

Final examination information

The final examination is on Thursday December 15 from 3:00–6:00pm in Amos Eaton 214. You may feel free to bring food as long as you clean up after yourself. The examination is closed book and closed notes. You may bring a calculator, but it is not necessary.¹

The examination will be designed to test both:

- conceptual understanding — the ideas behind the algorithms, which algorithm to apply to a problem and what the tradeoffs are
- detailed understanding — the intricacies of how an algorithm works and issues in its implementation.

There will be some questions involving factual recall or simple explanation of concepts or algorithms, but there will also be questions asking you to apply the course material to various problems and situations. You may be asked to extrapolate from course material or apply related concepts to new problems; I think one characteristic of a good examination is that students should learn something from it.

I expect the exam will have 6–9 sections, each with 2–6 questions. Each section focuses on a single topic (e.g., constraint satisfaction or neural networks). The questions in each section may be short-answer questions (i.e., asking for a definition, explanation of some concept, etc.); they may ask you to apply an algorithm to a problem; or some combination of the two.

I will release on the web page the midterm examinations for this class from Fall 1999 and 2000. This is so that you can get a feel for the format of the exam. I have never released previous years' final exams before; I will reconsider this, but I wouldn't count on it. I highly recommend reviewing your quizzes.

Formulas provided on the final examination

- Information

$$I(P(v_1), \dots, P(v_n)) = \sum_i -P(v_i) \log_2 P(v_i) \quad (\text{p. 659})$$

- Bayes classifiers

$$v = \operatorname{argmax}_{v_j \in V} \sum_{h_i \in H} P(v_j | h_i) P(h_i | D)$$

$$v = \operatorname{argmax}_{v_j \in V} P(v_j) \prod_i P(a_i | v_j)$$

$$h = \operatorname{argmax}_{h_i \in H} P(D | h_i) P(h_i)$$

- Sequential decision problems & reinforcement learning:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} T(s, a, s') U(s') \quad (17.5)$$

$$U^\pi(s) \leftarrow U^\pi(s) + \alpha (R(s) + \gamma U^\pi(s') - U^\pi(s)) \quad (21.3)$$

$$U(s) = \max_a Q(a, s) \quad (21.6)$$

$$Q(a, s) \leftarrow Q(a, s) + \alpha [R(s) + \gamma \max_{a'} Q(a', s') - Q(a, s)] \quad (21.8)$$

- Perceptron learning

$$\vec{W} \leftarrow \vec{W} + \alpha \times \vec{I} \times \text{Err}$$

$$W_j \leftarrow W_j + \alpha \times \text{Err} \times g'(in) \times x_j \quad (20.12)$$

- Backpropagation (see handout and text)

$$\vec{W}_j^{i,n} \leftarrow \vec{W}_j^{i,n} + \alpha \times \vec{a}_j \times \Delta_{i,n}$$

$$\vec{W}_k^{j,m} \leftarrow \vec{W}_k^{j,m} + \alpha \times \vec{a}_k \times \Delta_{j,m}$$

$$\Delta_{i,n} = \text{Err}_{i,n} \times g'(in_{i,n})$$

$$\Delta_{j,m} = g'(in_{j,m}) \sum_{n=1}^r W_{j,m}^{i,n} \times \Delta_{i,n}$$

$$g(x) = \frac{1}{1 + e^{-x}}$$

$$g'(x) = g(1 - g)$$

¹You will probably have to do some "simple arithmetic." This means either: multiplying a one digit and a one or two digit number and adding two digit numbers; or working with fractions whose numerator and denominator are one or two digits. If you find yourself doing more complicated arithmetic, you are probably doing something wrong. (Or should use fractions instead of decimals.)

Final examination topics

Introduction	
What is AI?	1.1
Agent structure & environment characteristics	2
Search	
Applying search algorithms to find a sequence of actions	
Formulating search problems	3.1–2
State space versus search tree	3.3
Optimality, completeness, time & space complexity	3.3
Avoiding repeated states (three approaches)	3.5
Blind search (BFS, DFS, DLS, IDS, UCS, BDS)	3.4
Heuristic search	
Greedy search	4.1
A* search	4.1
Heuristic functions	4.2
Admissibility, consistency, dominance	4.1–2
Creating heuristic functions	4.2
Memory bounded versions of A* (IDA*, RBFS, SMA*)	4.1
Local search (iterative improvement algorithms)	4.3
Hill climbing & variations	
Simulated annealing	
Local beam search	
Genetic algorithms	
Constraint satisfaction problems	
Assignment problems	5.1
Constructive approaches	5.2, slides
Blind search approaches	
Backtracking, forward checking	
Heuristics to improve blind search strategies	
Constraint propagation	
Repair approaches	5.3
Min-conflicts heuristic	
Structure-based approaches	5.4
linear ordering, cutset conditioning, tree decomposition	
Game playing search	
MINIMAX search	6.1–2
Perfect vs. imperfect decisions	6.4
Evaluation functions	6.4
Alpha-beta pruning	6.3
Probabilistic games (EXPECTIMAX)	6.5
Other search topics	
Beam search	
Online search: learning real-time A* (LRTA*)	4.5
Sensorless problems (and belief states)	3.6
Logic	
Knowledge representation & logical systems	7.1–3, slides
Inference & entailment	
Soundness & completeness	
Satisfiability	7.4

Logic, continued	
Propositional logic	7.4
Horn normal form	7.5
Conjunctive (and implicative) normal forms	7.5
Satisfiability algorithms (DPLL and WALKSAT)	7.5
Inference in propositional logic	7.5
Forward and backward chaining	
Resolution refutation proofs with set of support strategy	
First order logic	8.1–3
Quantifiers, Inference in first order logic	9.1
Conjunctive normal form	9.5
Skolemization	
Unification	9.2
Forward and backward chaining	9.3–4
Resolution refutation proofs with set of support strategy	9.5
Gödel’s completeness and incompleteness theorems	p. 295 (& 302)
Equality	9.5
Learning	
Introduction	18.1–2
Classification problems, supervised and unsupervised learning	
Decision trees	18.3–4, slides
Basic algorithm, information gain heuristic	
Missing attribute values	
Discretizing continuous-valued attributes	
Overfitting	
Gain ratio, χ^2 pruning	
Rule post pruning	
Probability and Bayesian learning/classifiers	slides, 20.1–2
Probability basics	13, slides
Bayes rule	
Conditional independence	
Combining evidence (Bayesian updating)	
Brute force classifier	
Optimal classifier	
Naive classifier	
m-estimates	
Reinforcement learning	
Introduction	21.1
Utility	16.1–3
Sequential decision problems	17.1–3
Value iteration	
Policy iteration	
Passive reinforcement learning	21.2
Direct utility estimation	
Adaptive dynamic programming	
Temporal differencing	
Active reinforcement learning	21.3
Exploration vs. exploitation	
Adapting passive RL methods	
Q-learning	
Generalization in reinforcement learning	21.4

Learning, continued

Neural networks	20.5
Perceptrons	
Perceptron learning rule	
Representational power of perceptrons	
Sigmoid units	
Multilayer feed-forward networks	
Backpropagation	20.5 & handout
Other learning methods	
Ensemble learning	18.4
Boosting, ADABOOST	
Instance-based learning	20.4
k-nearest neighbor	
Kernel estimator (kernel models)	
Concept learning	
Current-best hypothesis search	19.1
Candidate elimination (version space learning)	19.1

Key algorithms/techniques

The following are the most likely algorithms and techniques to be the subject of a question in which you are asked to solve a specific problem by applying that algorithm or technique.

- Blind search: BFS, DFS, DSL, IDS, UCS, BDS
- Heuristic search: Greedy search, A* search
- Constraint satisfaction search: Constructive methods (with forward checking, constraint propagation, and heuristics), Heuristic repair (with min-conflicts heuristic)
- Game playing search: MINIMAX, alpha-beta pruning, EXPECTIMAX
- Logic: Translating/transforming into logic sentences and into normal forms, forward chaining, backward chaining, resolution refutation proof with set of support strategy
- Decision trees: basic algorithm with information gain heuristic, rule post-pruning, gain ratio
- Artificial neural networks: Perceptron learning rule, backpropagation
- Bayes classifiers: Bayes naive classifier
- Reinforcement learning: value iteration, policy iteration, any of the passive RL methods, Q-learning