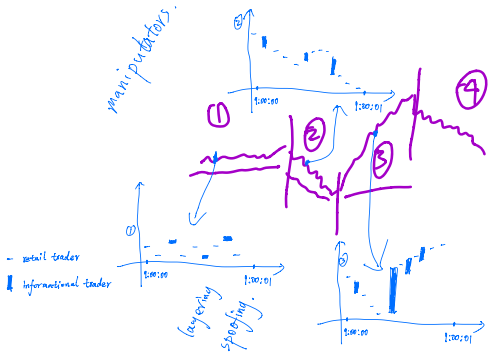
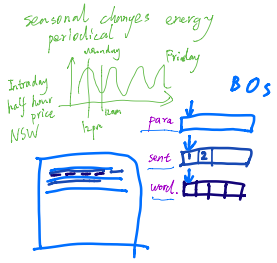
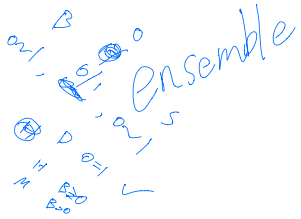


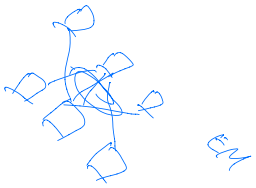


$\pi_{LPO}(\pi_{sg_1}(\pi_{sg_0}(M_t, H_t), D_T))$

$\pi_D \rightleftarrows \pi_m$

capable net layer.





与Hippo不同：0结构不同频率数据.

D
H
M

① 多层 subpolicy, subsubpolicy etc.

② disentangle

③ Option.

termination β. 是在sub-policy里. 还是oop里

④ Each level has own length of period

D: Null

M: 240/day

H: 4/day.

⑤ constraints on action space

DMN instead LLEP

MRFLSSVM.

Φ^H = DMN.

Φ^U Φ^P Φ^H

$\Phi \pi^M$ π^H π^0

EGarch volitility (duration)

Steven Energy Periodic.

(5) visiting prob? $r(\theta) = \frac{\pi_\theta ?}{\pi_{old} ?}$

(4) What happens when [done]?

How bound convertal in distribution?

L & U Bound.

① action space continuous. could action space be dynamic?

② actor_model : sampling
train_model : tarin_ops
} are these same model?

Yes. They are. re-used variables.

③ nsteps: sample steps? which is lstm steps?

① action & observation space.

obspace : determine input encode
discrete: one-hot - float
box: float.

box could be normalized

policies.py/build_policy( )
line 129 - 139

acspace : determine output distribution

policies.py / policywithvalue( )

common/distributions :
make_pdtype( ) → distribution( )

Categorical: catogorical

box: DiagGaussianPdType

policies.py

① PolicyWithValue.

__init__
self.state = constant

self.initial_state = None.

step

_evaluate [self.action, self.vf, self.state, self.neglogp]
sess.run

① LSTM difference?

add a break point to lstm's forward.
see when it's being called.
build_policy: policy-latent.

common/policies.py

PolicyWithValue.__init__( ):

line 39: self.__dict__.update(tensors)
↙ extra_tensors

policy_fn

extra_tensors {
rms 135 normalized things
recurrent_tensors 151
for lstm: { initial_state, S, M, state
}

(6) epoch? minibatch?

(7) value_net 'copy' 'share'

build_policy( ): policy_fn(,)

??? 165: todo vf not support recurrent net.

sample Advantage

$L^{VF} = V_\theta - V^{targ}$?
|| ?
496 V_θ - sampled_return.

415, $\text{sampled\_return}_t = V_t^{\theta old} + A_t^{\theta old}$

695, 698 $= r_t + \gamma V_{t+1}^{\theta old} + \gamma\lambda A_{t+1}^{\theta old}$
↑ return.

(8) How V^{targ} (return)

ppo2/runner.py line 65

mb_returns = mb_advs + mb_values.

$\text{return}_t = r_t + \gamma V(s_{t+1}) + \lambda\gamma\delta_{t+1} + \dots + (\lambda\gamma)^{T-t}\delta_T$

Settings :

if value_network : { None = 'share' (default) / 'copy' / callable }

by default share policy_net 's policy_latent

build_policy( ): policy_fn.

Notes:

policy has step( ) & value( )
↑ self.vf

policy_fn( ) → PolicyWithValue( )

receives X, value_network: callable

action is calculated by latent = policy_net(X)

action_distrib(fc(latent, nact)).sample( )

value is calculated by
vf_latent = { value_net(X) — value_net = 'copy' / callable ; policy_net(X) — value_net = 'shared' }

★ recurrent is not supported.

fc(vf_latent, 1)

policies.py 158-166

because vf_latent = _v_net(encoded_X).

_v_net { copy, lstm needs 'nenv' param ; callable, customized net need to handle nenv or nsteps }

How does build_policy models.py

nsteps = nbatch // nenv work?

nbatch = X.shape[0]

nbatch = nenv * nsteps = 6 × 20 = 120

models.py

M, S = placeholder

ms, xs

initial_state

nbatch_train > nsteps

$\frac{nenvs * ~~nsteps~~}{nmini} > ~~nsteps~~$

xs, ms list[X_{t_1}, ..., X_T]

[M_{t_1}, ... M_T]

Done ↓

④. mask : True / False

(1). In LSTM a2c/utils.py/ lstm( ):

cell state$_{t-1}$ / hidden state$_{t-1}$ * (1 - mask$_t$)

(2) In advs / returns ppo2/runner.py/run( )

nextnonterminal = 1 - self.dones. np.ndarray [1, nenv]

Sample:

Model. act-model differs from train_model in batch size & nsteps

parameters are reused.

model.step(X, States, M)

- X: [nenv, obs X-dim]
- States: [nenv, nlstm×2] self.dones
- M: [1, nenv] [(0/1), ...]

For sampling, nsteps always be 1.

① sample $a \sim \pi(a|X)$

1. build $p(a|X, \mu, \Sigma)$

build_policy/policy_fn( )

$\mu = fc_{\pi}(lstm(preproc(X), s, M))$

(Q) Σ_{π} = tf.get_var( ) 为何不以X为输入？不随X变？ 无偏估计. to be confirmed.

② estimate value $V(s_t)$

vf(s_t)

= fc_{vf}(lstm(preproc(X), s, M)) if copy

fc_{vf}(custom_value_net(X, S, M))

③ recurrent state: LSTM's state, not env's state

PolicyWithValue.__init__( ):

line 39: self.__dict__.update(tensors) ← extra_tensors

policy_fn

extra_tensors
- rms 135 normalized things
- recurrent_tensors 151
  - for lstm:
    - initial_state
    - S
    - M
    - state

state: concat [h_T, C_T]

Train:

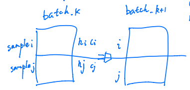
normalized advs:

$$advs = \frac{a - \bar{a}}{std(a) + 1e^{-8}}$$

Stateful LSTM

stateful: states from k batch will be used in k+1 batch matching index of samples

batch_k batch_k+1



Qs: 1. How blog frame batch (batch, timestep, feat_dim)

batch_input_size work in keras

2. LSTM size does not match input time_steps

it's cell size, not timesteps

KDD 20 Map

Imp { stable baseline, hippo, dev } by Sunday

Hippo Paper
GAE Book & Blogs.
↓
Option PPO + advantage

Stable baselines.

Init.

Model

Policy

Base RL Model

get policy

check env. { num, action, input space, VecEnv Ef.

ActorCritic Model

Setup_Model( ): nenv, n_steps, batch_size
act_model = policy(nenv, 1, nenv)
train_model = policy(nenv / nminibatch, n_steps, nenv * nsteps / nminibatch)

policies.py

q_value ( ) policies 430, distributions 242. q_value shape?
value_fn policies 428

<1> size
q_value: linear(rnn_output, '1', self.size) — n_actions
self._value_fn: linear(rnn_output, 'vf', 1)

self.neglogp: pd.neglogp(self.action)
self.value_flat: self.value_fn[:,0]
action: pd.sample
deterministic: pd.mode
-policy_proba: { Categorical, DiagGaussian: [mean, std]

distributions.py

self.action: pd_samples >
self.neglogp: self.pd.neglogp(action)
self.

Hippo

policy param.

$\pi(\theta)$

$\pi_{old}(\theta)$

Paths: → s, r, a, logp?
a = sample( )
s, r = Env(a..)

rllab/sampler/utils.py/rollout();

$\delta_t = r_t + r\delta_{t-1}$

$r_t = 9\,8\,7\,6\,5\,4\,3\,2\,1\,0$

$\delta_1 = r_1 + 0 \qquad 9$

$\delta_2 = r_2 + 0.5 \times 9 = 12.5$

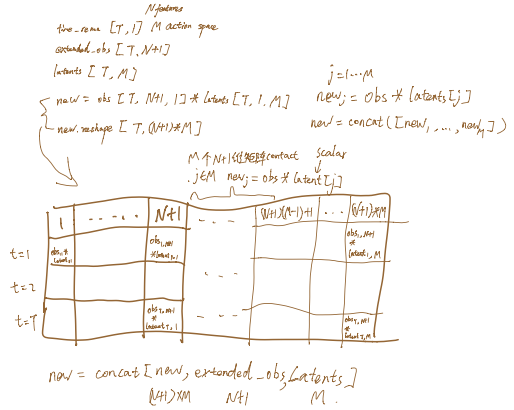
$\delta_3 = 7 + 0.5 \times 12.5 = 13.25$
6.25

$\delta_4 = 6 + 0.5 \times 13.25 = 12.625$
$6\ 6\ 2\ 5$

A: Base Sampler / process_samples:
A manager

Hier Batch Sampler / process_samples:
A_{skill}

hier_batch_sampler.py
skill_dependant_baseline
vf_skill



new = concat[new, extended_obs, latents]
(N+1)XM N+1 M

new [T, (N+1)XM + (N+1) + M]

vf_skill = vf_skill(new)

HIPPO

$\max \pi_\theta$ over $(obs, z, a)_{t=1 \dots T}$

$z_t = \pi^{manager}_{\theta^m_{old}}(obs_t) = \pi^m_{old} \ \theta^m_{old}$

$a_t = \pi^{skill}_{\theta^s_{old}}(obs, z_t) = \pi^s_{old}. \ \theta^s_{old}$

T total time-steps

H manager has been called H times $H = \{t_1, t_{15}, t_{32} \dots t_T\}$ (p = sample_period = 15)

P_h current skill total period $P_h \sim U(\text{period\_min}, \text{period\_max})$

P_h remaining time $p_h = \frac{P_h + t_h - t}{\text{max\_period}}$

$$Loss = \frac{\overset{E[1 \cdots H]}{L_{manager}}}{\text{mean(period)}} + \overset{E[1 \cdots T]}{L_{skill}}$$

$L_m = E[\frac{\pi^m_{new}}{\pi^m_{old}} A^m(s_t, z_t)] \quad t \in \mathcal{H}$

$= E[\exp(\log \pi^m_{new}(z_t, s_t) - \log \pi^m_{old}(z_t, s_t)) A^m(obs_t, z_t, a_t)] \quad t \in \mathcal{H}$

$\pi^m_{\theta^m_{old}}(z_t | s_t) \quad t \in \mathcal{H}$

$s_t = concat(obs_t, P_t)$

$z_t \sim \pi^m(z | s_t)$

neural_net.

$\pi^s_{\theta^s_{old}}(a_t | z_t, s_t) \quad t \in T$

$s_t = concat(obs_t, P_t)$

$a_t \sim \pi^s_{\theta^s_{old}}(a | z_t, s_t)$

$A^m_t(\tau) \quad \tau \sim (obs_t, z_t, a_t, r_t) \quad t \in \mathcal{H}$

for $t = 1:T$

$V^m_t = VF^m_{old}(obs_t)$

$\delta^m_t = r_t + gamma * V^m_{t+1} - V^m_t$

$A^m_t = \delta^m_t + \lambda * gamma * A^m_{t+1}$

$R^m_t = r^m_t + gamma * R^m_{t+1}$

$A^s_t(\tau) \quad \tau \sim (obs_t, z_t, a_t, r_t) \quad t \in T$

for $t = 1:T$

$s_t = concat[obs_t, P_t]$

$V^s_t = VF^s_{old}(concat[concat[s_t * w \text{ for } w \text{ in } z_t], s_t, z_t])$

$\delta^s_t = r_t + gamma * V^s_{t+1} - V^s_t$

$A^s_t = \delta^s_t + \lambda * gamma * A^s_{t+1}$

$R^s_t = R^m_t$

VF^m, VF^s

obj $\min (V_t \sim VF - R_t)^2$

In ppo with $(V - V_{targ})^2$
↑
changed every policy gradient iteration

Both hippo, ppo,

action : unchanged.

A : unchanged.

old_vpred_ph : unchanged . only used for clip_vf.

Implement normalize : obs . return, advantages.

② for recurrent_fs, time_remaining is not needed for low_policy hier_base.py, hier_batch_sampler.py

③ vf_manager, vf_skill fitted during sample period (process_samples( ))

should fit expected return

rather than reward directly.

Hier Options.

Markov Process.

Semi-MP

Off policy?? Sample Efficiency.

Mult Fractal Theory?

value, policy as GAN

GMM Graph HMM.

sutton 2000 $\frac{\partial Q}{\partial \theta}$ A？如何用，为何减var

O

[35] 92 $P(\cdot)R_t$

GAE 可否融合 2000, 92?

TRPO

$$\eta(\pi) = E_{s_0, a_0, \ldots}\left[\sum_{t=0}^{\infty} \gamma^t r(s_t)\right]$$

$$\eta(\pi_{new}) = \eta(\pi_{old}) + \sum_s \rho_{\pi_{new}}(s) \sum_a \pi_{new}(a|s) A_\pi(s,a)$$

Appendix A lemma1.
22, 23, 24

$t=0$ $\gamma V_1 - V_0$ $-\eta(\pi) = -E_{s_0}[V_\pi(s_0)]$

$t=1$ $\gamma V_2 - V_1$

$t=2$ $\gamma V_3 - V_2$.

$(\gamma-1)V_1$.

$$P(s_{t+1}, w_{t+1} \mid s_t, w_t) = \sum_a \pi_{w_t}(a \mid s_t)\, P(s_{t+1} \mid s_t, a)$$

$$\pi_{w_t}(a \mid s_t) = \sum_{\beta_t} P(\beta_t \mid w_t, s_t)\, P(a \mid s_t, \beta_t)$$

$$P(a \mid s_t, \beta_t) = \left[(1-\beta_{w_t}(s_{t+1}))\, \mathbb{1}_{w_t = w_{t+1}} + \beta_{w_t}(s_{t+1})\, \pi_{w_{t+1}}(a \mid s_{t+1})\right]$$

$$\beta_t \in \{0, 1\}$$

For option w_t

$$V1: \quad \sum_{\beta_t} P(\beta_t \mid w_t, s_{t+1})\, P(w_{t+1} \mid s_{t+1}, \beta_t)$$

$$P(w_{t+1} \mid s_{t+1}, \beta_t) = P(w_{t+1} \mid s_{t+1})^{\beta_t}$$

$$\sum_{\beta_t} \left[\mu_{\beta_t}(s_{t+1})\, P(w_{t+1} \mid s_{t+1})\right]^{\beta_t}$$

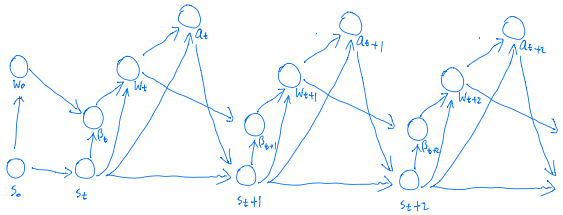
$$V2: \quad \sum_{\beta_t} P(\beta_t \mid w_t, s_{t+1})\, P(w_{t+1} \mid s_{t+1}, \beta_t)$$

$$P(w_{t+1} \mid s_{t+1}, \beta_t) = P(w_{t+1} \mid s_{t+1})^{\beta_t}$$

$$P(\beta_t \mid w_t, s_{t+1}) = \prod_{w_k} P(\beta_t \mid s_{t+1})^{w_k} \quad W \in R^{|\mathcal{O}|} \quad w \in \{0, 1\}^{|\mathcal{O}|}$$

$|\mathcal{O}|$ options

$$\sum_{\beta_t} \left[\prod_{w_k}^{k=1\ldots|\mathcal{O}|} \mu_k(s_{t+1})^{w_k}\, P(w_{t+1} \mid s_{t+1})\right]^{\beta_t}$$



$$P(s_0) P(s_1 \mid s_0) \qquad P(s_{t+1} \mid s_t, a_t)\, P(\beta_{t+1} \mid s_t, w_t)\, P(w_{t+1} \mid s_{t+1}, \beta_{t+1})\, P(a_{t+1} \mid w_{t+1}, s_{t+1})$$

$$P(s_0, w_0, s_1, \beta_1, w_1, a_1, \ldots, s_t, \beta_t, w_t, a_t)$$

$$= P(w_0 \mid s_0) \prod_{k=0}^{\infty} P(w_{k+1} \mid s_{k+1}, \beta_{k+1})\, P(a_{k+1} \mid w_{k+1}, s_{k+1})\, P(\beta_{k+1} \mid s_k, w_k)\, P(s_0)\, P(s_1 \mid s_0) \prod_{t=1}^{\infty} P(s_{t+1} \mid s_t, a_t)$$

$$P(s_0, s_1, a_1, \ldots, s_t, a_t) = \sum_{\beta} \sum_{w} P(s_0, w_0, s_1, \beta_1, w_1, a_1, \ldots, s_t, \beta_t, w_t, a_t)$$

$$= \sum_{w} \sum_{\beta} P(w_0 \mid s_0) \prod_{k=0}^{\infty} P(w_{k+1} \mid s_{k+1}, \beta_{k+1})\, P(a_{k+1} \mid w_{k+1}, s_{k+1})\, P(\beta_{k+1} \mid s_k, w_k)\, P(s_0)\, P(s_1 \mid s_0) \prod_{t=1}^{\infty} P(s_{t+1} \mid s_t, a_t)$$

$$= \sum_{w} P(w_0 \mid s_0) \prod_{k=0}^{\infty} \left[P(w_{k+1} \mid s_{k+1}, \beta_{k+1})\, P(a_{k+1} \mid w_{k+1}, s_{k+1}) \sum_{\beta} P(\beta_{k+1} \mid s_k, w_k)\right] \left[P(s_0)\, P(s_1 \mid s_0) \prod_{t=1}^{\infty} P(s_{t+1} \mid s_t, a_t)\right]$$

option - crictic

Jupyter.

temporal abstraction.

temporal in Dation time

not in obs space time

Poorway problem

房间 → 房间. temporal abstraction

abstract 掉的是 action

但房间本身没有 temporal 关系.

学习的是一个由多时间点完成的技能

但是技能间没有 temporal relationship.

技能内部也没有.

房间 → 房间 与 股票交易区别?

有无区别?

就他人没有.