Nicol Schraudolph, Peter Dayan and Terrence J. Sejnowski

Temporal Difference Learning of Position Evaluation in the Game of Go

T-61.6020 Reinforcement Learning
- Theory and Applications

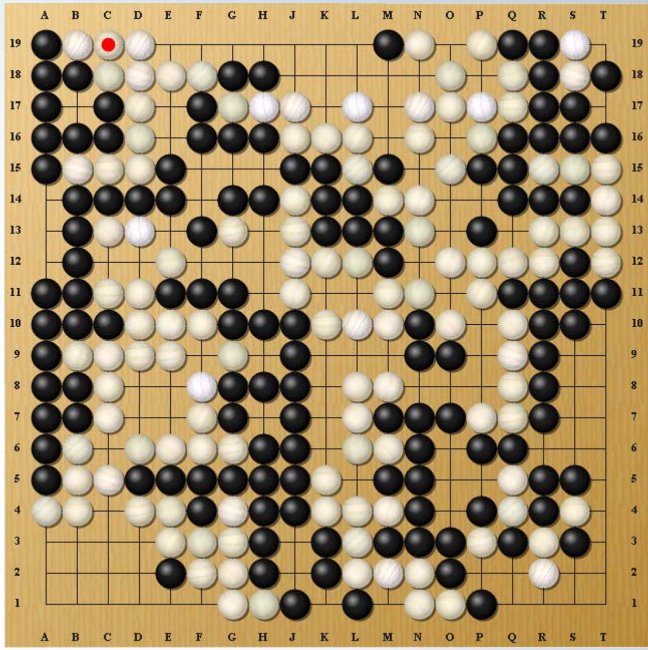
Shanzhen Chen

Outline

1. Background
2. Basic Ideas
3. Network Architecture
4. Training Strategies
5. Results

Background

- The game of Go, oldest board game
  - 19 by 19
  - Evaluation of positions
  - Look ahead
  - Rich information at the end



Background

- “Grant Challenge” for AI
  - – Tree Search not practical (*chess*)
  - – High branching factor ~200
  - – Deep look ahead ~60
- Conventional approach
  - Expert system
  - Need human for compiling domain knowledge
  - Barely above beginner level

Ideas

- Aim: Knowledge free
- Idea: Position Evaluation - Network
- From Tesauro's approach to backgammon
- Based on TD(λ) predictive learning algorithm
- Tesauro's program is trained by self-play (champion level)
- Trained by 3 programs

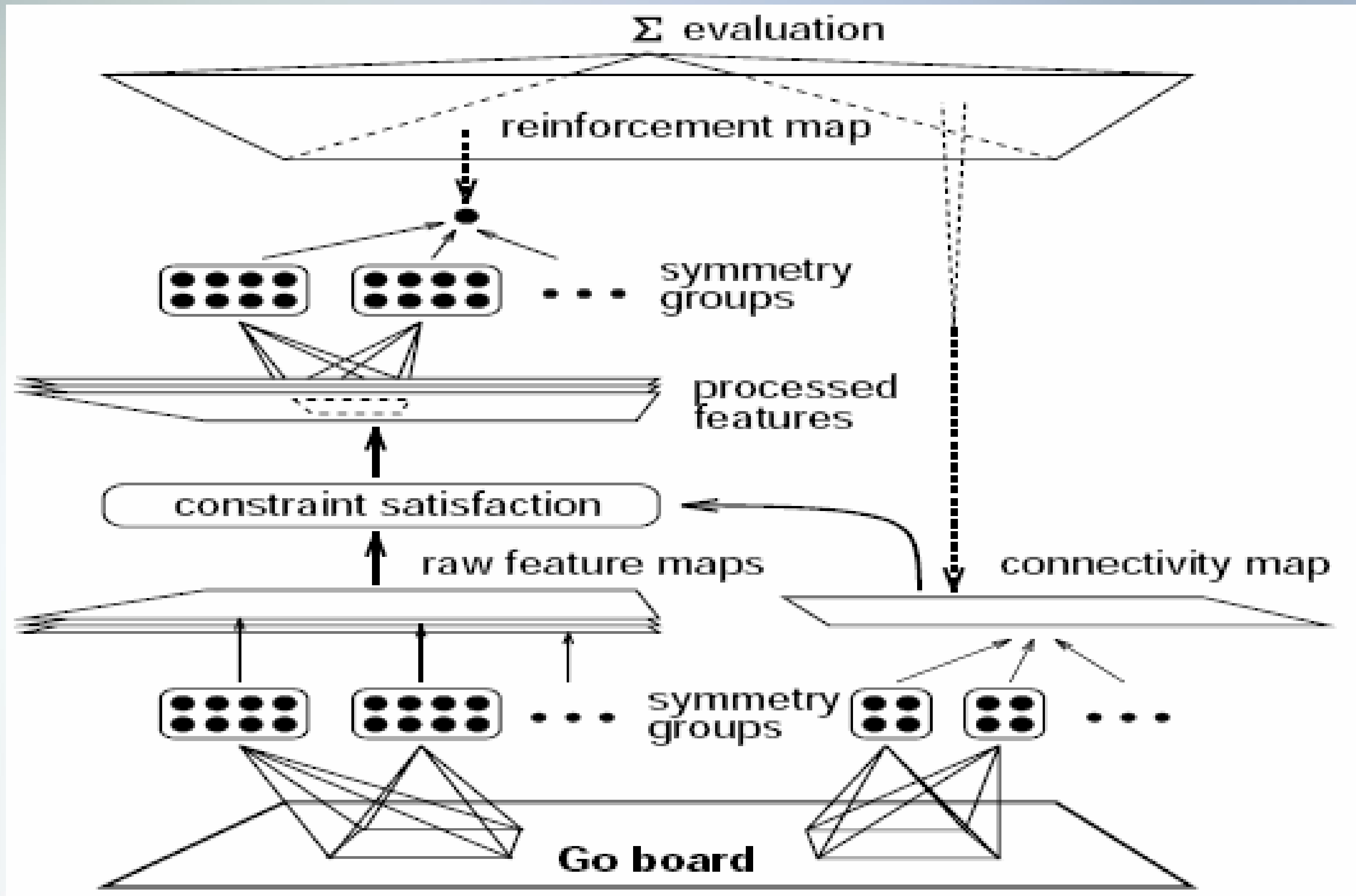
Network Architecture

- Final state richly informative
- Score is the sum of contribution of each point
- Predict the fate of each point
- Conventional program adopt certain input features ~30(*Wally*)
- RL approach take whatever set of features

Network Architecture

- Invariance helps to reduce number of features
  - Color reversal
  - Reflection of the board
  - Rotation
- Translation invariance
  - Convolution with a weight kernel
  - Each node has its own bias weight
  - Convolution kernel is twice the width

Network Architecture



Background

Ideas

Network

Training

Result

Further

Training Strategies

- Large number of games for training

- Criteria of training strategies
  - Computational efficiency
  - Quality of the play
  - Coverage of plausible positions

Training Strategies

- Tesauro trains TD-Gammon by self-playing
- Go is a deterministic game
- Self-training risks staying in suboptimal state
- Theoretically won't happen but it is a concern in practice
- Solution: Use Gibbs sampling to bring in randomness

Training Strategies

- Self-training alone not suitable
  - Computationally intensive
  - Sluggish bootstrap out of ignorance
- Use 3 computer opponents for training
  - Random move generator
  - Public-domain program – *Wally*
  - Commercial program – *The Many Faces of Go*
- The 2 programs are also used as measurement of the network

Training Strategies

- Random move generator
  - Low quality but fast
  - Effective to prime the network at the beginning
- Public-domain program – *Wally*
  - Slow and deterministic
  - Modified to include random moves
  - Randomness is reduced as network improves
- Commercial program – *The Many Faces of Go*
  - Use different standard *Go handicaps* to match the strength of the network

Result

- Many networks are trained with different methods
- A 9 by 9 network is trained through 3000 games to beat Many Faces (low level)
- The learned weight kernel offers a suitable biases for full-sized network

Result

- Comparison between self-training and against *Wally*
  - Similar at the beginning
  - The later over-perform the former soon
  - After 2000 games, overfit *Wally* and worsen against *Many Faces*
  - So the training partner is *Many Faces* after the agent reliably beats *Wally* ~1000 games
  - The self-training network edge-passes *Wally* in 3000 games

Discussion

- In general the network is more competent at the opening than further into the game
  - Reinforcement information did propagate back from the final position
  - Hard to capture the multiplicity of mid-game and complex of end-game
- Suggest hybrid approaches could be better

Further Improvements

- Adjust the input representation to a full translation- invariant network
- Train network on records of human players available on Internet

Thank you.

Questions?