The Divergence of Reinforcement Learning Algorithms with Value-Iteration and Function Approximation

Divergence with a greedy policy

M. Fairbank\textsuperscript{1} \hspace{1cm} E. Alonso\textsuperscript{1}

\textsuperscript{1}City University London
London

IJCNN 2012
Outline

1. Which algorithms did we make diverge?
2. Why do they diverge?
4. Divergence Results
5. Conclusions
Which RL/ADP algorithms can be forced to diverge with function approximation and a greedy policy?

- HDP (TD(0)): Yes
- DHP (VGL(0)): Yes
- TD(1): Yes (!)
- VGL(1): Yes (!)
- Sarsa(\(\lambda\)): Yes

This will hopefully provide a useful resource in developing convergent RL/ADP algorithms with a greedy policy and function approximation.
Why do they diverge?

HDP is
\[ \Delta \tilde{w} = \alpha \sum_t \left( \frac{\partial \tilde{J}}{\partial \tilde{w}} \right)_t \left( J'_t - \tilde{J}_t \right) \]
with \( J'_t \) defined by
\[ J'_t = U_t + \gamma \tilde{J}_{t+1} \]

DHP is
\[ \Delta \tilde{w} = \alpha \sum_t \left( \frac{\partial \tilde{G}}{\partial \tilde{w}} \right)_t \Omega_t \left( G'_t - \tilde{G}_t \right) \]
with \( G'_t \) defined by
\[ G'_t = \left( \frac{D U}{D \vec{x}} \right)_t + \gamma \left( \frac{D f}{D \vec{x}} \right)_t \tilde{G}_{t+1} \]

Disclaimer: These are the forms of the algorithms that we make diverge.

Are either of these gradient descent?
- No. For example,
  \[ -\frac{\partial}{\partial \tilde{w}} \left( \frac{1}{2} \sum_t \left( U_t + \gamma \tilde{J}_{t+1} - \tilde{J}_t \right)^2 \right) \neq \text{HDP} \]
- And similarly for DHP.

- Doing full gradient descent on an error function of this kind is known as the Galerkin-method by Werbos\(^1\) or the Residual Gradients method by Baird\(^2\), but we do not consider those methods here.

---


\(^2\)“Residual Algorithms: Reinforcement Learning with Function Approximation”, Leemon Baird, ICML 1995
Why do they diverge?

What about the $\lambda = 1$ Algorithms, TD(1) and VGL(1)?

- When the action network $A(\vec{x}, \vec{z})$ is fixed, we have
  \[ \frac{\partial}{\partial \vec{w}} \left( \frac{1}{2} \sum_t \left( J'_t - \tilde{J}_t \right)^2 \right) = \text{TD}(1). \]

- So TD(1) and VGL(1) are true gradient descent when a fixed policy is used - and so we can assure convergence with sufficiently small learning rate and smoothness on $E$.

- But when a greedy policy is used $J'$ is still a moving target.

- With a greedy policy, neither TD(1) nor VGL(1) is gradient descent, so there is no reason to assume they will converge.

- Whether divergence could happen replicably and analytically was previously an open question - our results here answer that question.
How did we do it? Brief Outline of method:

We analysed the $\text{VGL}(\lambda)$ weight update for a simple artificial problem.

- We created an example problem domain: A two-step problem, with state-vector $\vec{x} \in \mathbb{R}$, and with tunable cost constants. A linear/quadratic problem was sufficient for divergence.
- We created a quadratic function approximator for the critic $\tilde{J}(\vec{x}, \vec{w})$, with $\vec{w} = (w_1, w_2)$.
- We derived the greedy policy as a function of $(\vec{x}, t, \vec{w})$. Since the model and cost functions and critic were all quadratic functions, the greedy policy was possible to solve analytically.
Using the greedy policy, we expanded the trajectory states \((\vec{x}_1, \vec{x}_2)\) so that each was calculated as a pure function of \(\vec{w}\).

We calculated the critic gradients \(\tilde{G}_1\) and \(\tilde{G}_2\) in terms of \(\vec{w}\).

We calculated the target critic gradients \(G'_1\) and \(G'_2\) in terms of \(\vec{w}\).
How did we do it? Brief Outline of method:

- Having $\tilde{G}_1, \tilde{G}_2, G'_1$ and $G'_2$ as functions of $\bar{w}$ allowed us to write down the VGL($\lambda$) weight update as a function of $\bar{w}$. This gave us the whole VGL weight update as:

$$\Delta \bar{w} = \alpha F \left( -\frac{k(k\lambda+(c_2)^2+k(1-\lambda)c_2)}{(c_1+k)(c_2+k)^2} - \frac{k}{(c_1+k)} \right) F^T \bar{w}$$

or $\Delta \bar{w} = \alpha FA F^T \bar{w}$ for brevity.

- Here $c_1, c_2$ and $k$ were constants in the problem’s cost function. $\lambda$ is the VGL($\lambda$) parameter. $F$ is an arbitrary $2 \times 2$ constant matrix. We can vary these to try to make the system diverge.
How did we do it? Brief Outline of method:

$$\Delta \vec{w} = \alpha FAF^T \vec{w}$$

- For a sufficiently small learning rate $\alpha$, $\vec{w}$ is evolving according to a linear continuous time dynamic system.
- Divergence will happen if the real part of any eigenvalue of $FAF^T$ is positive. That’s what we looked for, to achieve divergence!
- The logic here is that if it is proven to diverge for a continuous time system, i.e. in the limit of an infinitesimal learning rate, then it would also diverge for any small finite learning rate too.
- Constants that achieved this are given in the paper.
The divergence results obtained for VGL(0) and VGL(1) are below:

DHP is the same thing as VGL(0), so this graph shows divergence of DHP with a greedy policy.

We used a different choice of constants to make VGL(1) diverge than those we used to make VGL(0) diverge.
Reminder of amazing fact:

- When the special $\Omega_t$ matrix is chosen as described in the paper (and the previous presentation), VGL(1) becomes equivalent to gradient descent on the total cost function $J$, and so should not diverge with small $\alpha$.
- We confirm this in the graph below:
We performed these divergences using exactly the same constants that we found for VGL(0) and VGL(1).

We then verified the divergence for TD(0) and TD(1) empirically.

Results are shown below:
To make the divergences easily replicable, source code is made available\(^3\).

When performing the TD\((\lambda)\) experiments we had to supplement the greedy policy with some small stochastic exploration, otherwise TD methods will converge to the wrong policy (as we showed in the previous presentation).

Why did the constants that made VGL\((\lambda)\) diverge also make TD\((\lambda)\) diverge? Because TD with stochastic exploration can be understood to be an approximation to a stochastic version of VGL. So we would expect a divergence example for VGL to cause divergence for TD\((\lambda)\) too.

\(^3\)In ancilliary files of this paper at eprint arXiv:1107.4606
Conclusions

- We achieved replicable and analytically derived divergence for DHP and VGL(1).
- We achieved replicable and empirically derived divergence for TD(0) and TD(1).
- All divergences were with a greedy policy and linear/quadratic problem and critic.
- The greedy policy was necessary to make these divergences happen.
- Introducing the special $\Omega_t$ matrix was sufficient to stop VGL(1) diverging, but not VGL(0) (DHP).
- Hopefully the analysis here can be useful in generating stronger convergence proofs and algorithms in the future.
Thank you!

Please email me for a copy of the slides or the paper to review at a slower pace.

michael.fairbank@virgin.net

Please chat with me after the presentation.