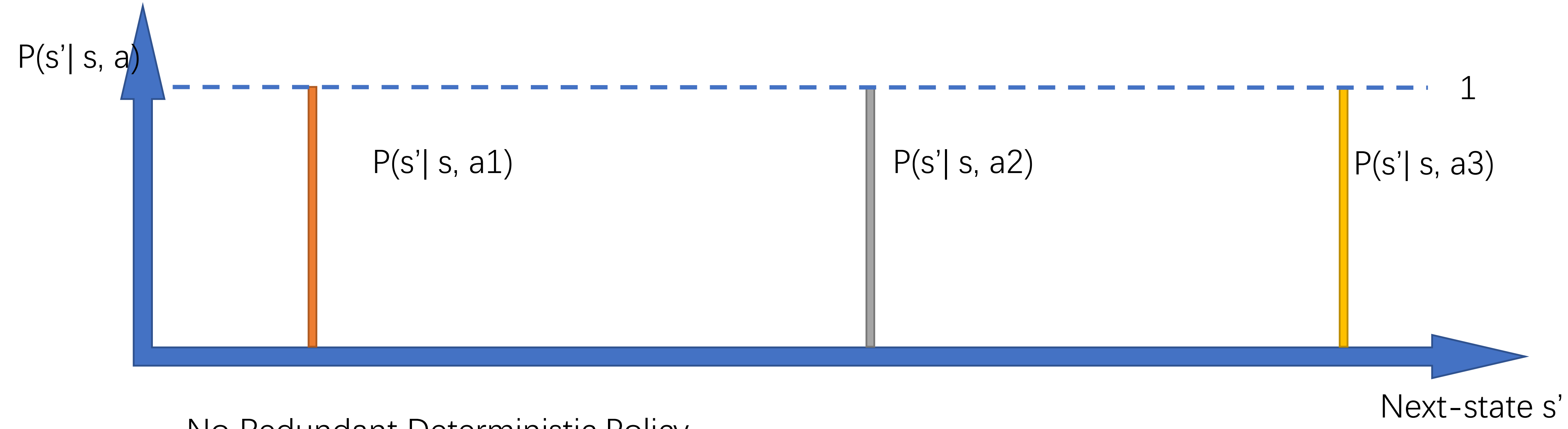
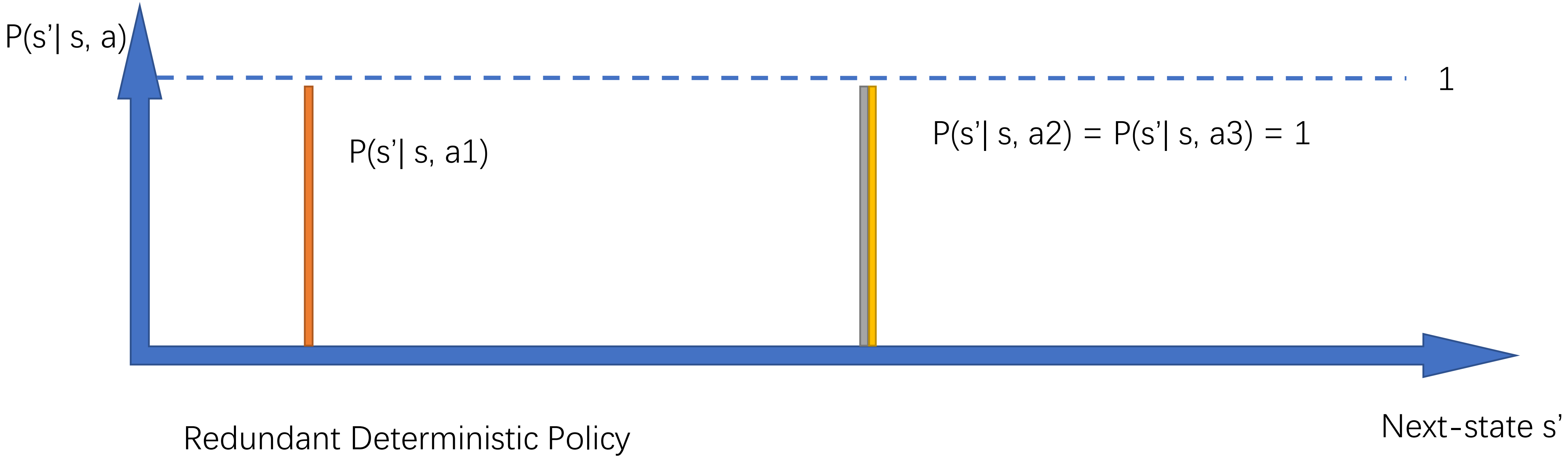
Explanation of No Redundant Deterministic Policy & Redundant Deterministic Policy

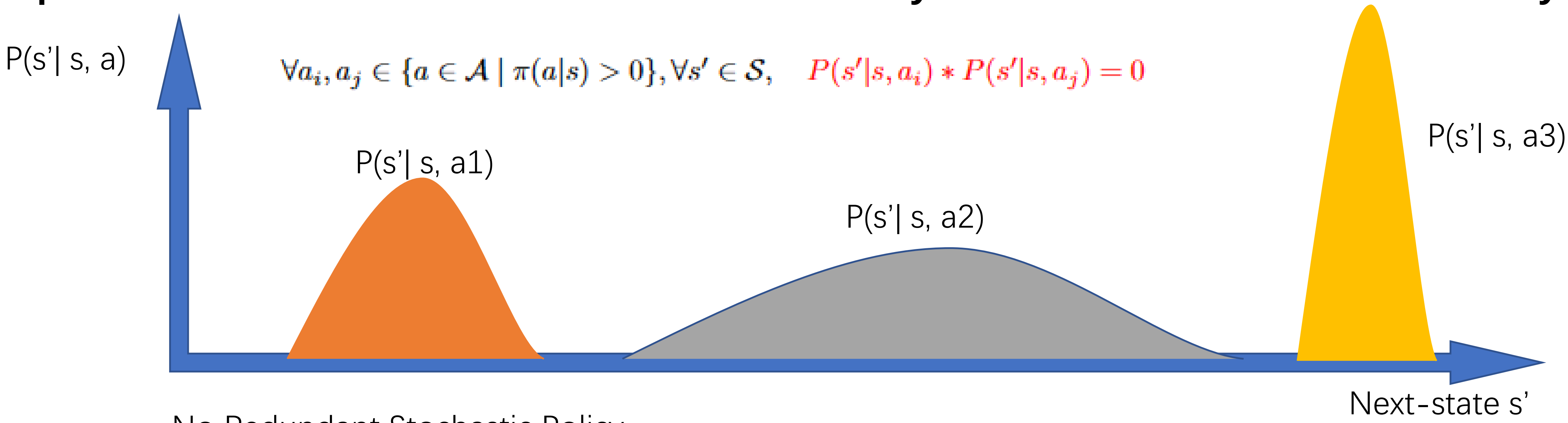


No Redundant Deterministic Policy

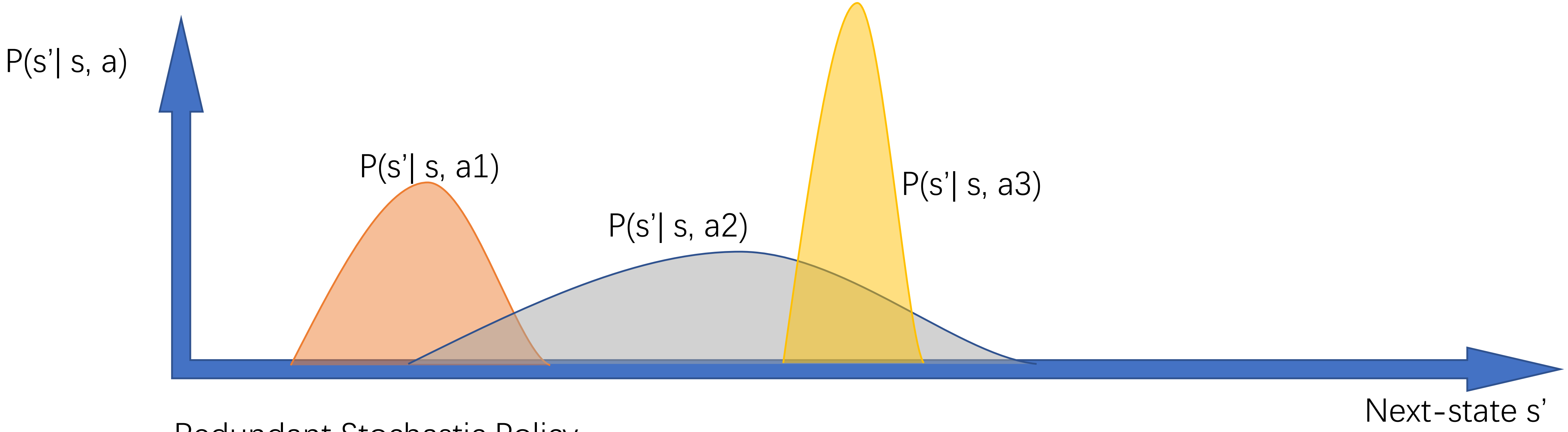


Redundant Deterministic Policy

Explanation of No Redundant Stochastic Policy & Redundant Stochastic Policy



No Redundant Stochastic Policy



Redundant Stochastic Policy

Incomplete beta

不完全 beta 函数

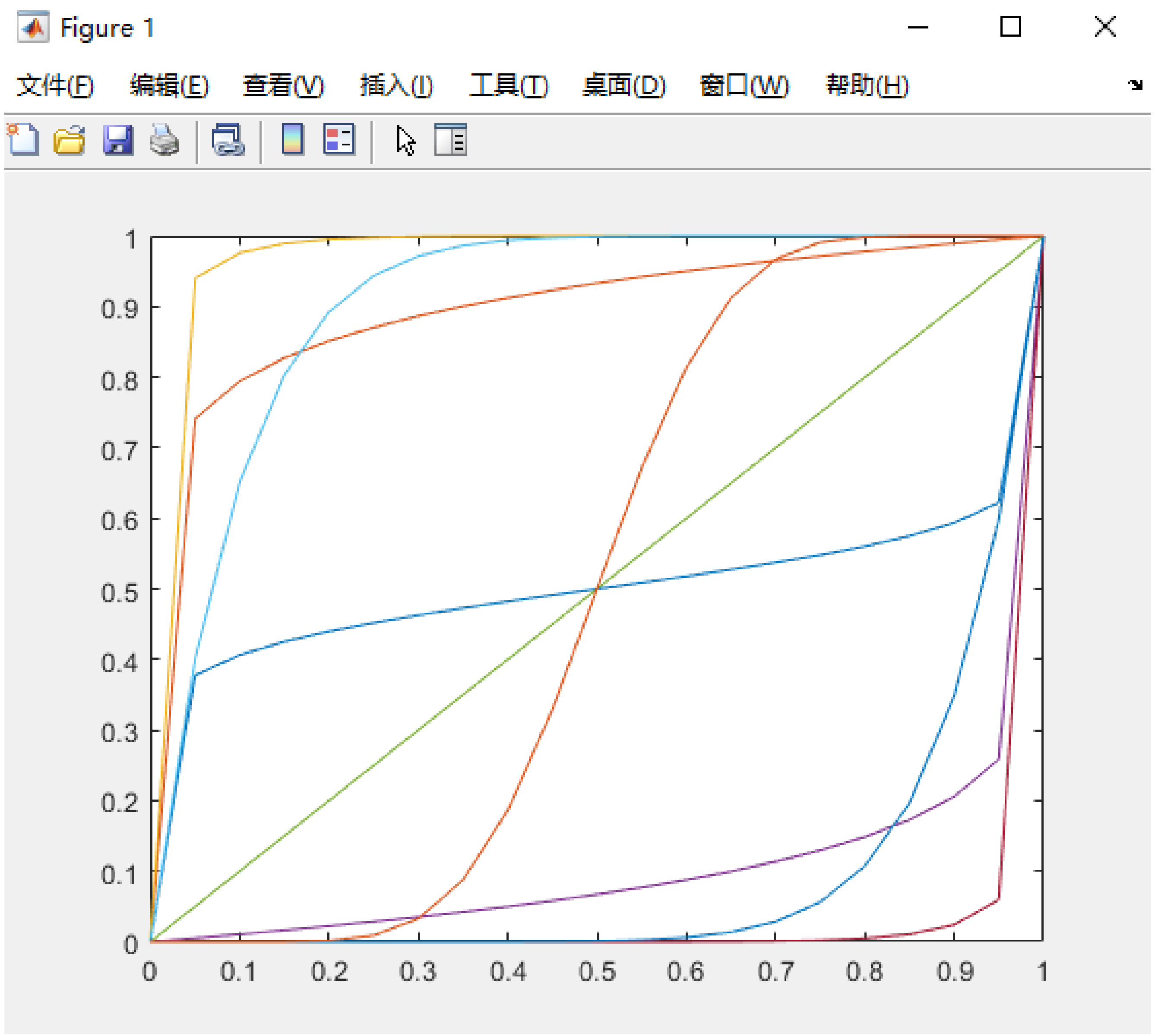
不完全 beta 函数为

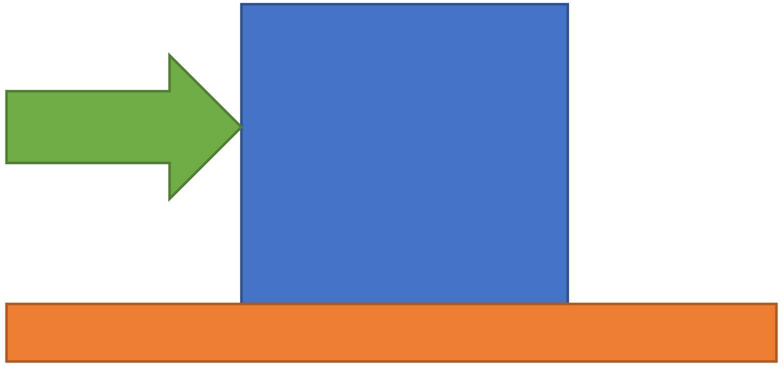
$$I_x(z,w)=\frac{1}{B(z,w)}\int_0^x t^{z-1}(1-t)^{w-1}dt$$

其中 $B(z,w)$，即 beta 函数，定义为

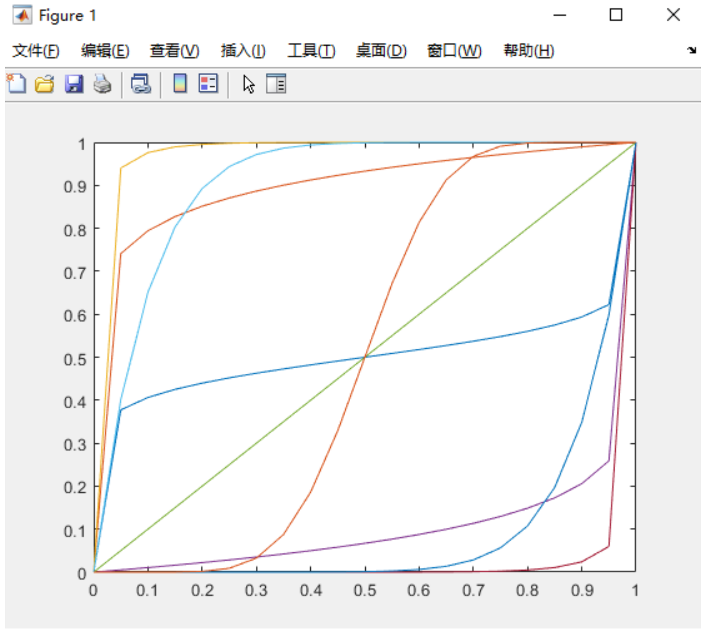
$$B(z,w)=\int_0^1 t^{z-1}(1-t)^{w-1}dt=\frac{\Gamma(z)\Gamma(w)}{\Gamma(z+w)}$$

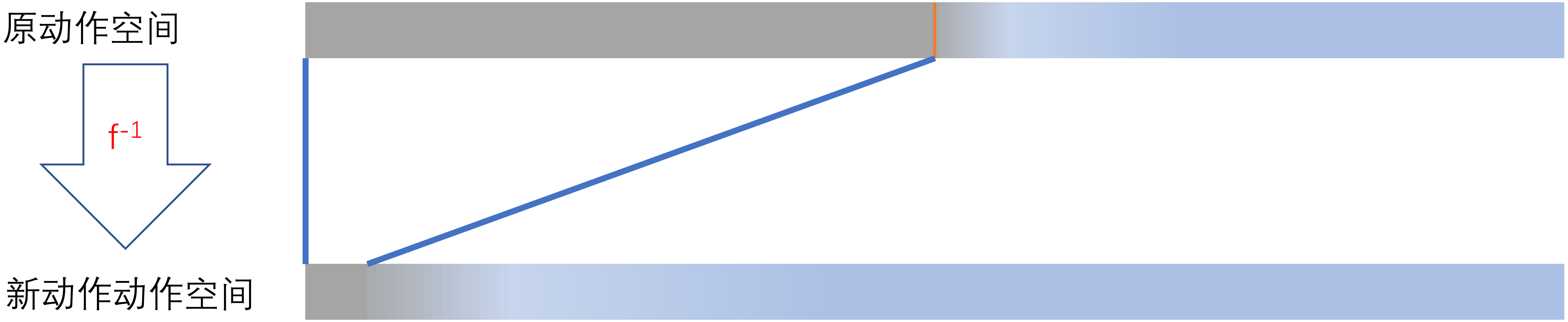
并且 $\Gamma(z)$ 为 gamma 函数。





由于有摩擦力，
0-0.5 N的力，小方块都不动
当力大于0.5N， 小方块才动





Method

$$\min_{\theta} J(\theta) = -\mathbb{E}_{e\sim\pi_i(\cdot|s)}\mathbb{E}_{s'\sim P^i(\cdot|s,e)} \log P^{\pi_i}(e|s,s')$$
$$= -\mathbb{E}_{a\sim\pi_o(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)} \log P(a|s,s')|\frac{\partial f(\theta,e)}{\partial e}|$$

最小化目标函数的理解:
1、考虑最特殊情况，如果P(a|s,s')非常小，那么说明a的冗余程度非常高，logP趋向于负无穷，这时 $\frac{\partial \mathrm{f}}{\partial e}$ 需要取一个较大值

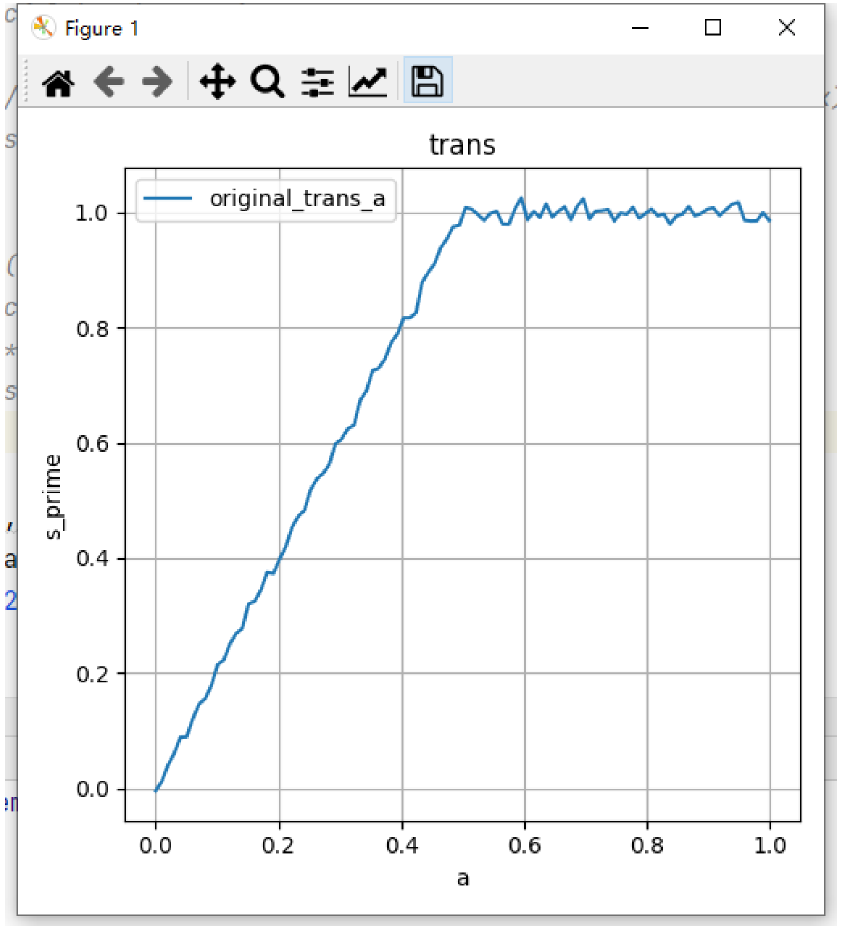
Gradient

$$\nabla_\theta J(\theta) = -\mathbb{E}_{a\sim\pi_o(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)} \left\{ -\frac{\partial \log |\frac{\partial f(\theta,e)}{\partial e}|}{\partial\theta} \left[\log P(a|s,s') + \log|\frac{\partial f(\theta,e)}{\partial e}|\right] + \frac{\partial \log |\frac{\partial f(\theta,e)}{\partial e}|}{\partial\theta} \right\}$$
$$= -\mathbb{E}_{a\sim\pi_o(\cdot|s)}\mathbb{E}_{s'\sim P(\cdot|s,a)} \left\{ \frac{\partial \log |\frac{\partial f(\theta,e)}{\partial e}|}{\partial\theta} \left[1 - \log P(a|s,s') - \log|\frac{\partial f(\theta,e)}{\partial e}|\right] \right\}$$

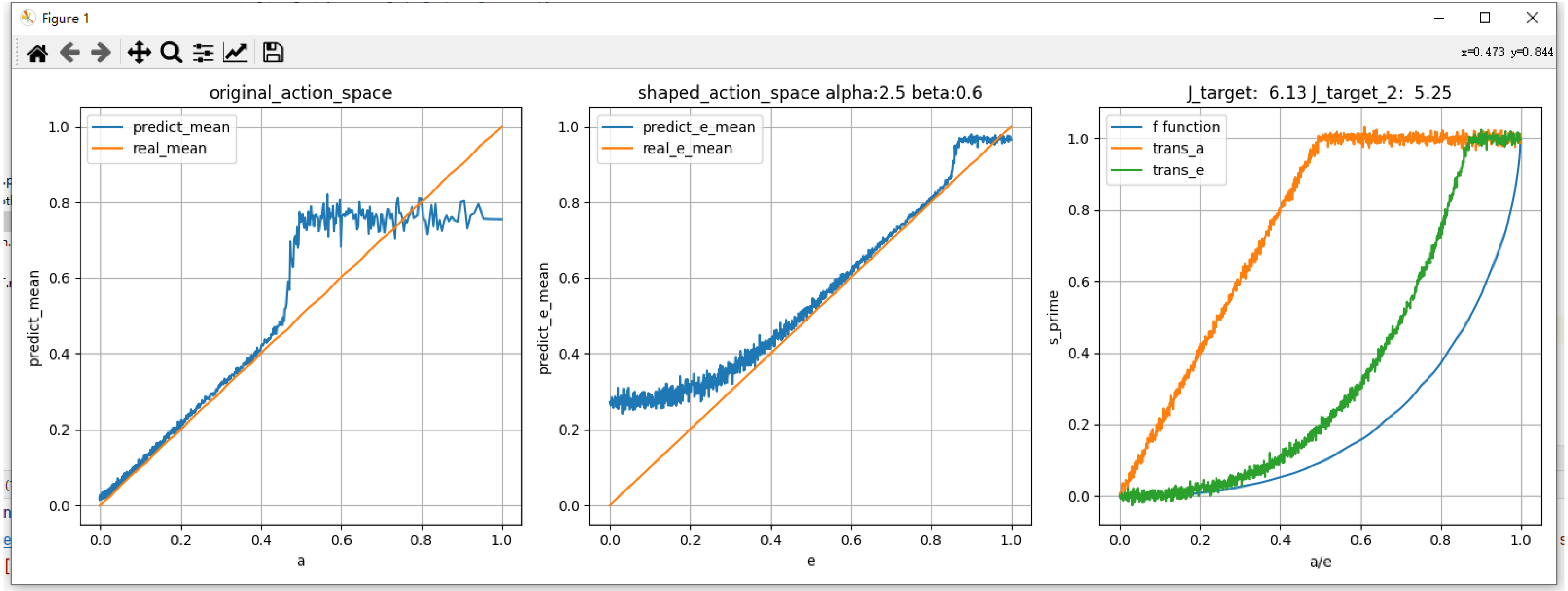
程序示例:

- 1、运行transition_function.py，查看当前的环境转移，为单步的MDP，初始状态为0，动作0-1，状态0-1

- 2、运行inverse_dynamics.py，训练完成RealInvNetwork.pth。该网络输入为s和s‘，输出为真实的动作a（用一个正态分布拟合）。



- 3、运行test_inverse.py。最左边示意图是原来的动作空间下，inverse网络预测的a与真正的a之间的差异。因为该mdp在动作>0.5之后的输出几乎一致，所以很难区别0.5-1.0之间的动作。

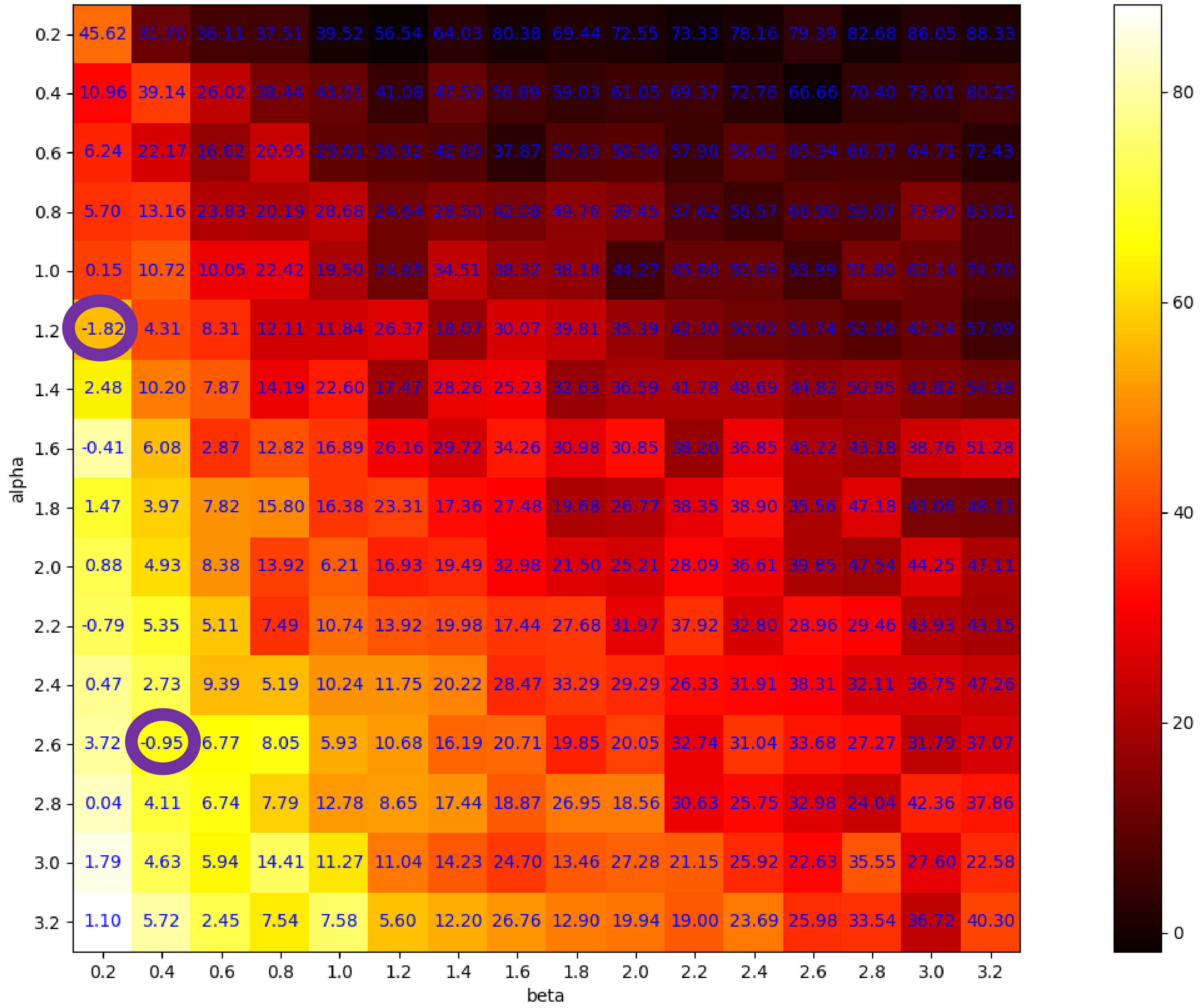


- 4、运行grid_search.py

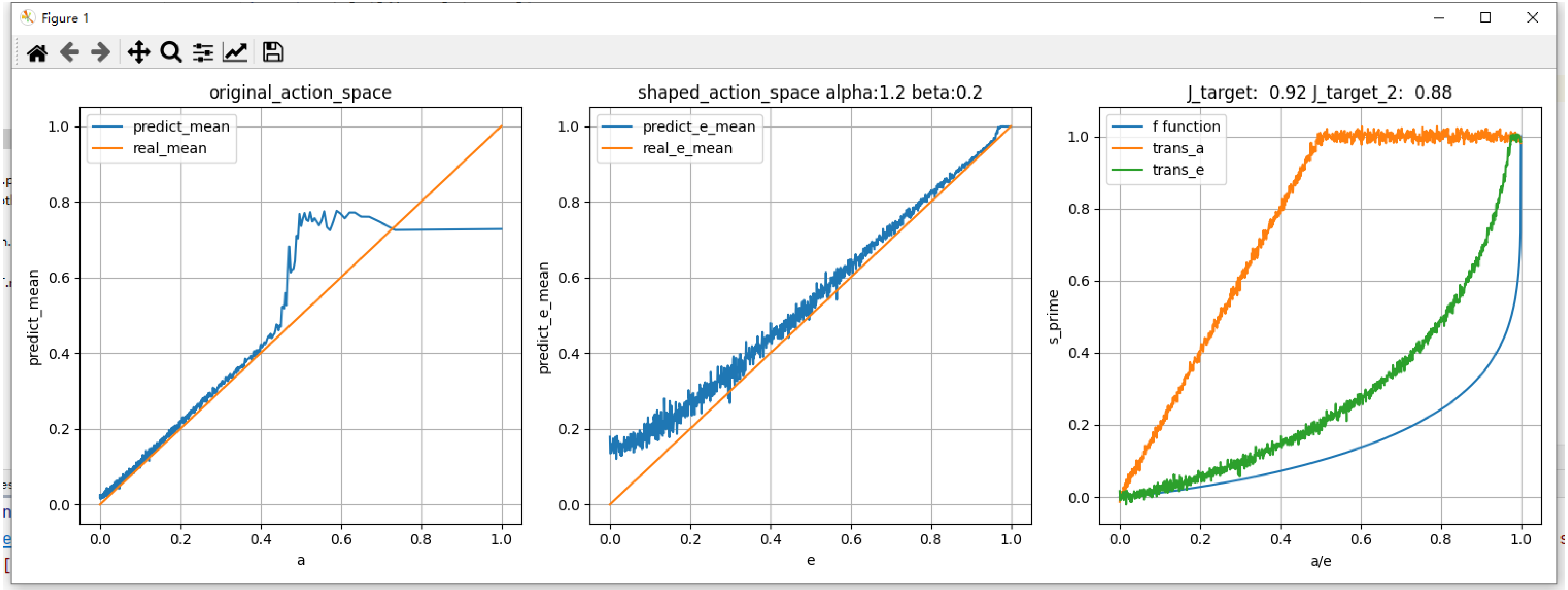
计算在alpha-beta取值不同参数的情况下的目标函数J的值，可以发现最小值在alpha=1.2,beta=0.2处取得

- 5、运行ar_train.py

利用简单的梯度方向进行迭代训练，目标函数逐渐下降，但是陷入局部最优值alpha=2.6, beta=0.4



- 6、运行test_inverse.py。查看不同参数下的f函数（最右图），以及均匀分布的e在不同的动作表征之后的状态转移（最右图），以及不同表征之后的预测误差



6、运行test_inverse.py。查看不同参数下的f函数（最右图），以及均匀分布的e在不同的动作表征之后的状态转移（最右图），以及不同表征之后的预测误差（中间图）