Neural Networks for Reinforcement Learning in Challenging State Spaces

Ludwig Winkler

Machine Learning Group
TU Berlin

November 8, 2017

Outline

AlphaGo

AlphaGo Zero

Hierarchical RL

AlphaGo - Go

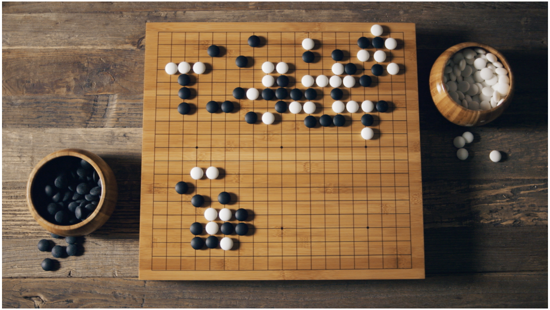
- Ancient board game from Asia on a 19×19 field
- More board configurations than atoms in the universe: $2 \cdot 10^{170}$

AlphaGo - Go

- Ancient board game from Asia on a 19×19 field
- More board configurations than atoms in the universe: $2 \cdot 10^{170}$
- Branching factor $b = 250$ and depth $d = 150$
- Chess: $b = 35$ and $d = 80$

AlphaGo - Go

- Ancient board game from Asia on a 19×19 field
- More board configurations than atoms in the universe: $2 \cdot 10^{170}$
- Branching factor $b = 250$ and depth $d = 150$
- Chess: $b = 35$ and $d = 80$
- Sheer number of positions makes brute force methods unrealistic



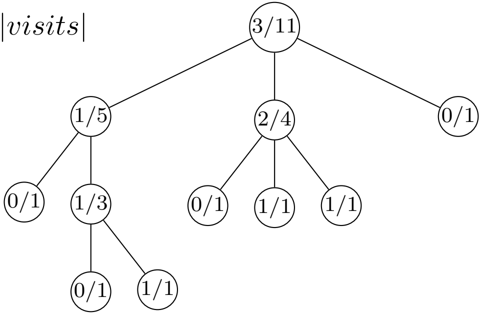
Monte Carlo Tree Search

- Go is a perfect information game
- All accesible information is available on the board
- Game tree of all possible board positions unfeasable
- MCTS approximates game tree with random play outs

Monte Carlo Tree Search

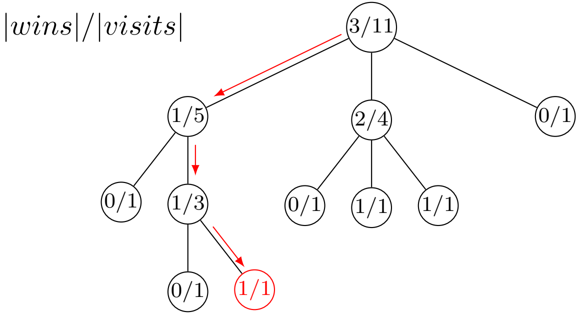
Game Tree

$|wins|/|visits|$



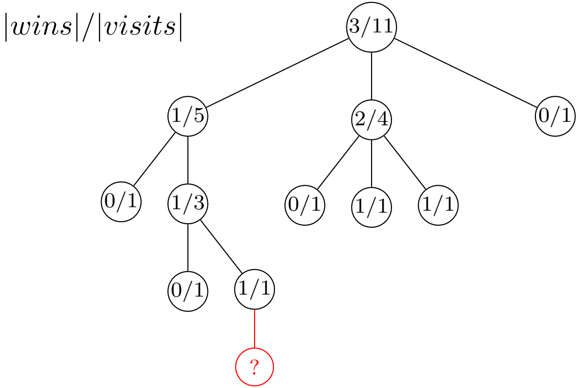
Monte Carlo Tree Search

Selection



Monte Carlo Tree Search

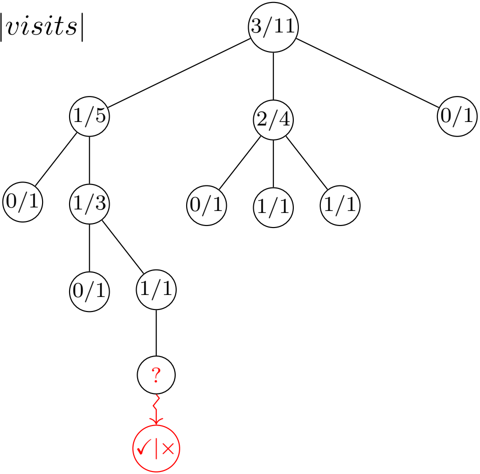
Selection → Expansion



Monte Carlo Tree Search

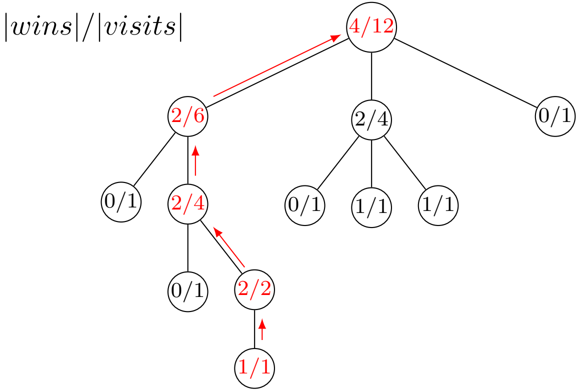
Selection → Expansion → Simulation

$|wins|/|visits|$



Monte Carlo Tree Search

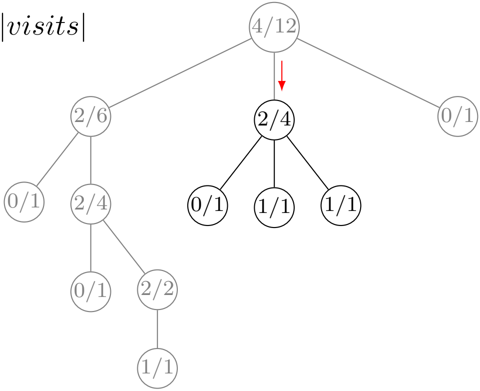
Selection → Expansion → Simulation → Backpropagation



Monte Carlo Tree Search

Selection → Expansion → Simulation → Backpropagation → Action

$|wins|/|visits|$



AlphaGo - Architecture

- Components of AlphaGo
  - Supervised policy network $p_{SL}(a|s)$
  - Reinforced policy network $p_{RL}(a|s)$
  - Value network $v(s)$
  - Fast rollout policy network $p_{FR}(a|s)$

AlphaGo - Architecture

- Components of AlphaGo
    - Supervised policy network $p_{SL}(a|s)$
    - Reinforced policy network $p_{RL}(a|s)$
    - Value network $v(s)$
    - Fast rollout policy network $p_{FR}(a|s)$
- Intialization with supervised learning
- Refinement with reinforcement learning

Policy and Value Network

- 12 convolutional layers with ReLu
- First layer $23 \times 23 \times 48$
- Padding of feature maps to $21 \times 21 \times 192$ for subsequent layers
- Final layer with softmax and individual bias
- Action from output probability $a_t \sim p(\cdot|s)$

Policy and Value Network

- 12 convolutional layers with ReLu
- First layer $23 \times 23 \times 48$
- Padding of feature maps to $21 \times 21 \times 192$ for subsequent layers
- Final layer with softmax and individual bias
- Action from output probability $a_t \sim p(\cdot|s)$

- Value network architecture similar to policy network
- FC and single *tanh*-unit for scalar output
- Predict win $z_T = +1$ or loss $z_T = -1$

Training of Policy Network

- Supervised training of policy network with 30 million expert positions
- Expert prediction with 57% accuracy

Training of Policy Network

- Supervised training of policy network with 30 million expert positions
- Expert prediction with 57% accuracy

- Reinforcement learning through self-play with previous versions
- Update with win $z_T = 1$ or loss $z_T = -1$

$$\Delta\theta \propto \nabla_\theta \log p_{RL}(a|s;\theta) \cdot \underbrace{z_T}_{\pm 1}$$

Training of Value Network

- Value function for strongest RL policy $p_{RL}(a|s)$
- Trained on state-outcome pairs

$$\Delta\theta \propto \nabla_\theta v(s;\theta) \cdot (z_T - v(s;\theta))$$

- Trained on self-play data between two identical policy networks

MCTS Program

- Each edge stores $Q(s,a)$, $N(a,s)$ and $p_{SL}(a|s) = P(a,s)$
- Choose action that maximizes the chance of winning

$$a = \max_{\tilde{a}} Q(s,\tilde{a}) + u(s,\tilde{a}), \qquad u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

MCTS Program

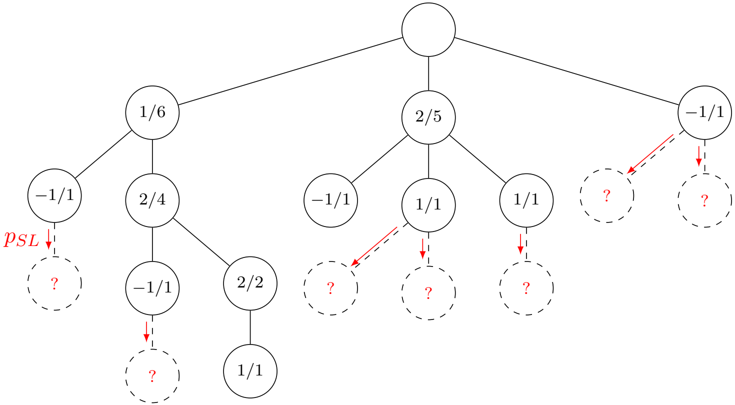
- Each edge stores $Q(s,a)$, $N(a,s)$ and $p_{SL}(a|s) = P(a,s)$
- Choose action that maximizes the chance of winning

$$a = \max_{\tilde{a}} Q(s,\tilde{a}) + u(s,\tilde{a}), \qquad u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$$

- $p_{SL}(a|s)$ selects more human-like search directions
- $Q(s,a)$ from mixed value network and rollouts evaluations

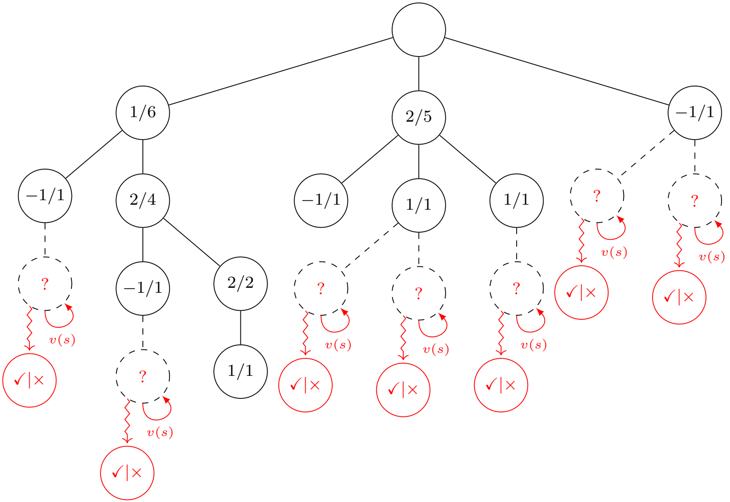
Play with MCTS Algorithm

Selection/Expansion



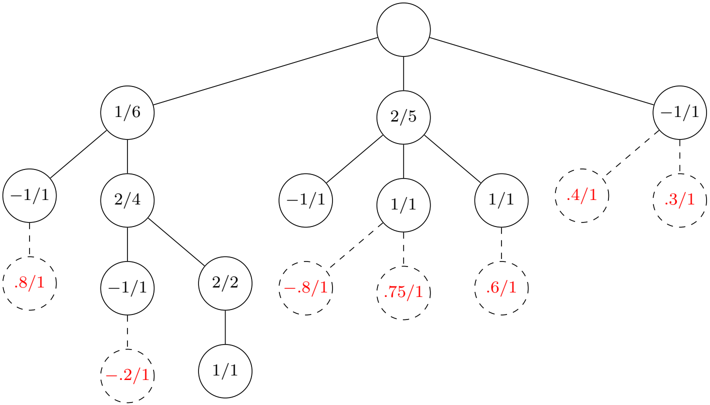
Play with MCTS Algorithm

Selection/Expansion



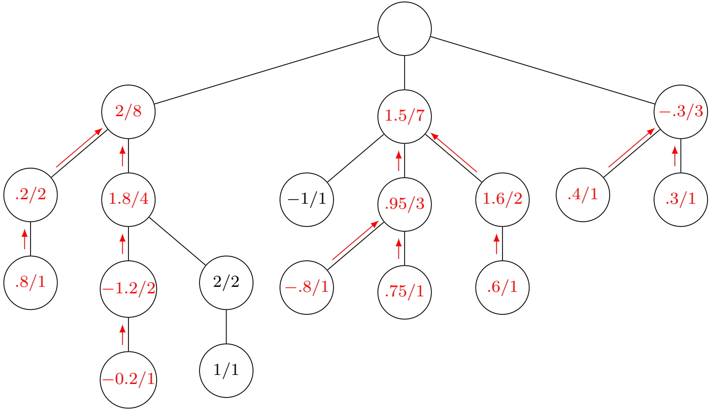
Play with MCTS Algorithm

Selection/Expansion → Simulation



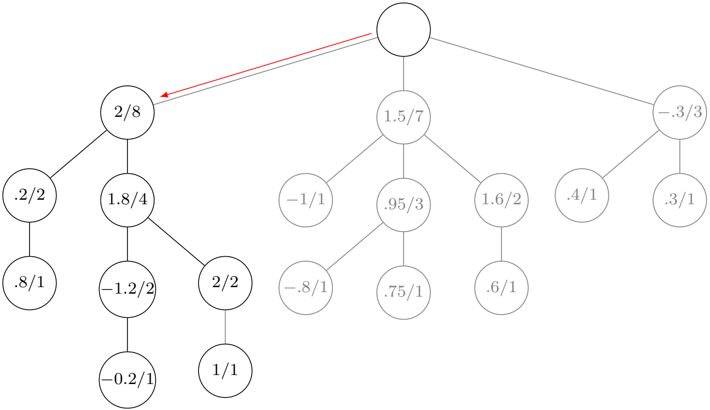
Play with MCTS Algorithm

Selection/Expansion → Simulation → Backpropagation

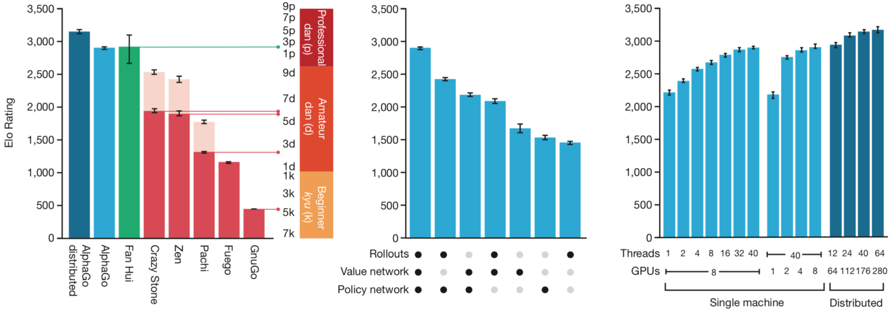


Play with MCTS Algorithm

Selection/Expansion → Simulation → Backpropagation → Action



Evaluation


Elo Rating
3,500
3,000
2,500
2,000
1,500
1,000
500
0
AlphaGo distributed
AlphaGo
Fan Hui
Crazy Stone
Zen
Pachi
Fuego
GnuGo
9p
7p
5p
3p
1p
9d
7d
5d
3d
1d
1k
3k
5k
7k
Professional dan (p)
Amateur dan (d)
Beginner kyu (k)
Rollouts
Value network
Policy network
Threads 1 2 4 8 16 32 40 40 12 24 40 64
GPUs 8 1 2 4 8 64 112 176 280
Single machine
Distributed

AlphaGo Zero - Motivation

- AlphaGo achieved super-human play strength
  - Supervised learning for pretraining
  - Reinforcement learning for policy network
  - Regression for value network
  - MCTS with policy/value network and rollout policy

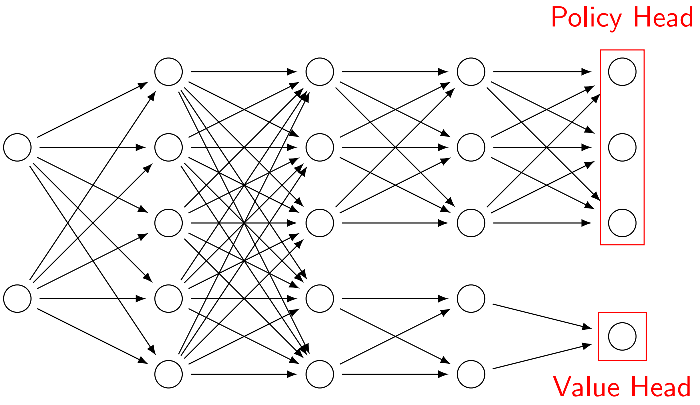
AlphaGo Zero - Motivation

- AlphaGo achieved super-human play strength
  - Supervised learning for pretraining
  - Reinforcement learning for policy network
  - Regression for value network
  - MCTS with policy/value network and rollout policy

- AlphaGo Zero streamlined the learning process
  - Training only through reinforcement learning
  - Multi-headed, single network for policy and evaluation
  - Residual blocks
  - MCTS with single network

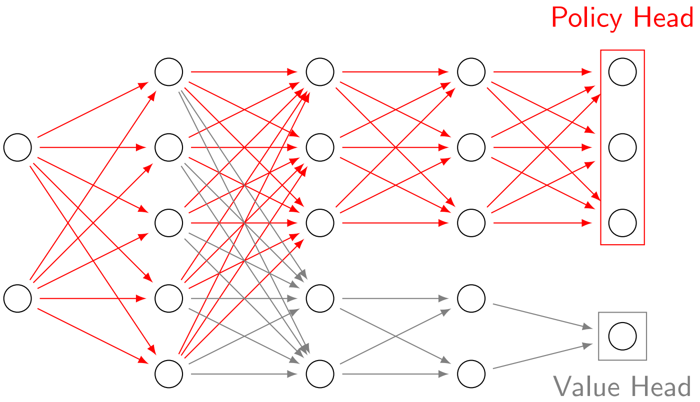
Combined Policy and Value Network

Policy and Value Head



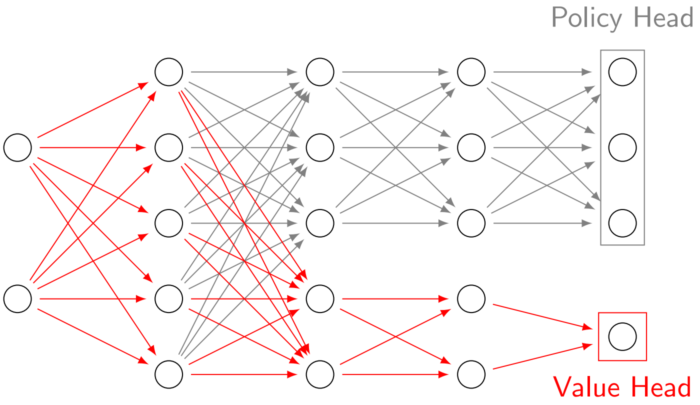
Combined Policy and Value Network

Policy Network



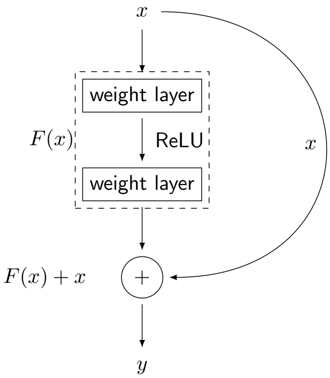
Combined Policy and Value Network

Value Network



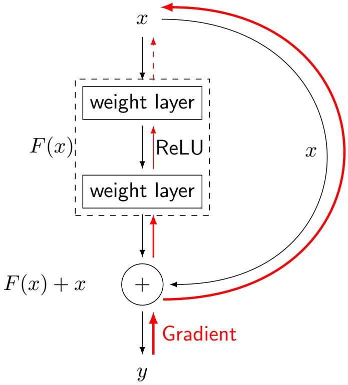
Residual Network

- Shortcuts over layer blocks
- Gradients can skip layers
- Propagation of gradient to first layer
- Allows for deeper networks
- Extensions: Highway Nets, Dense Nets



Residual Network

- Shortcuts over layer blocks
- Gradients can skip layers
- Propagation of gradient to first layer
- Allows for deeper networks
- Extensions: Highway Nets, Dense Nets

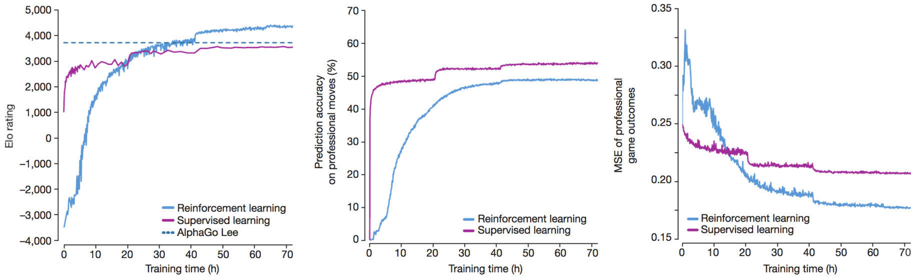


MCTS Program

- Policy head picks search beam directions in MCTS
- Value head evaluates positions in MCTS
- MCTS search probabilities used as policy head targets
- Self-play game outcome used for value head targets

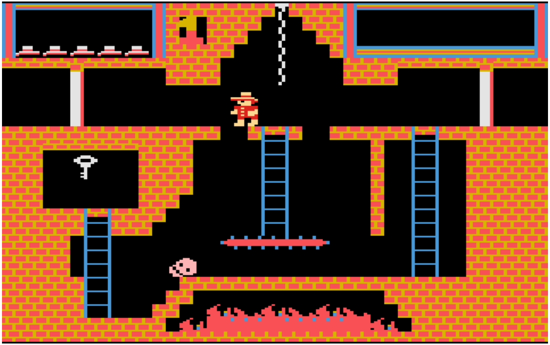
- MCTS can be interpreted as powerful policy improvement operator

Evaluation


5,000
4,000
3,000
2,000
1,000
0
−1,000
−2,000
−3,000
−4,000
Elo rating
0 10 20 30 40 50 60 70
Training time (h)
Reinforcement learning
Supervised learning
AlphaGo Lee
70
60
50
40
30
20
10
0
Prediction accuracy on professional moves (%)
0 10 20 30 40 50 60 70
Training time (h)
Reinforcement learning
Supervised learning
0.35
0.30
0.25
0.20
0.15
MSE of professional game outcomes
0 10 20 30 40 50 60 70
Training time (h)
Reinforcement learning
Supervised learning

Hierarchical RL - Motivation

- Long-term credit assignment problem
- Sparse rewards makes learning hard
- Sequences of sub-goals have to be fulfilled



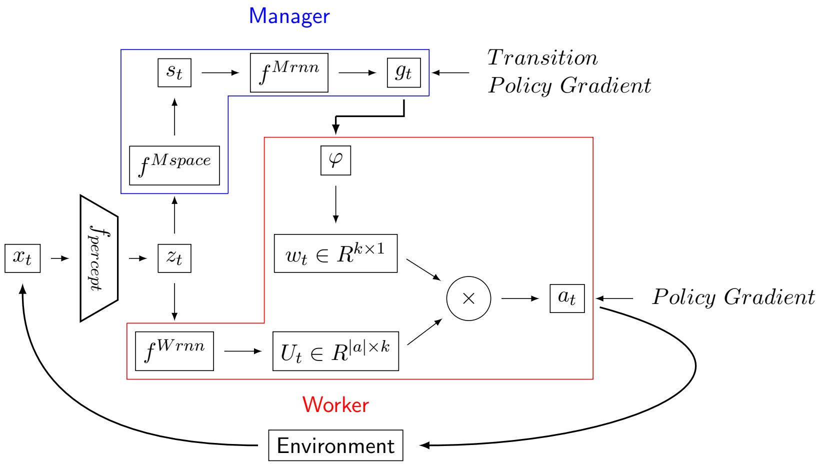
Architecture

- Hierarchy within agent
- Decoupling of goal setting from goal achievement
- Manager sets directional goals for worker in latent space

Architecture

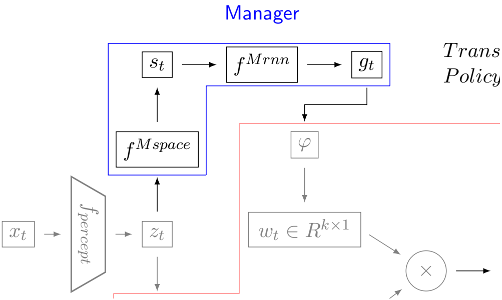
- Hierarchy within agent
- Decoupling of goal setting from goal achievement
- Manager sets directional goals for worker in latent space
- Manager works at lower temporal resolution
- Implementation with multiple neural networks

Architecture



Manager

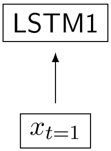
- Transforms joint z_t to internal s_t
- RNN f_{Mrnn} sets goal g_t for worker
- Dilated LSTMs for greater temporal reach
- Past goals are pooled



Dilated LSTM

- N LSTM's that activate every N steps
- Same parameters for all N LSTM's

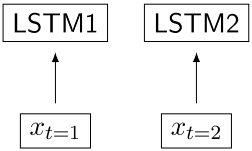
- E.g. 3 LSTM's that activate seperately in turn



Dilated LSTM

- N LSTM's that activate every N steps
- Same parameters for all N LSTM's

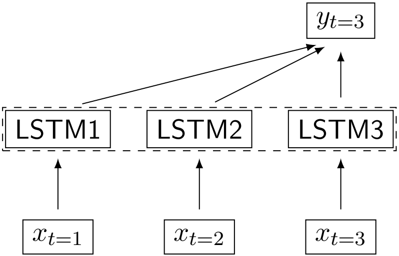
- E.g. 3 LSTM's that activate seperately in turn



Dilated LSTM

- N LSTM's that activate every N steps
- Same parameters for all N LSTM's

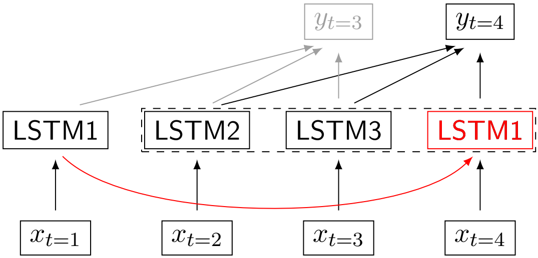
- E.g. 3 LSTM's that activate seperately in turn



Dilated LSTM

- N LSTM's that activate every N steps
- Same parameters for all N LSTM's

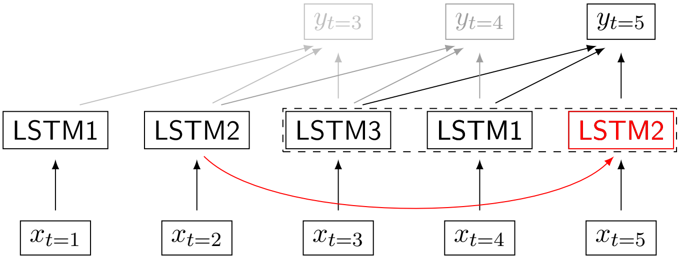
- E.g. 3 LSTM's that activate seperately in turn



Dilated LSTM

- N LSTM's that activate every N steps
- Same parameters for all N LSTM's

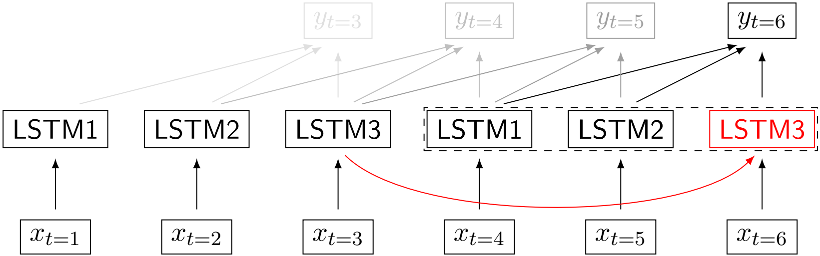
- E.g. 3 LSTM's that activate seperately in turn



Dilated LSTM

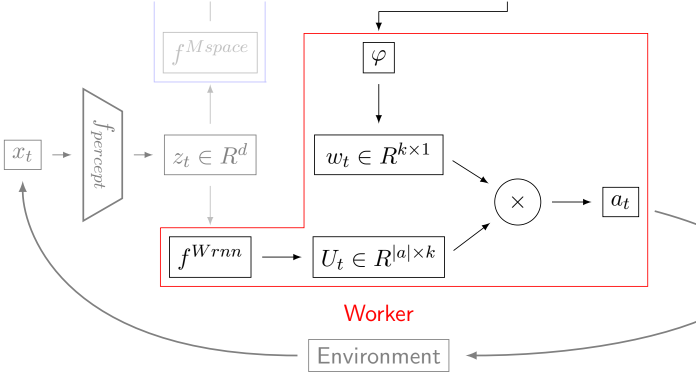
- N LSTM's that activate every N steps
- Same parameters for all N LSTM's

- E.g. 3 LSTM's that activate seperately in turn



Worker

- RNN f^{Wrnn} computes workers embedding matrix U_t
- φ projects bias-free goals to embedding w_t
- Workers output U_t is modulated by w_t



Training - Manager

- Independent training of Manager and Worker
    - No gradient between Manager and Worker

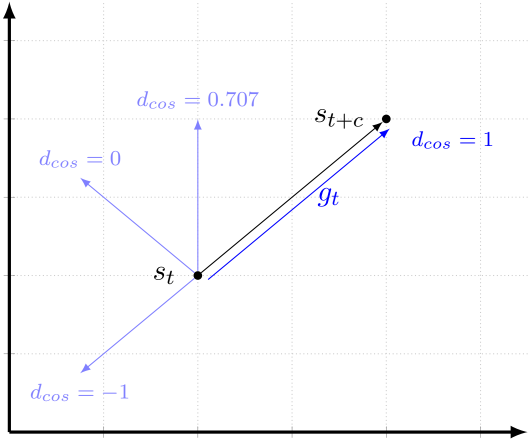
Training - Manager

- Independent training of Manager and Worker
  - No gradient between Manager and Worker
- Manager trained on advantageous directions in latent space

$$\nabla g_t = A^M(s,a)\ \nabla \underbrace{d_{cos}(s_{t+c} - s_t, g_t)}_{\text{Cost Function}}$$

- Advantage function $A^M(s,a)$ trained on external reward R_t
- Cosine similarity $d_{cos}(\underline{a}, \underline{b})$ measures alignment

Training - Manager



Training - Worker

- Worker trained on intrinsic and extrinsic reward

$$\nabla \pi_t = A^W(s,a)\ \nabla \log \pi(a_t|x_t;\theta)$$

Training - Worker

- Worker trained on intrinsic and extrinsic reward

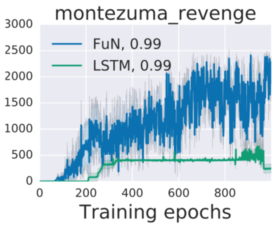
$$\nabla \pi_t = A^W(s, a)\ \nabla \log \pi(a_t | x_t; \theta)$$

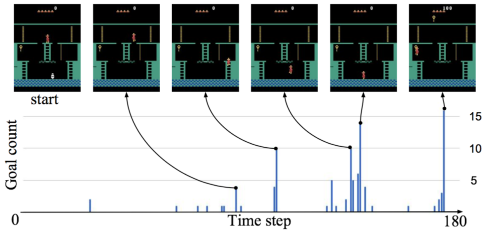
- Internal reward R_t^I measures alignment

$$\begin{aligned} R_t^W &= R_t + \alpha R_t^I \\ &= R_t + \alpha \frac{1}{c} \sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_{t-i}) \end{aligned}$$

Evaluation

- 1B observations for Montezuma and hyperparameter grid-search
- Strong in Montezumas Revenge, Enduro, Frostbite
- Similarly strong on Breakout, Seaquest, Space Invaders





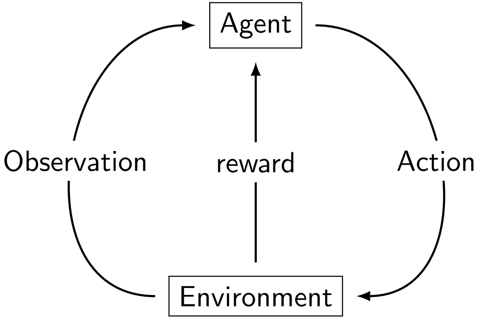
Sources

- 'Mastering the game of Go with deep neural networks and tree search', Silver et al.
- 'Mastering the game of Go without human knowledge', Silver et al.
- 'Reinforcement Learning', Sutton & Barto
- 'FeuDal Networks for Hierarchical Reinforcement Learning', Vezhenevets et al.

Thank you

RL Intuition

- In between supervised and unsupervised learning
- Take actions in an environment that maximize reward
  - Actions $\mapsto$ Policy
  - Environment $\mapsto$ States
  - Reward $\mapsto$ Feedback from environment



- State avlue $V(s)$ and action value $Q(s,a)$
- Minimize MSE between reward $R(s,a)$ and $V(s)$, $Q(s,a)$

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi\left[\left(R(s^{(t)}, a^{(t)}) + \gamma V(s^{(t+1)}) - V(s^{(t)})\right)^2\right]$$

$$J(\boldsymbol{\theta}) = \mathbb{E}_\pi\Big[\big(\underbrace{R(s^{(t)}, a^{(t)})}_{\text{Reward}} + \gamma \underbrace{Q(s^{(t+1)}, a^{(t+1)})}_{\text{Next Action}} - \underbrace{Q(s^{(t)}, a^{(t)})}_{\text{Action}}\big)^2\Big]$$

- State value function $V(s)$ as baseline
- Compare action value $Q(s,a)$ to state value $V(s)$
- Difference between the state-action value and the state value

$$A(s,a) = Q(s,a) - V(s)$$

- Better-than-average or worse-than-average actions

$$\begin{aligned} V(s) &= 100 \\ Q(s,a_1) = 90 \quad &\rightarrow \quad A(s,a_1) = -10 \\ Q(s,a_2) = 110 \quad &\rightarrow \quad A(s,a_2) = 10 \end{aligned}$$

- Directly learn a stochastic policy π_θ
- Total reward of following policy

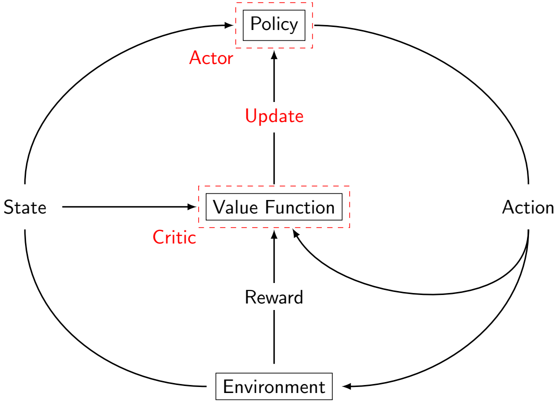
$$J(s) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\sum_{t=0}^{T} R(s^{(t)}, a^{(t)}) \right]$$

- Replace trajectory reward with action value $Q(s, a)$

$$J(s) \approx \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[Q(s, a) \right]$$
$$\nabla_{\boldsymbol{\theta}} J(s) \approx \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \left[\underbrace{Q(s, a)}_{\text{Critic Network}} \nabla_{\boldsymbol{\theta}} \log \left[\underbrace{\pi_{\boldsymbol{\theta}}(a|s)}_{\text{Policy Network}} \right] \right]$$

Deep Reinforcement Learning

Actor-Critic



Deep Reinforcement Learning

Training

- Training on stationary distributions in supervised learning
- Stable training with $i.i.d.$ data
- RL environments are non-stationary and highly correlated

- Aproximate stationary distribution with Experience Replay
- Store transitions (s, a, r, s) in buffer
- Sample mini-batches to decorrelate training data

- Different agents with separate exploration strategies and environments

$$J(\boldsymbol{\theta}) = \mathbb{E}_{\pi_{\boldsymbol{\theta}}}\left[\overbrace{(Q(s,a) - V(s))}^{A(s,a)} \nabla_{\boldsymbol{\theta}} \log\left[\pi_{\boldsymbol{\theta}}(a|s)\right]\right]$$