\)
(e) \(2 \times A > E\)
(f) \(B \leq E\)

Apply Backtracking Search with Forward Checking and Arc Consistency to solve the Constraint Satisfaction Problem. Assume that Backtracking Search selects variables and values in increasing lexicographic/numerical order. Add into the following table how the domains of the variables change after each assignment. Circle values that are assigned by Backtracking Search, leave-out values that are removed by Forward Checking, and cross-out values that are removed by Arc Consistency. Stop when you found the first solution.

8 pts.

<table>
<thead>
<tr>
<th>Assignment</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=1</td>
<td>(1)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

SOLUTION:

<table>
<thead>
<tr>
<th>Assignment</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>A=1</td>
<td>(1)</td>
<td>23</td>
<td>123</td>
<td>123</td>
<td>-1-</td>
</tr>
<tr>
<td>A=2</td>
<td>(2)</td>
<td>13</td>
<td>123</td>
<td>123</td>
<td>123</td>
</tr>
<tr>
<td>B=1</td>
<td>(2)</td>
<td>(1)</td>
<td>23</td>
<td>-1.23</td>
<td>1.23</td>
</tr>
<tr>
<td>C=2</td>
<td>(2)</td>
<td>(1)</td>
<td>23</td>
<td>23</td>
<td>23</td>
</tr>
<tr>
<td>D=2</td>
<td>(2)</td>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td>23</td>
</tr>
<tr>
<td>E=2</td>
<td>(2)</td>
<td>(1)</td>
<td>(2)</td>
<td>(2)</td>
<td>(2)</td>
</tr>
</tbody>
</table>

END SOLUTION
3 Decision Tree Learning

1. What is the worst-case time complexity of the “Top-Down-Induction-of-Decision-Tree” (TDIDT) algorithm discussed in class (e.g., using reduction of error rate as the splitting criterion). Assume you have $n$ training examples, and $N > n$ binary attributes (i.e., more attributes than training examples, as it is often the case in text classification). Give your answer in Big-O notation with respect to $n$ and $N$. Please explain your answer briefly.

SOLUTION:
Worst case is an unbalanced tree (i.e., list). For each split, you need to evaluate all $j$ remaining attributes by computing the errors it makes on the $i$ examples that are left. This has complexity $O(j \cdot i)$. This is repeated for all $n$ training examples, giving $N \cdot n + (N - 1) \cdot (n - 1) + \ldots + (N - n) \cdot 1$, which is $O(N \cdot n^2)$.

END SOLUTION
4 Linear Classifiers and SVMs

1. You are working on a particular learning task and cross-validation experiments indicate that your SVM is overfitting. Name one action that can help decrease overfitting in an SVM.

SOLUTION:
The following are possible answers:

- Decrease C
- Use less expressive kernel (e.g. smaller degree polynomial)
- Get more training data

END SOLUTION

2. Show that the Boolean function \((x_1 \land x_2) \lor (\neg x_1 \land \neg x_2)\) is not linearly separable (i.e. there is no linear classifier \(\text{sign}(w_1 * x_1 + w_2 * x_2 + b)\) that classifies all 4 possible input points correctly). Assume that “true” is represented by 1 and “false” is represented by –1.

SOLUTION:
Possible solutions:

- Assume it is linearly separable. Then:
  - a) \(w_1 + w_2 + b \geq 0\)
  - b) \(-w_1 - w_2 + b \geq 0\)
  - c) \(w_1 - w_2 + b \leq 0\)
  - d) \(-w_1 + w_2 + b \leq 0\)
  
  From these, we get \((a + b \implies b > 0)\) and \((c + d \implies b < 0)\). This is a contradiction. So it’s not separable.
- same as \(\neg(AXORB)\), which we know is not separable.
- graphical proof.

END SOLUTION

3. Show that there is a linear separator for this Boolean function when you use the kernel \(K(x_1, x_2) = (x_1 \cdot x_2)^2\). Give the weights and the value of \(b\) for one such separator.

SOLUTION:
This kernel implies the feature space \(x_1^2, x_1 * x_2, \) and \(x_2^2\). Giving weight “1” to \(x_1 * x_2\) and zero to the other features perfectly represents the Boolean function when choosing \(b = 0.5\).

END SOLUTION

5 Neural Networks

1. You have an application problem for which you need to decide whether to use a Two-Layer Neural Net or a Support Vector Machine with a non-linear Kernel. Name one argument in favor of the Net, and one other argument in favor of the SVM.

SOLUTION:
Pro SVM:
- no local optima
- have to pick fewer parameters (e.g. layers, number of nodes) than in net
- easier to do get estimate of generalization error, since number of support vectors is an upper bound on leave-one-out error

Pro Net:
- network structure allows modelling of prior knowledge
- can control size of the net, while size of SVM classifier grows with the number of training examples

END SOLUTION

2. Design a two-layer network that represents the boolean function \( \neg((A \lor B) \land (C \lor \neg D)) \). Assume that true and false are represented by the values 1 and 0. In your network use threshold activation functions that return 1 if the activation exceeds the threshold, and 0 otherwise. Draw the network. Label the edges with the weights and the nodes with the thresholds.

6 pts.

SOLUTION:
Layer 1:
- Neuron 1: \( A*1+B*1 \geq 0.5 \)
- Neuron 2: \( C*1+D*(-1) \geq -0.5 \)

Layer 2:
- Neuron 3: \( N1*(-1)+N2*(-1) \geq -1.5 \)

END SOLUTION
6 Statistical Learning Theory

An alien spaceship has landed in a remote town in Nevada and you are a secret government agent that has to track down the aliens among the 128 “people” you find in the town. The aliens don’t look exactly human, but neither do the regular inhabitants. Fortunately, you know the aliens have a gambling habit:

- they can predict random events like coin flips with perfect accuracy
- they never make a losing bet

So, you invite all 128 people in the town to a game, where they have to bet on coin tosses and you record who wins and who loses. Eventually, only aliens will have never lost a game, since humans lose a game with 50% probability. However, you need to know how many coin tosses you need to play before you can be 95% sure that only aliens are left.

Fortunately, you remember your CS472 class and the formula
\[ P(\text{Err}_P(\hat{h}) \geq \epsilon) \leq |H|e^{-\epsilon n} \]
which will help you save humankind.

1. What quantities do \(\epsilon\), \(|H|\), and \(n\) correspond to in your problem? 6 pts.

SOLUTION:
\(n\) is the number of coin tosses, \(|H|\) is the number of players (i.e. 128), and \(\epsilon\) is the error probability of the humans (i.e. 0.5). You can think about it in the following way. Each player is a hypothesis, and each coin flip is a training example. Humans are “bad” hypothesis (i.e. hypotheses with a high error rate of 50%). Aliens are “good” hypotheses (i.e. hypotheses with a low error rate of 0% 50%). What the bound tells you is the probability that there is any “bad” hypothesis that by chance made zero training errors after \(n\) coin flips. END SOLUTION

2. How many coin tosses you need to play before you can be at least 95% sure that only aliens are left? You do not need to compute the final result, just show your derivation of a lower bound on the number of coin tosses needed. 4 pts.

SOLUTION:
The formula bounds the probability that any hypothesis in the set of zero-training-error hypothesis has a prediction error greater or equal to \(\epsilon\). Humans have prediction error of 0.5. So it is sufficient to show that no hypothesis has prediction error greater equal 0.5.

\[ P(\text{Err}_P(\hat{h}) \geq 0.5) \leq |H|e^{-0.5n} \leq 0.05 \]
Solving for \(n\) gives \(-0.5n \leq -\ln(0.05)\) and \(n \geq 2\ln(\frac{0.05}{|H|}) \geq 15.7\)
END SOLUTION
7 Markov Decision Processes

The graph below depicts a Markov Decision Process (MDP) with 5 states $s_1, \ldots, s_5$. The arrows depict actions that take the agent from one state to the other. All actions are deterministic (i.e. the agent always goes to the state the arrow points to if the action is executed). Written inside each node is the reward $R(s)$ the agent receives in this state. Note that the MDP has no terminal state.

1. What is the optimal policy $\pi^*$, if the agent maximizes the sum of rewards with discount factor $\gamma = 0.1$? For each state, put a “check mark” next to the optimal action. 5 pts.

SOLUTION:
Optimal actions: a13, a23, a35, a45, a54
END SOLUTION

2. What is the utility $U^{\pi^*}(s)$ of each state under the policy $\pi^*$ from the previous question. Again, the agent maximizes the sum of rewards with discount factor $\gamma = 0.1$. 5 pts.

SOLUTION:
The utilities are:
- $U^{\pi^*}(s_1) = 0.21111...$
- $U^{\pi^*}(s_2) = -0.78888...$
- $U^{\pi^*}(s_3) = 2.1111...$
- $U^{\pi^*}(s_4) = 1.1111...$
- $U^{\pi^*}(s_5) = 1.1111...$

END SOLUTION
3. Why is Policy Evaluation not directly applicable in some situations where Temporal Difference learning is able to compute the utility function?

SOLUTION:
Policy Evaluation requires explicit knowledge or T and R, which TD learning does not.
END SOLUTION

4. You have a MDP task where you want to find the optimal policy. One question you are pondering is whether to compute and store $U(s)$ or $Q(s, a)$. Name one argument in favor of $U$, and one different argument in favor of $Q$.

SOLUTION:
In favor of $U$:
- need to store fewer values

In favor of $Q$:
- Less computation when executing policy
- Does not need T when executing policy

END SOLUTION
## 8 Knowledge-Based Systems

1. Let \(G(x), F(x), Z(x),\) and \(M(x)\) be the statements "\(x\) is a giraffe," "\(x\) is 15 feet or higher," "\(x\) is animal in this zoo," and "\(x\) belongs to me," respectively. Express each of the following statements in First-Order Logic using \(G(x), F(x), Z(x),\) and \(M(x)\).

   (a) Nothing, except giraffes, can be 15 feet or higher;
   (b) There is no animal in this zoo that does not belong to me;
   (c) I have no animals less than 15 feet high.
   (d) All animals in this zoo are giraffes.

**SOLUTION:**

The following are possible translations:

\[
\begin{align*}
&\forall x (\neg G(x) \rightarrow \neg F(x)) \text{ OR } \forall x (F(x) \rightarrow G(x)) \quad (5) \\
&\neg \exists x (Z(x) \land \neg M(x)) \text{ OR } \forall x (Z(x) \rightarrow M(x)) \quad (6) \\
&\forall x (M(x) \rightarrow F(x)) \quad (7) \\
&\forall x (Z(x) \rightarrow G(x)) \quad (8)
\end{align*}
\]

**END SOLUTION**

2. Show via resolution that \(D\) can be inferred from the following knowledge base:

\[
\begin{align*}
&\neg A \lor \neg B \lor C \\
&A \lor D \\
&B \\
&\neg C
\end{align*}
\]

For each step in the proof, show which disjunctions you are resolving and what the resolvent is.

**SOLUTION:**

Proof by contradiction, so we add \(\neg D\) to the knowledge base. Then apply resolution rule as follows.

(a) \(\text{resolve}(A \lor D, \neg D) = A\)
(b) \(\text{resolve}(A, \neg A \lor \neg B \lor C) = \neg B \lor C\)
(c) \(\text{resolve}(\neg B \lor C, B) = C\)
(d) \(\text{resolve}(C, \neg C) = \emptyset\) which shows that the knowledge base is unsatisfiable, so we can infer \(D\).

**END SOLUTION**
9 Planning

1. Five years from now, you have become a professor at Cornell and it is your turn to teach CS472. At the beginning of the semester, you make a plan of how you are going to teach the class. You know you want to teach “uninformed search”, “informed search”, “machine learning”, “logic”, and “planning”. Among these topics, there are dependencies. To understand planning, you need to know logic. And to understand theorem proving in logic, you need to know uninformed search. To understand informed search, you need to know uninformed search.

(a) Write start and goal for this planning problem in the STRIPS formalism. Use the predicate covered\( (x) \) to indicate that topic \( x \) was covered, and not\_covered\( (x) \) to indicate that it was not taught yet.

(b) Write the STRIPS planning operators, one for each topic.

SOLUTION:
Start: not\_covered\( (us) \), not\_covered\( (is) \), not\_covered\( (ml) \), not\_covered\( (l) \), not\_covered\( (p) \)
Start: covered\( (us) \), covered\( (is) \), covered\( (ml) \), covered\( (l) \), covered\( (p) \)
Operators:
teach\_us: Pre: \{not\_covered\( (us) \)\} Effect: \{\neg not\_covered\( (us) \), covered\( (us) \)\}
teach\_is: Pre: \{not\_covered\( (is) \), covered\( (us) \)\} Effect: \{\neg not\_covered\( (is) \), covered\( (is) \)\}
teach\_ml: Pre: \{not\_covered\( (ml) \)\} Effect: \{\neg not\_covered\( (ml) \), covered\( (ml) \)\}
teach\_l: Pre: \{not\_covered\( (l) \), covered\( (us) \)\} Effect: \{\neg not\_covered\( (l) \), covered\( (l) \)\}
teach\_p: Pre: \{not\_covered\( (p) \), covered\( (l) \)\} Effect: \{\neg not\_covered\( (p) \), covered\( (p) \)\}
END SOLUTION

2. Why would it be beneficial to use a planner that outputs a partially-ordered plan for this problem?

SOLUTION:
Only the necessary order of topics would be specified, but it would be evident how topics could still be moved around.
END SOLUTION