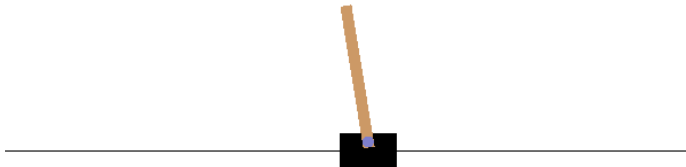


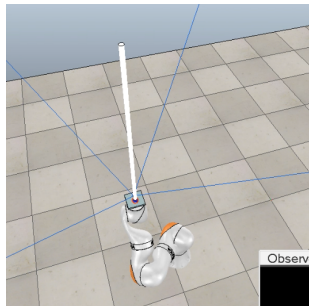
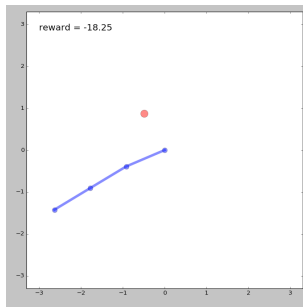
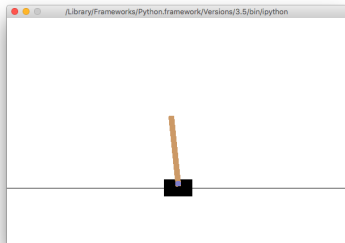
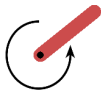
Model-Free Reinforcement Learning



Mathias Winther Madsen
mathias@micropsi-industries.com

March 6, 2017

Policy Gradient Methods



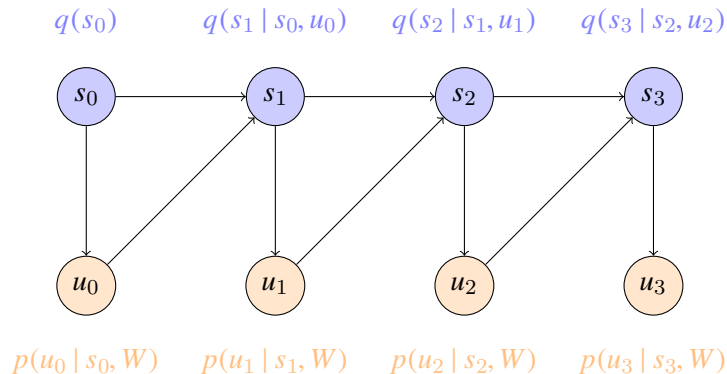
Policy Gradient Methods

The REINFORCE algorithm: Williams, “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning” (*Machine Learning*, 1992)

Task	Random	REINFORCE	TNPG	RWR	REPS	TRPO	CEM	CMA-ES	DDPG
Cart-Pole Balancing	77.1 ± 0.0	4693.7 ± 14.0	3986.4 ± 748.9	4861.5 ± 12.3	565.6 ± 137.6	4869.8 ± 37.6	4815.4 ± 4.8	2440.4 ± 568.3	4634.4 ± 87.8
Inverted Pendulum*	-153.4 ± 0.2	13.4 ± 18.0	209.7 ± 55.5	84.7 ± 13.8	-113.3 ± 4.6	247.2 ± 76.1	38.2 ± 25.7	-40.1 ± 5.7	40.0 ± 244.6
Mountain Car	-415.4 ± 0.0	-67.1 ± 1.0	-66.5 ± 4.5	-79.4 ± 1.1	-275.6 ± 166.3	-61.7 ± 0.9	-66.0 ± 2.4	-85.0 ± 7.7	-288.4 ± 170.3
Acrobot	-1904.5 ± 1.0	-508.1 ± 91.0	-395.8 ± 121.2	-352.7 ± 35.9	-1001.5 ± 10.8	-326.0 ± 24.4	-436.8 ± 14.7	-785.6 ± 13.1	-233.6 ± 5.8
Double Inverted Pendulum*	149.7 ± 0.1	4116.5 ± 65.2	4455.4 ± 37.6	3614.8 ± 368.1	446.7 ± 114.8	4412.4 ± 50.4	2566.2 ± 178.9	1576.1 ± 51.3	2863.4 ± 154.0
Swimmer*	-1.7 ± 0.1	92.3 ± 0.1	96.0 ± 0.2	60.7 ± 5.5	3.8 ± 3.3	96.0 ± 0.2	68.8 ± 2.4	64.9 ± 1.4	85.8 ± 1.8
Hopper	8.4 ± 0.0	714.0 ± 29.3	1155.1 ± 57.9	553.2 ± 71.0	86.7 ± 17.6	1183.3 ± 150.0	63.1 ± 7.8	20.3 ± 14.3	267.1 ± 43.5
2D Walker	-1.7 ± 0.0	506.5 ± 78.8	1382.6 ± 108.2	136.0 ± 15.9	-37.0 ± 38.1	1353.8 ± 85.0	84.5 ± 19.2	77.1 ± 24.3	318.4 ± 181.6
Half-Cheetah	-90.8 ± 0.3	1183.1 ± 69.2	1729.5 ± 184.6	376.1 ± 28.2	34.5 ± 38.0	1914.0 ± 120.1	330.4 ± 274.8	441.3 ± 107.6	2148.6 ± 702.7
Ant*	13.4 ± 0.7	548.3 ± 55.5	706.0 ± 127.7	37.6 ± 3.1	39.0 ± 9.8	730.2 ± 61.3	42.2 ± 5.9	17.8 ± 15.5	326.2 ± 20.8
Simple Humanoid	41.5 ± 0.2	128.1 ± 34.0	255.0 ± 24.5	93.3 ± 17.4	28.3 ± 4.7	269.7 ± 40.3	60.6 ± 12.9	28.7 ± 3.9	99.4 ± 28.1
Full Humanoid	13.2 ± 0.1	262.2 ± 10.5	288.4 ± 25.2	46.7 ± 5.6	41.7 ± 6.1	287.0 ± 23.4	36.9 ± 2.9	N/A ± N/A	119.0 ± 31.2
Cart-Pole Balancing (LS)*	77.1 ± 0.0	420.9 ± 265.5	945.1 ± 27.8	68.9 ± 1.5	898.1 ± 22.1	960.2 ± 46.0	227.0 ± 223.0	68.0 ± 1.6	
Inverted Pendulum (LS)	-122.1 ± 0.1	-13.4 ± 3.2	0.7 ± 6.1	-107.4 ± 0.2	-87.2 ± 8.0	4.5 ± 4.1	-81.2 ± 33.2	-62.4 ± 3.4	
Mountain Car (LS)	-83.0 ± 0.0	-81.2 ± 0.6	-65.7 ± 9.0	-81.7 ± 0.1	-82.6 ± 0.4	-64.2 ± 9.5	-68.9 ± 1.3	-73.2 ± 0.6	
Acrobot (LS)*	-393.2 ± 0.0	-128.9 ± 11.6	-84.6 ± 2.9	-235.9 ± 5.3	-379.5 ± 1.4	-83.3 ± 9.9	-149.5 ± 15.3	-159.9 ± 7.5	
Cart-Pole Balancing (NO)*	101.4 ± 0.1	616.0 ± 210.8	916.3 ± 23.0	93.8 ± 1.2	99.6 ± 7.2	606.2 ± 122.2	181.4 ± 32.1	104.4 ± 16.0	
Inverted Pendulum (NO)	-122.2 ± 0.1	6.5 ± 1.1	11.5 ± 0.5	-110.0 ± 1.4	-119.3 ± 4.2	10.4 ± 2.2	-55.6 ± 16.7	-80.3 ± 2.8	
Mountain Car (NO)	-83.0 ± 0.0	-74.7 ± 7.8	-64.5 ± 8.6	-81.7 ± 0.1	-82.9 ± 0.1	-60.2 ± 2.0	-67.4 ± 1.4	-73.5 ± 0.5	
Acrobot (NO)*	-393.5 ± 0.0	-186.7 ± 31.3	-164.5 ± 13.4	-233.1 ± 0.4	-258.5 ± 14.0	-149.6 ± 8.6	-213.4 ± 6.3	-236.6 ± 6.2	
Cart-Pole Balancing (SI)*	76.3 ± 0.1	431.7 ± 274.1	980.5 ± 7.3	69.0 ± 2.8	702.4 ± 196.4	980.3 ± 5.1	746.6 ± 93.2	71.6 ± 2.9	
Inverted Pendulum (SI)	-121.8 ± 0.2	-5.3 ± 5.6	14.8 ± 1.7	-108.7 ± 4.7	-92.8 ± 23.9	14.1 ± 0.9	-51.8 ± 10.6	-63.1 ± 4.8	
Mountain Car (SI)	-82.7 ± 0.0	-63.9 ± 0.2	-61.8 ± 0.4	-81.4 ± 0.1	-80.7 ± 2.3	-61.6 ± 0.4	-63.9 ± 1.0	-66.9 ± 0.6	
Acrobot (SI)*	-387.8 ± 1.0	-169.1 ± 32.3	-156.6 ± 38.9	-233.2 ± 2.6	-216.1 ± 7.7	-170.9 ± 40.3	-250.2 ± 13.7	-245.0 ± 5.5	
Swimmer + Gathering	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ant + Gathering	-5.8 ± 5.0	-0.1 ± 0.1	-0.4 ± 0.1	-5.5 ± 0.5	-6.7 ± 0.7	-0.4 ± 0.0	-4.7 ± 0.7	N/A ± N/A	-0.3 ± 0.3
Swimmer + Maze	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0
Ant + Maze	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	N/A ± N/A	0.0 ± 0.0

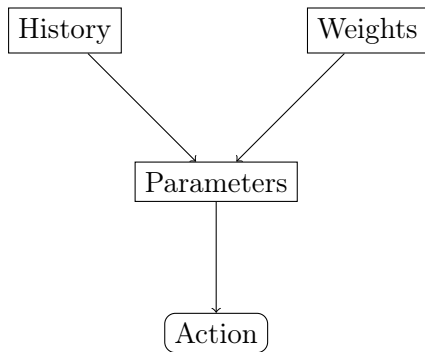
Table: Duan, Chen, Houthoofd, Schulman and Abbeel, “Benchmarking Deep Reinforcement Learning for Continuous Control,” *Proceedings of the ICML*, 2016.

Reinforcement Learning = Game Theory



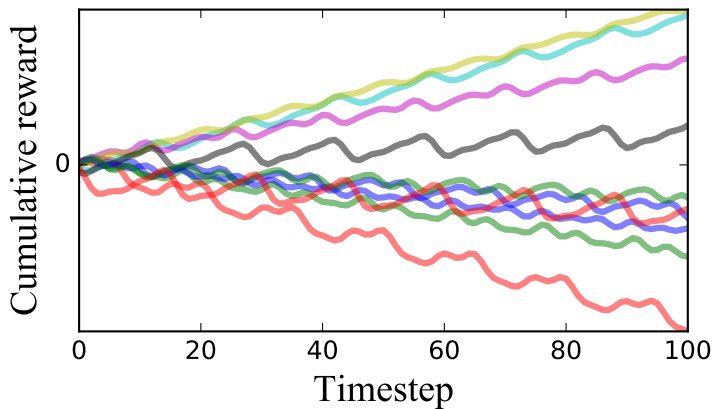
$$J(W) = E_W [R_0 + R_1 + \cdots + R_{T-1}], \text{ where } R_t = r(S_t, U_t)$$

Reinforcement Learning = Game Theory



$\text{Action} \sim \text{Distribution}(\text{History}, \text{Weights})$

The Policy Gradient Method



The Policy Gradient Method

For I epochs:

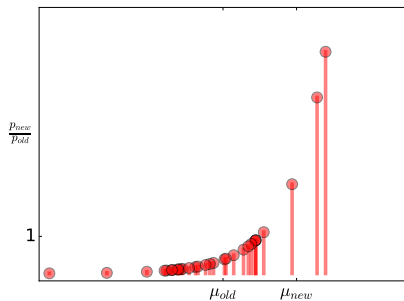
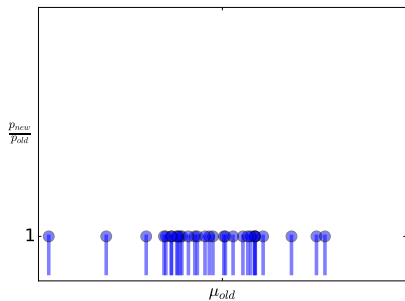
Collect N episodes, using
your stochastic policy;

Rate your actions according to
their (empirical) consequences.

Change W so that the good
actions become more probable.

Importance Weighting

$$E_{new}[X] = E_{old}\left[X \cdot \frac{p_{new}}{p_{old}}\right]$$



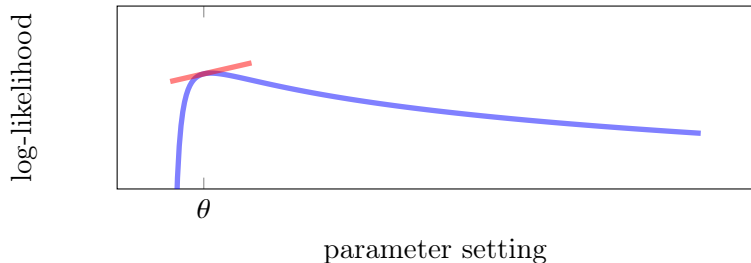
The Policy Gradient

$$E_W [R_0 + \cdots + R_{T-1}] = E_{W_0} \left[(R_0 + \cdots + R_{T-1}) \cdot \frac{p_W}{p_{W_0}} \right]$$

$$\nabla_W E_W [V] = E_{W_0} \left[V \cdot \frac{\nabla_W p_W}{p_{W_0}} \right]$$

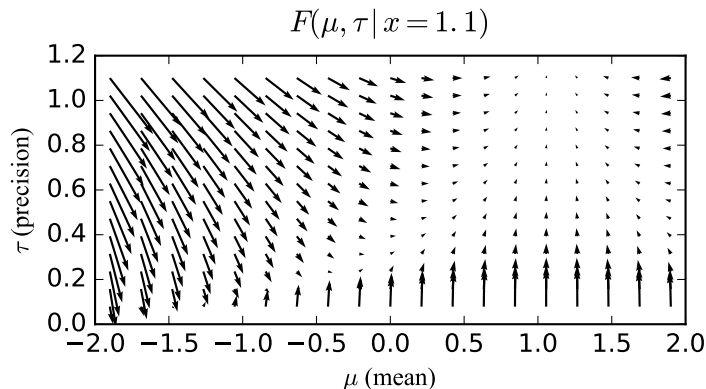
The Fisher Score

$$F(\theta | x) = \frac{\nabla_{\theta} p(x | \theta)}{p(x | \theta)} = \nabla_{\theta} \log p(x | \theta)$$



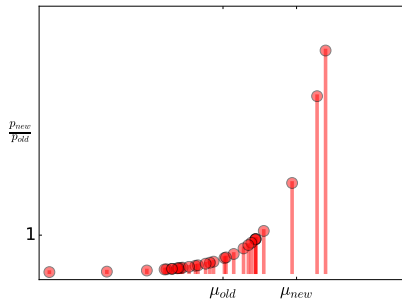
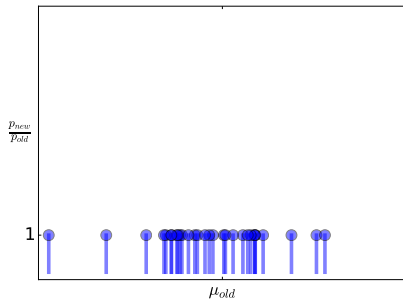
The Fisher Score

$$\nabla_{(\mu, \tau)} \log \left(\sqrt{\frac{\tau}{\pi}} \exp \left\{ -\tau(x - \mu)^2 \right\} \right) = \begin{pmatrix} 2\tau(x - \mu) \\ (2\tau)^{-1} - (x - \mu)^2 \end{pmatrix}$$

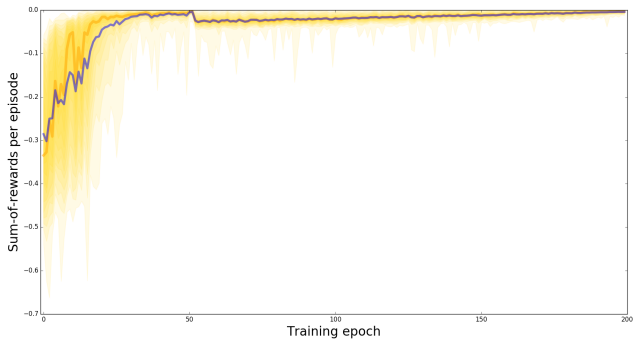


The Policy Gradient

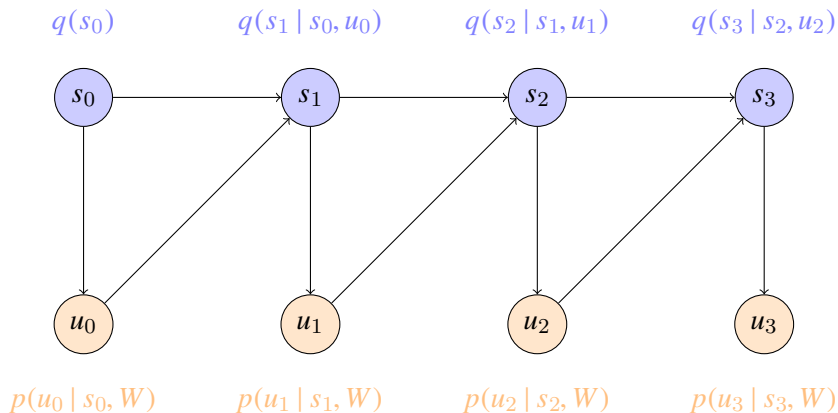
$$\nabla_{\mathbf{W}} E_{\mathbf{W}} [V] = E_{\mathbf{W}_0} [V \cdot F]$$



Dart-Throwing Game: $R(u) = -\|u - u^*\|^2$



Scores Given Rollouts



Scores Given Rollouts

$$\frac{\nabla (q_0 p_1 q_1 p_1 q_2 p_2 \cdots q_{T-1} p_{T-1})}{(q_0 p_1 q_1 p_1 q_2 p_2 \cdots q_{T-1} p_{T-1})} = \frac{\nabla (p_1 p_1 p_2 \cdots p_{T-1})}{(p_1 p_1 p_2 \cdots p_{T-1})}$$

Hence:

$$F = \nabla \log p_0 + \nabla \log p_1 + \nabla \log p_2 + \cdots + \nabla \log p_{T-1}$$

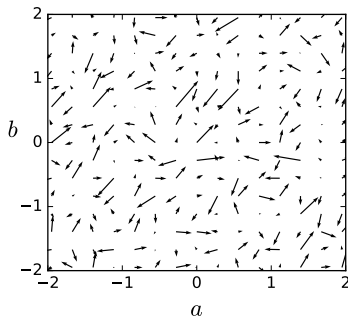
Repeat-After-Me Game: $R(s, u) = -\|s - u\|^2$

$$s \sim \mathcal{N}(1/2, 1)$$

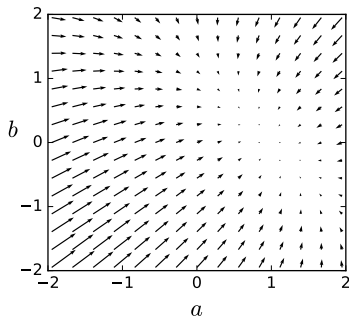
$$u \sim \mathcal{N}(as + b, 1)$$

$$F\left(\begin{matrix} a \\ b \end{matrix} \middle| s, u\right) = \begin{pmatrix} (as + b - u)s \\ (as + b - u) \end{pmatrix}$$

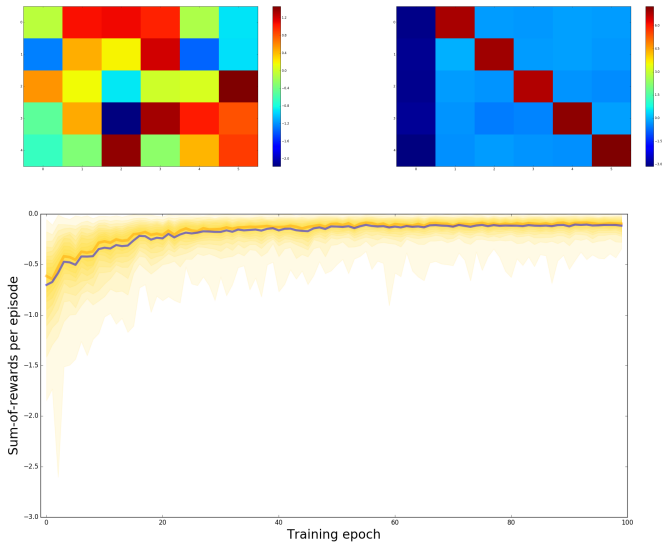
Score estimates



Gradient estimates



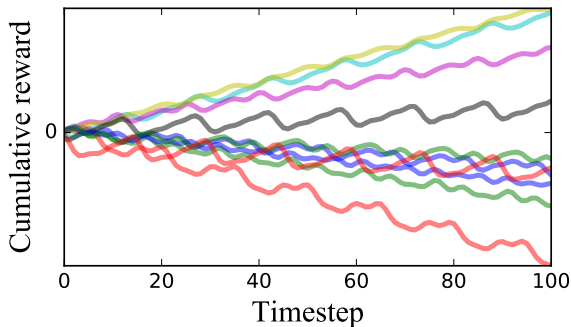
Repeat-After-Me Game: $R(s, u) = -\|s - u\|^2$



Apportioning Blame

$$V = R_0 + R_1 + R_2 + \cdots + R_{T-1}$$

$$F = \nabla \log p_0 + \nabla \log p_1 + \nabla \log p_2 + \cdots + \nabla \log p_{T-1}$$

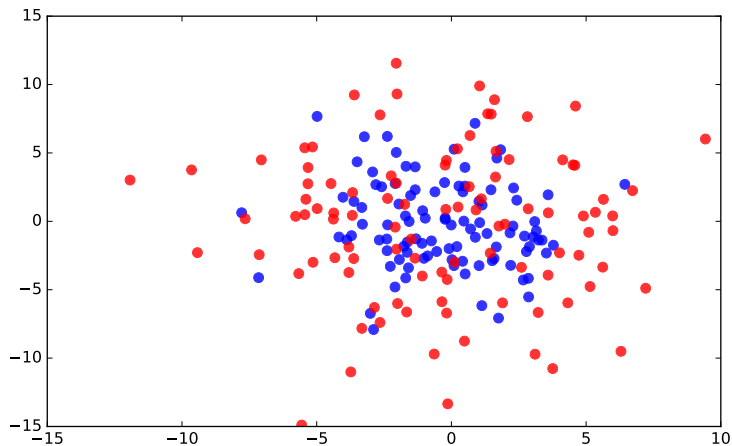


Apportioning Blame

$$E \begin{bmatrix} F_0 R_0 & F_0 R_1 & F_0 R_2 & \cdots & F_0 R_{T-1} \\ F_1 R_0 & F_1 R_1 & F_1 R_2 & \cdots & F_1 R_{T-1} \\ F_2 R_0 & F_2 R_1 & F_2 R_2 & \cdots & F_2 R_{T-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ F_{T-1} R_0 & F_{T-1} R_1 & F_{T-1} R_2 & \cdots & F_{T-1} R_{T-1} \end{bmatrix} =$$

$$E \begin{bmatrix} F_0 R_0 & F_0 R_1 & F_0 R_2 & \cdots & F_0 R_{T-1} \\ 0 & F_1 R_1 & F_1 R_2 & \cdots & F_1 R_{T-1} \\ 0 & 0 & F_2 R_2 & \cdots & F_2 R_{T-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & F_{T-1} R_{T-1} \end{bmatrix}$$

Apportioning Blame



Keeping or dropping
the zero-mean terms.

The Policy Gradient Method

For I epochs:

For N episodes:

Collect rollout, $[(S_t, U_t, R_t)]_{t=0}^{T-1}$

For each action U_t :

Compute score, $F_t = F(W | U_t)$;

Compute tailsum, $V_t = \sum_{k=t}^{T-1} R_k$;

Get gradient, $G_n = \sum_{t=0}^{T-1} V_t F_t$;

Average gradients, $G = \frac{1}{N} \sum_{n=0}^{N-1} G_n$;

Adjust weights, $W \leftarrow W + \alpha G$;

Future Complications

1. Short-lived dependencies:

$$V_t = R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \cdots$$

2. “Natural” directions:

$$G \leftarrow E[-\nabla F]^{-1} \cdot G$$

3. Line searches:

$$\alpha = \arg \max_{\alpha} J(W + \alpha G)$$

References

- ▶ Williams: “Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning” (*Machine Learning*, 1992).
- ▶ Amari: “Natural Gradient Works Efficiently in Learning” (*Neural Computation*, 1998).
- ▶ Schulman, Levine, Moritz, Jordan, and Abbeel: “Trust Region Policy Optimization” (*ICML*, 2015).
- ▶ Duan, Chen, Houthoofd, Schulman, and Abbeel, “Benchmarking Deep Reinforcement Learning for Continuous Control” (*ICML*, 2016).

`github.com/mathias-madsen/reinforce_tutorial`