Lecture 26: Exam 2 Review

Mark Hasegawa-Johnson

By 余壬、吴鉞描繪，徐顯卿題詠Yu Ren, Wu Yue - 北京故宮博物院藏 《徐顯卿宦跡圖》 The Official Career of Xu Xianqing, Peking Palace Museum, Public Domain, https://commons.wikimedia.org/w/index.php?curid=58327373

Overview

- Date, time, place; What to bring
- Material covered
  - Search
  - Minimax
  - AI safety
  - Theorem-proving
  - Transparency
  - HMM
  - Transformers

Date, time, place

- Monday, April 3, 2023
- 1:00pm
- Here (Lincoln Hall Theater)

- Conflict exams: e-mail address is on course web page

What to bring

- One 8.5x11 (or A4) sheet of notes, both sides, handwritten or printed in font comparable in size to handwriting
- Your UIUC ID

- Not allowed: calculator, computer, textbook, technological tutor

Overview

- Date, time, place; What to bring
- Material covered
  - Search
  - Minimax
  - AI safety
  - Theorem-proving
  - Bayes network
  - Transparency
  - HMM
  - Transformers

Search

Algorithm	Complete?	Optimal?	Time complexity	Space complexity	Implement the Frontier as a…
BFS	Yes	If all step costs are equal	$\mathcal{O}\{b^d\}$	$\mathcal{O}\{b^d\}$	Queue
DFS	No	No	$\mathcal{O}\{b^m\}$	$\mathcal{O}\{bm\}$	Stack
UCS	Yes	Yes	Number of nodes, n, with $g(n) \leq C^*$	Number of nodes, n, with $g(n) \leq C^*$	Priority Queue sorted by $g(n)$
Greedy	No	No	$\mathcal{O}\{b^m\}$	$\mathcal{O}\{b^m\}$	Priority Queue sorted by h(n)
A*	Yes	Yes	Number of nodes, n, with $f(n) \leq C^*$	Number of nodes, n, with $f(n) \leq C^*$	Priority Queue sorted by f(n)

b = branching factor, d = length of best path to goal, m = length of longest path to anywhere,
$g(n)$ =cost of best path from start to node n, C^* = cost of best path to goal,
$h(n)$ =heuristic underestimate of best path from node n to goal, $f(n) = g(n) + h(n)$

A* search

- A* search: Using a heuristic to help choose which node to expand
- Proof that Dijkstra's algorithm is optimal:
  - Expanding nodes in order of increasing cost $\Rightarrow$ first time goal node is expanded it will have the smallest possible cost of any path to goal
- Heuristics that allow A* to be optimal:
  - Consistent: $h(p) \leq d(p, r) + h(r)$
  - Admissible: $h(p) \leq d(p, Goal)$
- Design a consistent heuristic by relaxing constraints

Minimax

- Alternating two-player zero-sum games
  - $\wedge$ = a max node, $\vee$ = a min node
- Minimax search
  - Max: $v = \max(v, \text{child})$. Min: $v = \min(v, \text{child})$
- Limited-horizon computation and heuristic evaluation functions

$$v(s) = w_1 f_1(s) + w_2 f_2(s) + \cdots$$

- Alpha-beta search
  - Max: $v = \max(v, \text{child})$, $\alpha = \max(\alpha, \text{child})$, prune if $\alpha \geq \beta$.
  - Min: $v = \min(v, \text{child})$, $\beta = \min(\beta, \text{child})$, prune if $\alpha \geq \beta$.
- Computational complexity of minimax and alpha-beta
  - Minimax is $O\{b^d\}$. With optimal move ordering, alpha-beta is $O\{b^{d/2}\}$.

AI Safety: Variance Network

Given a dataset of examples $D = \{(x_0, y_0), \ldots, (x_{n-1}, y_{n-1})\}$, the network has two outputs, $f_1(x)$ and $f_2(x)$, trained to minimize:

$$\mathcal{L} = \frac{1}{n}\sum_{i=1}^{n}(f_1(x_i) - y_i)^2 + \frac{1}{n-1}\sum_{i=1}^{n}(f_2(x_i) - (f_1(x_i) - y_i)^2)^2$$

…learns to estimate the conditional mean and conditional variance:

$$f_1(x_i) \xrightarrow[n\to\infty]{} E[Y|X = x_i]$$

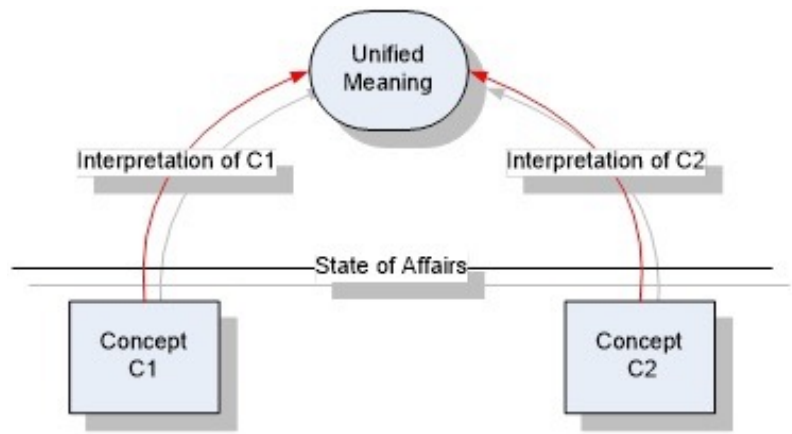
$$f_2(x_i) \xrightarrow[n\to\infty]{} \mathrm{Var}(Y|X = x_i)$$

Unification

Given a proposition P written in terms of the variables $\mathcal{V}_P$ and constants C, and a proposition Q written in terms of the variables $\mathcal{V}_Q$ and constants C, unification is a process with two outcomes:

- Find a substitution $S: \{\mathcal{V}_P, \mathcal{V}_Q\} \rightarrow \{\mathcal{V}_Q, C\}$ such that
- $S(P) = S(Q) = U$

… or prove that no such substitution exists.



Forward-chaining

Forward-chaining is a method of proving a theorem, T:

- Starting state: a database of known true propositions, $\mathcal{D} = \{P_1, P_2, \dots\}$
- Actions: the set of possible actions is defined by a set of rules, where each rule has the form $P \Longrightarrow Q$.
- Neighboring states: if P_1 unifies to P creating $\mathrm{S}(P) = \mathrm{S}(P_1)$, then create the new database $\mathcal{D}' = \{P_1, P_2, \dots, S(Q)\}$
- Termination: search terminates when we find a database containing T

Backward-chaining

Backward-chaining is a method of proving a theorem, T:

- Starting state: a goalset containing only one goal, the result to be proven, $\mathcal{G} = \{T\}$
- Actions: the set of possible actions is defined by rules of the form $P_1 \wedge P_2 \wedge \cdots \wedge P_n \Longrightarrow Q$
- Neighboring states: if Q unifies with some $Q' \in \mathcal{G}$ producing $\mathrm{S}(Q) = \mathrm{S}(Q')$ then:
  - Remove Q' from $\mathcal{G}$
  - Replace it with $\mathrm{S}(P_1) \wedge S(P_2) \wedge \cdots \wedge S(P_n)$
- Termination: search terminates if all propositions in the goalset are known to be true.

Bayesian Networks

- Bayesian network: Each variable is a node; An arrow between two nodes means that the child depends on the parent.
- Inference using a Bayesian network:

$$P(B = T, J = T) = \sum_{e=T}^{F} \sum_{a=T}^{F} P(B$$
$$= T)P(E = e)P(A = a|B = T, E = e)P(J = T|A = a)$$

- Key ideas:
  - Independent = no common ancestors
  - Conditionally independent = (1) no common descendants, and (2) none of the descendants of one are ancestors of the other

Relevance scoring in neural networks

- Unnormalized relevance:

$$\tilde{R}(f_c, x_d) = \frac{\partial f_c}{\partial x_d} x_d f_c$$

- Normalized relevance:

$$R(f_c, x_d) = \frac{\frac{\partial f_c}{\partial x_d} x_d}{\sum_{d'} \frac{\partial f_c}{\partial x_{d'}} x_{d'}} f_c$$

The summation is usually done over a layer, so that $\sum_d R(f_c, x_d) = f_c$ over each layer.

Hidden Markov Models

- HMM: Probabilistic reasoning over time

$$\pi_i = P(Y_0 = i)$$
$$a_{i,j} = P(Y_t = j | Y_{t-1} = i)$$
$$b_j(x_t) = P(X_t = x_t | Y_t = j)$$

- Viterbi algorithm

$$v_t(j) = \max_{i \in \mathcal{Y}} v_{t-1}(i)\, a_{i,j} b_j(x_t)$$
$$\psi_t(j) = \operatorname*{argmax}_{i \in \mathcal{Y}} v_{t-1}(i)\, a_{i,j} b_j(x_t)$$

Vector Semantics

- Skip-Gram:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c,j\neq 0}^{c} \ln P\left(w_{t+j}|w_t\right)$$

- Continuous Bag of Words (CBOW):

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c,j\neq 0}^{c} \ln P\left(w_t|w_{t+j}\right)$$

- Softmax probability, Dot-product similarity:

$$P\left(W_t = m|W_{t+j} = n\right) = \frac{\exp(v_m @ v_n)}{\sum_{m'} \exp(v_{m'} @ v_n)}$$

- Train using SGD:

$$v_m \leftarrow v_m - \eta \nabla_{v_m}\mathcal{L} = v_m + \frac{\eta}{T}\sum_{t:w_t=m}\sum_{j=-c,j\neq 0}^{c} \left(1 - P\left(W_t = m|w_{t+j}\right)\right) v_{w_{t+j}}$$

Transformer

- Stack up v_t, k_t, and q_i into matrices:

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix}, k = \begin{bmatrix} k_1 \\ \vdots \\ k_n \end{bmatrix}, q = \begin{bmatrix} q_1 \\ \vdots \\ q_m \end{bmatrix}$$

- $\alpha_{i,t}$ is the t[th] output of a softmax whose input vector is $q_i@k^T$:

$$\alpha_{i,t} = \text{softmax}_t(q_i@k^T) = \frac{\exp(q_i@k_t)}{\sum_\tau \exp(q_i@k_\tau)}$$

- c_i is the product of the vector $\text{softmax}(q_i@k^T)$ times the v matrix:

$$c_i = \text{softmax}(q_i@k^T)@v = \sum_t \alpha_{i,t}\, v_t$$

Overview

- Date, time, place; What to bring
- Material covered
    - Search
    - Minimax
    - AI safety
    - Theorem-proving
    - Bayes Network
    - Transparency
    - HMM
    - Transformers