# Comp 6321 - Machine Learning

## Using Neural Nets for playing othello

Federico O'Reilly Regueiro

Concordia University

November 30, 2016

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Problem statement

- Zero-sum, perfect-knowledge (no chance involved) competitive-game
  - A sandbox - toy-representation of reality
  - bounded problem space with clear goal and set of rules
  - bounded, but can be huge (ie, GO - $10^{761}$ possible games!) [1]
- Can a machine learn to play
  - One of the oldest questions in AI
  - The trick is in finding ways to narrow the search
  - Has been well answered, requiring less expert knowledge each time
- Without expert knowledge (rules or labels)...
  - ...the feedback becomes very sparse

# Approaches and solutions - common elements

- Classification problem
  - Dual class
    given a game-state, what are the odds of winning
    - Look ahead n moves - (n-ply) - then decide best path given leaf value
  - Multi-class
    given a game-state, what is the best next move
    - Learn a "policy" for action given a state $P(a|s)$

# Approaches and solutions - common elements

- Classification problem
  - Dual class
    given a game-state, what are the odds of winning
    - ★ Look ahead n-moves - (n-ply) - then decide best path given leaf ‘value’
  - Multi-class
    given a game-state, what is the best next move
    - Learn a ‘policy’ for action given a state $P(a|s)$

# Approaches and solutions - common elements

- Classification problem
  - Dual class
    given a game-state, what are the odds of winning
    - Look ahead n-moves - (n-ply) - then decide best path given leaf 'value'
  - Multi-class
    given a game-state, what is the best next move
    - Learn a 'policy' for action given a state $P(a|s)$

# Approaches and solutions - common elements

- Classification problem
  - Dual class
    given a game-state, what are the odds of winning
    - ⋆ Look ahead n-moves - (n-ply) - then decide best path given leaf 'value'
  - Multi-class
    given a game-state, what is the best next move
    - ⋆ Learn a 'policy' for action given a state $P(a|s)$

# Approaches and solutions - common elements

- Classification problem
  - Dual class
    given a game-state, what are the odds of winning
    - Look ahead n-moves - (n-ply) - then decide best path given leaf 'value'
  - Multi-class
    given a game-state, what is the best next move
    - Learn a 'policy' for action given a state $P(a|s)$

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning

    Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning

    Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning

    Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning

    Tesauro's TD-Backgammon - chance element

  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning

    Tesauro's TD-Backgammon - chance element

  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning

    Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - ⋆ Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities
    but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Approaches and solutions to the problem

- Rule-based approach dependent on expert knowledge
  - e.g. Deep Blue
- Supervised learning - collect labeled states and train
  - Also depends on human expert knowledge
  - Labor intensive collection and labeling
- Genetic optimizations - Evolutionary NNs
  - Does not exploit NNs learning capabilities but won't get stuck on local minima...
  - Slow to converge
  - Capable of finding innovative strategies [4] [2]
- Reinforcement learning
  - TD-learning
    - Tesauro's TD-Backgammon - chance element
  - Like having sparse and time-delayed labels
  - Credit assignment problem
  - explore-exploit dilemma

# Current state-of-the-art

- Alpha-go
  - Two policy convolutional networks - 1 large, 1 small - prune search tree $TD(\lambda)$
  - One Fully connected - predict win value
- DeepMind Atari deep reinforcement learning
  - Deep neural nets meet $TD(\lambda)$

# Current state-of-the-art

- Alpha-go
  - Two policy convolutional networks - 1 large, 1 small - prune search tree $TD(\lambda)$
  - One Fully connected - predict win value
- DeepMind Atari deep reinforcement learning
  - Deep neural nets meet $TD(\lambda)$

# Current state-of-the-art

- Alpha-go
  - Two policy convolutional networks - 1 large, 1 small - prune search tree $TD(\lambda)$
  - One Fully connected - predict win value
- DeepMind Atari deep reinforcement learning
  - Deep neural nets meet $TD(\lambda)$

# Current state-of-the-art

- Alpha-go
  - Two policy convolutional networks - 1 large, 1 small - prune search tree $TD(\lambda)$
  - One Fully connected - predict win value
- DeepMind Atari deep reinforcement learning
  - Deep neural nets meet $TD(\lambda)$

# Current state-of-the-art

- Alpha-go
  - Two policy convolutional networks - 1 large, 1 small - prune search tree $TD(\lambda)$
  - One Fully connected - predict win value
- DeepMind Atari deep reinforcement learning
  - Deep neural nets meet $TD(\lambda)$

# Current project state - discarded approaches

- Supervised Learning
  - Acquiring sets is a cumbersome task
    requires an overhead outside of ML - eg Edax
- Rule-based
  - Heuristic - Decision tree - in place but focus is on nets

# Current project state - discarded approaches

- Supervised Learning
  - Acquiring sets is a cumbersome task
    requires an overhead outside of ML - eg Edax
- Rule-based
  - Heuristic - Decision tree - in place but focus is on nets

# Current project state - discarded approaches

- Supervised Learning
  - Acquiring sets is a cumbersome task
    requires an overhead outside of ML - eg Edax
- Rule-based
  - Heuristic - Decision tree - in place but focus is on nets

# Current project state - discarded approaches

- Supervised Learning
  - Acquiring sets is a cumbersome task
    requires an overhead outside of ML - eg Edax
- Rule-based
  - Heuristic - Decision tree - in place but focus is on nets

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \lambda^{t-k}\nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \lambda^{t-k}\nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j)exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j)N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t)\sum_{k=1}^{t} \lambda^{t-k}\nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j)exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j)N_j(0,1)$

# Current project state - narrowed approaches

- TD learning
  - Similar to back propagation but recurses temporally
  - $\Delta_{w_t} = \alpha(P_{t+1} - P_t) \sum_{k=1}^{t} \lambda^{t-k} \nabla_w P_k, \quad 0 \leq \lambda \leq 1$
  - Based on Leouski and Utgoff's paper[3]
  - They use symmetry, rotation and weight sharing - 96 h.u. - turn into conv net
- ENN
  - Based on Chelapilla and Fogel[2]
  - Generation has 15 strategies, change vector $\sigma_i(j)$ for $j^{th}$ weight of $i^{th}$ strategy.
  - $\sigma_i'(j) = \sigma_i(j) exp(\tau N_j(0,1))$
  - $w_i'(j) = w_i(j) + \sigma_i(j) N_j(0,1)$

# References


Christopher Burger.
Google deepmind's alphago: How it works, 2016.

Kumar Chellapilla and David B Fogel.
Evolution, neural networks, games, and intelligence.
*Proceedings of the IEEE*, 87(9):1471–1496, 1999.

Anton V. Leouski and Paul E. Utgoff.
What a neural network can learn about othello, 1996.

David Moriarty and Risto Miikkulainen.
Evolving complex othello strategies using marker-based genetic encoding of neural networks.
Technical report, 1993.