Week 4 Briefing

Making robots learn by trial and error

Jack Cashman

July 19, 2024

Plan for presentation

Overview and Motivation

1. Last week I found that the beta distribution is a promising alternative as a prior for the policy in continuous and bounded action spaces.
2. This week, I've been experimenting in order to boost the performance of PPO-clip when using a beta policy.
3. Today I'll recap some of these experiments, as well as some issues and how I fixed them.

Overview and Motivation

1. Last week I found that the beta distribution is a promising alternative as a prior for the policy in continuous and bounded action spaces.
2. This week, I've been experimenting in order to boost the performance of PPO-clip when using a beta policy.
3. Today I'll recap some of these experiments, as well as some issues and how I fixed them.

Overview and Motivation

1. Last week I found that the beta distribution is a promising alternative as a prior for the policy in continuous and bounded action spaces.
2. This week, I've been experimenting in order to boost the performance of PPO-clip when using a beta policy.
3. Today I'll recap some of these experiments, as well as some issues and how I fixed them.

A Note On Numerical Instability

Given d independent, beta-distributed random variables $X_1, \ldots, X_d$, each with parameters α_i, β_i, the joint distribution, f, is:

$$f(x_1, \ldots x_d) \propto \prod_{i=1}^{d} {x_i}^{\alpha_i - 1}(1 - x_i)^{\beta_i - 1}$$

If either $\alpha_i < 1$ or $\beta_i < 1$, for some $1 \leq i \leq d$, then f is unbounded. This is problematic, as in PPO I need to call a method that returns $\log(f(\mathbf{x}))$ for a point $\mathbf{x} \in [0, 1]^d$. The distribution is bounded if we enforce $\alpha_i, \beta_i \geq 1$ for all i. I've used this restriction in all of the following implementations.

Restrictions on α and β

- For numerical stability I require $\alpha_i, \beta_i > 1$.
- In computation, the α_i and β_i that is output by the NN are relatively close to this lower bound.
- This distribution is *very* stochastic. For larger α_i and β_i, the policy becomes less stochastic.
- Perhaps introducing larger lower bounds on the α_i and β_i can improve performance?
- I've tested lower bounds of $1, 3, 5, 25$ on 5 seeds for 3 environments.

Restrictions on α and β

- For numerical stability I require $\alpha_i, \beta_i > 1$.
- In computation, the α_i and β_i that is output by the NN are relatively close to this lower bound.
- This distribution is *very* stochastic. For larger α_i and β_i, the policy becomes less stochastic.
- Perhaps introducing larger lower bounds on the α_i and β_i can improve performance?
- I've tested lower bounds of $1, 3, 5, 25$ on 5 seeds for 3 environments.

Restrictions on α and β

- For numerical stability I require $\alpha_i, \beta_i > 1$.
- In computation, the α_i and β_i that is output by the NN are relatively close to this lower bound.
- This distribution is *very* stochastic. For larger α_i and β_i, the policy becomes less stochastic.
- Perhaps introducing larger lower bounds on the α_i and β_i can improve performance?
- I've tested lower bounds of $1, 3, 5, 25$ on 5 seeds for 3 environments.

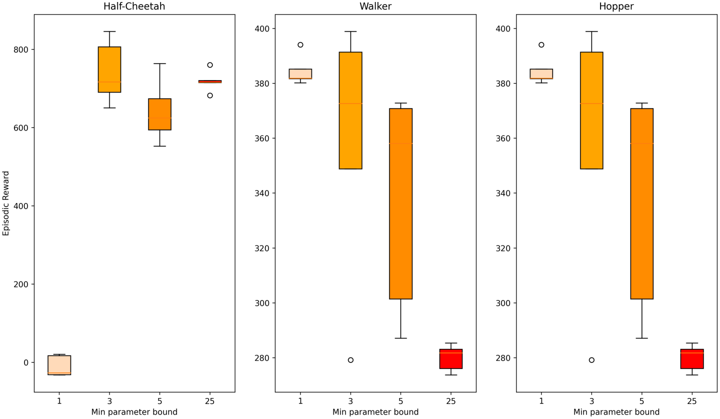
Restrictions on α and β

- For numerical stability I require $\alpha_i, \beta_i > 1$.
- In computation, the α_i and β_i that is output by the NN are relatively close to this lower bound.
- This distribution is *very* stochastic. For larger α_i and β_i, the policy becomes less stochastic.
- Perhaps introducing larger lower bounds on the α_i and β_i can improve performance?
- I've tested lower bounds of $1, 3, 5, 25$ on 5 seeds for 3 environments.

Restrictions on α and β

- For numerical stability I require $\alpha_i, \beta_i > 1$.
- In computation, the α_i and β_i that is output by the NN are relatively close to this lower bound.
- This distribution is *very* stochastic. For larger α_i and β_i, the policy becomes less stochastic.
- Perhaps introducing larger lower bounds on the α_i and β_i can improve performance?
- I've tested lower bounds of $1, 3, 5, 25$ on 5 seeds for 3 environments.

Restrictions on α and β - Results



It appears that requiring $\alpha_i, \beta_i \geq 3$ yields the best performance.

Different NN models

- I've been using 2 NNs to predict the vectors $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \dots & \alpha_d \end{pmatrix}^\intercal$ and $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \dots & \beta_d \end{pmatrix}^\intercal$
- To ensure that both vectors satisfy $\alpha_i, \beta_i > 3$, I push the output through the ReLU function, and then add 3
- Nan suggested that we may be able to instead use a single neural network. So, output $\begin{pmatrix} \alpha_1 & \dots & \alpha_d & \beta_1 & \dots & \beta_d \end{pmatrix}^\intercal$
- To test this, I've compared over 5 seeds for 3 separate `mujoco` environments; `Half-Cheetah`, `Hopper`, and `Walker-2d`

Different NN models

- I've been using 2 NNs to predict the vectors $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \dots & \alpha_d \end{pmatrix}^\intercal$ and $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \dots & \beta_d \end{pmatrix}^\intercal$
- To ensure that both vectors satisfy $\alpha_i, \beta_i > 3$, I push the output through the ReLU function, and then add 3
- Nan suggested that we may be able to instead use a single neural network. So, output $\begin{pmatrix} \alpha_1 & \dots & \alpha_d & \beta_1 & \dots & \beta_d \end{pmatrix}^\intercal$
- To test this, I've compared over 5 seeds for 3 separate `mujoco` environments; `Half-Cheetah`, `Hopper`, and `Walker-2d`

Different NN models

- I've been using 2 NNs to predict the vectors $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \dots & \alpha_d \end{pmatrix}^\mathsf{T}$ and $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \dots & \beta_d \end{pmatrix}^\mathsf{T}$
- To ensure that both vectors satisfy $\alpha_i, \beta_i > 3$, I push the output through the ReLU function, and then add 3
- Nan suggested that we may be able to instead use a single neural network. So, output $\begin{pmatrix} \alpha_1 & \dots & \alpha_d & \beta_1 & \dots & \beta_d \end{pmatrix}^\mathsf{T}$
- To test this, I've compared over 5 seeds for 3 separate `mujoco` environments; `Half-Cheetah`, `Hopper`, and `Walker-2d`

Different NN models

- I've been using 2 NNs to predict the vectors $\boldsymbol{\alpha} = \begin{pmatrix} \alpha_1 & \dots & \alpha_d \end{pmatrix}^\intercal$ and $\boldsymbol{\beta} = \begin{pmatrix} \beta_1 & \dots & \beta_d \end{pmatrix}^\intercal$
- To ensure that both vectors satisfy $\alpha_i, \beta_i > 3$, I push the output through the ReLU function, and then add 3
- Nan suggested that we may be able to instead use a single neural network. So, output $\begin{pmatrix} \alpha_1 & \dots & \alpha_d & \beta_1 & \dots & \beta_d \end{pmatrix}^\intercal$
- To test this, I've compared over 5 seeds for 3 separate `mujoco` environments; `Half-Cheetah`, `Hopper`, and `Walker-2d`

Different NN models - Results

Overall, using a single network to predict both the vector of α_i's and β_i's appears to be more effective.

	`Cheetah`	`Hopper`	`Walker`
Double Net	284 ± 46.6	358.2 ± 43.2	347 ± 20.3
Single Net	631 ± 48.4	383.4 ± 15.6	346.4 ± 10.8

Combining this with the aforementioned restriction that $\alpha_i, \beta_i > 3$, the algorithm performs similar to, or better than the `cleanrl` implementation that assume a diagonal Gaussian prior for the policy.

Conclusion

The performance when a Beta prior is assumed vs the performance when a digaonal Gaussian prior is assumed across 5 random seeds:

	`Cheetah`	`Hopper`	`Walker`
Gaussian Prior	32.5 ± 139.4	765.6 ± 237.5	365.7 ± 13.6
Beta Prior	742 ± 72.8	415.5 ± 59.6	384.6 ± 5

The beta distribution serves as a promising alternative to the traditional diagonal Gaussian in certain continuous and bounded action spaces.