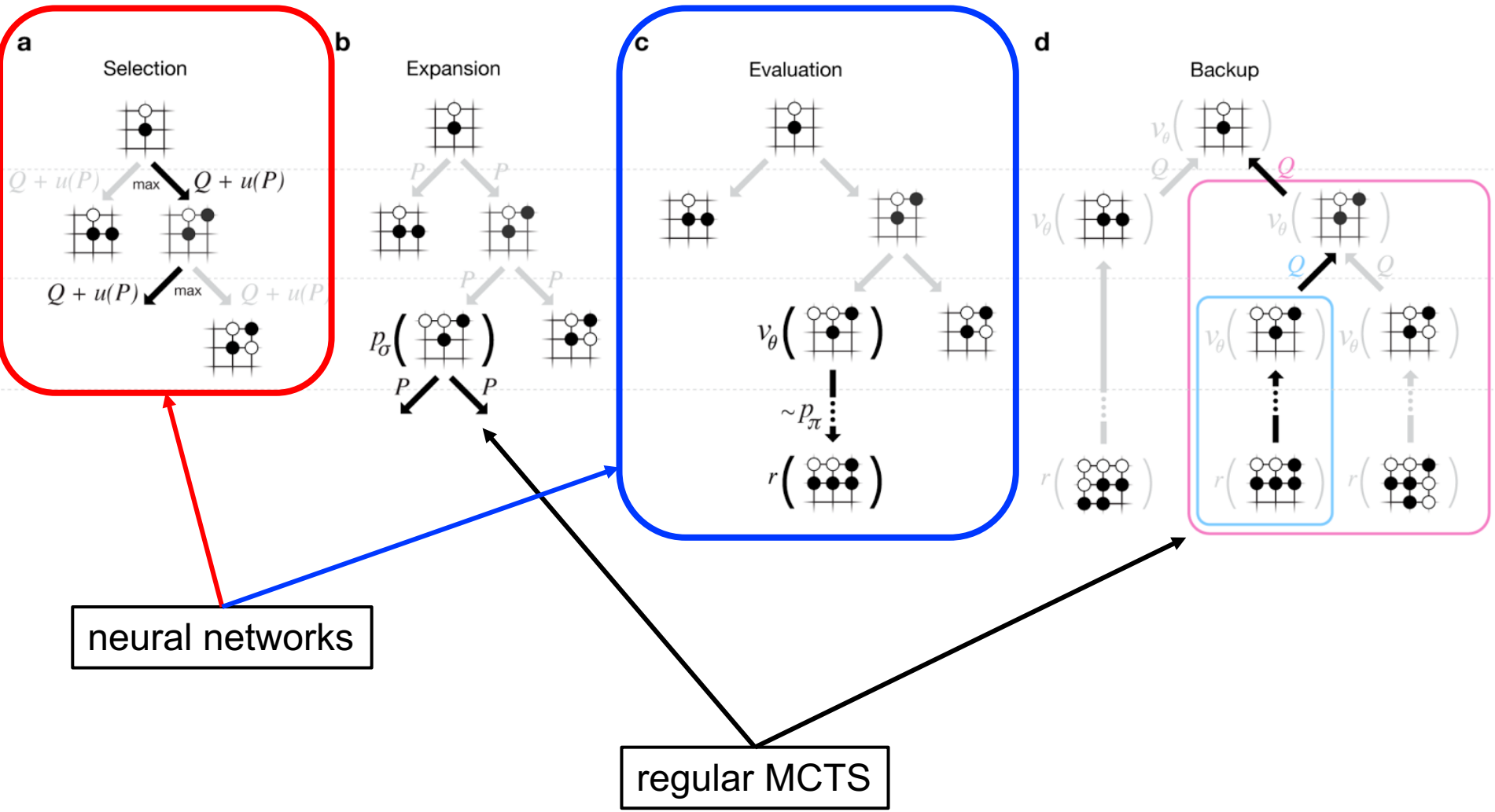
AlphaGo

2/17/17

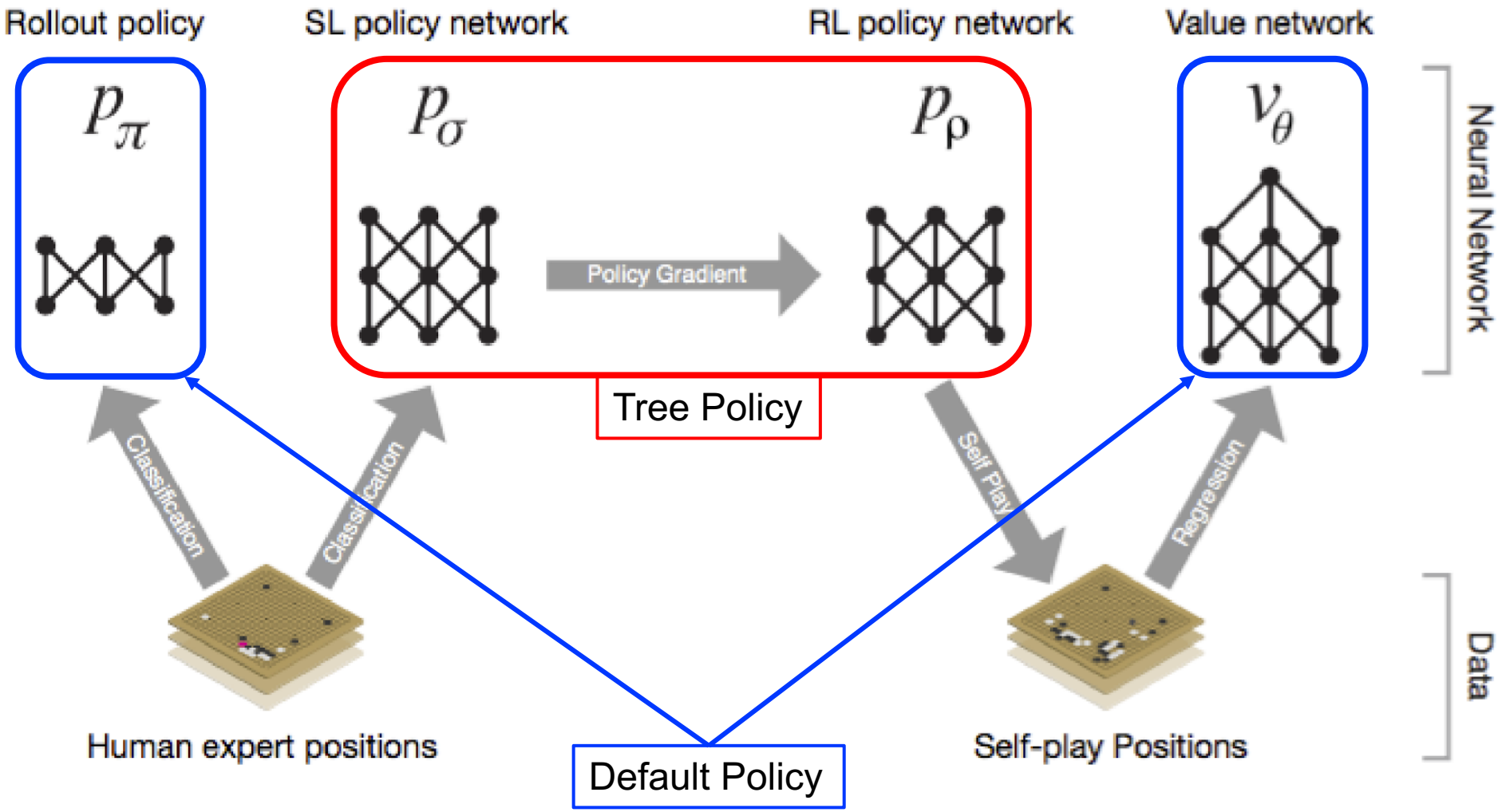
Video

https://www.youtube.com/watch?v=g-dKXOlsf98

Figure from the AlphaGo Paper



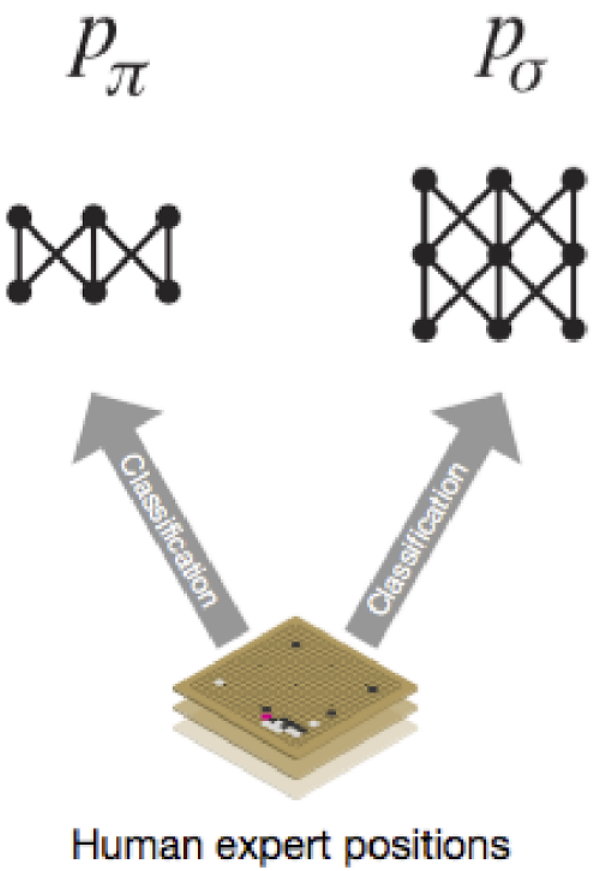
AlphaGo Neural Networks



Step 1: learn to predict human moves

- Used a large database of online expert games.

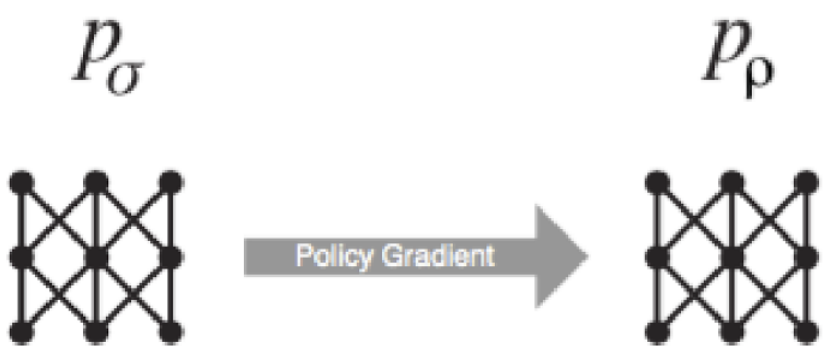
- Learned two versions of the neural network:
  - A fast network $\mathbf{P}_\pi$ for use in evaluation
  - An accurate network $\mathbf{P}_\sigma$ for use in selection.



CS63 topic
neural networks
weeks 8–9

Step 2: improve P_σ (accurate network)

- Run large numbers of self-play games.
- Update P_σ using reinforcement learning
  - weights updated by stochastic gradient descent

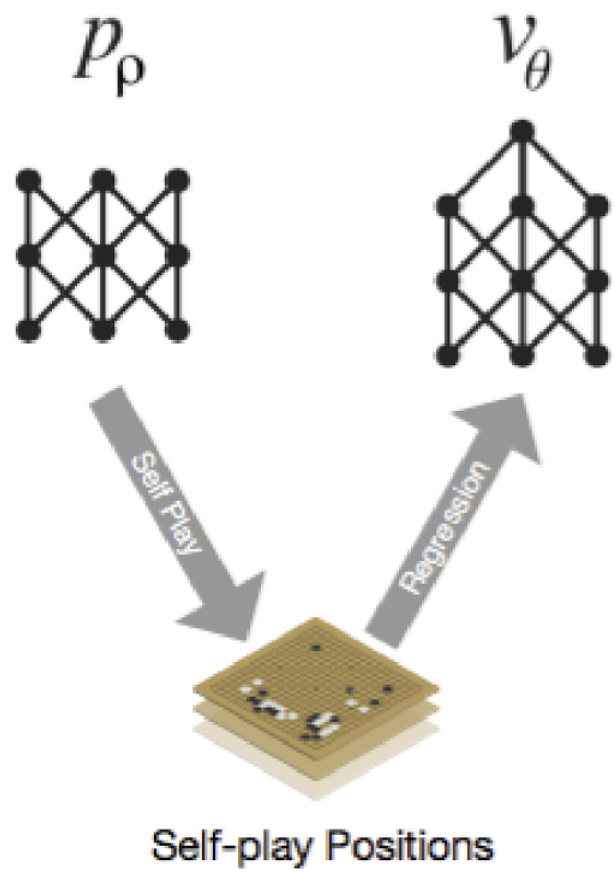


CS63 topic
reinforcement learning
weeks 6-7

CS63 topic
stochastic gradient descent
week 3

Step 3: learn a better boardEval V_θ

- use random samples from the self-play database
- prediction target: probability that black wins from a given board



CS63 topic
avoiding overfitting
weeks 9-10

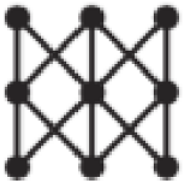
AlphaGo Tree Policy (selection)

- select nodes randomly according to weight:

$$value + \frac{prior}{1 + visits}$$

- ***prior*** is determined by the improved policy network

$$p_\rho$$



AlphaGo Default Policy (simulation)

When expanding a node, its initial value combines:

- an evaluation from value network V_θ
- a rollout using fast policy P_π

$$v(s) = \lambda * outcome \quad + \quad (1 - \lambda) * eval$$

A rollout according to P_π selects random moves with the estimated probability a human would select them instead of uniformly randomly.

AlphaGo Results

- Played Fan Hui (October 2015)
  - World #522.
  - AlphaGo won 5-0.
- Played Lee Sedol (March 2016)
  - World #5, previously world #1 (2007-2011).
  - AlphaGo won 4-1.
- Played against top pros (Dec 2016 – Jan 2017)
  - Included games against the word #1-4.
  - Games played online with short time limits.
  - AlphaGo won 60-0.

MCTS vs Bounded Min/Max

UCT / MCTS

- Optimal with infinite rollouts.
- Anytime algorithm (can give an answer immediately, improves its answer with more time).
- A heuristic is not required, but can be used if available.
- Handles incomplete information gracefully.

MinMax/Backward Induction

- Optimal once the entire tree is explored or pruned.
- Can prove the outcome of the game.
- Can be made anytime-ish with iterative deepening.
- A heuristic is required unless the game tree is small.
- Hard to use on incomplete information games.

Discussion: why use MCTS for go?

- We're using MCTS in lab because we don't want to write new heuristics for every game.
  - AlphaGo is all about heuristics. They're learned by neural networks, but they're still heuristics.

- MCTS handles randomness and incomplete information better than Min/Max.
  - Go is a deterministic, perfect information game.

So why does MCTS make so much sense for go?