Rapport de laboratoire du TP3

Présenté à

Alexandre Piché

Par

Justine Pepin 1789244

Gr. 1 de laboratoire

Dans le cadre du cours de I.A. : tech. probabilistes et d'apprentissage

INF8225

18 mars 2018

PARTIE I – GRADIENT DE LA POLITIQUE

Toutes les formules et les notions théoriques utilisées pour répondre à ces questions ont été obtenues dans le manuel suggéré aux sections 5 et 13. (http://incompleteideas.net/book/bookdraft2018jan1.pdf)

a) Dérivation des gradients pour une politique gaussienne

$$\nabla_{\theta\mu} = \ln \pi(a|s;\theta)$$
$$\nabla_{\theta\mu} = \frac{\nabla_{\theta\mu}\pi(a|s;\theta)}{\pi(a|s;\theta)}$$

$$\nabla_{\theta\sigma} = \ln \pi(a|s;\theta)$$
$$\nabla_{\theta\sigma} = \frac{\nabla_{\theta\sigma}\pi(a|s;\theta)}{\pi(a|s;\theta)}$$

$$\pi(a|s;\theta) = \frac{1}{\sigma(s,\theta)\sqrt{2\pi}} e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}$$
$$\mu(s,\theta) = \theta_\mu^T s_\mu$$
$$\sigma(s,\theta) = e^{(\theta_\sigma^T s_\sigma)}$$

Pour $\nabla_{\theta\mu}$:

$$\nabla_{\theta\mu} = \frac{\nabla_{\theta\mu}\left(\frac{1}{\sigma(s,\theta)\sqrt{2\pi}} e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}\right)}{\frac{1}{\sigma(s,\theta)\sqrt{2\pi}} e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}}$$

$$\nabla_{\theta\mu} = \frac{\nabla_{\theta\mu}\left(e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}\right)}{e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}}$$

$$\nabla_{\theta\mu} = \frac{e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)} \nabla_{\theta\mu}\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}{e^{\left(\frac{(a-\mu(s,\theta))^2}{2\sigma^2(s,\theta)}\right)}}$$

$$\nabla_{\theta\mu} = \frac{1}{2\sigma^2(s,\theta)} \nabla_{\theta\mu}\left((a-\mu(s,\theta))^2\right)$$
$$\nabla_{\theta\mu} = \frac{2(a-\mu(s,\theta))}{2\sigma^2(s,\theta)} \nabla_{\theta\mu}(-\mu(s,\theta))$$

$$\nabla_{\theta\mu} = \frac{\big(a - \mu(s,\theta)\big)s}{\sigma^2(s,\theta)}$$

Pour $\nabla_{\theta\sigma}$:

$$\nabla_{\theta\sigma} = \frac{\frac{1}{\sqrt{2\pi}}\nabla_{\theta\sigma}\left(\frac{1}{\sigma(s,\theta)}e^{\left(\frac{\left(a-\mu(s,\theta)\right)^2}{2\sigma^2(s,\theta)}\right)}\right)}{\frac{1}{\sqrt{2\pi}}\frac{1}{\sigma(s,\theta)}e^{\left(\frac{\left(a-\mu(s,\theta)\right)^2}{2\sigma^2(s,\theta)}\right)}}$$

$$\nabla_{\theta\sigma} = \frac{\frac{1}{\sigma(s,\theta)}e^{\left(\frac{\left(a-\mu(s,\theta)\right)^2}{2\sigma^2(s,\theta)}\right)}\nabla_{\theta\sigma}\left(\frac{\big(a-\mu(s,\theta)\big)^2}{2\sigma^2(s,\theta)} - \theta_\sigma^T s_\sigma\right)}{\frac{1}{\sigma(s,\theta)}e^{\left(\frac{\left(a-\mu(s,\theta)\right)^2}{2\sigma^2(s,\theta)}\right)}}$$

$$\nabla_{\theta\sigma} = \frac{\big(a-\mu(s,\theta)\big)^2}{2}\nabla_{\theta\sigma}\left(\frac{1}{\sigma^2(s,\theta)}\right) - s$$

$$\nabla_{\theta\sigma} = \frac{-\big(a-\mu(s,\theta)\big)^2}{\sigma^2(s,\theta)}s - s$$

$$\nabla_{\theta\sigma} = -\left(\frac{\big(a-\mu(s,\theta)\big)^2}{\sigma^2(s,\theta)} + 1\right)s$$

b) Gradient de la politique avec les retours suivants

Retour de Monte Carlo

$$\theta = \theta + \alpha\gamma^t \sum_{i=0}^{T} \big(\gamma^i r_{t+i}\big)\, \nabla_\theta \ln \pi(a_t|s_t;\theta)$$

Retour de Monte Carlo et fonction d'utilité comme variable de contrôle

$$\theta = \theta + \alpha\gamma^t \sum_{i=0}^{T} \big(\gamma^i r_{t+i} - V(s_t)\big)\, \nabla_\theta \ln \pi(a_t|s_t;\theta)$$

Retour estimé

$$\theta = \theta + \alpha\gamma^t(r_t - V(s_{t+1}))\nabla_\theta \ln \pi(a_t|s_t;\theta)$$

Autre Politique

$$\theta = \theta + \alpha\gamma^t(r_t - V(s))\nabla_\theta \ln \pi(a_t|s_t;\theta)$$

$$V(s) \cong \frac{\sum_{t\in\tau(s)} \rho_{t:T(t)-1} G_t}{|\tau(s)|}$$

$$\rho_{t:T(t)-1} \cong \prod_{k=t}^{T-1} \frac{\varphi(a_t|s_t)}{\pi(a_t|s_t)}$$

Pour cette autre politique, on doit utiliser le ratio d'*importance-sampling* entre les deux politiques afin d'utiliser le retour de Monte Carlo pour estimer l'utilité d'un état. Dans le calcul de l'utilité, on tient compte de l'ensemble $\tau(s)$ des temps auxquels on a visité chacun des états visités à ce jour.

c) Commentaires sur la variance des gradients obtenus en b)

En général, le retour de Monte Carlo a toujours une variance assez haute. Utiliser une variable de contrôle qui correspond à l'utilité de l'état permet de réduire un peu la variance, parce que les états qui sont moins avantageux par rapport à d'autres états auront un pointage négatif et ceux qui seront nettement plus avantageux auront un pointage positif tout de même plus élevé que les autres états avantageux. Lorsqu'on utilise l'estimation avec Monte Carlo, on associe la récompense de l'état présent à l'utilité du prochain état (*bootstrapping* aux états à venir). Ainsi, on fait l'introduction d'un biais qui réduit la variance tout en accélérant l'apprentissage. C'est donc le meilleur choix des quatre façons d'obtenir le gradient de la politique qui sont présentées ici. Finalement, utiliser le retour Monte Carlo et le ratio d'*importance-sampling* avec une autre politique est un très mauvais choix pour réduire la variance puisque celle-ci est infinie (*unbounded*) étant donné que le ratio entre les deux politiques l'est.

PARTIE II – APPRENTISSAGE AVEC DOUBLES RÉSEAUX Q

Les pseudocodes suivis pour l'implémentation de Double DQN et Double Dueling DQN sont disponibles dans les articles recommandés suivants :

[1] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: arXiv preprint arXiv:1312.5602 (2013).

[2] Hado Van Hasselt, Arthur Guez, and David Silver. "Deep Reinforcement Learning with Double Q-Learning." In: AAAI. Vol. 16. 2016, pp. 2094–2100.

[3] ZiyuWangetal. "Dueling network architectures for deep reinforcement learning".In:arXiv preprint arXiv:1511.06581 (2015).

LISTING 1 : Agent qui implémente Double DQN et Double Dueling DQN

```
class Agent(object):
    def __init__(self, gamma=0.99, batch_size_=128):
        self.target_Q = DQN()
        self.Q = DQN()
        self.gamma = gamma
        self.duelling = True
        self.batch_size_ = batch_size_
        self.lr = 0.001
        hard_update(self.target_Q, self.Q)
        self.optimizer = torch.optim.Adam(self.Q.parameters(), self.lr)

    def act(self, x, epsilon_=0.1):
        if random.random() < epsilon_:
            return
Variable(torch.from_numpy(np.array([env.action_space.sample()])).type(t
orch.LongTensor))
        else:
            value, indice = torch.max(self.Q.forward(x), 0)
            return indice

    def backward(self, transitions):
        my_batch = Transition(*zip(*transitions))
        state_batch = Variable(torch.cat(my_batch.state))
        next_state_batch = Variable(torch.cat(my_batch.next_state))
        action_batch = Variable(torch.cat(my_batch.action))
        reward_batch = Variable(torch.cat(my_batch.reward))
        done_batch = Variable(torch.cat(my_batch.done))
        state_action_values = self.Q(state_batch).gather(1,
action_batch.view(self.batch_size_, 1))

        mask = np.logical_not(done_batch.data.numpy()) *
np.ones(self.batch_size_)
        mask = Variable(torch.from_numpy(mask).type(torch.FloatTensor))
        if self.duelling is False:
            next_state_action_values =
[self.target_Q.forward(Variable(j)).max().data for j in
```

```
my_batch.next_state]
            next_state_action_values =
Variable(torch.cat(next_state_action_values))
        else:
            next_state_actions = [self.Q.forward(Variable(j)).data for
j in my_batch.next_state]
            values, next_state_actions =
Variable(torch.cat(next_state_actions)).detach().max(1)
            next_state_action_values =
self.target_Q(next_state_batch).gather(
                1, next_state_actions.view(self.batch_size_, 1)).view(-
1).detach()

        expected_state_action_values = next_state_action_values *
self.gamma * mask + reward_batch
        loss = F.smooth_l1_loss(state_action_values,
expected_state_action_values)
        self.optimizer.zero_grad()
        loss.backward()
        self.optimizer.step()
        soft_update(self.target_Q, self.Q, self.lr)
```

Figure 1 : Total des récompenses obtenues à chaque cycle avec Double DQN

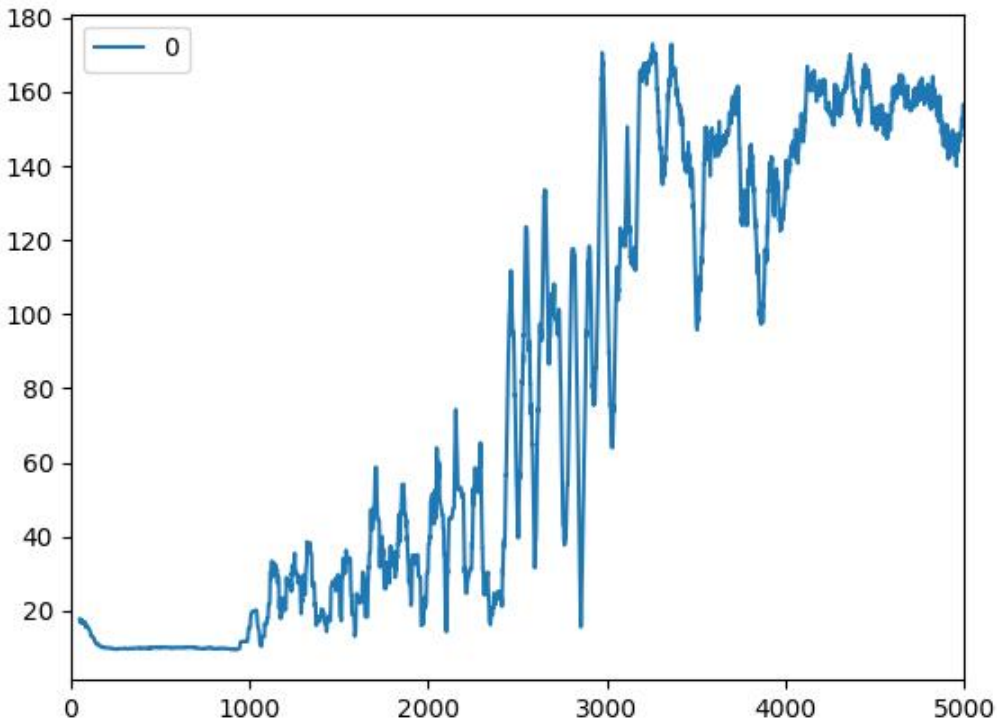

Figure 2 : Total des récompenses obtenues à chaque cycle avec Double Dueling DQN