Artificial Intelligence

CS 165A

Dec 10, 2020

Instructor: Prof. Yu-Xiang Wang

→ Final Review
Logistic notes

• ESCI Survey: Please go and submit your feedback!
  – Deadline is approaching. This is my final reminder.

• Final: Next Tuesday 9:00 am - Wednesday 11:59 pm.
  – 27 hours in total; for an exam that will take roughly 3 hours max.
  – Submit your take home final on gradescope.
  – Open book. NO collaboration allowed. We will check for similarities. Your questions might be subtly different from your peers.
  – Covers topics up to First Order Logic (but before FOL inference)
  – About 80% will be on topics after the midterm, 20% on earlier topics. (Note that you might be asked to apply ML or PGM on topics in the second half of the lecture!)
  – There
Tips for studying for the final

• Focus on the Lectures and HWs

• For any concepts that you are confused, check the textbook (Again, books are random access, you don’t have to read chapters from the beginning to the end)
  – Stick to AIMA book for Search and Logic
  – Stick to the Sutton and Barto book for RL.
We’ve come a long way…

<table>
<thead>
<tr>
<th>Week</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Course Overview &amp; Intelligent Agents</td>
</tr>
<tr>
<td>2</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>2</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>2</td>
<td>Machine Learning</td>
</tr>
<tr>
<td>2</td>
<td>Probabilistic Graphical Models</td>
</tr>
<tr>
<td>3</td>
<td>Probabilistic Graphical Models</td>
</tr>
<tr>
<td>3</td>
<td>Search: Problem solving with search</td>
</tr>
<tr>
<td>4</td>
<td>Search: Search algorithms</td>
</tr>
<tr>
<td>4</td>
<td>Search: Minimax search and game playing</td>
</tr>
<tr>
<td>5</td>
<td>Midterm Review</td>
</tr>
<tr>
<td>5</td>
<td>Midterm (take-home)</td>
</tr>
<tr>
<td>6</td>
<td>RL: Intro / Markov Decision Processes</td>
</tr>
<tr>
<td>6</td>
<td>RL: Solving MDPs</td>
</tr>
<tr>
<td>7</td>
<td>RL: Bandits and Exploration</td>
</tr>
<tr>
<td>7</td>
<td>RL: Reinforcement Learning Algorithms</td>
</tr>
<tr>
<td>8</td>
<td>RL: Reinforcement Learning Algorithms</td>
</tr>
<tr>
<td>8</td>
<td>Logic: Propositional Logic</td>
</tr>
<tr>
<td>9</td>
<td>Thanksgiving break</td>
</tr>
<tr>
<td>9</td>
<td>Logic: First Order Logic</td>
</tr>
<tr>
<td>10</td>
<td>Responsible AI</td>
</tr>
<tr>
<td>10</td>
<td>Final Review</td>
</tr>
<tr>
<td>11</td>
<td>Final Exam (take-home)</td>
</tr>
</tbody>
</table>

- **Machine Learning**
- **Probabilistic Reasoning**
- **Search**
- **Reinforcement Learning**
- **Logic**
- **Responsible AI**
Lecture 1: AI Overview

- AI for problem solving
- Rational agents
- Examples of AI in the real world
- Modelling-Inference-Learning Paradigm
Modeling-Inference-Learning

Modeling

Inference

Learning
Structure of the course

Probabilistic Graphical Models / Deep Neural Networks

Reflex Agents
- Classification / Regression
- Bandits

Planning Agents
- Search
- game playing

Reasoning agents
- Markov Decision Processes
- Reinforcement Learning
- Logic, knowledge base
- Probabilistic inference

Low-level intelligence

High-level intelligence

Machine Learning

Potential question in the final: what type of agents are suitable to a given problem?
Our view of AI

• So this course is about designing rational agents
  – Constructing $f$
  – For a given class of environments and tasks, we seek the agent (or class of agents) with the “best” performance
  – Note: Computational limitations make complete rationality unachievable in most cases

• In practice, we will focus on problem-solving techniques (ways of constructing $f$), not agents per se
Different Ways of Looking at the AI

• Agent types / level of intelligence
  – Low-level: Reflex agents
  – Mid-level: Goal-based / Utility-based agents: planning agents
  – High-level: Knowledge-based: Logic agents

• Optimization view
  – Everything is an optimization problem

• Theoretical aspects
  – Time/space complexity
  – Algorithms and data structures
  – Statistical properties: # of free parameters / how easily can we learn them with data
Optimization perspective of AI

• A rational agent \( \max_{a_1, \ldots, a_T} \text{Utility}(a_1, \ldots, a_T) \)
  – Modelling tools: Features / Hypothesis, PGM, State-space abstraction, agent categorization
  – Constraints: Computation, Data, Storage

• Discrete optimization \( \min_{\mathcal{P} \in \text{Paths}} \text{Distance}(\mathcal{P}) \)
  – Algorithmic tool: Search / Dynamic programming

• Continuous optimization \( \min_{\theta \in \mathbb{R}^d} \text{TrainingError}(\theta) \)
  – Algorithmic tool: Gradient descent / Stochastic gradient descent
Different objectives to optimize in AI (first half of the course)

• PGM:
  – MLE: Maximize the log-likelihood function
  – Classifier / decision: max the posterior distribution

• Search and planning:
  – Find valid solutions with smallest path cost.
  – Minimax search / games: Maximize your worst-case pay-off (assuming your opponent plays optimally)

• Machine Learning
  \[
  \min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i)) + \lambda r(\theta)
  \]
  – (Regularized) Empirical Risk Minimization (ERM):
  – But the goal is to minimize the (unseen) expected loss.
Different objectives to optimize in AI (second half of the course)

• Markov Decision Processes / RL
  – Maximize the cumulative expected reward of “decision policy”
  – Balance Exploration and Exploitation.

• Logic / Knowledge based agent:
  – Solve a feasibility problem, find a “proof”.
  – Determining “Valid, Satisfiable, Unsatisfiable”
A lot of these problems are computationally / statistically hard, but so what?

• Get help from human:
  – Use a model
  – Use abstractions at the right level
  – Use features
  – Use heuristic functions

• Get help from mathematics and computer science theory:
  – Solve an approx. solution
  – Approximate inference of a PGM
  – More iterations with less accuracy per SGD
  – A* Search
  – TD learning and Bootstrapping
Lecture 2-4: Machine Learning

• Machine learning
  – Types of machine learning models
  – Focus on Supervised Learning ---- classifier agents.

• What is a feature vector?
  – Feature engineering, feature extraction

• Hypothesis class and free parameters
  – How many are there? How to evaluate a classifier?
  – Error? On training data or on new data?
  – Overfitting, underfitting?

• How to learn (optimize)?
  – Surrogate losses, Gradient Descent, SGD
Example of a feature vector of dimension 4

Contains hyperlinks

Proportion of misspelled words

Whether the contact list

Length of the message

<table>
<thead>
<tr>
<th>Contains hyperlinks</th>
<th>Proportion of misspelled words</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.0375</td>
</tr>
<tr>
<td>80</td>
<td></td>
</tr>
</tbody>
</table>

Step 1 in Modelling
Feature extractor:
Converting the object of interest to a vector of numerical values.
Example: Linear classifiers

• Score(x) = \( w_0 + w_1 \times 1(\text{hyperlinks}) + w_2 \times 1(\text{contact list}) + w_3 \times \text{misspelling} + w_4 \times \text{length} \)

• A linear classifier: \( h(x) = 1 \) if \( \text{Score}(x) > 0 \) and 0 otherwise.

• Question: What are the “free-parameters” in a linear classifier?
  – If we redefine \( \mathcal{Y} = \{-1, 1\} \)
  – A compact representation:

    \[
    h(x) = \text{sign}(\mathbf{w}^T [1; x])
    \]
Just “relax”: relaxing a hard problem into an easier one

$$\min_{w \in \mathbb{R}^d} \text{Error}(w) = \frac{1}{n} \sum_{i=1}^{n} 1(\text{sign}(w^T x_i) \neq y_i)$$

$$\min_{w \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(w^T x_i, y_i).$$
Illustration of GD vs SGD

Observation: With the time gradient descent taking one step, SGD would have already moved many steps.
One natural stochastic gradient to consider in machine learning

• Recall that

\[
\min_{\theta \in \mathbb{R}^d} \frac{1}{n} \sum_{i=1}^{n} \ell(\theta, (x_i, y_i))
\]

• Pick a **single** data point \(i\) uniformly at random

  – Use \(\nabla_{\theta} \ell(\theta, (x_i, y_i))\)

  – Show that this is an unbiased estimator!
  – Know which part of the expression is random!
  – Know how to apply the definition of expectation.
Lecture 5-6: Probabilities and BayesNet

• Modelling the world with a joint probability distribution
  – Number of parameters?

• CPTs
  – Count number of independent numbers to represent a CPT

• Conditional, Marginal, Probabilistic Inference with Bayes Rule

• Read off conditional independences from the graph
  – d-separation
  – Bayes ball algorithm
  – Markov Blanket
Tradeoffs in our model choices

Fully Independent
\[ P(X_1, X_2, \ldots, X_N) = P(X_1) P(X_2) \cdots P(X_N) \]
\[ O(N) \]

Fully general
\[ P(X_1, X_2, \ldots, X_N) \]
\[ O(e^N) \]

Space / computation efficiency
Expressiveness

Idea:
1. Independent groups of variables?
2. Conditional independences?
Question: How to fill values into these CPTs?
Ans: Specify by hands. Learn from data (e.g., MLE).

What are the CPTs? What are their dimensions?
Big question: Is there a general way that we can answer questions about conditional independences by just inspecting the graphs?

- Turns out the answer is “Yes!”
What are the probabilistic graphical models for topics we learned in the second half?

- Expectimax
- MDP
- Bandits / Contextual Bandits
- Reinforcement Learning
Lecture 7-10: Search

• Problem solving by search
  – Abstraction, problem formulation
  – State-space diagram
  – Count the number of states, number of actions.

• Uniformed Search algorithms
  – Four evaluation criteria

• Informed (heuristic) search
  – Admissible / consistent heuristics
  – Tree-search vs graph search

• Minimax Search and Game playing
  – Know how to do minimax / expectimax by hand!
  – Pruning
Problem Formulation and Search

- Problem formulation
  - State-space description $< \{S\}, S_0, \{S_G\}, \{O\}, \{g\} >$
    - $S$: Possible states
    - $S_0$: Initial state of the agent
    - $S_G$: Goal state(s)
      - Or equivalently, a goal test $G(S)$
    - $O$: Operators $O: \{S\} \Rightarrow \{S\}$
      - Describes the possible actions of the agent
    - $g$: Path cost function, assigns a cost to a path/action
  - At any given time, which possible action $O_i$ is best?
    - Depends on the goal, the path cost function, the future sequence of actions….
- Agent’s strategy: Formulate, Search, and Execute
  - This is offline problem solving
State-Space Diagrams

- State-space description can be represented by a state-space diagram, which shows
  - States (incl. initial and goal)
  - Operators/actions (state transitions)
  - Path costs
State Space vs. Search Tree (cont.)

Search tree (partially expanded)
Minimax example

Which move to choose?

The \textbf{minimax decision} is move $A_1$
Alpha pruning

\[ A \geq 10 \]

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

\[ \begin{align*}
  &10 \\
  &25 \\
  &15 \\
  &5
\end{align*} \]
Beta pruning

![Diagram of a decision tree with nodes A, B, C, and D, illustrating beta pruning.](Image)

- Node A is the root with a MIN operator and a value of 25.
- Node B, with a value of 25, is connected to nodes 10, 25, and 15.
- Node C, with a value of 50, is connected to nodes 25 and 15.
- Node D, with a MAX operator, is connected to nodes 25 and 15, with the left branch pruned at 25.

The pruned subtree includes nodes 10 and 25, with a value of 25.
Your opponent behave randomly with a given probability distribution,

If you move left, your opponent will select actions with probability [0.5,0.5]

If you move right, your opponent will select actions with [0.6,0.4]
Lecture 11-16 Reinforcement Learning

• Markov Decision Processes
  – New concepts: reward, value function, policy, transition dynamics
  – Bellman equations
  – Iterative algorithms for finding the optimal policy

• Bandits / Contextual bandits
  – The notion of regret
  – Explore-exploit

• Reinforcement Learning
  – Model-based learning
  – Model-free learning
  – Bootstrapping with Temporal Difference Learning
Reinforcement learning problem setup

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents might not even observe the state
Reinforcement learning problem setup

- State, Action, Reward and Observation
  \[ S_t \in S \quad A_t \in A \quad R_t \in \mathbb{R} \quad O_t \in O \]

- Policy:
  - When the state is observable: \( \pi : S \rightarrow A \)
  - Or when the state is not observable
    \[ \pi_t : (O \times A \times \mathbb{R})^{t-1} \rightarrow A \]

- Learn the best policy that maximizes the expected reward
  - Finite horizon (episodic) RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{T} R_t \right] \)
  - Infinite horizon RL: \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right] \)

\( \gamma \): discount factor

\( T \): horizon
Reinforcement learning is, arguably, the most general AI framework.

- **RL**: State, Action, Reward, Nothing is known.

- **Simplified RL models:**
  - iid state $\rightarrow$ Contextual bandits
  - No state, tabular action $\rightarrow$ Multi-arm bandits
  - iid state, no reward $\rightarrow$ Supervised Learning
  - Known dynamics / reward $\rightarrow$ Markov Decision Processes (Control/Cybernetics)
  - No reward / Unknown dynamics $\rightarrow$ System Identification
Let us tackle different aspects of the RL problem one at a time

• Markov Decision Processes:
  – Dynamics are given no need to learn

• Bandits: Explore-Exploit in simple settings
  – RL without dynamics

• Full Reinforcement Learning
  – Learning MDPs
Tabular MDP

- **Discrete State, Discrete Action, Reward and Observation**
  
  \[ S_t \in \mathcal{S} \quad A_t \in \mathcal{A} \quad R_t \in \mathbb{R} \quad O_t \in \mathcal{O} \]

- **Policy:**
  - When the state is observable: \( \pi : \mathcal{S} \rightarrow \mathcal{A} \)
  - Or when the state is not observable
    \[ \pi_t : (\mathcal{O} \times \mathcal{A} \times \mathbb{R})^{t-1} \rightarrow \mathcal{A} \]

- **Learn the best policy that maximizes the expected reward**
  - **Finite horizon (episodic) MDP:** \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{T} R_t \right] \)
  - **Infinite horizon MDP:** \( \pi^* = \arg \max_{\pi \in \Pi} \mathbb{E} \left[ \sum_{t=1}^{\infty} \gamma^{t-1} R_t \right] \)

\( \gamma \): discount factor
Reward function and Value functions

- Immediate reward function $r(s, a, s')$
  - expected immediate reward
  \[ r(s, a, s') = \mathbb{E}[R_1 | S_1 = s, A_1 = a, S_2 = s'] \]
  \[ r^\pi(s) = \mathbb{E}_{a \sim \pi(a | s)}[R_1 | S_1 = s] \]

- state value function: $V^\pi(s)$
  - expected long-term return when starting in $s$ and following $\pi$
  \[ V^\pi(s) = \mathbb{E}_\pi[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s] \]

- state-action value function: $Q^\pi(s, a)$
  - expected long-term return when starting in $s$, performing $a$, and following $\pi$
  \[ Q^\pi(s, a) = \mathbb{E}_\pi[R_1 + \gamma R_2 + ... + \gamma^{t-1} R_t + ... | S_1 = s, A_1 = a] \]
Bellman equations – the fundamental equations of MDP and RL

- An alternative, recursive and more useful way of defining the $V$-function and $Q$ function
  - $V^\pi$ function Bellman equation
    \[ V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^\pi(s')] \]
  - $Q^\pi$ function Bellman equation
    \[ Q^\pi(s, a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \sum_{a'} \pi(a'|s')Q^\pi(s', a')] \]
  - $V^*$ function Bellman (optimality) equation
    \[ V^*(s) = \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V^*(s')] \]
  - $Q^*$ function Bellman (optimality) equation
    \[ Q^*(s, a) = \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma \max_{a'} Q^*(s', a')] \]
Let's work out the Value function for a specific policy

actions: UP, DOWN, LEFT, RIGHT

- reward +1 at [4,3], -1 at [4,2]
- reward -0.04 for each step

\[
V^\pi(s) = \sum_a \pi(a|s) \sum_{s'} P(s'|s, a)[r(s, a, s') + \gamma V^\pi(s')] = \sum_a \pi(a|s)Q^\pi(s, a)
\]

\[
1.0 + 0.8 \times (+1 - 0.04 + 0) \\
+ 0.1 \times (-0.04 + V^\pi([3,2])) \\
+ 0.1 \times (-0.04 + V^\pi([3,3]))
\]
Inference problem: given an MDP, how to compute its optimal policy?

• It suffices to compute its Q* function, because:

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

• It suffices to compute its V* function, because:

$$Q^*(s, a) = \sum_{s'} P(s'|s, a)[r(s, a, s') + \gamma V^*(s')]$$
MDP inference problem: Policy Evaluation (prediction) vs Policy Optimization (control)

• Policy Evaluation (prediction):
  – Simulate Bellman equation w.r.t. policy $\pi$ until it converges

• Policy Optimization (control):
  – Policy evaluation, policy improvement, PE, PI, …
  – Value iterations: simulate Bellman optimality equation
How to calculate value functions given MDP parameters? Policy Iterations and Value Iterations

• What are these algorithms for?
  – Algorithms of computing the $V^*$ and $Q^*$ functions from MDP parameters

• Policy Iterations

  \[ \pi_0 \xrightarrow{E} V^{\pi_0} \xrightarrow{I} \pi_1 \xrightarrow{E} V^{\pi_1} \xrightarrow{I} \ldots \xrightarrow{I} \pi^* \xrightarrow{E} V^* \]

• Value iterations

  \[ V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s, a) [r(s, a, s') + \gamma V_k(s')] \]

• How do we make sense of them?
  – Recursively applying the Bellman equations until convergence.
Multi-arm bandits: Problem setup

• No state. k-actions \( a \in \mathcal{A} = \{1, 2, \ldots, k\} \)

• You decide which arm to pull in every iteration

\[ A_1, A_2, \ldots, A_T \]

• You collect a cumulative payoff of \( \sum_{t=1}^{T} R_t \)

• The goal of the agent is to maximize the expected payoff.
  – For future payoffs?
  – For the expected cumulative payoff?
How do we measure the performance of an online learning agent?

- The notion of “Regret”:
  - I wish I have done things differently.
  - Comparing to the best actions in the hindsight, how much worse did I do.

- For MAB, the regret is defined as follow

\[
T \max_{a \in [k]} \mathbb{E}[R_t | a] - \sum_{t=1}^{T} \mathbb{E}_{a \sim \pi}[\mathbb{E}[R_t | a]]
\]
Regret of different MAB algorithms

- Greedy: $O(T)$
- ExploreFirst: $O(T^{2/3} k^{1/3})$
- eps-Greedy: $O(T^{2/3} k^{1/3})$
- Upper Confidence Bound: $O(T^{1/2} k^{1/2})$
RL algorithms

- Model-based approach (plug-in an empirically estimated MDP, run VI / PI)

- Model-free approach:
  - Monte Carlo (average converges to mean) e.g., First visit Monte Carlo
  - Combining Monte Carlo with Dynamic Programming (e.g., VI)
  - Temporal difference learning
Revisit the dynamic programming approach

- **Policy Evaluation**

\[ V_{k+1}^\pi(s) \leftarrow \sum_a \pi(a|s) \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^\pi(s')] \]

- **Policy improvement**

\[ \pi'(s) = \arg \max_a Q^\pi(s,a) = \arg \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k^\pi(s')] \]

- **Value iterations**

\[ V_{k+1}(s) \leftarrow \max_a \sum_{s'} P(s'|s,a)[r(s,a,s') + \gamma V_k(s')] \]

*We do not have the MDP parameters in RL!*
Reinforcement learning agents have “online” access to an environment

- State, Action, Reward
- Unknown reward function, unknown state-transitions.
- Agents can “act” and “experiment”, rather than only doing offline planning.
TD policy optimization (TD-control)

- **SARSA (On-Policy TD-control)**
  - Update the Q function by bootstrapping Bellman Equation
    \[ Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma Q(S', A') - Q(S, A)] \]
  - Choose the next A' using Q, e.g., eps-greedy.

- **Q-Learning (Off-policy TD-control)**
  - Update the Q function by bootstrapping Bellman Optimality Eq.
    \[ Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)] \]
  - Choose the next A' using Q, e.g., eps-greedy, or any other policy.
Q-Learning with function approximation

\[ Q(s, a) = w_1 f_1(s, a) + w_2 f_2(s, a) + \ldots + w_n f_n(s, a) \]

• Q-learning with linear Q-functions:

  \[
  \text{transition} = (s, a, r, s') \\
  \text{difference} = [r + \gamma \max_{a'} Q(s', a')] - Q(s, a) \\
  Q(s, a) \leftarrow Q(s, a) + \alpha \text{[difference]} \quad \text{Exact Q’s} \\
  w_i \leftarrow w_i + \alpha \text{[difference]} f_i(s, a) \quad \text{Approximate Q’s}
  \]

• Intuitive interpretation:
  – Adjust weights of active features
  – E.g., if something unexpectedly bad happens, blame the features that were on: disprefer all states with that state’s features

• Formal justification: online least squares (Read the textbook!)
Policy gradient

- Let’s not worry about states, dynamics, Q function.
  - We might not even observe the true state.
  - Let’s specify a class of parametrized policy and hope to compare to the best within this class

- Objective function to maximize: \( J(\theta) = v_{\pi_\theta}(s_0) \),

- Do SGD: \( \theta_{t+1} = \theta_t + \alpha \nabla J(\theta_t) \),

- Policy gradient theorem:
  \[
  \nabla J(\theta) = \sum_s d^\pi(s) \sum_a Q^\pi(s, a) \nabla_\theta \pi(a | s, \theta)
  \]

*Note how this theorem is non-trivial… The first two terms depends on \( \pi \), but we did not take the gradient w.r.t. them.*
Special case of policy gradient theorem in contextual bandits?

• Gradient of the IPS-estimator (or Importance Sampling estimator that you’ve seen) w.r.t. the parameter $\theta$ of the policy $\pi$

$$\hat{\nabla}_\pi \widehat{\nu}_{\text{IPS}} = \frac{1}{n} \sum_{i=1}^{n} \frac{\pi(a_i|x_i)}{\mu(a_i|x_i)} r_i$$

• Connections to the policy gradient?
Most important concepts in MDP / RL

• Make sure you understand
  – Problem setup, evaluation criteria
  – Definition of the policy, state, action, immediate reward, value, value function …

• Bellman equations

• Policy / Value iterations
  – Finding fixed points of Bellman equations
  – Finding eigenvector of a matrix

• SARSA, Q-Learning
  – SGD-style Stochastic simulation of Bellman equations
Lecture 16-17: Logic

• Logic agent
  – Know how to play, e.g., Minesweeper and know how to explain your reasons.

• Knowledge Base
  – Tell operation
  – Ask operation

• Components of a formal mathematical logic system
  – Syntax, Semantics

• Inference Algorithms.
KB Agents

- Need a formal logic system to work
- Need a data structure to represent known facts
- Need an algorithm to answer ASK questions
Syntax and semantics

• Two components of a logic system

• Syntax --- How to construct sentences
  – The symbols
  – The operators that connect symbols together
  – A precedence ordering

• Semantics --- Rules the assignment of sentences to truth
  – For every possible worlds (or “models” in logic jargon)
  – The truth table is a semantics
Entailment

A is entailed by B, if A is true in all possible worlds consistent with B under the semantics.
Inference procedure

• Inference procedure
  – Rules (algorithms) that we apply (often recursively) to derive truth from other truth.
  – Could be specific to a particular set of semantics, a particular realization of the world.

• Soundness and completeness of an inference procedure
  – Soundness: All truth discovered are valid.
  – Completeness: All truth that are entailed can be discovered.
Propositional Logic

• Syntax:
  – True, false, propositional symbols
  – ( ), ¬ (not), ∧ (and), ∨ (or), ⇒ (implies), ⇔ (equivalent)

• Semantics:
  – Five rules (the following truth table)

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>Q</td>
<td>¬P</td>
<td>P ∧ Q</td>
<td>P ∨ Q</td>
<td>P ⇒ Q</td>
<td>P ⇔ Q</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>----</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
<td>--------</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>False</td>
</tr>
<tr>
<td>True</td>
<td>True</td>
<td>False</td>
<td>True</td>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
</tbody>
</table>

• Inference rules:
  – Modus Pronens etc. Most important: Resolution
Propositional logic agent

• Representation: Conjunctive Normal Forms
  – Represent them in a data structure: a list, a heap, a tree?
  – Efficient TELL operation

• Inference: Solve ASK question
  – Use “Resolution” only on CNFs is Sound and Complete.
  – Equivalent to SAT, NP-complete, but good heuristics / practical algorithms exist

• Possible answers to ASK:
  – Valid, Satisfiable, Unsatisfiable
First order logic

• More expressive language
  – Relations and functions of objects.
  – Quantifiers such as, All, Exists.

• Easier to construct a KB.
  – Need much smaller number of sentences to capture a domain.

• Follow the same structure: Symbols, Semantics

• Dedicated inference algorithms

• (FOL inference is not covered in the Final)
Potential types of questions in FOL

- Translate FOL sentence to natural language or the other way round.

- Translate the rule of a simple game to FOL.
Lecture 18: Responsible AI

• What are the typical pitfalls in AI applications
  – Privacy: Data sharing, data use, data ownership
  – Fairness of AI Decision making: Recidivism prediction, Admission / Recruiting
  – Polarizing effects of recommendation systems
  – Fake news / fake videos
  – Social impacts: unemployment / rich gets richer
  – Robustness and Safety: adversarial examples, self-driving cars

• What are your thoughts on overcoming them?
  – Potentially a case / short essay question.

• Your thoughts on Weak AI vs Strong AI.
  – And artificial general Intelligence…
Thank you and stay in touch!

- It’s a challenging quarter for everyone of us.
- It’s my pleasure to work with you!
- I hope the course is / will be useful.

- **AI Research at UCSB**
  - Machine Learning Lab
  - Natural Language Processing Lab
  - Center for Responsible Machine Learning
  - Center for Information Technology and Society
  - The Mellichamp Initiative in Mind & Machine Intelligence
  - Data Science Initiative