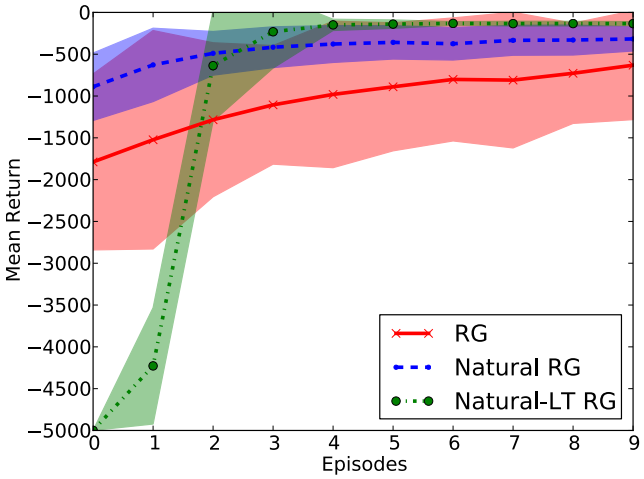

Figure 2: Mountain Car (Residual Gradient)

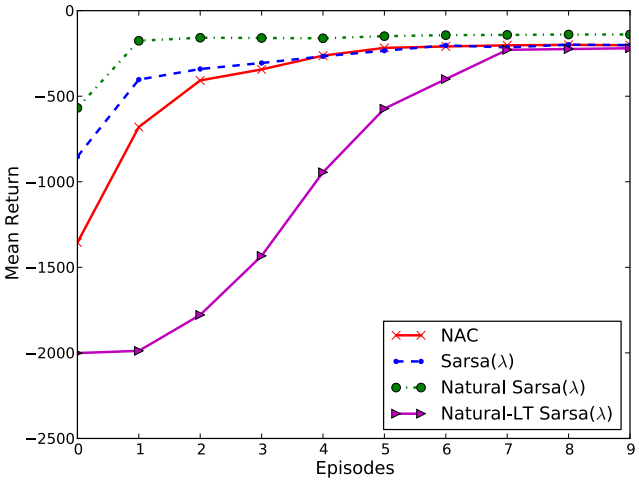

Figure 3: Mountain Car (Sarsa(λ))

over 10 trials, again with standard deviations shown by the shaded regions. For the Sarsa(λ) experiments we include results for Natural Actor-Critic (Peters and Schaal 2008), to provide a comparison with another approach to applying natural gradients to reinforcement learning. However, for these experiments we do not include the standard deviations because they make the figures much harder to read. We used a soft-max policy with Natural Actor-Critic (NAC).

Mountain Car

Mountain car is a simple simulation of an underpowered car stuck in a valley; full details of the domain can be found in the work of Sutton and Barto (1998). Figures 2 and 3 give the results for each algorithm on mountain car. The linear time natural residual gradient and Sarsa(λ) algorithms take longer to learn good policies than the quadratic time natural algorithms. One reason for the slower initial learning of the linear algorithms is that they must first build up an estimate of the w vector before updates to the value function

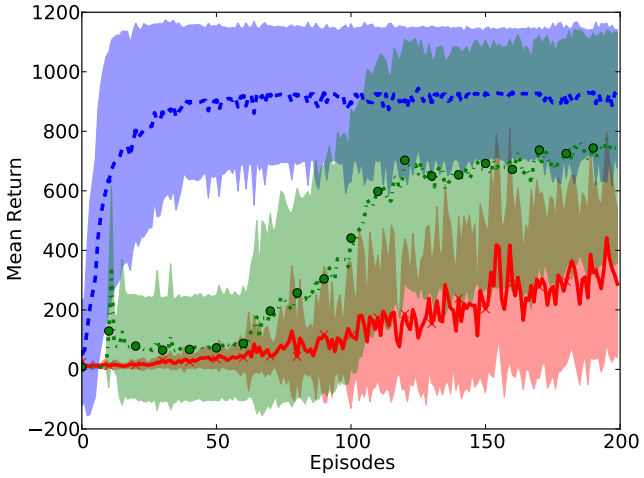

Figure 4: Cart Pole (Residual Gradient). Same legend as Figure 2

parameters become meaningful. Out of all the algorithms we found that the quadratic time Natural Sarsa(λ) algorithm performed the best in mountain car, reaching the best policy after just two episodes.

Cart Pole Balancing

Cart pole balancing simulates a cart on a short one dimensional track with a pole attached with a rotational hinge, and is also referred to as the inverted pendulum problem. There are many varieties of the cart pole balancing domain, and we refer the reader to Barto, Sutton, and Anderson (1983) for complete details. Figures 4 and 5 give the results for each algorithm on cart pole balancing. In the cart pole balancing domain the two quadratic algorithms, Natural Sarsa(λ) and Natural RG perform the best. Again, the linear algorithm, takes a slower start as it builds up an estimate of w, but converges well above the non-natural algorithms and very close to the quadratic ones. Natural Sarsa(λ) reaches a near optimal policy within the first couple of episodes, and compares favorably with the heavily optimized Sarsa(λ), which does not even reach the same level of performance after 100 episodes.

Visual Tic-Tac-Toe

Visual Tic-Tac-Toe is a novel challenging decision problem in which the agent plays Tic-tac-toe (Noughts and crosses) against an opponent that makes random legal moves. The game board is a 3×3 grid of handwritten letters (X, O, and B for blank) from the UCI Letter Recognition Data Set (Slate 1991), examples of which are shown in Figure 8. At every step of the episode, each letter of the game board is drawn randomly with replacement from the set of available handwritten letters (787 X's, 753 O's, and 766 B's). Thus, it is easily possible for the agent to never see the same handwritten "X", "O", or "B" letter in a given episode. The agent's state features are the 16 integer valued attributes for each of the letters on the board. Details of the data set and the attributes can be found in the UCI repository.

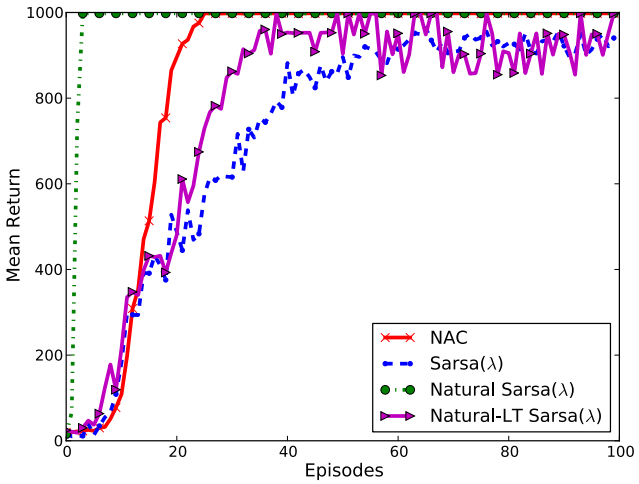

Figure 5: Cart Pole (Sarsa(λ))

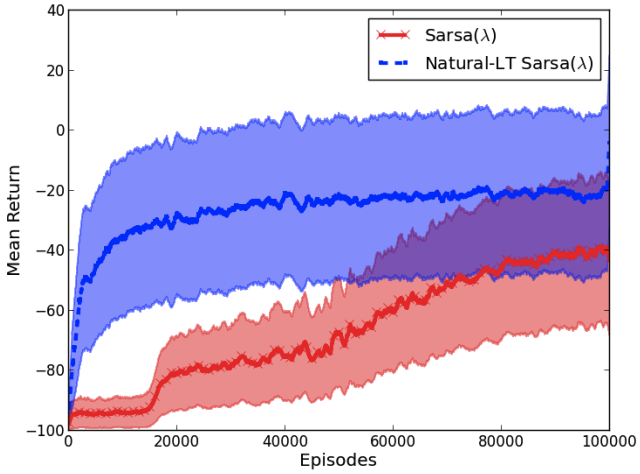

Figure 6: Visual Tic-Tac-Toe Experiments

There are nine possible actions available to the agent, but attempting to play on a non-blank square is considered an illegal move and results in the agent losing its turn. This is particularly challenging because blank squares are marked by a "B", making recognizing legal moves challenging in and of itself. The opponent only plays legal moves, but chooses randomly among them. The reward for winning is 100, -100 for losing, and 0 otherwise.

Figure 6 gives the results comparing Natural-LT Sarsa and Sarsa(λ) on the visual Tic-tac-toe domain using the artificial neural network described previously. These results show linear natural Sarsa(λ) in a setting where it is able to account for the shape of a more complex value function parameterization, and thus confer greater improvement in convergence speed over non-natural algorithms. We do not compare quadratic time algorithms due to computational limits.

Figure 8: Visual Tic-Tac-Toe example letters

Acrobot

Acrobot is another commonly studied reinforcement learning task in which the agent controls a two-link under actuated robot by applying torque to the lower joint with the goal of raising the tip of the lower link above a certain point. See Sutton and Barto (1998) for a full specification of the domain and its equations of motion. To evaluate the off-policy Natural TDC algorithm we first generated a fixed policy by online training of a hand tuned Sarsa(λ) agent for 200 episodes. We then trained TDC and Natural TDC for 10000 episodes on Acrobot following the previously learned fixed policy. We evaluated an algorithm's learned value function every 100 episodes by sampling states and actions randomly and computing the true expected undiscounted return using Monte Carlo rollouts following the fixed policy. Figure 7 shows the MSE between the learned values and the true expected return.

Natural TDC clearly out performs TDC, and in this experiment converged to much lower MSE. Additionally, we found TDC to be sensitive to the step-sizes used, and saw that Natural TDC was much less sensitive to these parameters. These results show that the benefits of natural temporal difference learning, already observed in the context of control learning, extend to TD-learning for value function estimation as well.

Discussion and Conclusion

We have presented the natural residual gradient algorithm and proved that it is covariant. We suggested that the temporal difference learning metric tensor, derived for natural residual gradient, can be used to create other natural tempo-

ral difference learning algorithms like natural Sarsa(λ) and natural TDC. The resulting algorithms begin with the identity matrix as their estimate of the (inverse) metric tensor. This means that before an estimate of the (inverse) metric tensor has been formed, they still provide meaningful updates—they follow estimates of the non-natural gradient.

We showed how the concept of compatible function approximation can be leveraged to create linear-time natural residual gradient and natural Sarsa(λ) algorithms. However, unlike the quadratic-time variants, these linear-time variants do not provide meaningful updates until the natural gradient has been estimated. As a result, learning is initially slower using the linear-time algorithms.

In our empirical studies, the natural variants of all three algorithms outperformed their non-natural counterparts on all three domains. Additionally, the quadratic-time variants learn faster initially, as expected. Lastly, we showed empirically that the benefits of natural gradients are amplified when using non-linear function approximation.

Appendix A

Proof of Covariant Theorem: The following theorem and its proof closely follow and extend the foundations laid by Bagnell and Schneider (2003) and later clarified by Peters and Schaal (2008) when proving that the natural policy gradient is covariant.

No algorithm can be covariant for all parameterizations. Thus, constraints on the parameterized functions that we consider are required.

Property 1. *Functions $g : \Phi \times X \to \mathbb{R}$, and $h : \Theta \times X \to \mathbb{R}$ are two instantaneous loss functions parameterized by $\phi \in \Phi$ and $\theta \in \Theta$ respectively. These correspond to the loss functions $\hat{g}(\phi) = \mathbb{E}_{x \in X}[g(\phi, x)]$ and $\hat{h}(\theta) = \mathbb{E}_{x \in X}[h(\theta, x)]$. For brevity, hereafter, we suppress the x inputs to g and h. There exists a differentiable function, $\Psi : \Phi \to \Theta$, such that for some $\phi \in \Phi$, we have $g(\phi) = h(\Psi(\phi))$ and the Jacobian of Ψ is full rank.*

Definition 1. *Algorithm $\mathcal{A}$ is covariant if, for all g, h, Ψ, and ϕ satisfying Property 1,*

$$g(\phi + \Delta\phi) = h(\Psi(\phi) + \Delta\theta), \tag{13}$$

where $\phi + \Delta\phi$ and $\Psi(\phi) + \Delta\theta$ are the parameters after an update of algorithm $\mathcal{A}$.

Lemma 1. *An algorithm $\mathcal{A}$ is covariant for sufficiently small step-sizes if*

$$\Delta\theta = \frac{\partial \Psi(\phi)}{\partial \phi} \Delta\phi. \tag{14}$$

Proof. Let $J_{\Psi(\phi)}$ be the Jacobian of $\Psi(\phi)$, i.e., $J_{\Psi(\phi)} = \frac{\partial \Psi(\phi)}{\partial \phi}$. As such, it maps tangent vectors of h to tangent vectors of g, such that

$$\frac{\partial g(\phi)}{\partial \phi} = J_{\Psi(\phi)} \frac{\partial h(\Psi(\phi))}{\partial \Psi(\phi)}, \tag{15}$$

when $g(\phi) = h(\Psi(\phi))$, as $J_{\Psi(\phi)}$ is a tangent map (Lee 2003, p. 63).

Taking the first order Taylor expansion of both sides of (13), we obtain

$$\begin{array}{lll} h(\Psi(\phi)) & + \frac{\partial h(\Psi(\phi))^\intercal}{\partial \Psi(\phi)} \Delta\theta & \\ & + O(\|\Delta\theta\|^2) & = \end{array} \begin{array}{ll} g(\phi) & + \frac{\partial g(\phi)^\intercal}{\partial \phi} \Delta\phi \\ & + O(\|\Delta\phi\|^2). \end{array}$$

For small step-sizes, $\alpha > 0$, the squared norms become negligible, and because $g(\phi) = h(\Psi(\phi))$, this simplifies to

$$\begin{aligned} \frac{\partial h(\Psi(\phi))^\intercal}{\partial \Psi(\phi)} \Delta\theta &= \frac{\partial g(\phi)^\intercal}{\partial \phi} \Delta\phi, \\ &= \left(J_{\Psi(\phi)} \frac{\partial h(\Psi(\phi))}{\partial \Psi(\phi)} \right)^\intercal \Delta\phi, \\ &= \frac{\partial h(\Psi(\phi))^\intercal}{\partial \Psi(\phi)} J_{\Psi(\phi)}^\intercal \Delta\phi. \end{aligned} \tag{16}$$

Notice that (16) is satisfied by $\Delta\theta = J_{\Psi(\phi)}^\intercal \Delta\phi$, and thus if this equality holds then $\mathcal{A}$ is covariant. □

Theorem 1. *The natural gradient update $\Delta\theta = -G_\theta^{-1} \nabla h(\theta)$ is covariant when the metric tensor G_θ is given by*

$$G_\theta = \mathop{\mathbb{E}}_{x \in X} \left[\frac{\partial h(\theta)}{\partial \theta} \frac{\partial h(\theta)^\intercal}{\partial \theta} \right]. \tag{17}$$

Proof. First, notice that the metric tensor G_ϕ is equivalent to G_θ with $J_{\Psi(\phi)}$ twice as a factor,

$$\begin{aligned} G_\phi &= \mathop{\mathbb{E}}_{x \in X} \left[\frac{\partial g(\phi)}{\partial \phi} \frac{\partial g(\phi)^\intercal}{\partial \phi} \right], \\ &= \mathop{\mathbb{E}}_{x \in X} \left[(J_{\Psi(\phi)} \frac{\partial h(\Psi(\phi))}{\partial \theta})(J_{\Psi(\phi)} \frac{\partial h(\Psi(\phi))}{\partial \theta})^\intercal \right], \\ &= \mathop{\mathbb{E}}_{x \in X} \left[J_{\Psi(\phi)} \frac{\partial h(\Psi(\phi))}{\partial \theta} \frac{\partial h(\Psi(\phi))}{\partial \theta}^\intercal J_{\Psi(\phi)}^\intercal \right], \\ &= J_{\Psi(\phi)} \mathop{\mathbb{E}}_{x \in X} \left[\frac{\partial h(\Psi(\phi))}{\partial \theta} \frac{\partial h(\Psi(\phi))}{\partial \theta}^\intercal \right] J_{\Psi(\phi)}^\intercal, \\ &= J_{\Psi(\phi)} G_\theta J_{\Psi(\phi)}^\intercal. \end{aligned} \tag{18}$$

We show that the right hand side of (14) is equal to the left, which, by Lemma 1, implies that the natural gradient update is covariant.

$$\begin{aligned} J_{\Psi(\phi)}^\intercal \Delta\phi &= J_{\Psi(\phi)}^\intercal \alpha G_\phi^{-1} \nabla g(\phi), \\ &= J_{\Psi(\phi)}^\intercal \alpha G_\phi^{+} \nabla g(\phi), \\ &= \alpha J_{\Psi(\phi)}^\intercal \left(J_{\Psi(\phi)} G_\theta J_{\Psi(\phi)}^\intercal \right)^+ J_{\Psi(\phi)} \nabla h(\Psi(\phi)), \\ &= \alpha J_{\Psi(\phi)}^\intercal (J_{\Psi(\phi)}^\intercal)^+ G_\theta^+ J_{\Psi(\phi)}^+ J_{\Psi(\phi)} \nabla h(\Psi(\phi)). \end{aligned} \tag{19}$$

Since $J_{\Psi(\phi)}$ is full rank, $J_{\Psi(\phi)}^+$ is a left inverse, and thus

$$\begin{aligned} J_{\Psi(\phi)}^\intercal \Delta\phi &= \alpha G_\theta^{-1} \nabla h(\Psi(\phi)), \\ &= \Delta\theta. \end{aligned}$$ □

Notice that, unlike the proof that the natural actor-critic using LSTD is covariant (Peters and Schaal 2008), our proof does not assume that $J_{\Psi(\phi)}$ is invertible. Our proof is therefore more general, since it allows $|\phi| \geq |\theta|$.

References

Amari, S., and Douglas, S. 1998. Why natural gradient? In *Proceedings of the 1998 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '98)*, volume 2, 1213–1216.

Amari, S. 1998. Natural gradient works efficiently in learning. *Neural Computation* 10:251–276.

Bagnell, J. A., and Schneider, J. 2003. Covariant policy search. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1019–1024.

Baird, L. 1995. Residual algorithms: reinforcement learning with function approximation. In *Proceedings of the Twelfth International Conference on Machine Learning*.

Barto, A. G.; Sutton, R. S.; and Anderson, C. W. 1983. Neuronlike adaptive elements that can solve difficult learning control problems. *IEEE Transactions on Systems, Man, and Cybernetics* 13(5):834–846.

Bergstra, J., and Bengio, Y. 2012. Random search for hyper-parameter optimization. In *Journal of Machine Learning Research*.

Bhatnagar, S.; Sutton, R. S.; Ghavamzadeh, M.; and Lee, M. 2009. Natural actor-critic algorithms. *Automatica* 45(11):2471–2482.

Degris, T.; Pilarski, P. M.; and Sutton, R. S. 2012. Model-free reinforcement learning with continuous action in practice. In *Proceedings of the 2012 American Control Conference*.

Kakade, S. 2002. A natural policy gradient. In *Advances in Neural Information Processing Systems*, volume 14, 1531–1538.

Konidaris, G. D.; Kuindersma, S. R.; Grupen, R. A.; and Barto, A. G. 2012. Robot learning from demonstration by constructing skill trees. volume 31, 360–375.

Kushner, H. J., and Yin, G. 2003. *Stochastic Approximation and Recursive Algorithms and Applications*. Springer.

Lee, J. M. 2003. *Introduction to Smooth Manifolds*. Springer.

Morimura, T.; Uchibe, E.; and Doya, K. 2005. Utilizing the natural gradient in temporal difference reinforcement learning with eligibility traces. In *International Symposium on Information Geometry and its Application*, 256–263.

Peters, J., and Schaal, S. 2008. Natural actor-critic. *Neurocomputing* 71:1180–1190.

Slate, D. 1991. UCI machine learning repository.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Sutton, R. S.; McAllester, D.; Singh, S.; and Mansour, Y. 2000. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems 12*, 1057–1063.

Sutton, R. S.; Maei, H. R.; Precup, D.; Bhatnagar, S.; Silver, D.; Szepesvári, C.; and Wiewiora, E. 2009. Fast gradient-descent methods for temporal-difference learning with linear function approximation. In *Proceedings of the 26th Annual International Conference on Machine Learning*, 993–1000. ACM.