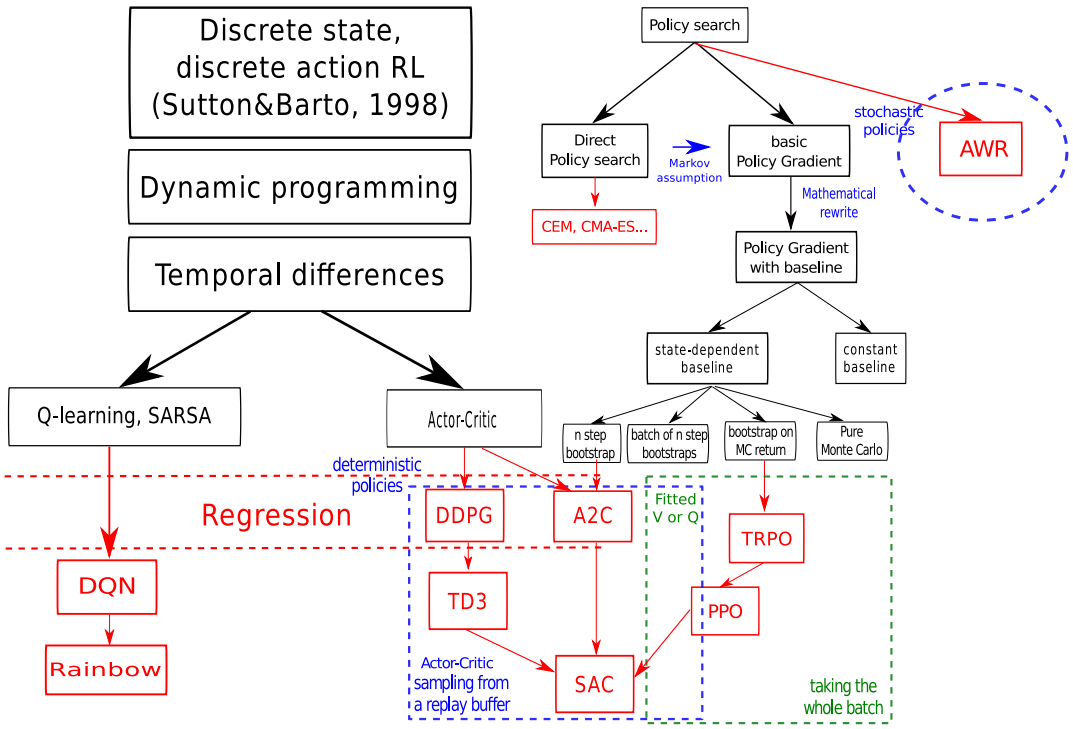
Advantage Weighted Regression

Olivier Sigaud

Sorbonne Université
http://people.isir.upmc.fr/sigaud

Motivation



- The AWR family is an outlier in the policy search landscape
- With nice properties: partly off-policy, easy to code, safe w.r.t. the deadly triad (no bootstrap)

From policy gradient to off-policy learning

- Consider the policy search setting where you have a set of trajectories τ and the corresponding rewards $r(\tau)$.
- In the policy gradient setting, you consider that you know the distribution of policies that generated these trajectories, and you write

$$P(\tau^{(i)}, \theta) = \prod_{t=1}^{H} p(s_{t+1}^{(i)} | s_t^{(i)}, a_t^{(i)}).\pi_\theta(a_t^{(i)} | s_t^{(i)})$$

- What if you do not know the distribution of policies? What can you do?
- This is the essence of the off-policy setting

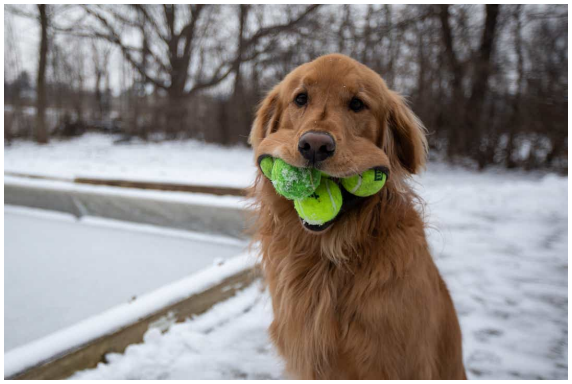
Learning a policy from regression

- An obvious thing that you can do is learn a policy from the trajectories using regression
- The learned policy should perform as the observed trajectories
- Provided rich enough trajectories, it should generalize to unseen states
- This is a form of learning from demonstration
- But the obtained policy does not perform better than observed trajectories
- Can we do better?

Weighted imitation = Batch reinforcement learning

- Using the previous regression approach, we can weight the samples based on the return of the trajectories
- This is similar to one policy gradient iteration, but from off-policy data
- The obtained policy should perform better than the observed trajectories
- If we perform just one iteration, this is batch reinforcement learning
- An open question is Under what condition on the batch data can we obtain an optimal policy this way?
- If the condition does not hold, can we still do better?

Advantage Weighted Regression

- The AWR algorithm performs the above weighted regression
- But from the policy + some exploration, it generates new data
- And it iterates from this new data
- Through exploration, better and better trajectories might be discovered
- Until the optimal policy is eventually found

Algorithm

Algorithm 1: Advantage Weighted Regression

Input : A buffer of trajectories $\mathcal{D}$;

Output: A policy $\pi_{k_{max}}$

for $k = 1, \ldots, k_{max}$ **do**

the return

approximates the value function of the current iteration

$$V_k^{\mathcal{D}} \leftarrow \operatorname*{argmin}_V \mathbb{E}_{s,a\sim\mathcal{D}}[||R_{s,a}^{\mathcal{D}} - V(s)||^2]$$

it minimizes the expectation of the difference

approximate the policy

$$\pi_k \leftarrow \operatorname*{argmax}_\pi \mathbb{E}_{s,a\sim\mathcal{D}}[\log\pi(a|s)\exp(\frac{R_{s,a}^{\mathcal{D}} - V_k^{\mathcal{D}}(s)}{\beta})]$$

log of the policy

generate new buffer $\mathcal{D}$ sampling trajectories $\{\tau_k\}$ using π_k;

- The initial buffer of trajectories $\mathcal{D}$ can come from:
  - A random policy π_0 (as in the paper)
  - Some expert trajectories (imitation learning)
- If no iteration, this is Batch Reinforcement Learning, and that's MARWIL

Peng, X. B., Kumar, A., Zhang, G., and Levine, S. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019

Estimating $V_k^{\mathcal{D}}$

$$V_k^{\mathcal{D}} \leftarrow \underset{V}{\operatorname{argmin}}\, \mathbb{E}_{s,a\sim\mathcal{D}}[||R_{s,a}^{\mathcal{D}} - V(s)||^2]$$

- The expectation is approximated over a batch:

$$V_k^{\mathcal{D}} \leftarrow \underset{V}{\operatorname{argmin}}\, \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}[||R^{\mathcal{D}}(s_t^{(i)}, a_t^{(i)}) - V(s_t)||^2]$$

- Rather than using pure Monte Carlo, $R_{s,a}^{\mathcal{D}}$ is estimated using $TD(\lambda)$ with $V_{k-1}^{\mathcal{D}}$ as bootstrap
- Thus estimate with λ-return rather than MC estimate or pure batch TD estimate ($\lambda = 0.95$)

Estimating π_k

$$\pi_k \leftarrow \underset{\pi}{\text{argmax}}\, \mathbb{E}_{s,a\sim\mathcal{D}}[\log\pi(a|s)\exp(\frac{R^{\mathcal{D}}_{s,a} - V^{\mathcal{D}}_k(s)}{\beta})]$$

- Again, the expectation is approximated over a batch:

$$\pi_k \leftarrow \underset{\pi}{\text{argmax}}\, \frac{1}{m}\sum_{i=1}^{m}\sum_{t=1}^{H}[\log\pi(a|s)\exp(\frac{R^{\mathcal{D}}_{s,a} - V^{\mathcal{D}}_k(s)}{\beta})]$$

- Enforces a trust region (see derivation)
- The weights $w^{\mathcal{D}}_{s,a} = \exp(\frac{R^{\mathcal{D}}_{s,a} - V^{\mathcal{D}}_k(s)}{\beta})$ are clipped: $w^{\mathcal{D}}_{s,a} = min(w^{\mathcal{D}}_{s,a}, w_{max})$ $(w_{max} = 20)$
- In [Peters & Schaal(2007)Peters and Schaal], tuning β is explained. Here, they take a fixed $\beta = 0.05$

Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007

Related papers

- RL approach:
  - RWR: Same idea but reward weighted instead of advantage, and no deep network

    Peters, J. and Schaal, S. Reinforcement learning by reward-weighted regression for operational space control. In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007
  - LAWER: Not applied to deep networks

    Neumann, G. and Peters, J. R. Fitted Q-iteration by advantage weighted regression. In *Advances in neural information processing systems*, pp. 1177–1184, 2009
- Imitation approach:
  - MARWIL: Just one iteration of AWR = Batch Reinforcement Learning

    Wang, Q., Xiong, J., Han, L., Liu, H., Zhang, T., et al. Exponentially weighted imitation learning for batched historical data. In *Advances in Neural Information Processing Systems*, pp. 6288–6297, 2018
  - SIL: Combines A2C + the AWR policy regression. Link to Soft-Q learning.

    Oh, J., Guo, Y., Singh, S., and Lee, H. Self-imitation learning. *arXiv preprint arXiv:1806.05635*, 2018

Experimental setup

- Batch size $=$ 2000 per iteration
- $\mathcal{D}$ contains the 50K most recent samples, thus ~ 25 previous policies
- Thus partly off-policy
- They use SGD with momentum of 0.9, mini-batch size $=$ 256
- The learning rate of the policy and value function are 5.10^{-5} and 10^{-4}
- The value function is updated with 200 gradient steps per iteration
- The policy is updated with 1000 gradient steps per iteration
- Sensitivity to these 200, 1000?
- No study of overfiting, no validation set...

Performance overview

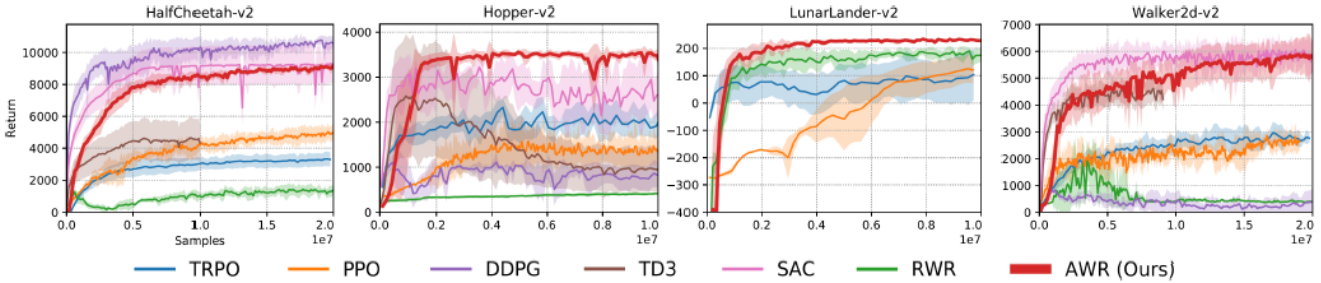


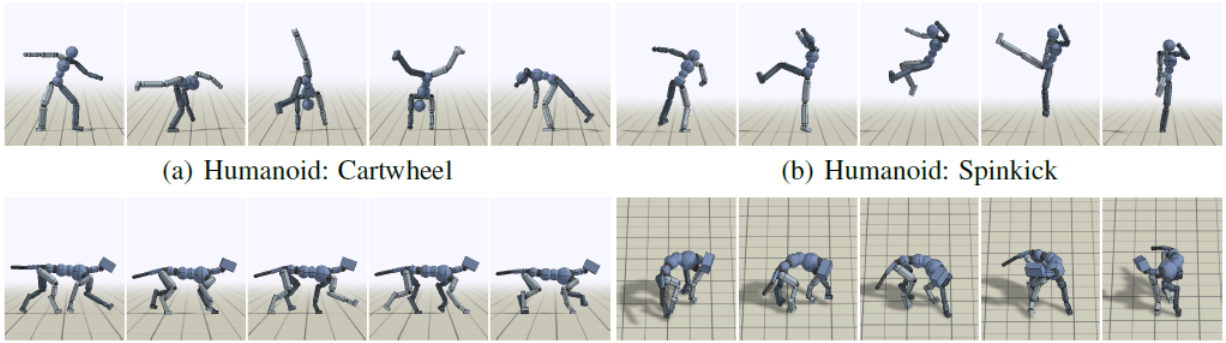
Figure 3: Learning curves of the various algorithms when applied to OpenAI Gym tasks. Results are averaged across 5 random seeds. AWR is generally competitive with the best current methods.

Task	TRPO	PPO	DDPG	TD3	SAC	RWR	AWR (Ours)
Ant-v2	2901 ± 85	1161 ± 389	72 ± 1550	4285 ± 671	$\mathbf{5909 \pm 371}$	181 ± 19	5067 ± 256
HalfCheetah-v2	3302 ± 428	4920 ± 429	$\mathbf{10563 \pm 382}$	4309 ± 1238	9297 ± 1206	1400 ± 370	9136 ± 184
Hopper-v2	1880 ± 337	1391 ± 304	855 ± 282	935 ± 489	2769 ± 552	605 ± 114	$\mathbf{3405 \pm 121}$
Humanoid-v2	552 ± 9	695 ± 59	4382 ± 423	81 ± 17	$\mathbf{8048 \pm 700}$	509 ± 18	4996 ± 697
LunarLander-v2	104 ± 94	121 ± 49	–	–	–	185 ± 23	$\mathbf{229 \pm 2}$
Walker2d-v2	2765 ± 168	2617 ± 362	401 ± 470	4212 ± 427	$\mathbf{5805 \pm 587}$	406 ± 64	$\mathbf{5813 \pm 483}$

Table 1: Final returns for different algorithms on the OpenAI Gym tasks, with $\pm$ corresponding to one standard deviation of the average return across 5 random seeds. In terms of final performance, AWR generally performs comparably or better than prior methods.

- AWR is consistently better than RWR
- Performance is more consistent over benchmarks than competing methods

Imitation performance



(a) Humanoid: Cartwheel (b) Humanoid: Spinkick

(c) Dog: Trot (d) Dog: Turn

Figure 5: Snapshots of 34 DoF humanoid and 64 DoF dog trained with AWR to imitate reference motion recorded from real world subjects. AWR is able to learn sophisticated skills with characters with large numbers of degrees of freedom.

Task	PPO	RWR	AWR (Ours)
Humanoid: Cartwheel	0.76 ± 0.02	0.03 ± 0.01	$\mathbf{0.78 \pm 0.07}$
Humanoid: Spinkick	0.70 ± 0.02	0.05 ± 0.03	$\mathbf{0.77 \pm 0.04}$
Dog: Canter	0.76 ± 0.03	0.78 ± 0.04	$\mathbf{0.86 \pm 0.01}$
Dog: Trot	$\mathbf{0.86 \pm 0.01}$	$\mathbf{0.86 \pm 0.01}$	$\mathbf{0.86 \pm 0.03}$
Dog: Turn	0.75 ± 0.02	0.75 ± 0.03	$\mathbf{0.82 \pm 0.03}$

Table 2: Performance statistics of algorithms on the motion imitation tasks. Returns are normalized between the minimum and maximum possible returns per episode.

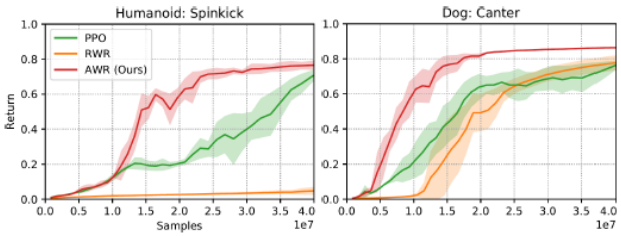


Figure 6: Learning curves on motion imitation tasks. On these challenging tasks, AWR generally learns faster than PPO and RWR.

- Efficiency of imitation learning
- Watch the nice video

https://sites.google.com/view/awr-supp/

Sensitivity to hyperparameters

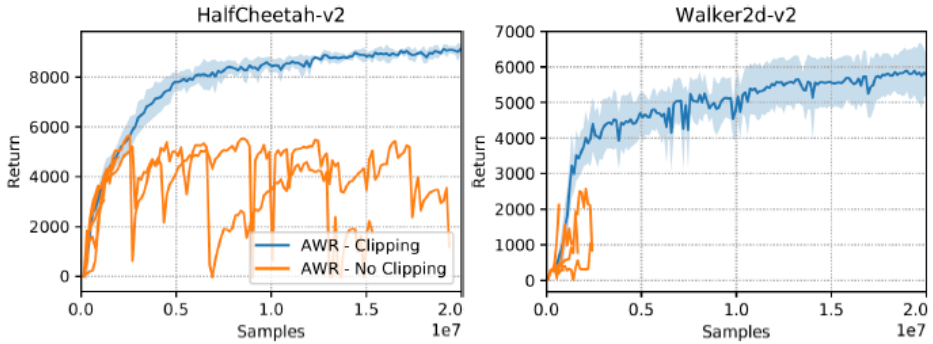


Figure 10: Learning curves comparing AWR policies trained with and without weight clipping. Weight clipping is vital for ensuring stable training with AWR. Policies trained without weight clipping are susceptible to exploding gradients due to excessively large weights.

- Sensitivity to w_{max}: weight clipping is vital!
- Raises doubts on the validity of exponentiating the advantage
- Sensitivity to λ, β?

Ablation study

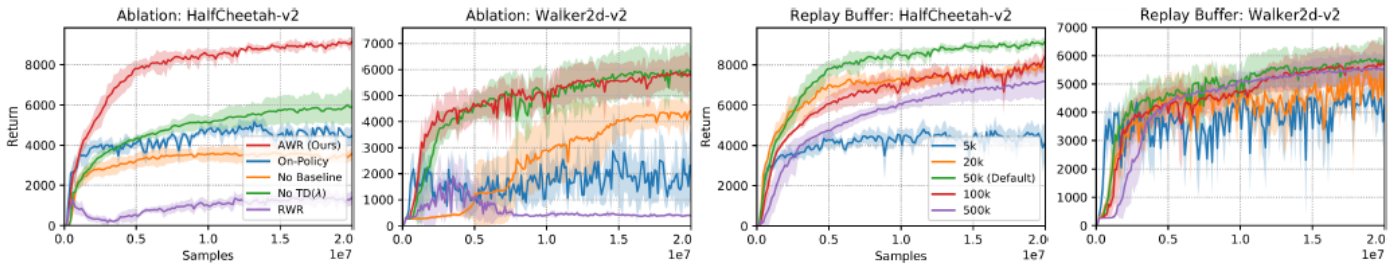


Figure 4: **Left:** Learning curves comparing AWR with various components removed. Each component appears to contribute to improvements in performance, with the best performance achieved when all components are combined. **Right:** Learning curves comparing AWR with different capacity replay buffers. AWR remains stable with large replay buffers containing primarily off-policy data from previous iterations of the algorithm.

- Using advantage versus reward (RWR) is critical
- Using the $V_k^{\mathcal{D}}(s)$ baseline is important
- Using data just from the previous policy is worse (on-policy)
- No major impact of using $TD(\lambda)$ instead of MC in Walke-2D, large impact in HalfCheetah
- Thus sensitivity to λ does not matter much

Properties

- Even if $R^{\mathcal{D}}_{s,a}$ is estimated with bootstrap, $V^{\mathcal{D}}_{k}$ is still estimated with regression
- Not a bootstrap method, so should avoid the deadly triad issue
- But still more sample efficient than MC methods (TRPO, PPO)
- In principle, AWR could be truly off-policy
- But in the derivation, the current policy should not be too distant from the previous one (trust region)
- Derivation quite similar to the standard policy gradient derivation
- Derived from an EM approach

Difference to policy gradient

- Regression computes an internal gradient, so is it truly different?

$$\mathbb{E}_{s\sim d_\pi(s)}\mathbb{E}_{a\sim\pi(a|s)}[\nabla_\theta \log\pi(a|s)(R^\pi_{s,a} - V^\pi(s))] \qquad (PG)$$

$$\mathbb{E}_{s\sim d_\mu(s)}\mathbb{E}_{a\sim\mu(a|s)}[\nabla_\theta \log\pi(a|s)\exp(\frac{R^\mu_{s,a} - V^\mu(s)}{\beta}) \qquad (AWR)$$

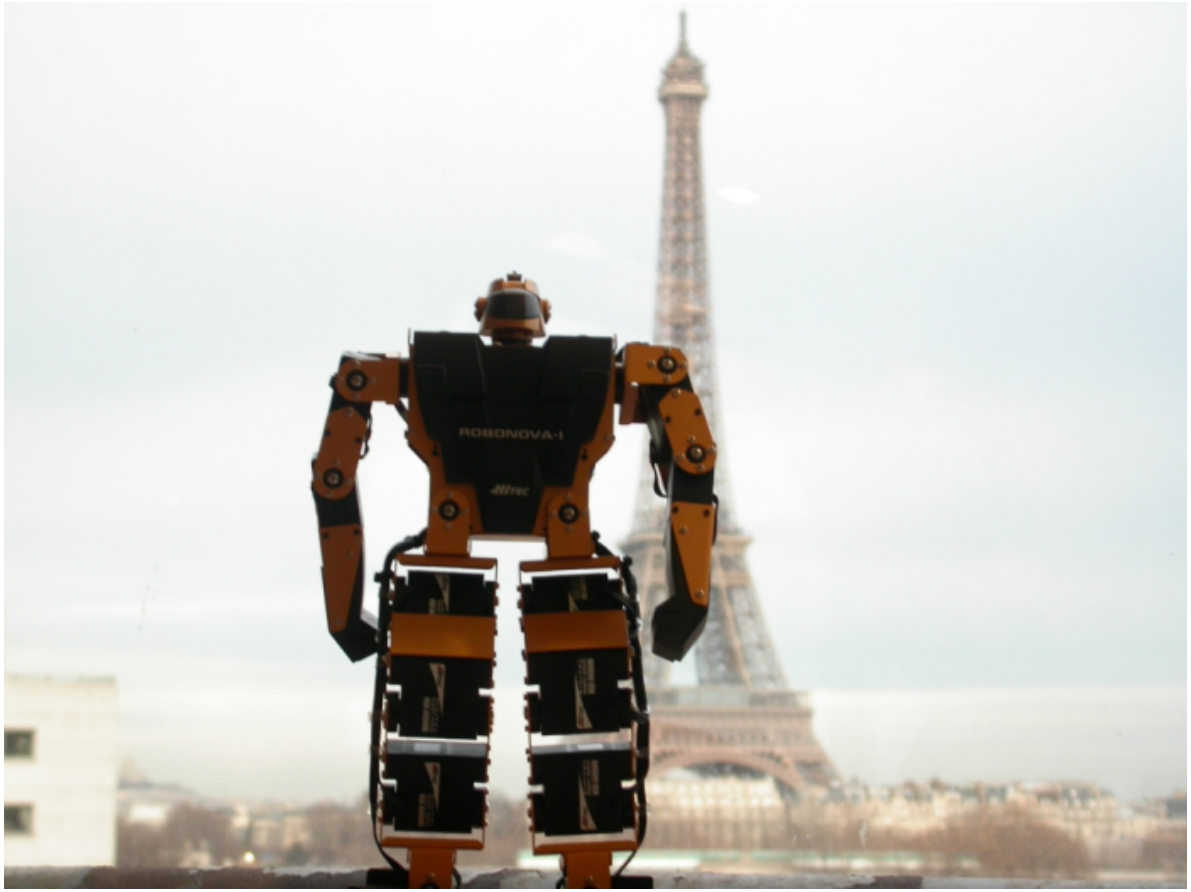
- In PG, from the data we compute a gradient related to and applied to the current policy.
- In AWR, from the data we compute a whole new policy

Role of exponentiation

- From the derivation, enforces a trust region
- In PG, a negative step size can make learning unstable (see CACLA)
- By exponentiating, guarantee to always get it positive
- PG with exponentiated step size should work better (check with a negative reward problem)

Van Hasselt, H. and Wiering, M. A. Reinforcement learning in continuous action spaces. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 272–279, 2007

Any question?



Send mail to: Olivier.Sigaud@upmc.fr

Gerhard Neumann and Jan R Peters.

Fitted q-iteration by advantage weighted regression.
In *Advances in neural information processing systems*, pp. 1177–1184, 2009.

Junhyuk Oh, Yijie Guo, Satinder Singh, and Honglak Lee.

Self-imitation learning.
arXiv preprint arXiv:1806.05635, 2018.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine.

Advantage-weighted regression: Simple and scalable off-policy reinforcement learning.
arXiv preprint arXiv:1910.00177, 2019.

Jan Peters and Stefan Schaal.

Reinforcement learning by reward-weighted regression for operational space control.
In *Proceedings of the 24th international conference on Machine learning*, pp. 745–750, 2007.

Hado Van Hasselt and Marco A. Wiering.

Reinforcement learning in continuous action spaces.
In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning (ADPRL)*, pp. 272–279, 2007.

Qing Wang, Jiechao Xiong, Lei Han, Han Liu, Tong Zhang, et al.

Exponentially weighted imitation learning for batched historical data.
In *Advances in Neural Information Processing Systems*, pp. 6288–6297, 2018.