RL: Policy Search

Finite Difference

Marius Lindauer

Leibniz Universität Hannover

Automated Machine Learning Hannover

Policy Gradient

- Assume episodic MDPs
- Policy gradient algorithms search for a local maximum in $V(s_0, \theta)$ by ascending the gradient of the policy, w.r.t parameters θ

$$\Delta\theta = \alpha \nabla_\theta V(s_0, \theta)$$

where α is the learning rate (step-size) and $\nabla_\theta V(s_0, \theta)$ is the policy gradient

$$\nabla_\theta V(s_0, \theta) = \begin{pmatrix} \frac{\partial V(s_0, \theta)}{\partial \theta_1} \\ \vdots \\ \frac{\partial V(s_0, \theta)}{\partial \theta_n} \end{pmatrix}$$

Automated Machine Learning Hannover

Simple Approach: Compute Gradients by Finite Differences

- To evaluate policy gradient of $\pi_\theta(s, a)$
- For each dimension $k \in [1, n]$
  - Estimate k-th partial derivative of objective function wrt θ
  - By pertubating θ by small amount ϵ in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon u_k) - V(s_0, \theta)}{\epsilon}$$

where u_k is a unit vector with 1 in k-th component and 0 elsewhere

Automated Machine Learning Hannover

Simple Approach: Compute Gradients by Finite Differences

- To evaluate policy gradient of $\pi_\theta(s, a)$
- For each dimension $k \in [1, n]$
  - Estimate k-th partial derivative of objective function wrt θ
  - By pertubating θ by small amount ϵ in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon u_k) - V(s_0, \theta)}{\epsilon}$$

where u_k is a unit vector with 1 in k-th component and 0 elsewhere

- ϵ should be
  - large enough to observe a change in the policy and value function
  - small enough to have a good gradient approximation

Simple Approach: Compute Gradients by Finite Differences

- To evaluate policy gradient of $\pi_\theta(s, a)$
- For each dimension $k \in [1, n]$
  - Estimate k-th partial derivative of objective function wrt θ
  - By pertubating θ by small amount ϵ in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon u_k) - V(s_0, \theta)}{\epsilon}$$

  where u_k is a unit vector with 1 in k-th component and 0 elsewhere

- ϵ should be
  - large enough to observe a change in the policy and value function
  - small enough to have a good gradient approximation
- Uses $\geq n$ evaluations to compute policy gradient in n dimensions
  - $\rightsquigarrow$ fairly inefficient for doing a single update!
  - $\rightsquigarrow$ weight space should be small — no computer vision

Automated Machine Learning Hannover

Simple Approach: Compute Gradients by Finite Differences

- To evaluate policy gradient of $\pi_\theta(s, a)$
- For each dimension $k \in [1, n]$
  - Estimate k-th partial derivative of objective function wrt θ
  - By pertubating θ by small amount ϵ in k-th dimension

$$\frac{\partial V(s_0, \theta)}{\partial \theta_k} \approx \frac{V(s_0, \theta + \epsilon u_k) - V(s_0, \theta)}{\epsilon}$$

  where u_k is a unit vector with 1 in k-th component and 0 elsewhere
- ϵ should be
  - large enough to observe a change in the policy and value function
  - small enough to have a good gradient approximation
- Uses $\geq n$ evaluations to compute policy gradient in n dimensions
  - $\rightsquigarrow$ fairly inefficient for doing a single update!
  - $\rightsquigarrow$ weight space should be small — no computer vision
- Simple, noisy, inefficient – but sometimes effective
- Works for arbitrary policies, even if policy is not differentiable