Ph.D. Thesis Proposal

Study of Reinforcement Learning Methods with Generalization Capabilities

Bohdana Ratitch

SOCS, McGill University

Supervisors:
Denis Therien and Doina Precup

November, 2000

Motivation

- **Reinforcement Learning** (RL) **-** a successful framework for learning in dynamic environments, based on the theory of Markov Decision Processes.
- Large problems require the incorporation of **function approximation** (FA) techniques (from supervised learning) into RL methods.

But:

- Limited theoretical results exist for RL with FA, mostly for simple approximators.
- Mixed evidence of success/failure in practice.

Examples of unanswered questions:

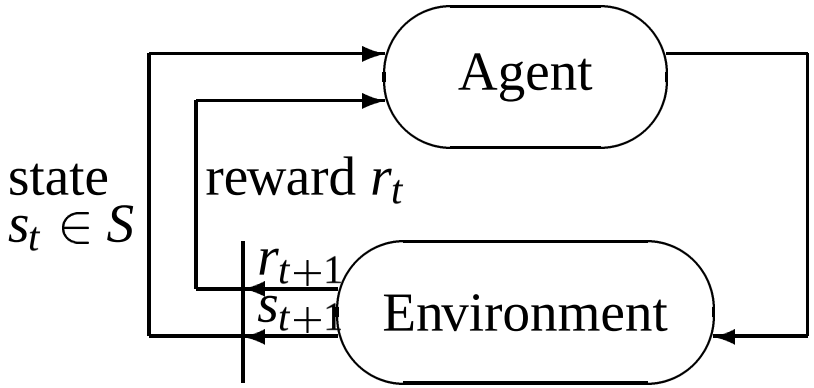
- Which combinations of RL and FA methods are successful/ unstable ?
- How do properties of the problem affect the stability of learning?

My goal: elucidate the effect of domain and FA dynamics on RL algorithms.

Outline

- Reinforcement Learning.
  - RL problem.
  - RL algorithms.
- Function Approximation.
  - Approximation architectures.
  - Training algorithms.
- Reinforcement Learning with Function Approximation.
  - Two approaches to using FA in RL.
  - Available theoretical results and outstanding questions.
- Proposed Research.

RL Problem



S - state space
$A(s_t)$-actions
available at state s_t

- Markov system model:
  - Transition probabilities: $P^a_{s,s'}$
  - Expected value of the immediate reward: $R^a_{s,s'}$
- Policy: $\pi : S \times A \to [0,1]$.
  $\pi(s,a)$ - probability of taking action a in state s.
- Goal: find a policy that optimizes a long-term performance criterion.
- Return $R_t(s)$: cumulative long-term reward from state s.
- Continuing, discounted task: $R_t(s) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$, where $\gamma \in (0,1]$.

Value Functions

Most RL algorithms estimate value functions:

- State-value function for policy π:

$$V^{\pi}(s) = E_{\pi}\{R_t | s_t = s\} = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s\},\ \forall s \in S$$

- Action-value function for policy π:

$$Q^{\pi}(s,a) = E_{\pi}\{\sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a\}\ , \forall s \in S, a \in A(s)$$

- Optimal state-value function: $V^*(s) = \max_{\pi} V^{\pi}(s)\ , \forall s \in S$.
- Optimal policy π^* - a policy with the maximal value function $V^*(s)$.
- In finite MDPs, there is a unique V^* and a deterministic π^* that achieves V^*.
- For small, discrete state spaces, the value functions can be represented by a table, with one value entry for each state (or state action pair).

Dynamic Programming

Compute value functions based on Bellman equations:

- **Bellman equation**: $V^{\pi}(s) = \sum_a \pi(s,a) \sum_{s'} P^a_{s,s'} [R^a_{s,s'} + \gamma V^{\pi}(s')] \ , \forall s \in S$
- **Bellman optimality equation**:
  $V^*(s) = \max_{a \in A(s)} \sum_{s'} P^a_{s,s'} [R^a_{s,s'} + \gamma V^*(s')] \ , \forall s \in S$
- **Policy Iteration** algorithm
  1. **Policy Evaluation**: turn Bellman equation into an update rule.
  2. **Policy Improvement**: find deterministic policy greedy w.r.t. V^{π}
     $\pi'(s) = argmax_{a \in A(s)} \sum_{s'} P^a_{s,s'} [R^a_{s,s'} + \gamma V^{\pi}(s')] \ , \forall s \in S$
- Tabular DP methods converge with probability 1 in the limit.

Monte Carlo Methods

- Estimate value functions as the sample mean of observed returns.
- No knowledge of the system model required!
  Learn directly from the interaction of the agent with its environment.
- Converges to the actual value function in the limit.
- Need for sufficient **exploration** when estimating the action-value function: $\pi(s,a) > 0\ , \forall s, a$.

  E.g. ε-greedy policy: - choose a greedy action with probability $1 - \varepsilon$;

  - choose any other action with probability ε.
- On-policy and Off-policy methods.
  - On-policy methods estimate the value function for the policy used behave.
  - Off-policy methods use one policy to behave but estimate the value function for a different policy (e.g. the optimal policy π^*).

Temporal Difference (TD) Learning

- Updates of the value function are performed after *each observed transition*.
- Learning directly from experience!

Prediction problem: estimate the value of a given policy.

- **One-step TD** : $V(s_t) \leftarrow V(s_t) + \alpha_t \underbrace{[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)]}_{\textbf{Temporal difference } \delta_t}$

  where $\alpha_t \in [0,1]$ **-** learning rate parameter.

- **Eligibility traces -** establish the "influence" of past events in the current updates:

$$e_t(s) = \begin{cases} \gamma\lambda e_{t-1}(s) & \text{if } s \neq s_t \\ \gamma\lambda e_{t-1}(s) + 1 & \text{if } s = s_t \end{cases} \qquad \lambda \in [0,1]$$

- **TD(λ) algorithm**: $V(s) \leftarrow V(s) + \alpha\delta_t e_t(s)\ , \forall s \in S$

TD Learning (2)

Control Problem : find the optimal policy.

There are different on-policy and off-policy algorithms. E.g.

Q-learning (Watkins, 1989):

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha_t [r_{t+1} + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)] \ , \forall t$$

Agent evaluates the optimal policy but follows an arbitrary stochastic policy.

- Almost all TD-style learning algorithms converge in the limit with probability 1 to correct values if value functions are represented as tables, based on general stochastic approximation results.
- Tabular representation is infeasible for large problems:
  - space issue
  - exploration issue

  $\Longrightarrow$ Need for FA to represent value functions.

Function Approximation

Given m training samples $(\vec{x}_i, y_i), y_i = f(\vec{x}_i)$.
Find an approximate mapping $\tilde{f}$, by minimizing some error function, e.g.

$$e = \sum_{i=1}^{m} [y_i - \tilde{f}(\vec{x}_i)]^2$$

Approximation Architectures

- Common approach: $\tilde{f}(\vec{x}) = \sum_{j=0}^{M} a_j B_j(\vec{x}, \vec{\theta}_j)$ - **"dictionary methods"**
- **Linear** methods learn coefficients a_j; **non-linear** methods learn a_j and $\vec{\theta}_j$.
- Basis functions $B_j(\vec{x}, \vec{\theta}_j)$ can be defined **globally** or **locally**.

E.g.:

- Radial Basis Functions $B_j(\vec{x}) = e^{-\|\vec{x}-c_j\|^2/2\sigma_j^2}$:
  - local basis functions, can form linear or non-linear architecture.
- Single layer feed-forward neural nets: non-linear architecture, global basis functions.

Training Methods

- **Gradient descent**: update parameters $\vec{w}$ of the approximation architecture to descent on the error function.
  - **Batch steepest descent**:

$$\vec{w} \leftarrow \vec{w} + \alpha \sum_{i=1}^{m} \nabla \tilde{f}(\vec{x}_i, \vec{w})[y_i - \tilde{f}(\vec{x}_i, \vec{w})]$$

    where $\alpha \in (0,1)$ **-** learning rate.
  - **Incremental gradient descent**: update $\vec{w}$ after each training sample.
- **Memory-based (lazy) learning**:
  - Store all data points without processing
  - To make a prediction, interpolate on the stored data close to the query
  - Good for non-stationary target functions

  E.g. k-nearest neighbor, locally weighted learning.

RL with FA (1)

Value-based approach : FA is used to represent value functions.

- RL methods provide FA with targets or error functions for a subset of states.

  E.g. - Sample returns collected by Monte Carlo method

  - Updates to the value estimates computed by DP and TD methods

  - Temporal difference $\delta_t = r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$

  - Bellman error $e_t = \frac{1}{2} E^2 \{Q(s_t, a_t) - [r_{t+1} + \gamma \max_{a \in A(s_{t+1})} Q(s_{t+1}, a)]\}$

- FA provides RL methods with value estimates generalized to the entire state space.

- Policy is deterministic, greedy w.r.t. approximation of the value function.

Policy-based approach : FA is used to represent the stochastic policy.

- Learn the optimal policy directly.

- FA parameters are updated in the direction of the gradient of the expected return for the current stochastic policy.

RL with FA (2)

FA is much more difficult in the context of RL!

- Samples of the target value functions are not available right away.
- The target function appears non-stationary.

Convergence of approximate RL methods to the optimal solution is not expected.

General convergence questions:

- Does the algorithm converge?
- If so, how close is the resulting solution to the optimal one?
- If not, does it oscillate in some bounded region or does it diverge?

RL with FA (3)

Theoretical results are very limited!

Prediction Problem	Control Problem
TD(λ) converges with probability 1 for *linear architectures*. Monte Carlo and Bellman Error methods converge to a locally optimal solution.	Value iteration and Q-learning converge with *averagers*. Q-learning converges with *soft-state aggregation*. Grow-Support algorithm is safe from divergence. Policy Gradient and VAPS converge to a locally optimal solution.
TD(0) was shown to diverge for a particular non-linear architecture.	Oscillatory behavior was detected for Policy iteration and its variants. Value iteration was shown to diverge for a particular non-linear architecture.

RL with FA (4)

Some of the outstanding questions:

- Which RL/FA combinations are successful/ unstable?
- How common is unstable learning in practice? Does it have oscillatory or divergent nature?
- What is the sample complexity of RL/FA methods?
- How accurate the approximations of the value functions should be to find good policies?
- What are the relative merits of different RL/FA algorithms in practice? No comparative study exists up to date!
- How to choose appropriate algorithms for problems with different properties?
- How to automatically choose successful algorithms' parameters to avoid trial-and-error design process.

Main Thesis Directions

- Define attributes that characterize RL problems and create a benchmark suite of RL problems.
- Conduct an empirical study of RL/FA methods.
- Partially automate the design of RL systems by developing algorithms for automatic parameter setting and for choosing FA, based on domain properties.
- Improve understanding of the specific outstanding questions about properties and relative merits of RL/FA algorithms.

Proposed Research (1)

- Define attributes to characterize the properties of RL tasks, e.g.:
  - Average entropy - a measure of the uncertainty of state transitions.
  - Controllability - a measure of the effect of the agent's actions on state transitions.

  Prior evidence that they influence performance of algorithms.
- Develop a benchmark suite of artificial and real RL problems, using attributes to characterize them. None is available at present!
- Conduct an empirical study of the RL/FA methods, focusing on:
  - Stability of the learning process
  - Quality of the solution
  - Speed of convergence
  - Sample complexity
  - Sensitivity to user-tunable parameters (e.g. α, λ)
  - Performance for problems with different characteristics

Proposed Research (2)

Investigate the following issues:

- Impact of different exploration strategies
- On-policy vs. off-policy algorithms
- Bootstrapping vs. non-bootstrapping methods:
  - Bias-variance trade-off by varying λ in TD(λ)
  - Possibly formal analysis
- Value-based vs. policy-based algorithms
- How accurate should the value function approximation be for different methods and problems

Proposed Research (3)

- Develop boosting algorithms for RL.
  - Boosting - combine several approximators, specialized on a subset of training samples, to obtain a better overall performance.
  - Boosting can be valuable for RL - possibility to combine simple but reliable approximators to boost the overall performance while preserving a well-behaved learning.
- Extend the work of my Master Thesis on the Fuzzy Cellular Automata approximator and apply it to RL.
  - Local method, similar to memory-based learning.
  - Related to other FAs successful for RL.