

Oct 5 Decision-making

Let's recap last time! What does the Minimax Algorithm do?

Announcements and To-Dos

Announcements:

1. “Exam” planned to be posted tomorrow or Wednesday.

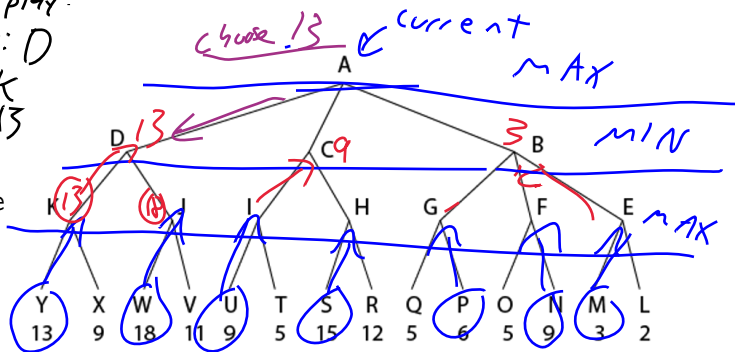
Last time we learned:

1. An in-class notebook, playing with Genetic Algorithm.

Minimax Cont'd

Game play:
 MAX: D
 MIN: K
 MAX: 13

The minimax decision in any given state s is the action that is the optimal choice for Max/Min, because it leads to the highest/lowest minimax value.



We define:

$$\text{minimax}(s) = \begin{cases} \text{utility}(s) & \text{if } s \text{ is terminal} \\ \max_{a \in \text{actions}(s)} & \text{if it is Max's turn} \\ \min_{a \in \text{actions}(s)} & \text{if it is Min's turn} \end{cases}$$

Minimax Code

```
def minimax_decision(state):
    all_actions = what are the available actions?
    best_action = action that maximizes min_value(result(action, state))
    return best_action

def min_value(state):
    if terminal_state(state):
        return utility(state)
    value = infinity
    for action in all available actions:
        value = min(value, max_value(result(action, state)))
    return value

def max_value(state):
    if terminal_state(state):
        return utility(state)
    value = -infinity
    for action in all available actions:
        value = max(value, min_value(result(action, state)))
    return value
```

Minimax Code

```
function MINIMAX-DECISION(state) returns an action
  return  $\arg \max_{a \in \text{ACTIONS}(s)} \text{MIN-VALUE}(\text{RESULT}(\text{state}, a))$ 
```

```
function MAX-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow -\infty$ 
  for each a in ACTIONS(state) do
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a)))$ 
  return v
```

```
function MIN-VALUE(state) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
   $v \leftarrow \infty$ 
  for each a in ACTIONS(state) do
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a)))$ 
  return v
```

Figure 5.3 An algorithm for calculating minimax decisions. It returns the action corresponding to the best possible move, that is, the move that leads to the outcome with the best utility, under the assumption that the opponent plays to minimize utility. The functions MAX-VALUE and MIN-VALUE go through the whole game tree, all the way to the leaves, to determine the backed-up value of a state. The notation $\arg \max_{a \in S} f(a)$ computes the element *a* of set *S* that has the maximum value of *f(a)*.

Minimax Complexity

The number of games states is exponential in the number of moves at any given moment: this set of actions will serve as a *branching factor* for the state tree.

So we **don't examine every node**.

The **Alpha-Beta** pruning algorithm take similar ideas to e.g. Uniform-Cost expansion and applies it to the minimax problem.

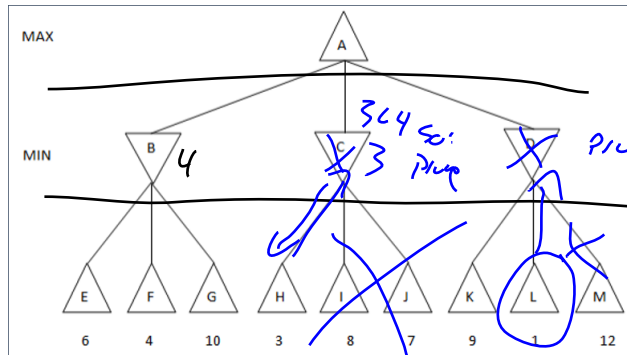
1. Remove branches that do not influence our final decision.
2. Idea; you can bracket the best/worst (or highest/lowest) values at each node, even before all of its successors have been evaluated.

Alpha-Beta Pruning

To avoid doing a full search - which is $O(b^m)$ with maximum depth m and branching factor b - we know that under optimal play some branches can be discarded.

We call that *pruning* those branches.

On the tree to the right, the MIN value of B is 4. When we consider node C , we discover a candidate min of $H = 3$ which makes C worse to move to (from A) *regardless* of what's in I, J . We can prune those branches.



Alpha-Beta Formalism

Using Alpha-Beta pruning in practice:

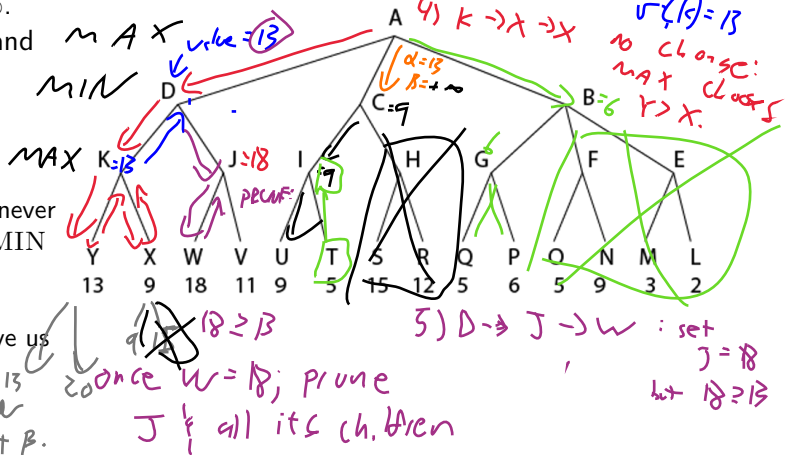
1. Start with $\alpha = -\infty$, $\beta = \infty$.

These are the current best and worst case estimates for the problem as a whole.

2. Prune if:

2.1 While exploring a MIN, $v \leq \alpha$. This means we'd never choose to move to that MIN

2.2 While exploring a MAX, $v \geq \beta$. This means MIN would never choose to give us that option.



Alpha-Beta Formalism

```
def alphabeta_search(state):
    alpha = -infinity
    beta = +infinity
    value = max_value(state, alpha, beta)
    best_action = action that has utility=value to Max
    return

def max_value(state, alpha, beta):
    if terminal_state(state):
        return utility(state)
    value = -infinity
    for action in all available actions:
        value = max(value, min_value(result(action, state), alpha, beta))
        if value >= beta: return value
        alpha = max(value, alpha)
    return value

def min_value(state, alpha, beta):
    if terminal_state(state):
        return utility(state)
    value = +infinity
    for action in all available actions:
        value = min(value, max_value(result(action, state), alpha, beta))
        if value <= alpha: return value
        beta = min(value, beta)
    return value
```

Alpha-Beta Formalism

function ALPHA-BETA-SEARCH(*state*) **returns** an action
 $v \leftarrow \text{MAX-VALUE}(\text{state}, -\infty, +\infty)$
return the *action* in $\text{ACTIONS}(\text{state})$ with value v

function MAX-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow -\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \geq \beta$ **then return** v
 $\alpha \leftarrow \text{MAX}(\alpha, v)$
return v

function MIN-VALUE(*state*, α , β) **returns** a utility value
if $\text{TERMINAL-TEST}(\text{state})$ **then return** $\text{UTILITY}(\text{state})$
 $v \leftarrow +\infty$
for each a **in** $\text{ACTIONS}(\text{state})$ **do**
 $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(\text{RESULT}(s, a), \alpha, \beta))$
if $v \leq \alpha$ **then return** v
 $\beta \leftarrow \text{MIN}(\beta, v)$
return v

Figure 5.7 The alpha-beta search algorithm. Notice that these routines are the same as the MINIMAX functions in Figure 5.3, except for the two lines in each of MIN-VALUE and MAX-VALUE that maintain α and β (and the bookkeeping to pass these parameters along).

α := value of best (highest) choice for MAX we have found so far (over all actions from the root). Initially $-\infty$.

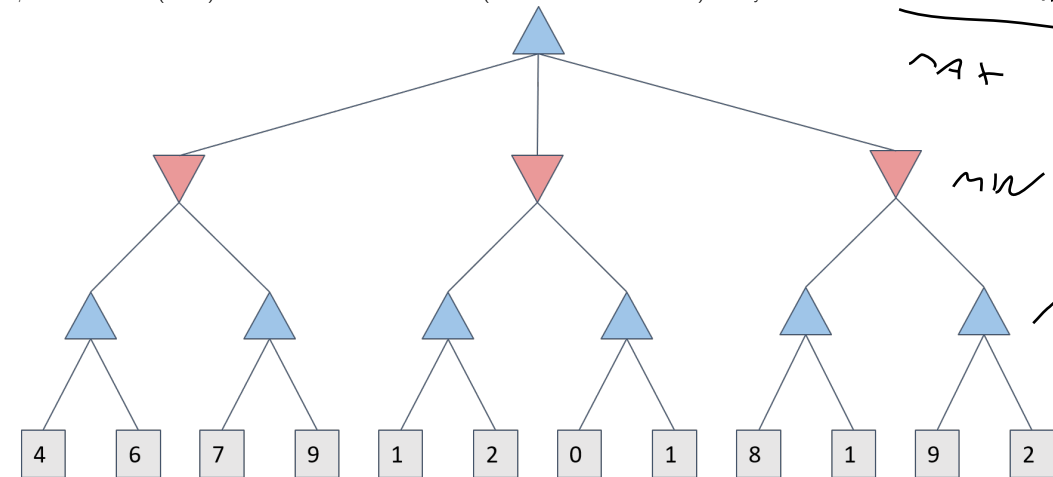
β := value of best (lowest) choice for MIN we have found so far (over all actions from the root) Initially ∞ .

Next time

MAX

MIN

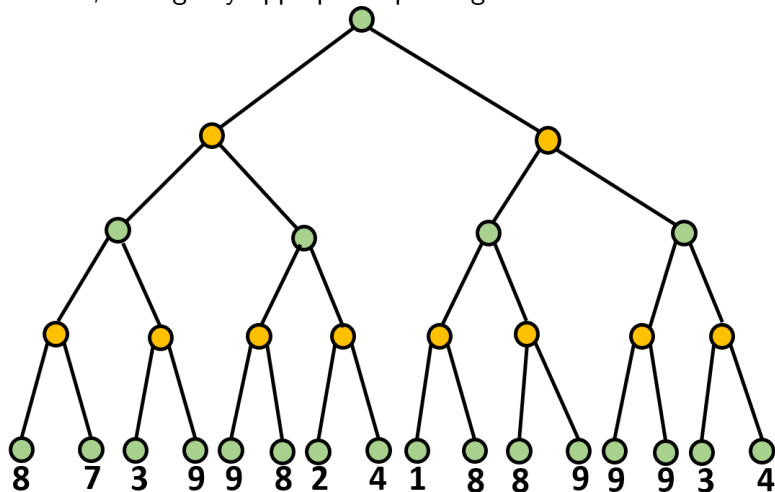
MAX



4 8
6 10

Another Example

Consider the game tree below. Find the resulting value of the root node by following the minimax, noting any appropriate pruning actions.



AB Complexity

What's the payoff of the Alpha-beta pruning?

1. Generally doesn't affect final results.
2. Entire subtrees can be pruned early on: not just leaves.
3. On typical problems, can look twice as deep as minimax in the same amount of time.
4. Both algorithms still have exponential complexity in theory. $b^m \rightarrow b^{2m}$
 \rightarrow complexity of same tree
5. ... what do we do if we can never reach a true goal state? Those values might be *utility* values or even heuristics!

Decision-making

When we traveled across the Eastern US, what mattered to us?

1. Distance-to-destination
2. Time-to-destination
 - Is time always the same?



17

Decision-making

When we traveled across the Eastern US, what mattered to us?

1. Distance-to-destination
2. Time-to-destination
 - Is time always the same?

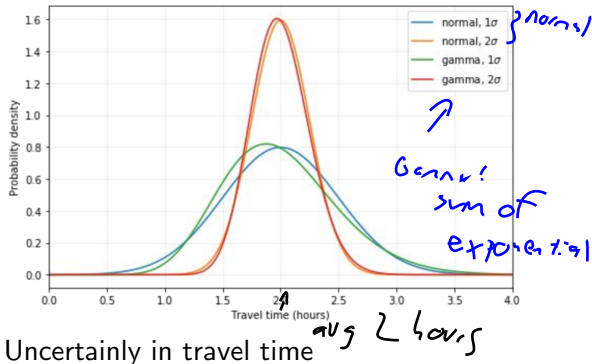
Full decision-making for an agent both requires us specify *what we value* but also uncertainty: and our *interpretation of uncertainty*.



Probability Theory

Probability Theory is where we move beyond *ontological* commitment and into epistemological commitments.

- *Ontology* is the study of being. An *ontological commitment* is a deterministic assertion of existence: think of it as a true/false assertion, a 0 or 1, or a result of propositional logic.
- *Epistemology* is the study of knowledge. An *epistemological commitment* quantifies our **degree of belief** in all the possible outcomes. These will take values as **probabilities** from 0 to 1.

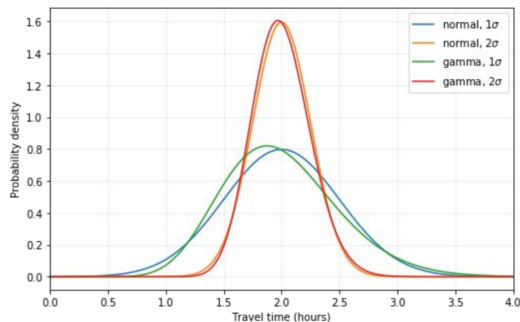
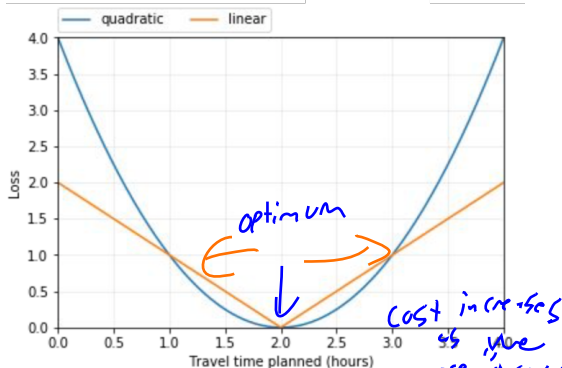


Probability Theory and Loss

For **Decision Theory**, we marry:

Probability Theory: a model for the uncertainty of results (*given* decisions) with

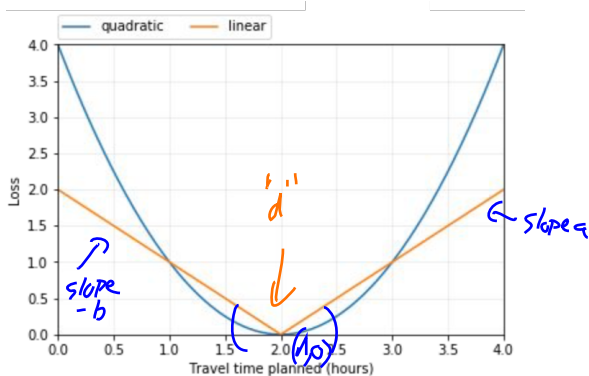
Utility Theory: a model for the cost of being wrong or suboptimal.



Linear and Quadratic Loss functions.

Uncertainty in travel time

Loss



Linear and Quadratic Loss functions.

A *loss function* determines the falloff in utility as we move further and further away from the sweet spot. Suppose the proper value is d . Then the loss incurred when the true value is x might be:

1. Linear:

$$L_l(d, x) = \begin{cases} a(x - d) & x \geq d \\ b(d - x) & x < d \end{cases}$$

$\sim |x - d|$

2. Quadratic:

Ex: Comfort w.r.t temperature

$$L_q(d, x) = k(d - x)^2$$

3. Or something else: e.g. sum of squared/absolute errors for discrete-valued probabilities, etc.

Decision-making and uncertainty

A quick probability review: what did the **expected value** or $E[\cdot]$ function do?

1. For a random variable X with pdf x , $E[X] = \int_x x f(x) dx$. (if continuous)
2. We think of $E[\cdot]$ as the average value of an underlying population, and sometimes denote it μ_X . 😊
3. It's just the sample mean. of an infinite population
4. ~~$E[X]$ is a random number.~~

NO

EVPI and EVIU


P.I: we know weather
 Bayes: we can predict average weather
 Ignoring: we don't care about weather
 done this!

We introduce two major measures for how and when to include uncertainty in a model:

1. The **Expected Value of Perfect Information** (EVPI) focuses on how much uncertainty is driving our loss: how much do we expect to lose *because* there is uncertainty.
2. The **Expected Value of Including Uncertainty** (EVIU) is a measure for our decisions: how much better do they get if we describe and use uncertainty at all.
3. For quantifying both, we need to set up an overall framework. Consider 3 types of decisions:

EVPI and EVIU

To understand EVPI and EVIU, we have to understand how our decisions might change when faced with uncertainty.

1. d_{Bayes} will be the “best” on-average decision, called the Bayes' decision. It minimizes *expected* loss. We will try to find this decision in some future algorithms.
2. We could also make other decisions: d_{pi} might be the decision we make with perfect information: *after* observing the “uncertain” elements.
3. We could also make other decisions: d_{iu} might be the decision we make ignoring all uncertainty.


EVIU

The **expected value of including uncertainty** is the difference in expected (loss) value between the decision we make with when we include uncertainty and the decision we make while ignoring it.

$$\begin{aligned}
 \overset{\text{IV}}{\cancel{EV}}\overset{\text{no uncertainty}}{\downarrow}PI &= E[\text{Loss from } d_{iu} - \text{Loss from } d_{Bayes}] \\
 &= \\
 &= \\
 &=
 \end{aligned}$$

as "average" decision

Where x holds all of the possible states, and $f(x)$ is the uncertainty pdf for x : our probability (density) for each of those states.

EVIU

The **expected value of including uncertainty** is the difference in expected (loss) value between the decision we make with when we include uncertainty and the decision we make while ignoring it.

$$\begin{aligned}
 EVPI &= E[\text{Loss from } d_{iu} - \text{Loss from } d_{Bayes}] \\
 &= \int_x (L(d_{iu}, x) - L(d_{Bayes}, x)) f(x) dx \\
 &= \int_x L(d_{iu}, x) dx - \int_x L(d_{Bayes}, x) f(x) dx \\
 &=
 \end{aligned}$$

Where x holds all of the possible states, and $f(x)$ is the uncertainty pdf for x : our probability (density) for each of those states.

EVIU

The **expected value of including uncertainty** is the difference in expected (loss) value between the decision we make with when we include uncertainty and the decision we make while ignoring it.

$$\begin{aligned}
 \text{EV} \cancel{\text{PI}}^{\text{IU}} &= E[\text{Loss from } d_{iu} - \text{Loss from } d_{Bayes}] \\
 &= \int_x (L(d_{iu}, x) - L(d_{Bayes}, x)) f(x) dx \\
 &= \int_x L(d_{iu}, x) dx - \int_x L(d_{Bayes}, x) f(x) dx \\
 &= E_x[L(d_{iu}, x)] - E_x[L(d_{Bayes}, x)]
 \end{aligned}$$

Where x holds all of the possible states, and $f(x)$ is the uncertainty pdf for x : our probability (density) for each of those states.

EVPI

The **expected value of perfect information** is the difference in expected (loss) value between the decision we make with perfect information and the decision we make “just doing our best.”

$$EVPI = E[\text{Loss from } d_{Bayes} - \text{Loss from } d_{pi}]$$

$$=$$

$$=$$

$$=$$

Where again, x holds all of the possible states, and $f(x)$ is the uncertainty pdf for x : our probability (density) for each of those states.

EVPI

The **expected value of perfect information** is the difference in expected (loss) value between the decision we make with perfect information and the decision we make “just doing our best.”

$$\begin{aligned} EVPI &= E[\text{Loss from } d_{Bayes} - \text{Loss from } d_{pi}] \\ &= \int_x (L(d_{Bayes}, x) - L(d_{pi}, x)) f(x) dx \\ &= \int_x L(d_{Bayes}, x) dx - \int_x L(d_{pi}, x) f(x) dx \\ &= \end{aligned}$$

Where again, x holds all of the possible states, and $f(x)$ is the uncertainty pdf for x : our probability (density) for each of those states.

EVPI

The **expected value of perfect information** is the difference in expected (loss) value between the decision we make with perfect information and the decision we make “just doing our best.”

$$\begin{aligned} EVPI &= E[\text{Loss from } d_{Bayes} - \text{Loss from } d_{pi}] \\ &= \int_x (L(d_{Bayes}, x) - L(d_{pi}, x)) f(x) dx \\ &= \int_x L(d_{Bayes}, x) dx - \int_x L(d_{pi}, x) f(x) dx \\ &= E_x[L(d_{Bayes}, x)] - E_x[L(d_{pi}, x)] \end{aligned}$$

Where again, x holds all of the possible states, and $f(x)$ is the uncertainty pdf for x : our probability (density) for each of those states.

EVUI

Property: The EVUI for the quadratic loss function is zero!

Proof:

EVUI

Property: The EVUI for the quadratic loss function is zero!

Proof:

- 1) Recall the definition:

$$EVUI = E_x[L(d_{iu}, x)] - E_x[L(d_{Bayes}, x)]$$

- 2) What's the Bayes' decision? d_{Bayes} is the decision that *minimizes expected loss*. So we choose d_{bayes} to be the decision d such that $E_x[L(d, x)]$ is as small as possible. Then:

EVUI

Property: The EVUI for the quadratic loss function is zero!

Proof:

$$\begin{aligned} E_x[L(d, x)] &= E_x[k(d - x)^2] \\ &= k \cdot E_x[d^2 - 2dx - x^2] = k(d^2 - 2dE_x[x] + E_x[x^2]) \end{aligned}$$

EVUI

Property: The EVUI ($E_x[L(d_{iu}, x)] - E_x[L(d_{Bayes}, x)]$) for the quadratic loss function is zero!

Proof: We have that $E_x[L(d, x)] = k (d^2 - 2dE_x[x] - E_x[x^2])$.

EVUI

Property: The EVUI ($E_x[L(d_{iu}, x)] - E_x[L(d_{Bayes}, x)]$) for the quadratic loss function is zero!

Proof: We have that $E_x[L(d, x)] = k(d^2 - 2dE_x[x] + E_x[x^2])$. We're choosing d , so we differentiate with respect to d and set equal to zero.

$$\begin{aligned}\frac{dE_x[L(d, x)]}{dd} &= \frac{d}{dd}k(d^2 - 2dE_x[x] + E_x[x^2]) \\ &= 2kd - 2kE_x[x] \\ 0 &\stackrel{Set}{=} 2kd - 2kE_x[x] \\ \implies d &= E_x[x]\end{aligned}$$

So we have $d_{Bayes} = E_x[x]$: when the loss is quadratic, the Bayes' decision is the one that returns the average value of the error of X itself!

EVUI

Property: The EVUI ($E_x[L(d_{iu}, x)] - E_x[L(d_{Bayes}, x)]$) for the quadratic loss function is zero!

Proof: We have that $d_{Bayes} = E_x[x]$.

By a symmetric argument, $d_{iu} = x = E_x[x]$: we choose the minimizer of the loss function $k(d - x)^2$. The difference is then 0!

We assumed that, neglecting uncertainty, the “best” estimate of x is $E_x[x]$. If we’re risk-averse and/or asymmetric loss, might make a different assumption.

EVPI

Property: Under quadratic loss, EVPI will likely not be zero.

Moving Forward

- ▶ This Week:

- 1. nb day Friday

- ▶ Next time: Decision-Making!