Pokerbots 2025

Lecture 9: Decision Trees

Sponsors

HRT
hudson river trading

Jump
TRADING

VIRTU
FINANCIAL

CITADEL | CITADEL Securities

DRW

MIDPOINT
MARKETS

FIVE RINGS

imc

TREX
QUANT

Jane
Street®

{...} codeium

CTC®
CHICAGO TRADING COMPANY®

2σ
TWO SIGMA

SUSQUEHANNA

Announcements

Last Night's Lightning Tournament #2

1. Pineapple	$400
2. Encore 1/2 Regs	$300
3. heyo394	$225
4. pink fluffy sheep	$175
5. DKE juniors	$150

Week 3 Bot Deadline!

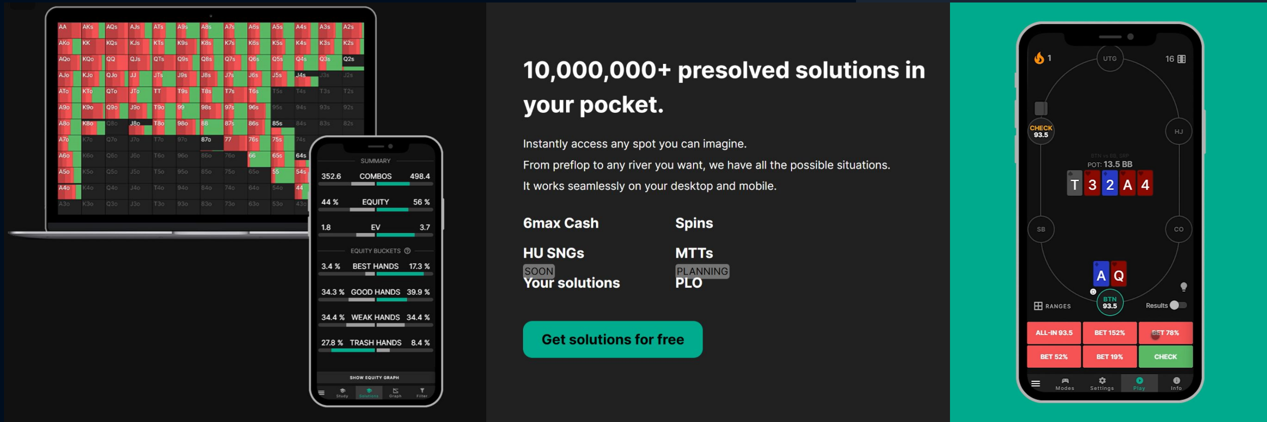
- Third pokerbot due Tomorrow 1/24, 11:59PM EST
- Submission on scrimmage server
- Mini-tournament 3 will occur shortly after

Next Week…

Monday 1/27

Guest Lecture: GTO Wizard

- Current state of the art Poker Solver and Study tool
- Developed by team of top AI researchers and engineers
- Core team will be presenting their journey and research
- RSVP at **pkr.bot/gto**



Tuesday 1/28

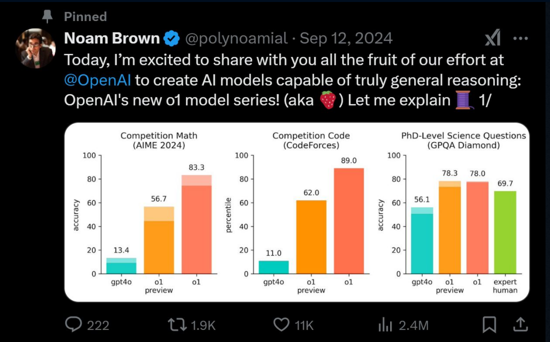
Team Strategy Reports due 11:59PM EST

- No Lecture
- 3-5 pages, double spaced, ≥1 member must submit
- Assignment details and submission on Canvas: **pkr.bot/canvas**
- Description also available in syllabus: **pkr.bot/syllabus**

Wednesday 1/29

Guest Lecture: Dr. Noam Brown

- Won Pokerbots and co-created Deep CFR while PhD at CMU
- Created Libratus and Pluribus, the world's first superhuman poker AIs
- Research scientist currently at OpenAI, previously worked at Meta AI
- Leading role in developing OpenAI's latest GPT o1 and o3 LLMs



Wednesday 1/29 (cont.)

Poker Social After!

- During office hours block
- Come play with Noam!

Wednesday 1/29 (cont.)

Final Bot Submission due 11:59PM EST

- Upload and select bot as active on scrimmage server
- Both report and bot needed to pass this class!
- Bot will compete in last and final Pokerbots tournament
- Non-secret prize amounts listed on syllabus

Final Tournament Prizes	
First place	$10,000
Second place	$6,500
Third place	$3,500
Fourth place	$2,000
Fifth place	$1,000
First place in language (Python, Java, or C++)	$500 x 3
Second place in language (Python, Java, or C++)	$250 x 3
Third place in language (Python, Java, or C++)	$125 x 3
Best freshman-majority (>51%) team	$2,000

Friday 1/31

Pokerbots Final Event 4:30-7PM in Kresge Auditorium

- Presentation of Awards
- Closing Ceremony
- Sponsor Event and Puzzle Hunt
- Lots of free merch and raffle prizes!
- Dinner provided 😋

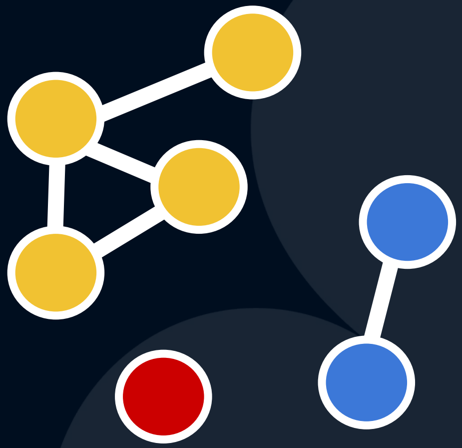
All in all...

1/24 Tomorrow	Week 3 Bot Due
1/27 Monday	GTO Wizard Talk
1/28 Tuesday	Final Report Due
1/29 Wednesday	Noam Brown Talk Social Event Final Bot Due
1/31 Friday	Final Event

Giveaways!

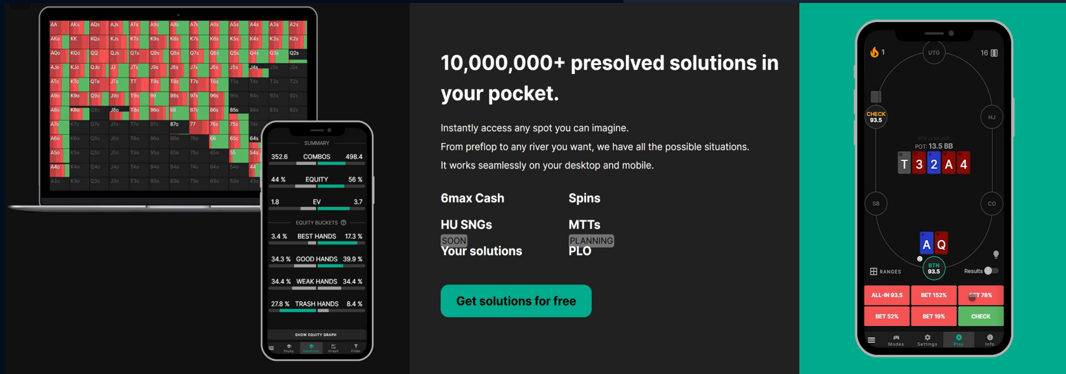
Connection Game: pkr.bot/connect

- List the kerbs of up to 3 other classmates in your submission
- Two people are **connected** if they list each other (strings must match exactly)
- A **connected component** is a group of people who are all directly or indirectly connected
- **Four winners** are randomly drawn from the largest connected component(s)!



Feel free to get up and travel!
But be back to your seat in 4 min

Prize: One year's worth of GTO Wizard Full Subscription <u>each</u>



RSVP for GTO Wizard: **pkr.bot/gto**

- Indicate whether you plan to attend next Monday's talk
- Winner selected at random
- Prize: Sony XM4

Decision Trees

From last lecture...

Model training steps on a large representation (tabular CFR) causes issues

→ Use compressed approximation method like a neural network to store model,
Making training require less steps and inference (or evaluation) less costly

Why we use neural networks:

- Expressive
- Flexible over different data types
- Does well with large amounts of data

What other approximation methods could we use?

What are some processes we'd like to be able to express?

From pot odds bot:

```
if RaiseAction in legal_actions:
    if random.random() < 0.5:
        if strength > 2*pot_odds:
            raise_amount = int(min_raise + 0.1 * (max_raise - min_raise))
            return RaiseAction(raise_amount)
        return RaiseAction(min_raise)
if CheckAction in legal_actions:  # check-call
    return CheckAction()
if random.random() < 0.25:
    return FoldAction()
return CallAction()
```

What are some processes we'd like to be able to express?

We'd like:

- Qualitative selection of variables to condition a decision on
  - Pot odds
  - Legal actions
  - Hole strength
  - Randomly generated values

- Quantitative tuning of how to decide based on that variable
  - Pot odds threshold
  - Ratio of p:pot odds
  - Threshold for random values

Can neural networks do this?

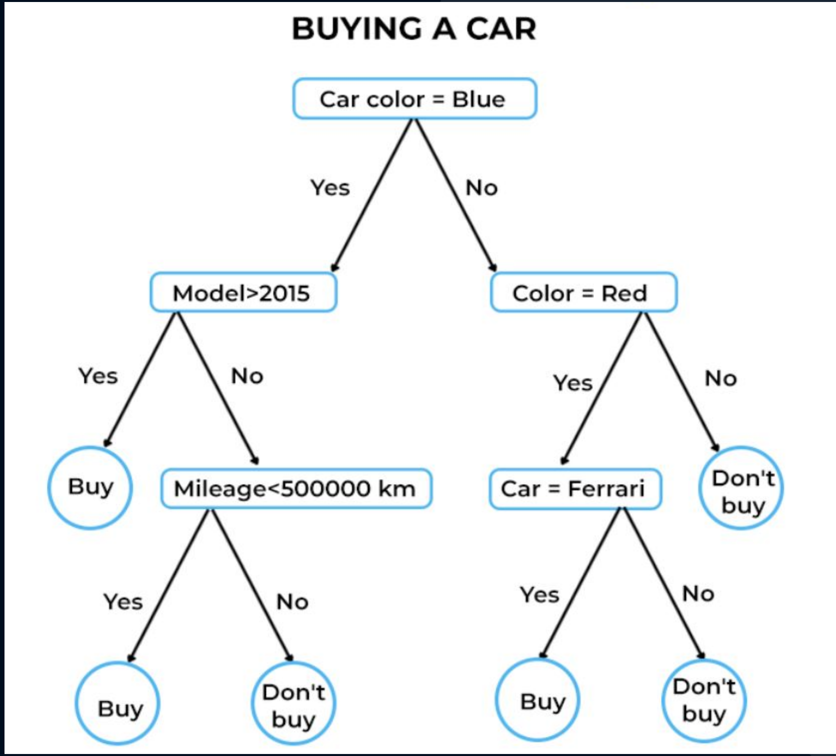
Well, yes…

- Nonlinear activations like ReLU and Sigmoid can capture qualitative regions of similar behavior (‘step functions’)
- Linear layers can learn weights for specific thresholds needed
- So for a large enough neural network, all the if statements and thresholds could be encoded as linear transformations and nonlinear activations

But this seems like overkill if we desire to capture a simple relationship…

Decision Trees

- Definition: A tree-like model used to make decisions by splitting data based on feature values.
- Structure:
  - Nodes: Represent a feature to split on.
  - Edges: Represent decision outcomes.
  - Leaves: Represent the final prediction (class or value).
- Example: Classifying if we'd like to buy a car based on various characteristics



How Decision Trees are Trained

- Start with some tree (initially root node), that sends the dataset into a number of leaves with a label on each leaf
- Performance of the tree is related to how many dataset examples are sent to a leaf with a good prediction. We'd like to consider how we can add additional splits at each leaf to improve this performance.
- Split data on a feature that maximizes information gain or reduces impurity (e.g., Gini index, entropy).
- Repeat recursively until a stopping condition is met (e.g., maximum depth, minimum samples per leaf).
- Predict based on majority class (classification) or average value (regression).

Feature Selection

Incredibly important in decision trees

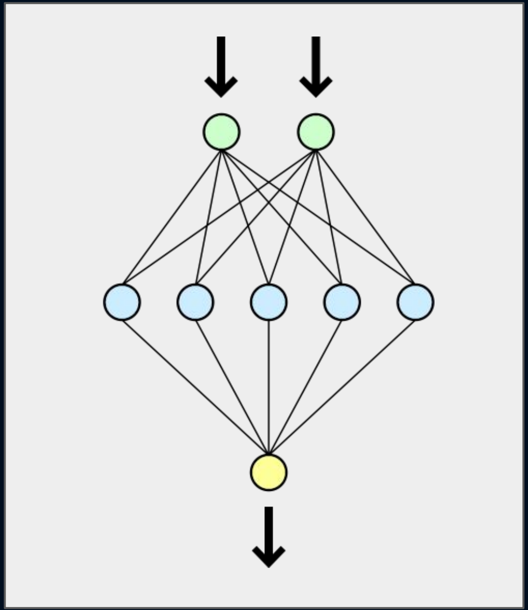
- If we want a quality of data to play a factor in our output, we need to construct it as a feature.
- Ex: We want to raise if $p > c * pot\_odds$, for a learned constant c.
  - This condition is not immediately representable in terms of features over/under a constant
  - We have to *transform* existing features to make it easier for the decision tree to pick up on this condition.
  - New feature: p/pot_odds
  - Raise if this new feature is greater than a learned constant c

Disadvantages to NNs

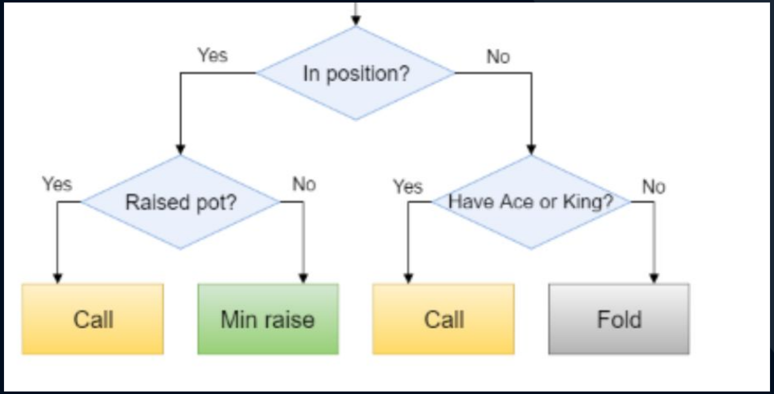
- Have to construct features
- Prone to overfitting or variance if not enough examples
    - Creating a leaf for each data point is too much fitting
    - A single data point can greatly sway behavior
- Meanwhile neural nets create their own features and do well at smoothly reacting to each data point via gradient descent.

Advantages

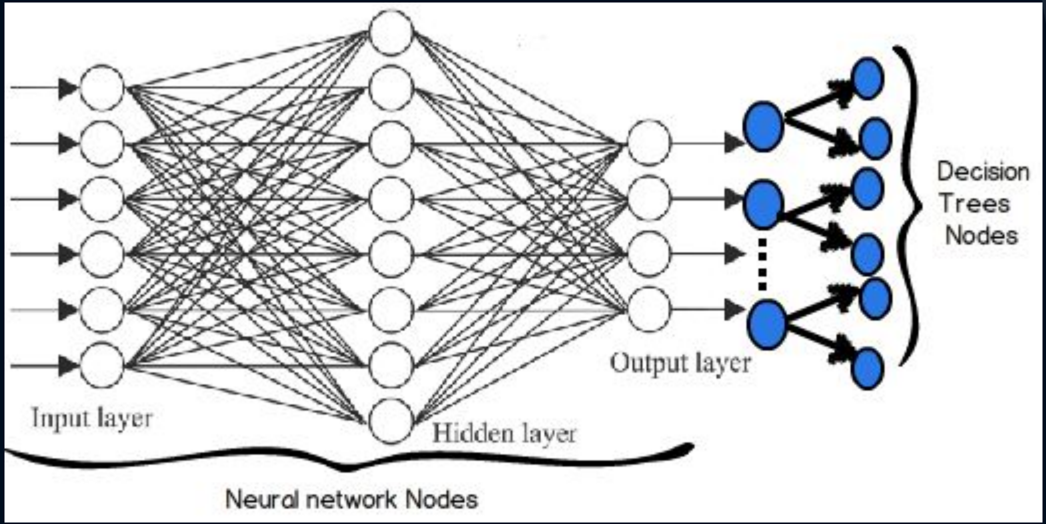
Interpretable!



??



What about both?

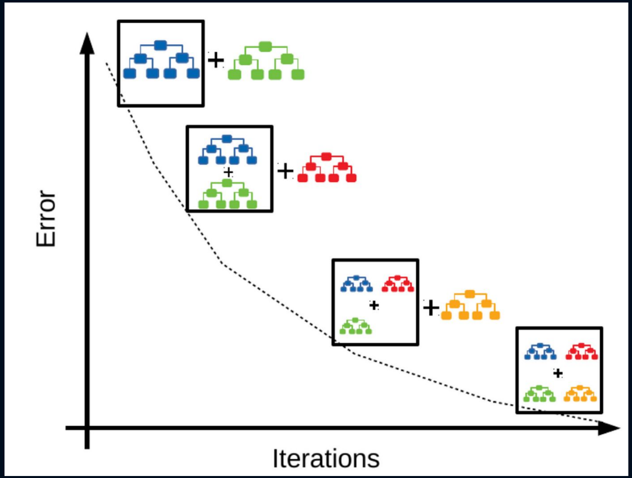


Boosting

Another way to resolve disadvantages

- An ensemble technique that builds multiple weak learners (e.g., decision trees) iteratively, where each new tree corrects errors of the previous one.
- Key Idea: Combine many weak models to form a strong one.
- Loss Minimization:
- Fit each tree to minimize the residual error of the previous trees.

Boosting



- Result: smooths out variance while being able to make more complex decisions
- General result that can be used to improve additive models

Lunch 😋

Leave any type of feedback at **pkr.bot/feedback**!

reference-9-2025

Giveaway Winners

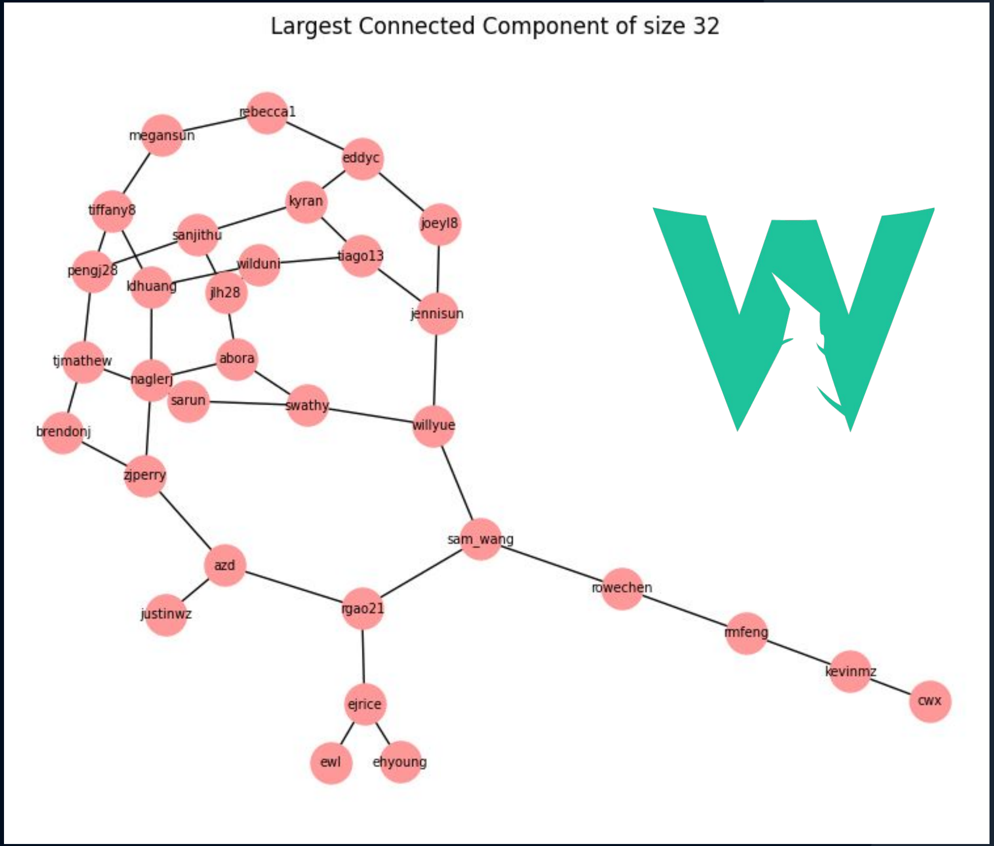
GTO RSVP Raffle: kerb “cleve195”

Connection Game: kerbs

['kyran', 'ehyoung', 'abora', 'tiago13']

N=48 total entries
44 successful connections
15 failed connections

```
{'jorgevas', 'jiange12', 'eposondu'}
{'mandoj', 'mattzhou'}
{'emilyc', 'rlsalas'}
{'merey'}
{'kjiang77'}
{'ljkeller'}
{'gsjau'}
{'decoudav'}
{'tjwshu'}
{'akshaya7'}
{'bocchi'}
{'pky'}
```



Closing Remarks…

Thanks for coming!

Slides will be posted on **pkr.bot/resources**

Repo will be pushed to **pkr.bot/github**

Make sure to check **pkr.bot/piazza** for updates

Lecture recordings at **pkr.bot/panopto**

Leave feedback at **pkr.bot/feedback**!