CS221 Problem Workout

Week 5

Stanford University

Zero Sum Turn Based Games

- Zero Sum (Adversarial)
  - Only one player can win
  - One player loses by the amount the other player wins
- Turn based
  - Only one player takes an action at a time

Image Credit: Chess.com

Game Tree

- In order to reason about games we make a Game Tree
- Enumerate all the possible actions by a given player on their turn
- Allows us to compute expected value of the game based on players policies

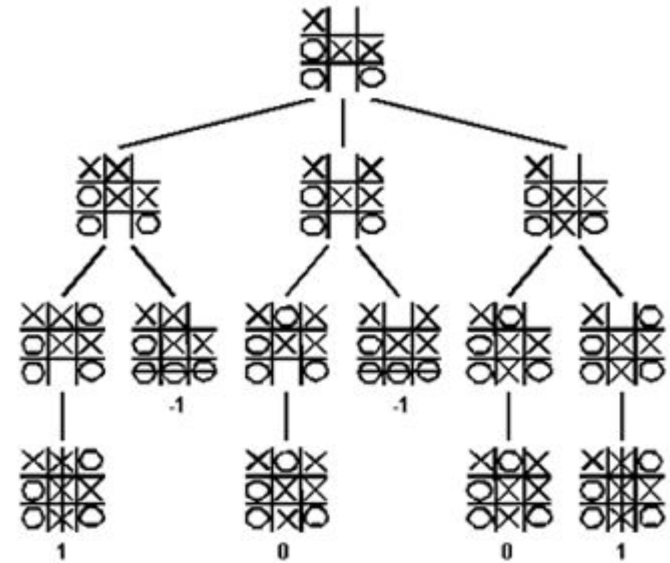
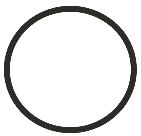
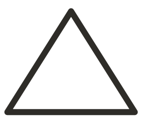
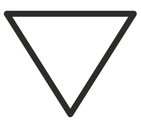
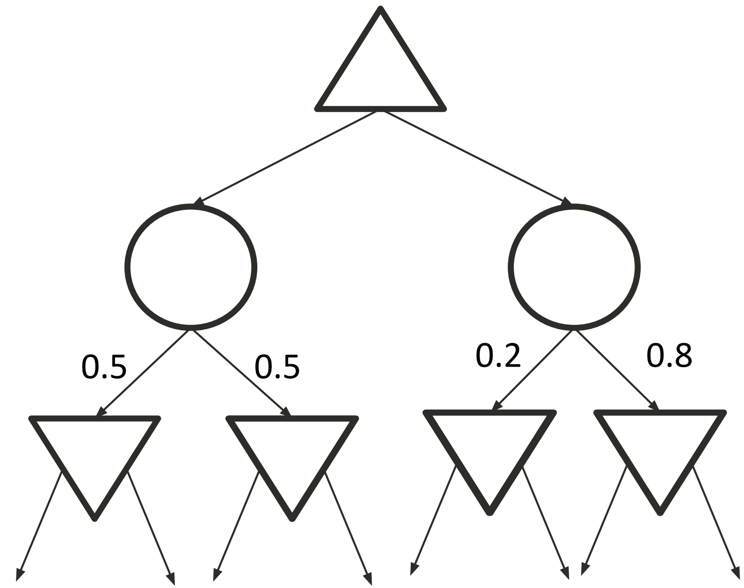


Image Credit: USC

Game Tree

- Helps to represent players based on their policy
-  = Probabilistic
-  = Maximizer
-  = Minimizer
- It is important to consider that a minimizer player is “maximizing” the opponent reward (their reward) in a zero sum game!



Finding Optimal Policy

- We need to evaluate the expected utility of each game state

- Depending on the game we can use:
    - Expectimax: Fixed Random Opponent
    - Minimax: Minimizer Opponent
    - Expectiminimax: Minimizer Opponent with randomness in the game

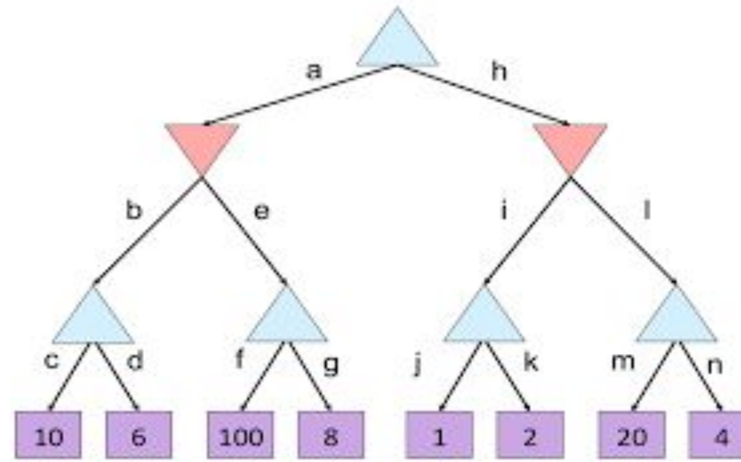


Image Credit: [UC Berkeley]

Finding Optimal Policy

- We need to evaluate the expected utility of each game state

- Depending on the game we can use:
  - Expectimax: Fixed Random Opponent
  - Minimax: Minimizer Opponent
  - Expectiminimax: Minimizer Opponent with randomness in the game

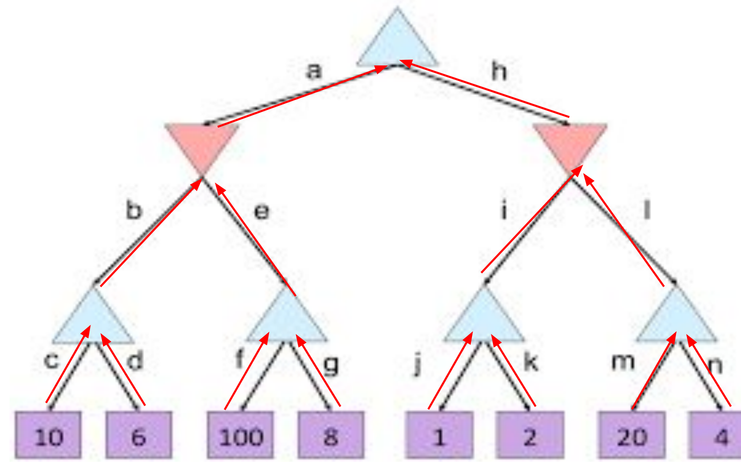


Image Credit: [UC Berkeley]

Improve Efficiency: Evaluation Functions

- Sometimes we can't possibly enumerate the whole search tree
- We can perform a depth limited search to a certain depth in the tree
- We can then define Eval(s) functions which take in a state and return a predicted value of that state

Example: chess

$$\text{Eval}(s) = \text{material} + \text{mobility} + \text{king-safety} + \text{center-control}$$

$$\text{material} = 10^{100}(K - K') + 9(Q - Q') + 5(R - R') + 3(B - B' + N - N') + 1(P - P')$$

$$\text{mobility} = 0.1(\text{num-legal-moves} - \text{num-legal-moves}')$$

...

Improve Efficiency: Alpha Beta Pruning

a_s: lower bound on the value that a max node can contribute upwards (increases with updates)

b_s: upper bound on the value that a min node can contribute upwards (decreases with updates)

alpha_s: maximum a that we know of from currNode to root

beta_s: minimum b that we know of from currNode to root

A max node only has a chance of being on the optimal path if **a_s ≤ beta_s**

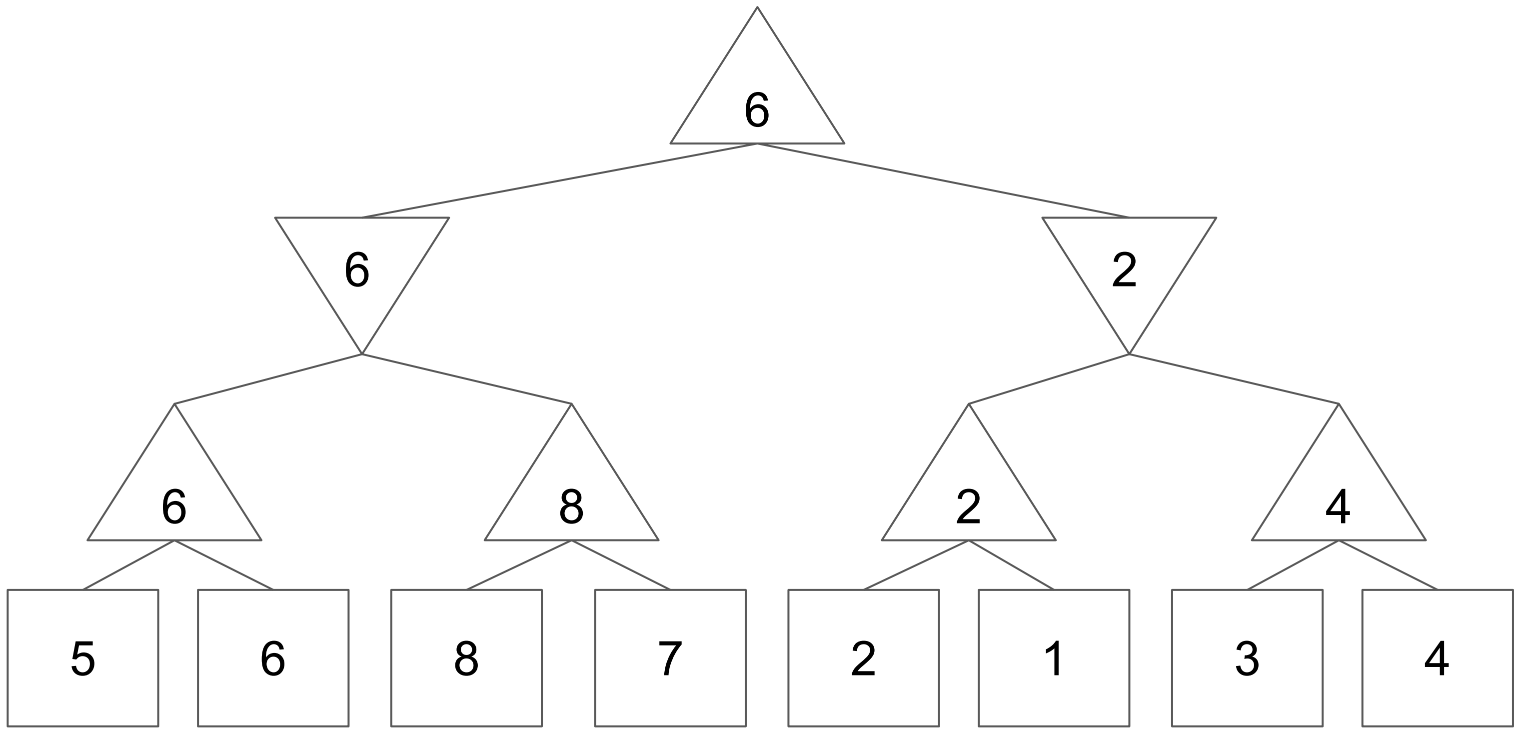
- "My value will be at least a_s, my min ancestors will let through at most beta_s"

If we see a max node where a_s > beta_s: we can prune all of its unexplored children!

- Exploring more children will only increase the max node's value, which is already not feasible through the min ancestors

Work this out for min nodes!

Improve Efficiency: Alpha Beta Pruning



2) "I am the Lorax who speaks for the [game] trees, which you seem to be [alpha-beta pruning] as fast as you please!" - The Lorax

(a) Evaluate the following game (Figure 1) where the edges are probabilities:

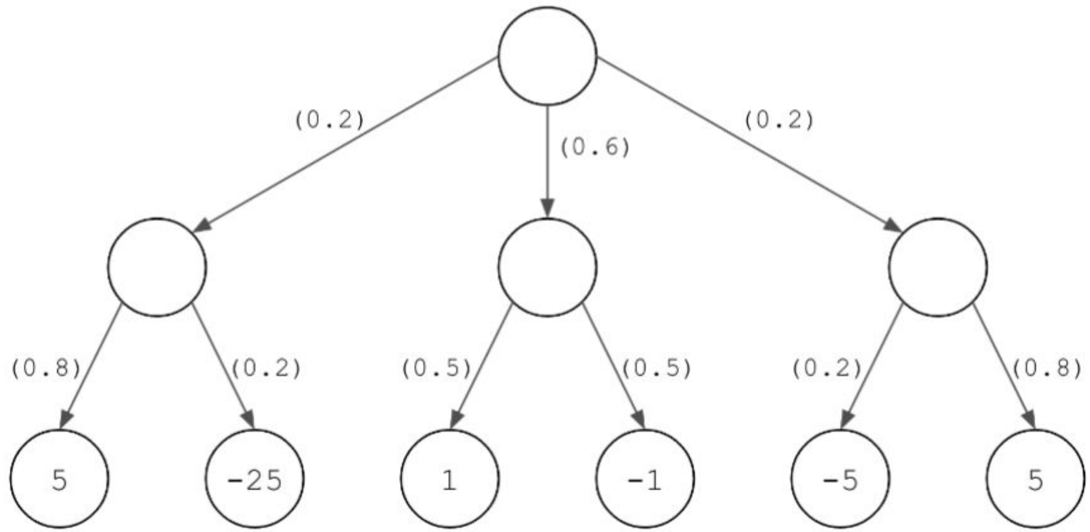


Figure 1

Pretend the top node is now a maximizing player. Under expectimax which action should they take (left, center, or right) and what is the value of the game.

(b) Evaluate the game in Figure 2 using the minimax strategies for both players, with $x = -5$. Recall that upwards pointing triangles is the maximizing player and downwards pointing is the minimizing player.

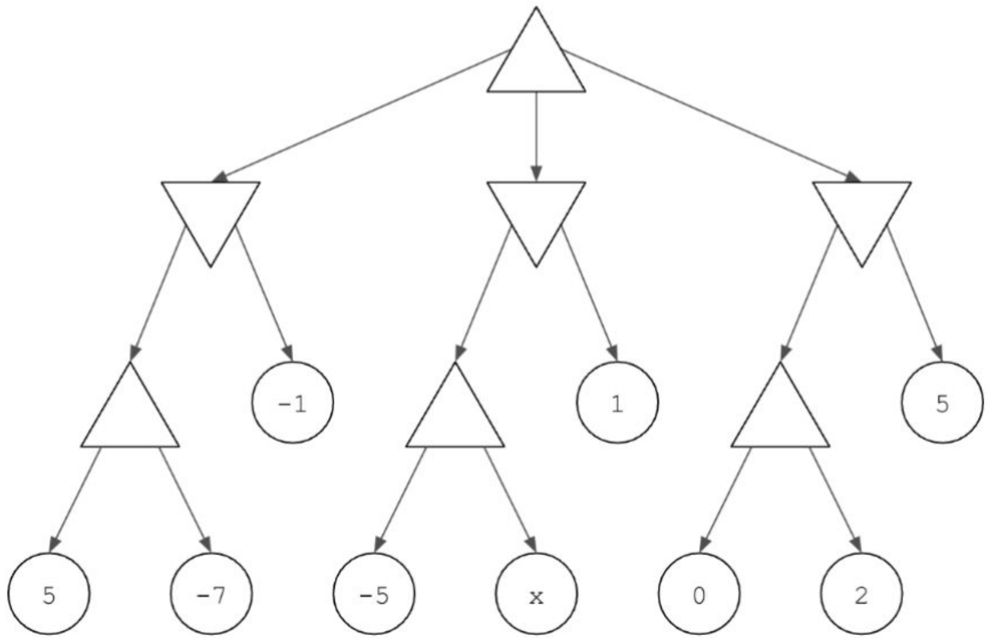


Figure 2

Can we pick x so that the maximizing player loses? Why or why not.

(c) Can either player do better by deviating from minimax assuming the other stays?

(d) Evaluate the game in Figure 3 under the expectiminimax strategy, using $x = -5$. Write down a funny answer for who the third player playing the circles is.

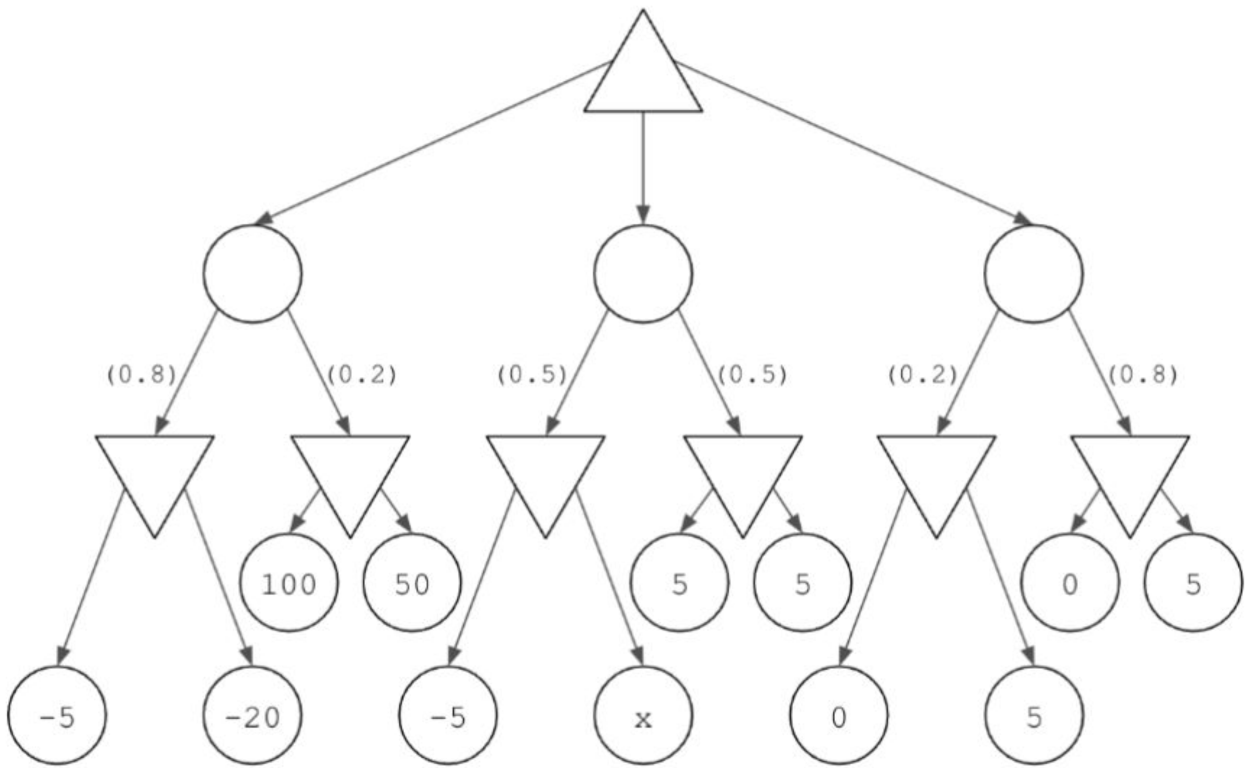


Figure 3

(e) In the previous problem, is there a value of x we can choose so that the game does not end in a draw?

(f) Assume that in the case of a tie in the value of multiple options, the maximizing player chooses the rightmost tied-value action. Still referring to (d) and Figure 3 with $x = -5$, explain, in your own words, why expectiminimax always chooses to draw the game given this choice of tie-breaking. Is there a better way of breaking ties?

Problem 1: General ML Review

Problem 1: General ML Review

“Linear Regression” with Feature Maps

“Linear Regression” with Feature Maps

We have a trained linear regression model $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$. In your own words, explain why we call this model “linear”. Is it linear in x? Linear in $\phi(x)$? Linear in $\mathbf{w}$? Note that linearity for some generic function g means that $g(x + y) = g(x) + g(y)$ and $g(\alpha x) = \alpha g(x)$ for all parameters α.

“Linear Regression” with Feature Maps

We have a trained linear regression model $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$. In your own words, explain why we call this model “linear”. Is it linear in x? Linear in $\phi(x)$? Linear in $\mathbf{w}$? Note that linearity for some generic function g means that $g(x + y) = g(x) + g(y)$ and $g(\alpha x) = \alpha g(x)$ for all parameters α.

- Is it linear in x? NO!
- Linear in $\phi(x)$? Yes
- Linear in $\mathbf{w}$? Yes

"Linear Regression" with Feature Maps

We have a trained linear regression model $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$. In your own words, explain why we call this model "linear". Is it linear in x? Linear in $\phi(x)$? Linear in $\mathbf{w}$? Note that linearity for some generic function g means that $g(x+y) = g(x) + g(y)$ and $g(\alpha x) = \alpha g(x)$ for all parameters α.

- Is it linear in x? NO!
- Linear in $\phi(x)$? Yes
- Linear in $\mathbf{w}$? Yes

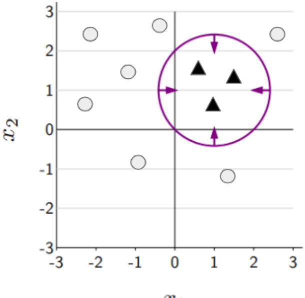
Key Takeaway: Feature maps let us express / model non-linear functions within linear regression!

"Linear Regression" with Feature Maps

We have a trained linear regression model $f_{\mathbf{w}}(x) = \mathbf{w} \cdot \phi(x)$.

- Is it linear in x? NO!
- Linear in $\phi(x)$? Yes
- Linear in $\mathbf{w}$? Yes

Key Takeaway: Feature maps let us express non-linear functions within linear classification models, e.g. quadratic features:



Problem 1: General ML Review

Linear Classification Decision Boundaries

Linear Classification Decision Boundaries

We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$. What is the decision boundary? What does $\mathbf{w} \cdot \phi(x)y = -1000$ imply about how well our model classified the point (x, y)? What does $\mathbf{w} \cdot \phi(x)y = 0.1$ imply about how well our model classified the point (x, y)?

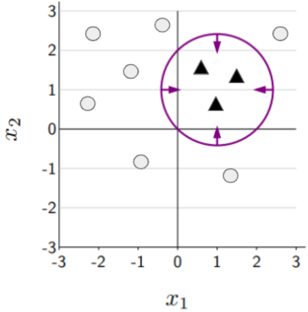
Linear Classification Decision Boundaries

We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$.

What is the decision boundary?

Linear Classification Decision Boundaries

We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$.
What is the decision boundary?



Recall our definition: The decision boundary is $w \cdot \phi(x) = 0$.
Key Takeaway: Decision boundaries let us separate data into different groups!

Linear Classification Decision Boundaries

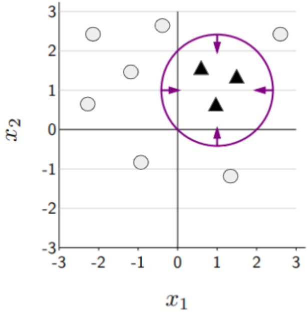
We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$.

What does $\mathbf{w} \cdot \phi(x) y = -1000$ imply about how well our model classified the point (x, y)?

Linear Classification Decision Boundaries

We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$.

What does $\mathbf{w} \cdot \phi(x)y = -1000$ imply about how well our model classified the point (x, y)?



Our model is confident in the classification (far from the decision boundary), but incorrect in the classification (note the sign).

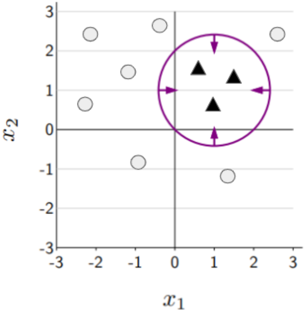
Linear Classification Decision Boundaries

We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$.

What does $\mathbf{w} \cdot \phi(x) y = 0.1$ imply about how well our model classified the point (x, y)?

Linear Classification Decision Boundaries

We are working with a classification model $f_{\mathbf{w}}(x) = \text{sign}(\mathbf{w} \cdot \phi(x))$.
What does $\mathbf{w} \cdot \phi(x)y = 0.1$ imply about how well our model classified the point (x, y)?



Our model is not very confident in the classification (close to the decision boundary), but correct in the classification (same signs).

Problem 1: General ML Review

Loss Functions

Loss Functions

Additionally, you consider using the following loss function

$$1[(\mathbf{w} \cdot \phi(x))y \leq 0]$$

for gradient descent. Explain why using this loss function is a bad idea.

Loss Functions

Additionally, you consider using the following loss function

$$1[(\mathbf{w} \cdot \phi(x))y \leq 0]$$

for gradient descent. Explain why using this loss function is a bad idea.

This is the zero-one loss function, which has zero gradient almost everywhere!

Key Takeaway: We want our loss function to have a meaningful gradient for gradient descent!

Loss Functions

After solving the prior problem, you realize the zero-one loss function is a bad idea and instead decide to use the logistic loss function. Your data is $y \in \{0, +1\}$, so you define the logistic loss as follows

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

where f has a range of $[0, 1]$. Before picking f, you'd like to differentiate L with respect to $\mathbf{w}$. Is this possible, and if so, what is $\frac{\partial L}{\partial \mathbf{w}}$?

Loss Functions

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

Yes! We use the chain rule:

$$\begin{aligned}\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} &= -y \frac{1}{f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} + (1 - y) \frac{1}{1 - f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} \\ &= \left(\frac{f(x; \mathbf{w}) - y}{f(x; \mathbf{w})(1 - f(x; \mathbf{w}))} \right) \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}}\end{aligned}$$

Key Takeaway: Be prepared to take derivatives of any loss function!

Loss Functions

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

Yes! We use the chain rule:

$$\begin{aligned}\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} &= -y \frac{1}{f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} + (1 - y) \frac{1}{1 - f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} \\ &= \left(\frac{f(x; \mathbf{w}) - y}{f(x; \mathbf{w})(1 - f(x; \mathbf{w}))} \right) \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}}\end{aligned}$$

(Food for thought: how would the derivative change if it were over a summation?)

Loss Functions

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

Yes! We use the chain rule:

$$\begin{aligned}\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} &= -y \frac{1}{f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} + (1 - y) \frac{1}{1 - f(x; \mathbf{w})} \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}} \\ &= \left(\frac{f(x; \mathbf{w}) - y}{f(x; \mathbf{w})(1 - f(x; \mathbf{w}))} \right) \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}}\end{aligned}$$

(Food for thought: how would the derivative change if it were over a summation?)

Same process, just with indexing!

Loss Functions

For your function f in the above loss function, you can't decide between using the sigmoid function,

$$g(x; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T x}}$$

or the shifted tanh function,

$$h(x; \mathbf{w}) = \frac{1}{2}\tanh(\mathbf{w}^T x) + \frac{1}{2} \quad \text{with} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

in place of f. How would the derivative from Part (c) look like with function g above in place of f, and with function h above in place of f?

Loss Functions

Sigmoid function,

$$g(x; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T x}}$$

$$\frac{\partial g(x; \mathbf{w})}{\partial \mathbf{w}} =$$

Loss Functions

Sigmoid function,

$$g(x; \mathbf{w}) = \frac{1}{1 + e^{-\mathbf{w}^T x}}$$

$$\begin{aligned}
\frac{\partial g(x; \mathbf{w})}{\partial \mathbf{w}} &= -(1 + e^{-\mathbf{w}^T x})^{-2} \frac{\partial}{\partial \mathbf{w}} \left(1 + e^{-\mathbf{w}^T x}\right) \\
&= \frac{x e^{-\mathbf{w}^T x}}{(1 + e^{-\mathbf{w}^T x})^2} \quad \text{(this is a valid answer)} \\
&= x \frac{1}{(1 + e^{-\mathbf{w}^T x})} \frac{e^{-\mathbf{w}^T x}}{(1 + e^{-\mathbf{w}^T x})} \\
&= x g(x; \mathbf{w})(1 - g(x; \mathbf{w}))
\end{aligned}$$

Remember: $\sigma(\mathbf{w})(1 - \sigma(\mathbf{w}))\frac{\partial \mathbf{w}}{\partial x}$ form to save time and work!

Loss Functions

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

$$\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} = \left(\frac{f(x; \mathbf{w}) - y}{f(x; \mathbf{w})(1 - f(x; \mathbf{w}))} \right) \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}}$$

For sigmoid g:

$$\begin{aligned} \frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} &= \left(\frac{g(x; \mathbf{w}) - y}{g(x; \mathbf{w})(1 - g(x; \mathbf{w}))} \right) \frac{\partial g(x; \mathbf{w})}{\partial \mathbf{w}} \\ &= \left(\frac{g(x; \mathbf{w}) - y}{g(x; \mathbf{w})(1 - g(x; \mathbf{w}))} \right) g(x; \mathbf{w})(1 - g(x; \mathbf{w}))x \\ &= x(g(x; \mathbf{w}) - y) \end{aligned}$$

Loss Functions

tanh function,

$$h(x;\mathbf{w}) = \frac{1}{2}\tanh(\mathbf{w}^T x) + \frac{1}{2} \quad \text{with} \quad \tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$$

$$\begin{aligned}
\frac{\partial h(x;\mathbf{w})}{\partial \mathbf{w}} &= \frac{1}{2}\frac{\partial \tanh(\mathbf{w}^T x)}{\partial \mathbf{w}} \\
&= \frac{1}{2}\frac{\partial}{\partial w}\frac{(e^{\mathbf{w}^T x} - e^{-\mathbf{w}^T x})}{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})} \\
&= \frac{1}{2}\left[\frac{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})}{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})} - \frac{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})^2}{(e^{\mathbf{w}^T x} + e^{-\mathbf{w}^T x})^2}\right] x \\
&= \frac{1}{2}(1 - \tanh(\mathbf{w}^T x)^2)x
\end{aligned}$$

Loss Functions

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

$$\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} = \left(\frac{f(x; \mathbf{w}) - y}{f(x; \mathbf{w})(1 - f(x; \mathbf{w}))} \right) \frac{\partial f(x; \mathbf{w})}{\partial \mathbf{w}}$$

For tanh h:

$$\frac{\partial L(x, y; \mathbf{w})}{\partial \mathbf{w}} = \left(\frac{h(x; \mathbf{w}) - y}{h(x; \mathbf{w})(1 - h(x; \mathbf{w}))} \right) \frac{\partial h(x; \mathbf{w})}{\partial \mathbf{w}}$$

$$= \left(\frac{h(x; \mathbf{w}) - y}{\frac{1}{2}(\tanh(\mathbf{w}^T x) + 1)(1 - \frac{1}{2}(\tanh(\mathbf{w}^T x) + 1))} \right) \frac{1}{2}(1 - \tanh(\mathbf{w}^T x)^2) x$$

$$= \left(\frac{h(x; \mathbf{w}) - y}{(\tanh(\mathbf{w}^T x) + 1)\frac{1}{2}(1 - \tanh(\mathbf{w}^T x))} \right) (1 - \tanh(\mathbf{w}^T x)^2) x$$

$$= 2x(h(x; \mathbf{w}) - y)$$

Problem 1: General ML Review

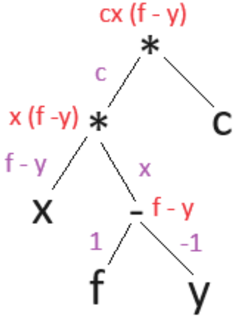
Backpropagation

Backpropagation

Explain why writing the derivative of the loss function in the form of $cx(f(x;\mathbf{w}) - y)$ is very convenient for backpropagation.

Backpropagation

Explain why writing the derivative of the loss function in the form of $cx(f(x;\mathbf{w}) - y)$ is very convenient for backpropagation.



Very straightforward arithmetic operations involving known values!

Key Takeaway: Backpropagation breaks down derivatives into a simple structure for a computer to do!

Problem 1: General ML Review

Reusing Derivatives

Reusing Derivatives

Unfortunately your model has poor performance for both sigmoid and tanh. You decide to make your model a neural network to hopefully fix that.

Let

$$N(x; A, B) = B \max\{Ax, 0\} = z$$

The loss function is now:

$$\begin{aligned} L(x, y; A, B, \mathbf{w}) = -y \log(f(N(x; A, B); \mathbf{w})) - \\ (1 - y) \log(1 - f(N(x; A, B); \mathbf{w})) \end{aligned}$$

Can we we reuse our result from before for $\frac{\partial L}{\partial w}$?

Reusing Derivatives

Let

$$N(x; A, B) = B \max\{Ax, 0\} = z$$

The loss function is now:

$$L(x, y; A, B, \mathbf{w}) = -y \log(f(N(x; A, B); \mathbf{w})) - (1 - y) \log(1 - f(N(x; A, B); \mathbf{w}))$$

Can we we reuse our result from before for $\frac{\partial L}{\partial w}$?

Replace x with $z = N(x; A, B)$!

We were differentiating with respect to $\mathbf{w}$, not x, so the process doesn't change! N is simply a constant in this context.

Key Takeaway: Be careful of what you're differentiating with respect to!

Problem 1: General ML Review

Regularization

Regularization

Food for thought: suppose we figure that our model's poor performance was due to overfitting instead. Why might L_2 regularization help, and how would it change our loss function?

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

Regularization

Food for thought: suppose we figure that our model's poor performance was due to overfitting instead. Why might L_2 regularization help, and how would it change our loss function?

$$L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w}))$$

Key Takeaway: L_2 regularization penalizes our weights $\mathbf{w}$ when we take a minimization:

$$\min_{\mathbf{w}} \left[L(x, y; \mathbf{w}) = -y \log(f(x; \mathbf{w})) - (1 - y) \log(1 - f(x; \mathbf{w})) + \frac{\lambda}{2} ||\mathbf{w}||_2^2 \right]$$

Search Problem (from Week 3)

Search Problem (from Week 3)

Defining the Search Problem

Defining the Search Problem

In 16th century England, there were a set of $N+1$ cities $C = \{0, 1, 2, \ldots, N\}$. Connecting these cities were a set of bidirectional roads R: $(i, j) \in R$ means that there is a road between city i and city j. Assume there is at most one road between any pair of cities, and that all the cities are connected. If a road exists between i and j, then it takes $T(i, j)$ hours to go from i to j.

Romeo lives in city 0 and wants to travel along the roads to meet Juliet, who lives in city N. They want to meet.

Search problems typically require a lot of reading... try to break it down to the important parts.

Defining the Search Problem

Search problems typically require a lot of reading... try to break it down to the important parts.

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N

Defining the Search Problem

- $N+1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N

To reduce confusion, they will reconnect after each traveling a road. For example, if Romeo travels from city 3 to city 5 in 10 hours at the same time that Juliet travels from city 9 to city 7 in 8 hours, then Juliet will wait 2 hours. Once they reconnect, they will both traverse the next road (neither is allowed to remain in the same city). Furthermore, they must meet in the end in a city, not in the middle of a road. Assume it is always possible for them to meet in a city.

Defining the Search Problem

- $N+1$ cities $C = \{0, 1, 2, \dots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N

- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e.
  Cost of $(r, j) \to (r', j') = \max(T(r, r'), T(j, j'))$

Defining the Search Problem

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r, j) \to (r', j') = \max(T(r, r'), T(j, j'))$

States: $s = (r, j)$ where $r \in C$ and $j \in C$ are the cities Romeo and Juliet currently in

Defining the Search Problem

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r, j) \to (r', j') = \max(T(r, r'), T(j, j'))$

$\text{Actions}((r, j)) = \{(r', j') : (r, r') \in R, (j, j') \in R\}$ corresponds to both traveling to a connected city

Defining the Search Problem

- $N+1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i,j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i,j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r,j) \to (r',j') = \max(T(r,r'), T(j,j'))$

$\text{Cost}((r,j),(r',j')) = \max(T(r,r'), T(j,j'))$ is the maximum over the two times

Defining the Search Problem

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r, j) \rightarrow (r', j') = \max(T(r, r'), T(j, j'))$

$\text{Succ}((r, j), (r', j')) = (r', j')$: just the next pair of cities the two end up at

Defining the Search Problem

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r, j) \to (r', j') = \max(T(r, r'), T(j, j'))$

IsGoal$((r, j)) = 1[r = j]$ (whether the two are in the same city)

Search Problem (from Week 3)

Redefining for a Heuristic

Redefining for a Heuristic

- $N+1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i,j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i,j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r,j) \to (r',j') = \max(T(r,r'), T(j,j'))$

Uniform Cost Search to compute $M(i,k)$, the minimum time it takes one person to travel from city i to city k for all pairs of cities $i, k \in C$.

Give a consistent A* heuristic for the search problem. Your heuristic should take $O(N)$ time to compute, assuming that looking up $M(i,k)$ takes $O(1)$ time.

Redefining for a Heuristic

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r, j) \to (r', j') = \max(T(r, r'), T(j, j'))$
- UCS precompute $M(i, k)$, minimum time to go from any city i to any city k; takes $O(1)$ to look up

How to relax the search problem to make use of the additional info? Is there a contradiction anywhere in the criteria?

Redefining for a Heuristic

- $N + 1$ cities $C = \{0, 1, 2, \ldots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- ~~Romeo and Juliet will wait for the other to finish traveling before moving again, i.e. Cost of $(r, j) \to (r', j') = \max(T(r, r'), T(j, j'))$~~
- UCS precompute $M(i, k)$, minimum time to go from any city i to any city k; takes $O(1)$ to look up

How to relax the search problem to make use of the additional info? Is there a contradiction anywhere in the criteria?

Redefining for a Heuristic

- $N + 1$ cities $C = \{0, 1, 2, \dots, N\}$
- R: $(i, j) \in R$ is a road between city i and j
- Only 1 road between any 2 cities
- $T(i, j)$ hours to go along road from i to j
- Romeo starts at 0, Juliet starts at N
- Romeo and Juliet will only wait for the other to finish traveling if they make it to the goal
- UCS precompute $M(i, k)$, minimum time to go from any city i to any city k; takes $O(1)$ to look up

Key Takeaway: How to relax the search problem to make use of the additional info? Is there a contradiction anywhere in the criteria?

Redefining for a Heuristic

Romeo and Juliet will only wait for the other to finish traveling if they make it to the goal

$$h((r,j)) = \min_{c \in C} \max\{M(r,c), M(j,c)\}.$$

A* heuristic $h(s)$ is consistent if

$$h(s) \leq \text{Cost}(s,a) + h(\text{Succ}(s,a)).$$

so the following needs to be true

$\min_{c \in C} \max\{M(r,c), M(j,c)\} \leq$

$$\text{Cost}((r,j),(r',j')) + \min_{c' \in C} \max\{M(r',c'), M(j',c')\}$$

Redefining for a Heuristic

Romeo and Juliet will only wait for the other to finish traveling if they make it to the goal

$$h((r,j)) = \min_{c \in C} \max\{M(r,c), M(j,c)\}.$$

A* heuristic $h(s)$ is consistent if the following is true

$\min_{c \in C} \max\{M(r,c), M(j,c)\} \leq$

$$\text{Cost}((r,j),(r',j')) + \min_{c' \in C} \max\{M(r',c'), M(j',c')\}$$

The cost on the right-hand side is the original cost function, which has Romeo/Juliet wait at every stop. That makes the right-hand side larger!