Memory-Augmented Monte Carlo Tree Search

Chenjun Xiao, Jincheng Mei and Martin Müller

University of Alberta

UNIVERSITY OF ALBERTA

NSERC CRSNG

RL & AI

Contributions

- Framework for Online Value Approximation
- Theoretical Analysis
- Design Memory and Integrate with MCTS
- Experiments in the Game of Go

Monte Carlo Tree Search

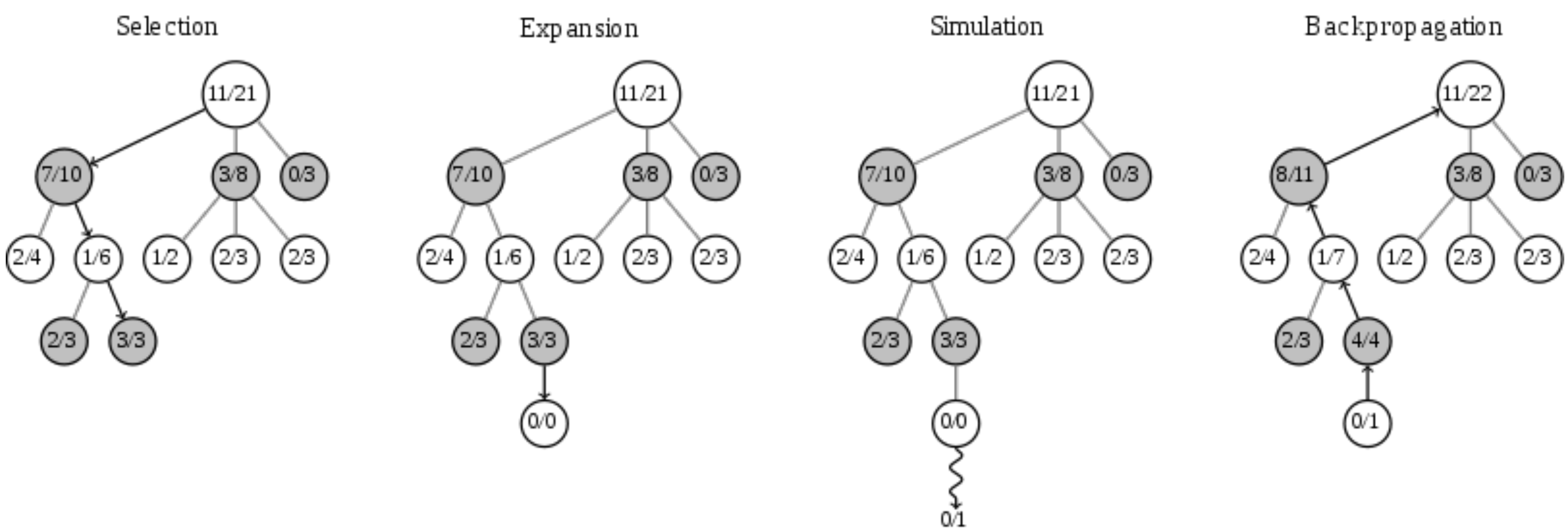
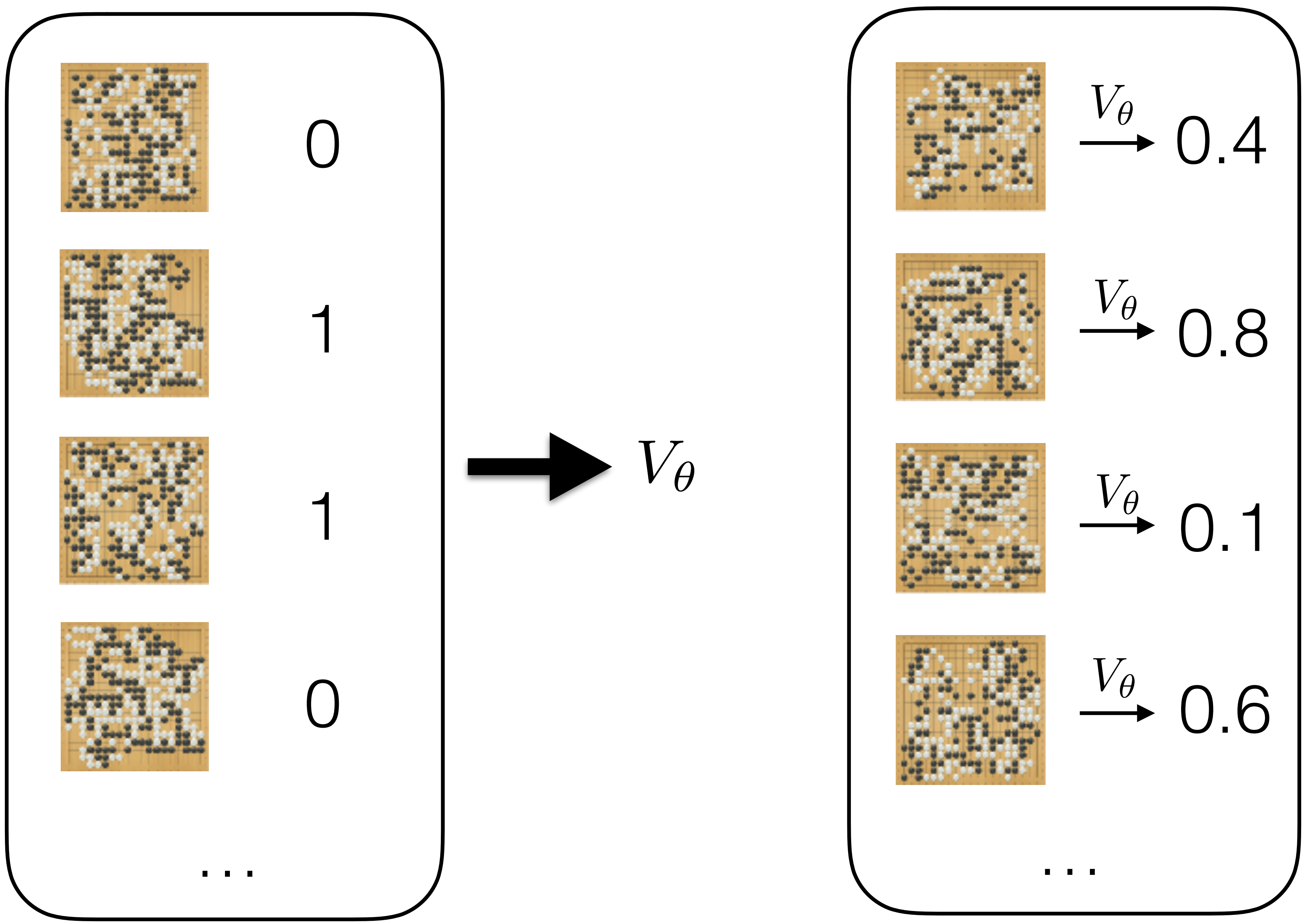


Image source: http://en.wikipedia.org/wiki/Monte-Carlo_tree_search

Value Approximation

Generalization is the key!



MCTS in AlphaGo

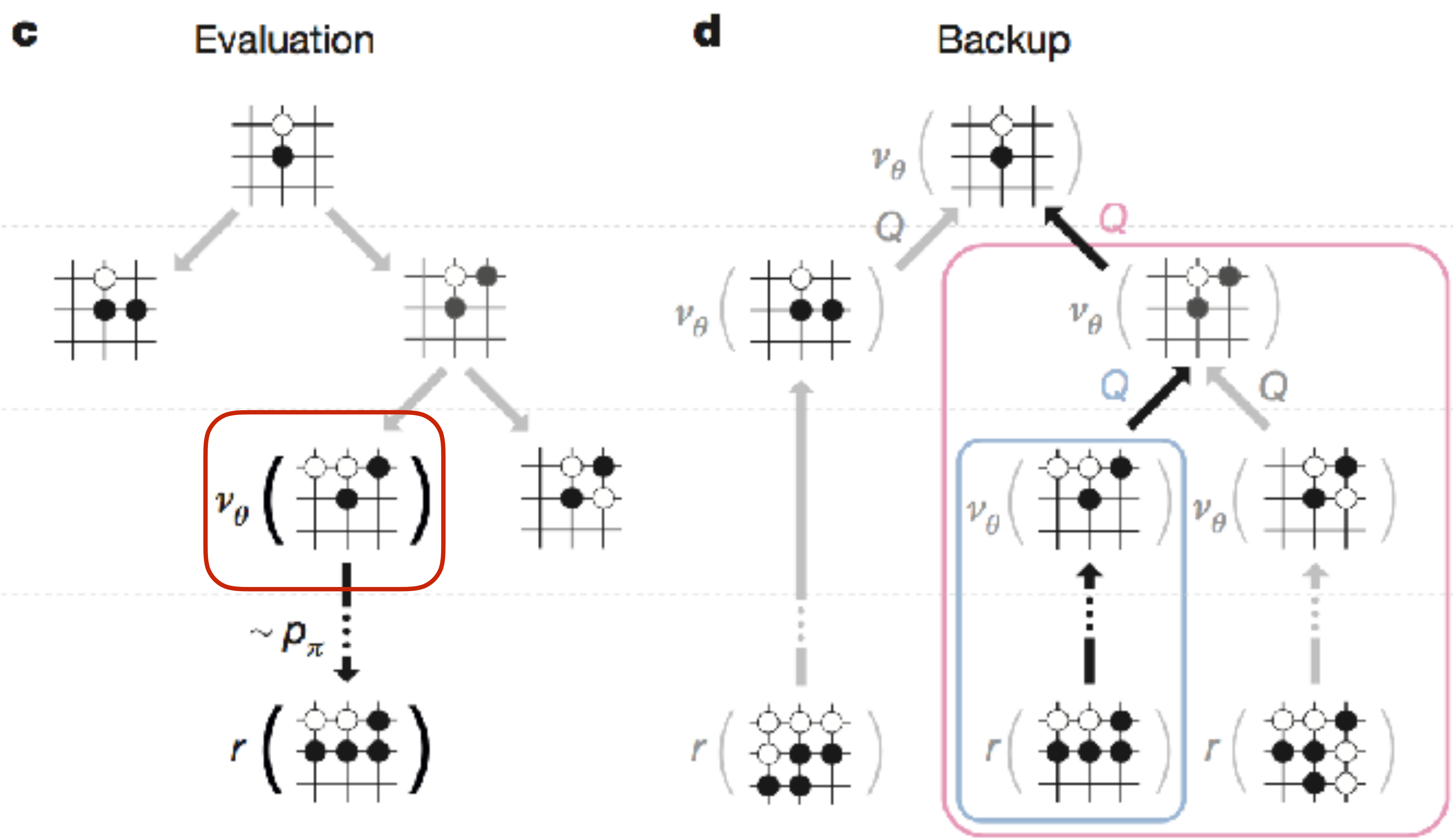
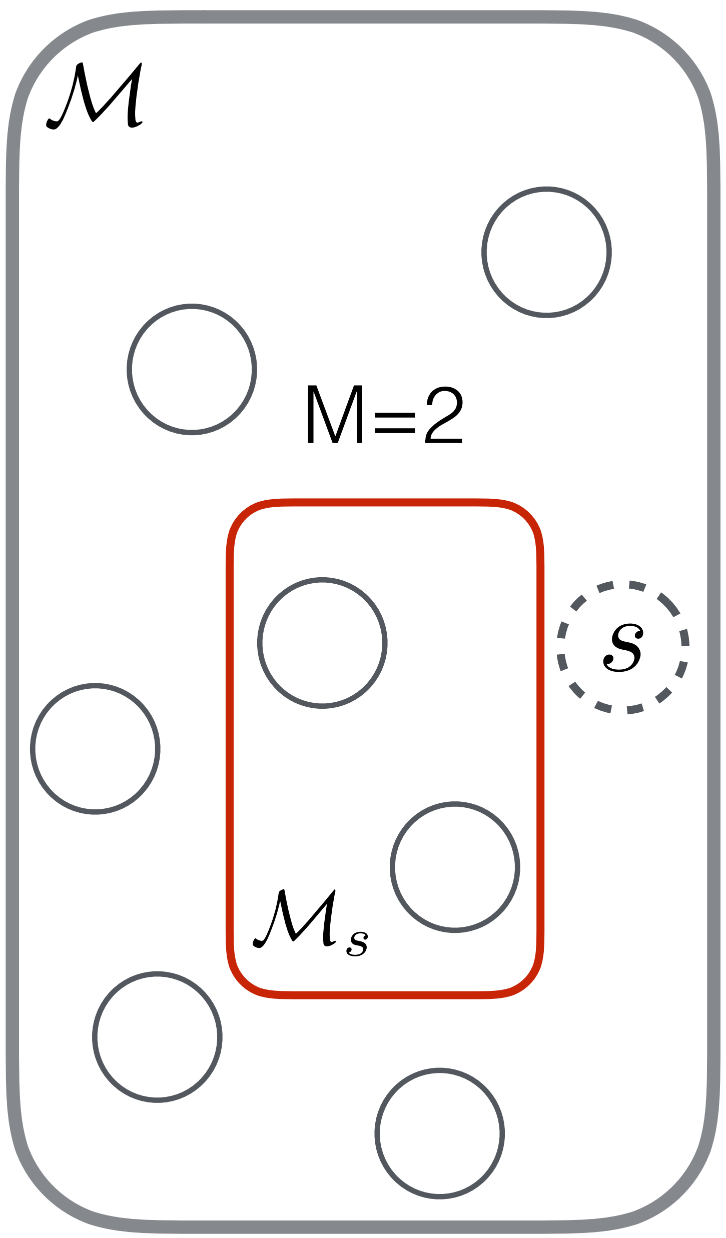


Image source: Mastering the game of Go with deep neural networks and tree search

Online Value Approximation



$$\delta_s = |\hat{V}(s) - V^*(s)|$$

$$\varepsilon_{s,x} = |V^*(s) - V^*(x)|$$

Assumption:

δ_s is sub-gaussian

$$\varepsilon_M = \max_{i \in \mathcal{M}_s} \varepsilon_{s,i} \in [0, \varepsilon]$$

Online Value Approximation

- Memory Value:

$$\hat{V}_{\mathcal{M}}(x) = \sum_{i=1}^{M} w_i(x)\hat{V}(i) \quad s.t. \sum_{i=1}^{M} w_i(x) = 1$$

- Memory Value error:

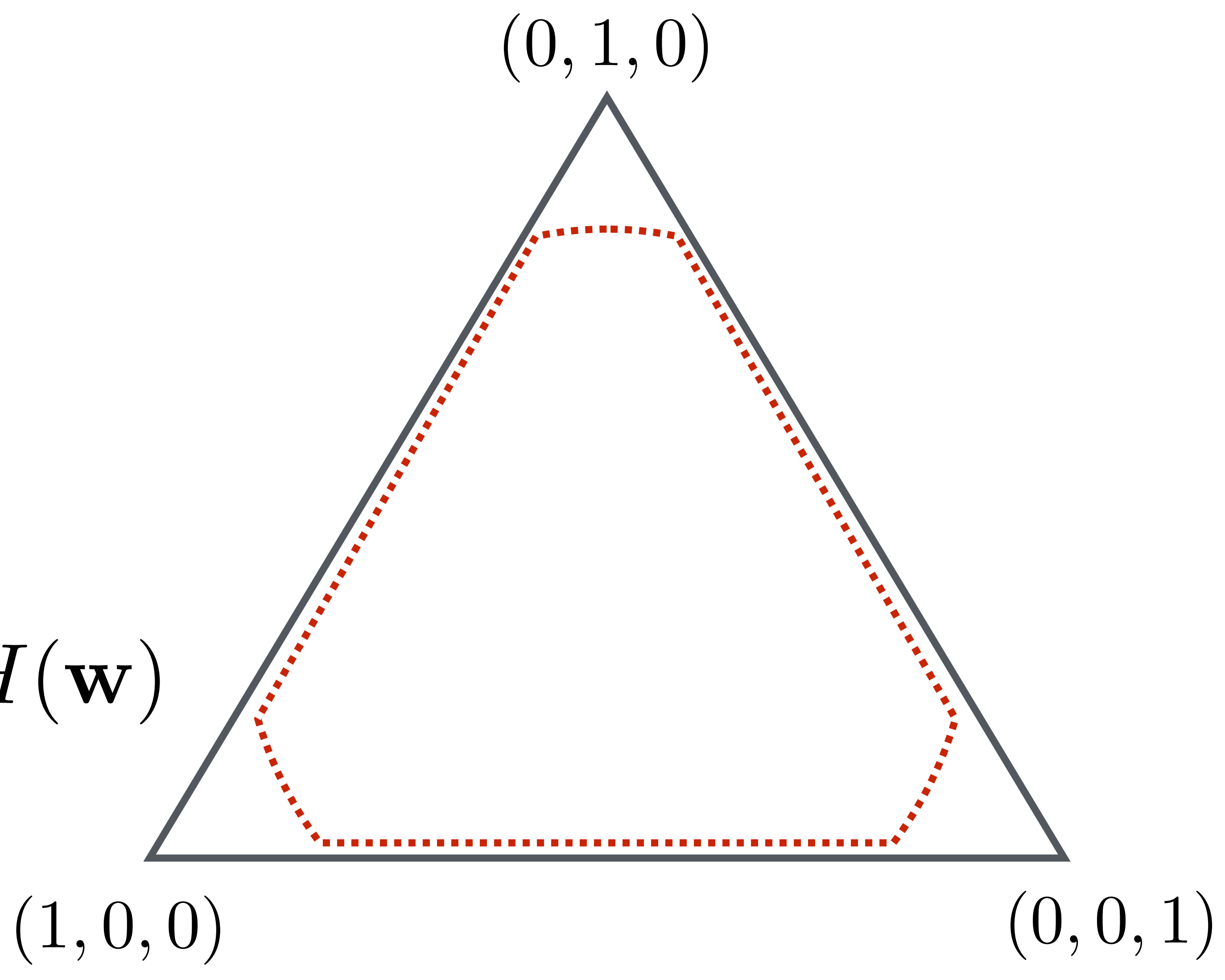
$$|\sum_{i=1}^{M} w_i(x)\hat{V}(i) - V^*(x)| \leq \sum_{i=1}^{M} w_i(x)(\delta_i + \varepsilon_{i,x})$$

Entropy Regularized Optimization

Let $q_i = -(\delta_i + \varepsilon_i)$

- $\max_{\mathbf{w} \in \Delta} \mathbf{w} \cdot \mathbf{q}$

- $\max_{\mathbf{w} \in \Delta} \mathbf{w} \cdot \mathbf{q} + \tau H(\mathbf{w})$



Entropy Regularized Optimization

- The "softmax" : $F_\tau(\mathbf{q}) = \tau \log(\sum_{i=1}^{M} e^{q_i/\tau})$
- The "soft indmax" : $f_\tau(\mathbf{q}) = \frac{e^{\mathbf{q}/\tau}}{\sum_{i=1}^{M} e^{q_i/\tau}} = e^{(\mathbf{q}-F_\tau(\mathbf{q}))/\tau}$

Lemma. *(Nachum et al. 2017; Haarnoja et al. 2017; Ziebart 2010)*

$$
\begin{aligned}
F_\tau(\mathbf{q}) &= \max_{\mathbf{w}\in\Delta}\{\mathbf{w}\cdot\mathbf{q} + \tau H(\mathbf{w})\} \\
&= f_\tau(\mathbf{q})\cdot\mathbf{q} + \tau H(f_\tau(\mathbf{q}))
\end{aligned}
$$

Main Theorem

- Choose weights $\mathbf{w} = f_\tau(-\mathbf{c})$
- For states with large sampling error $\delta_x > \varepsilon$
- With large enough number of simulations of "addressed" neighbour states $n = \sum_{i=1}^{M} N_i$
- Memory value is better than MC value with high probability

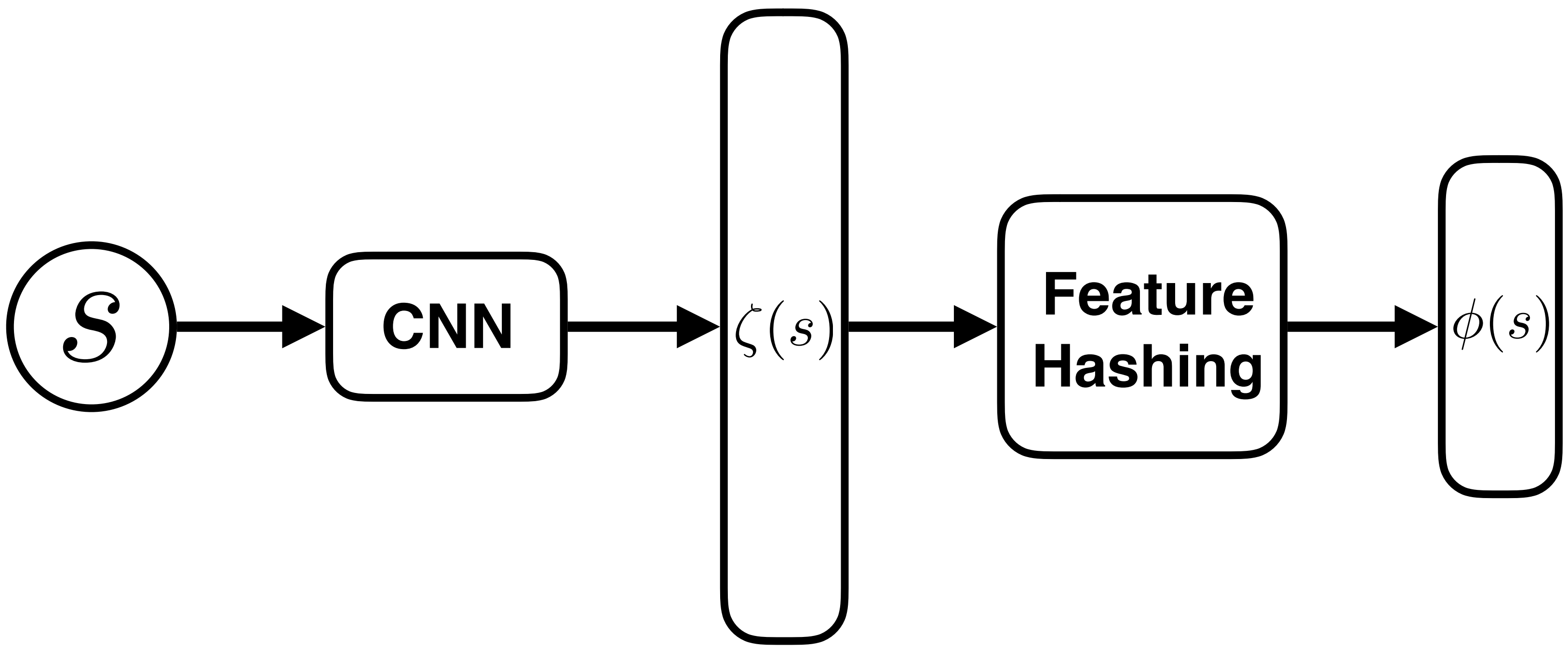
From Theory to Application

- Approximate optimal weights
- Design of memory and operations
- Integrate memory in MCTS

Approximate Optimal Weight

- Approximate simulation error: $\delta_i \propto 1/N_i$
- Approximate similarity: $\varepsilon_{i,x} \approx d(i,x) = -\cos(\phi(i), \phi(x))$
- Approximate weight: $w_i(x) = \frac{N_i \exp(-d(i,x)/\tau)}{\sum_{j=1}^{M} N_j \exp(-d(j,x)/\tau)}$

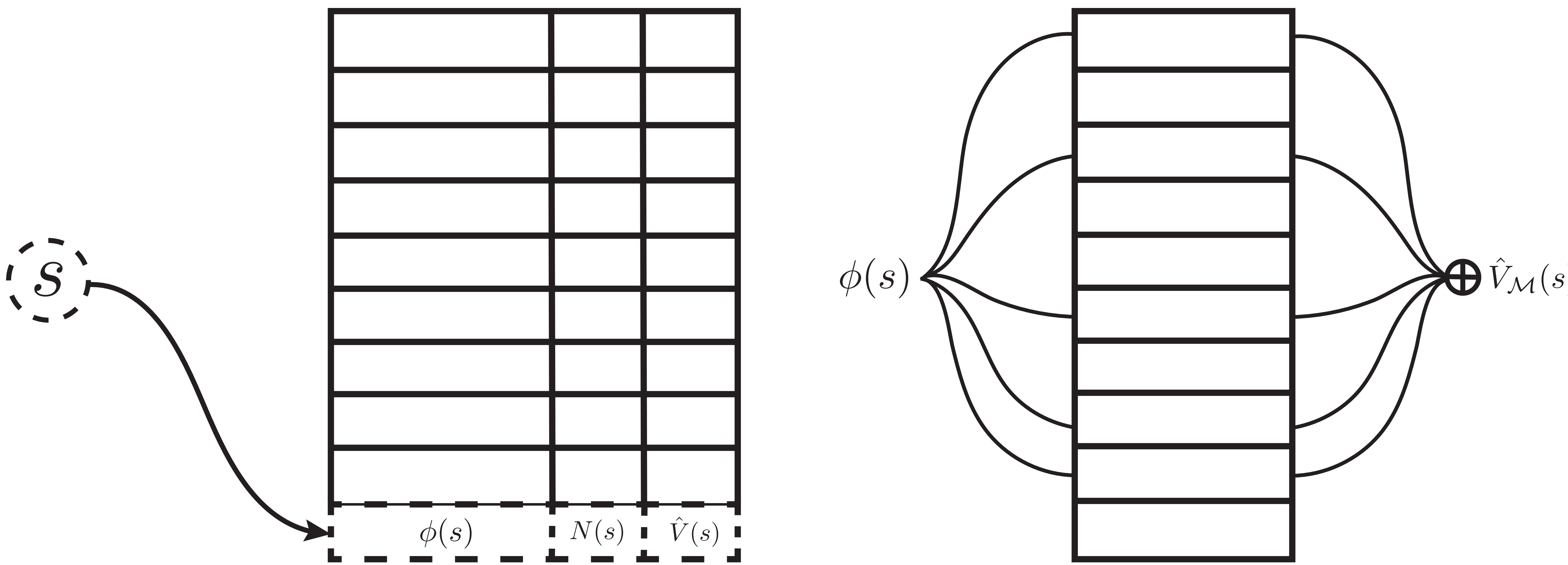
Feature Representation



- Unbiased property of Feature Hashing (Weinberger et al. 2009):

$$\mathbb{E}[\cos(\phi(s), \phi(x))] = \cos(\zeta(s), \zeta(x))$$

Design of Memory

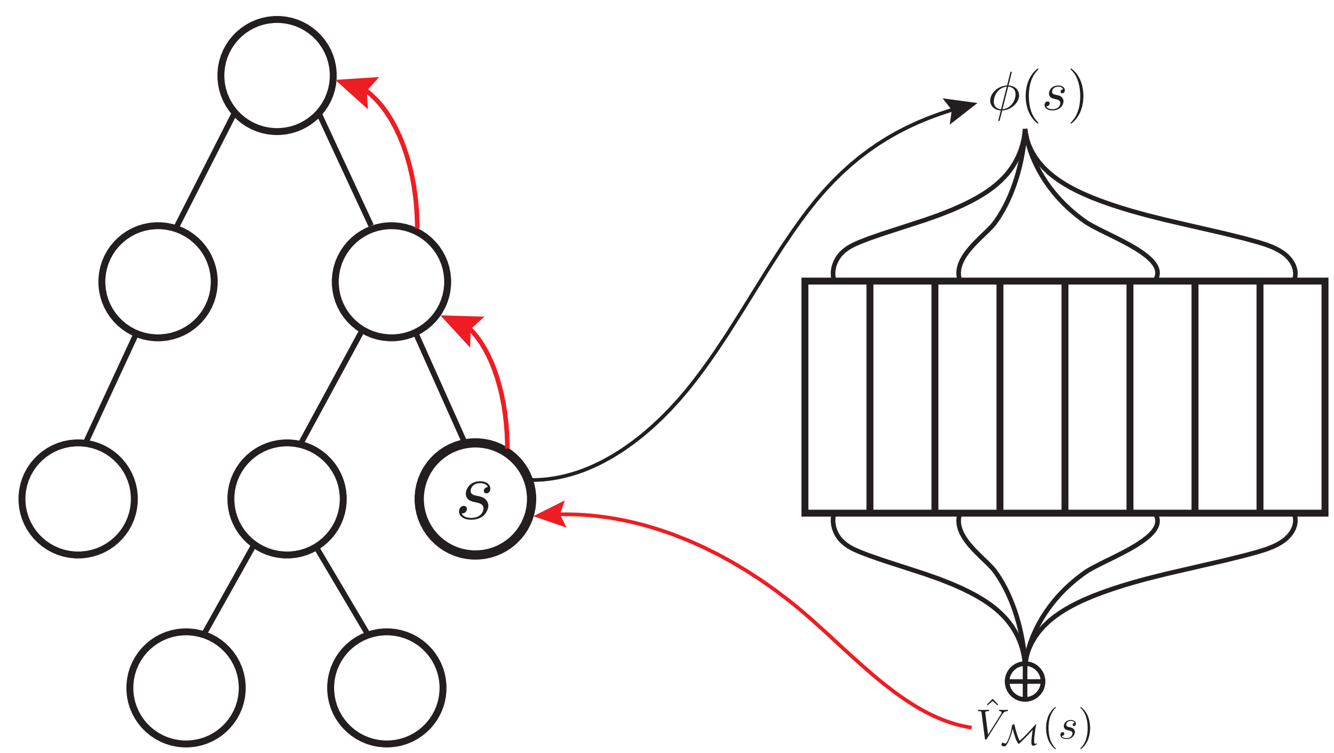


Add/Update

Query

M-MCTS

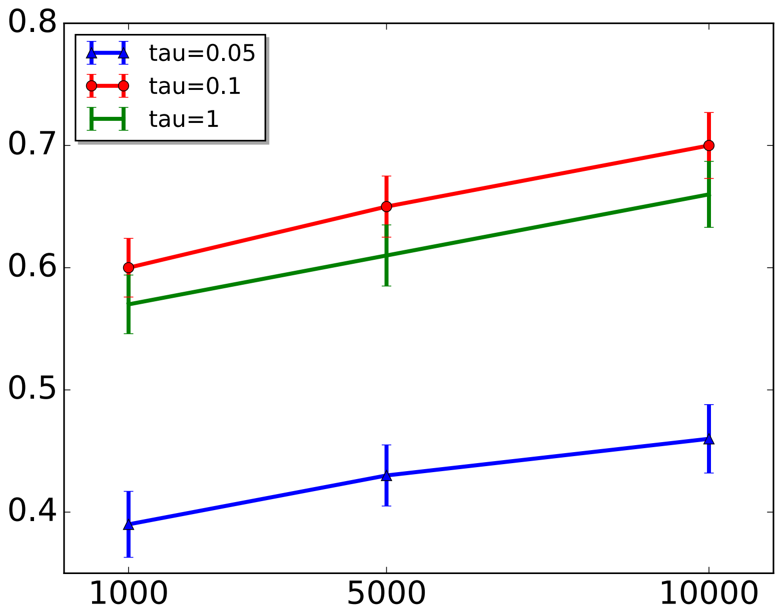
- Selection: compute state value by $V(s) = (1 - \lambda_s)\hat{V}_s + \lambda_s \hat{V}_{\mathcal{M}}$
- Evaluation: evaluate states by both MC and memory
- Backup: update MC value and memory value in tree



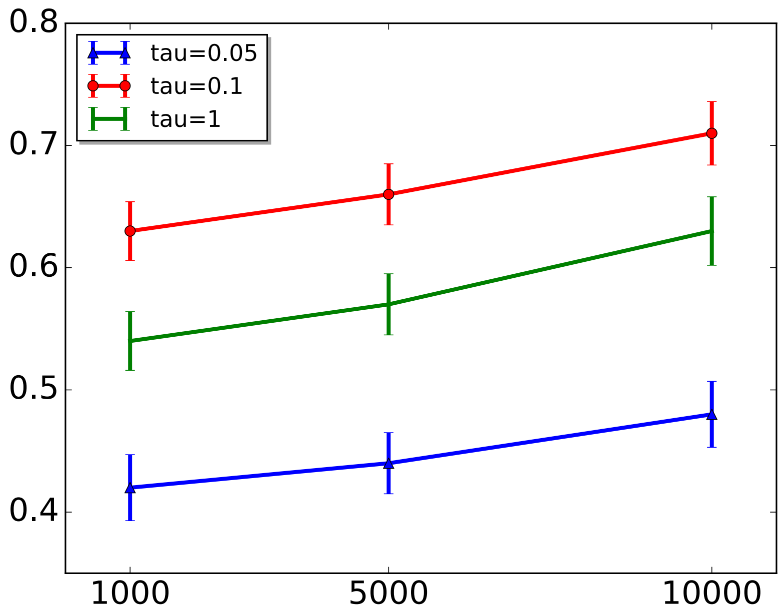
Experiments

- Implementation based on Go program Fuego
- Baseline: Fuego + Policy network (CNN)
- Two tests:
    - Test neighbourhood size M and temperature τ
    - Test the size of memory
- Test scaling with number of simulations

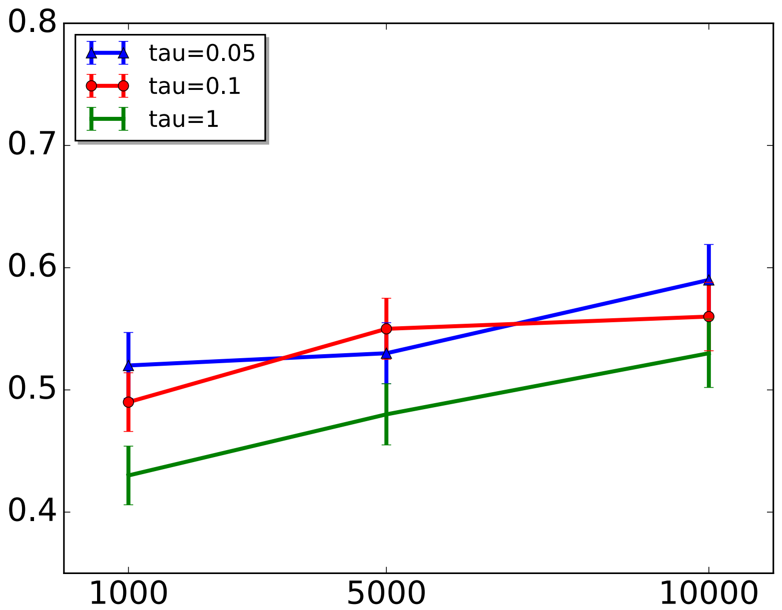
Varying M and τ



M=20

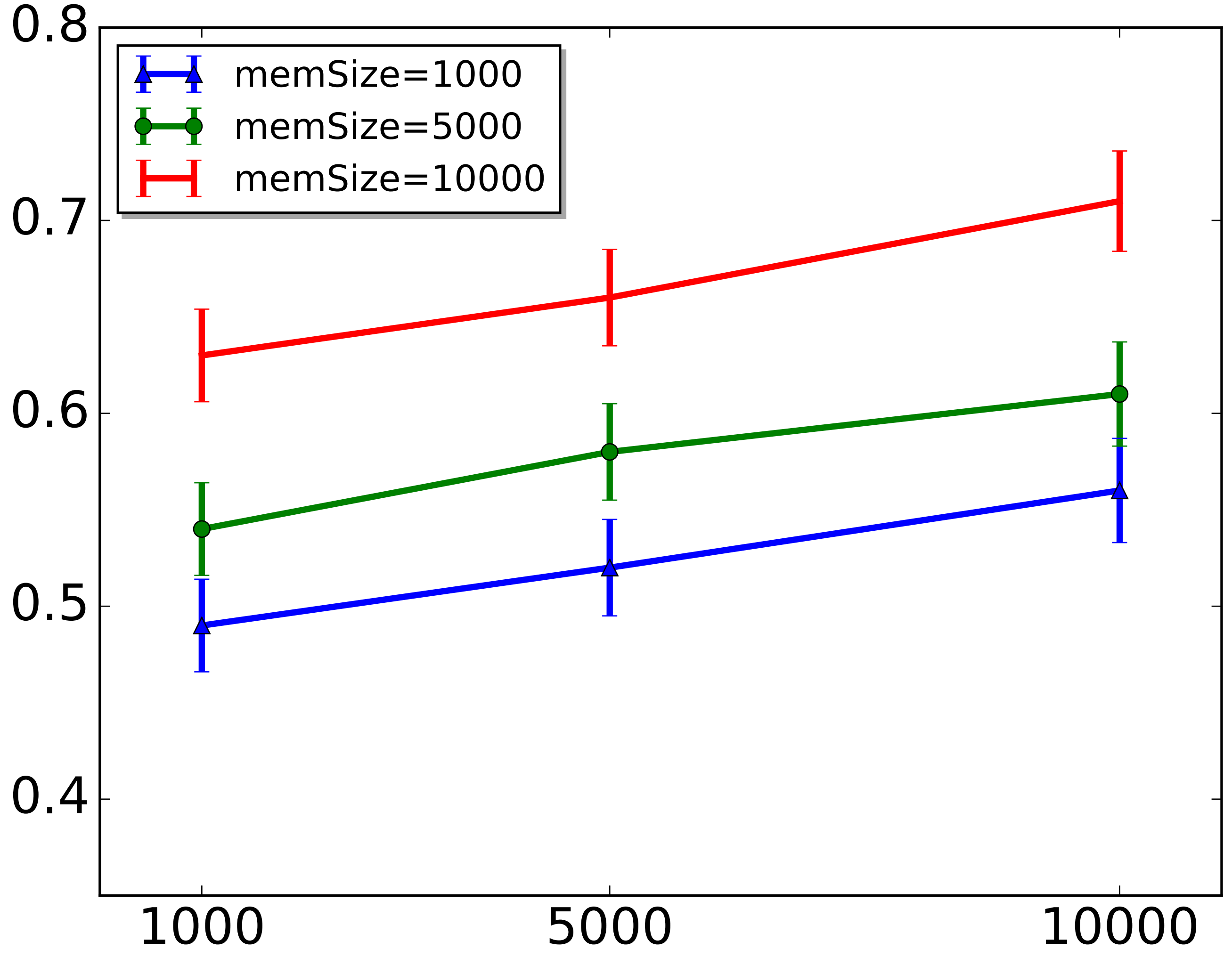


M=50



M=100

Varying Memory Size



M=50, tau=0.1

Future Work

- Combine with Value Network evaluation
- Learn feature representation for similarity
- Investigate online generalization in other methods, such as model-based RL

Thanks! Questions?