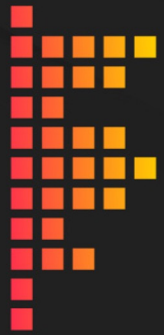


Reinforcement learning

Proximal Policy Optimization

otus.ru

Проверить, идет ли запись

Меня хорошо видно && слышно?

Ставим “+”, если все хорошо
“-”, если есть проблемы

Правила вебинара

Активно
участвуем

Off-topic обсуждаем
в учебной группе

Задаем вопрос
в чат

Вопросы вижу в чате,
могу ответить не сразу

Условные обозначения

Индивидуально

Время, необходимое
на активность

Пишем в чат

Говорим голосом

Документ

Ответьте себе или
задайте вопрос

Тема вебинара

Reinforcement learning

Proximal Policy Optimization

Игорь Стурейко

Руководитель курсов: Reinforcement Learning, ML Professional, ML Basic, MLOps, FinML

Teamlead, главный инженер проекта,
Физический факультет МГУ, PhD теоретическая физика

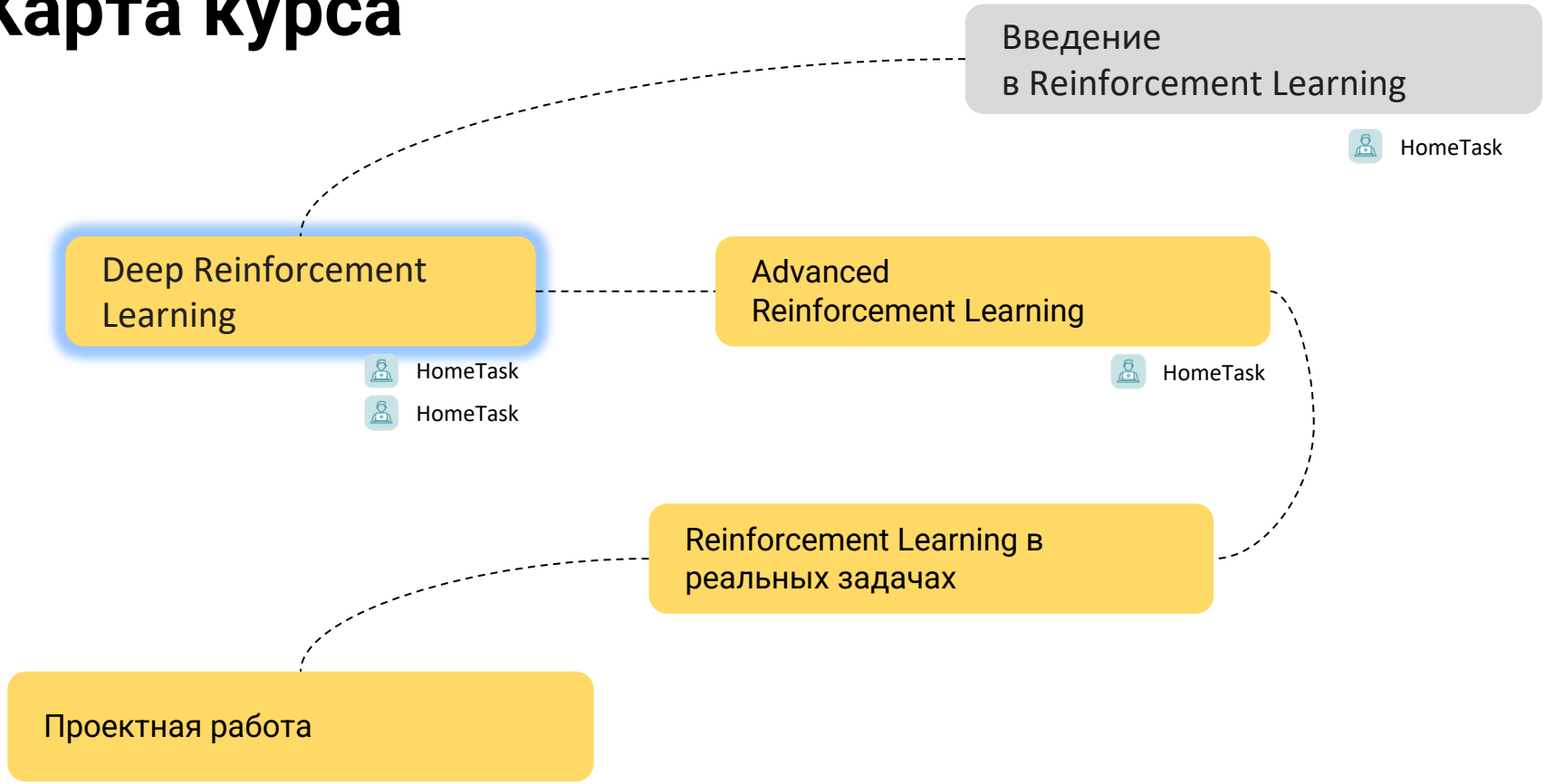
Опыт:
Более 15 лет занимался прикладной математикой и мат моделированием (Data Scientist) (Python, C++) в НИИ ПАО Газпром

@stureiko (TG)

LinkedIn: igor-stureiko

@rl_fintech (Мой канал посвященный финансовым моделям)

Карта курса


Введение
в Reinforcement Learning
HomeTask
Deep Reinforcement
Learning
HomeTask
HomeTask
Advanced
Reinforcement Learning
HomeTask
Reinforcement Learning в
реальных задачах
Проектная работа

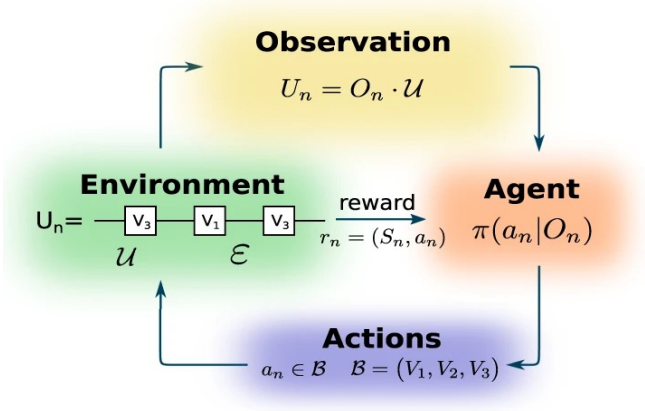
Программа курса

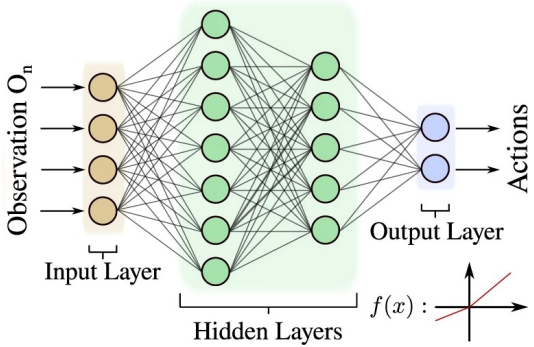
Deep Reinforcement Learning

- Введение в Deep Reinforcement Learning
- Deep Q-Network (DQN) алгоритм  HomeTask
- Policy Gradient (PG) алгоритм
- Actor-Critic алгоритм  HomeTask
- TRPO -> PPO
- DDPG -> TD3 -> LSTM-TD3

Маршрут вебинара

- Знакомство
- Постановка задачи
- Policy Gradient (PG)
- Natural Policy Gradient (NPG)
- Trust Region Policy Optimization (TRPO)
- Proximal Policy Optimization (PPO)
- Алгоритм PPO





Цели вебинара

К концу занятия вы сможете

1. Понимать пути улучшения алгоритма Policy Gradient
2. Понять сложности прямого вычисления расстояния Куллбека-Лейбнера
3. Понять схему алгоритма Trust Region Policy Optimization (TRPO)
4. Понять схему алгоритма Proximal Policy Optimization (PPO)
5. Реализовать алгоритм Proximal Policy Optimization в коде на Python
6. Сравнить собственную реализацию с работой фреймворка StableBaselain3

Смысл

Зачем вам это уметь

1. Программировать задачи обучения с подкреплением для окружений большой размерности и непрерывного пространства действий
2. Понимать работу алгоритма PPO и использовать его в своих проектах

Алгоритм PPO

PG -> NPG -> TRPO -> PPO

Модели

Обновление весов модели:

$$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) G_t$$ - REINFORCE

$$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) F_t$$ - Advantage REINFORCE

$$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) Q(S_t, A_t)$$ - Actor-Critic

$$\eta \leftarrow \eta - \alpha \nabla_\eta \ln \pi^\eta(A_t|S_t) (R_{t+1} + \gamma V(S_{t+1}) - V(S_t))$$ - Advantage Actor-Critic

$$\eta \leftarrow \eta - \alpha \nabla_\eta \left(\frac{1}{N} \sum_{i=1}^{N} Q^\theta(s_i, \pi^\eta(s_i)) \right)$$ - DDPG (Deep deterministic Policy Gradient)

Policy Gradient (PG)

Политика π - это просто функция, которая возвращает действие a при заданном состоянии s.

Функция приближающая π определяется набором весов θ.

Изменяя эти веса и наблюдая разницу в наградах мы обновляем θ в том направлении, которое дает большую награду. Этот механизм лежит в основе всех методов градиента политики.

$$J(\theta) = \mathbb{E}_{\tau \sim \pi_\theta} R(\tau) = \sum_{\tau} P(\tau, \theta) R(\tau)$$

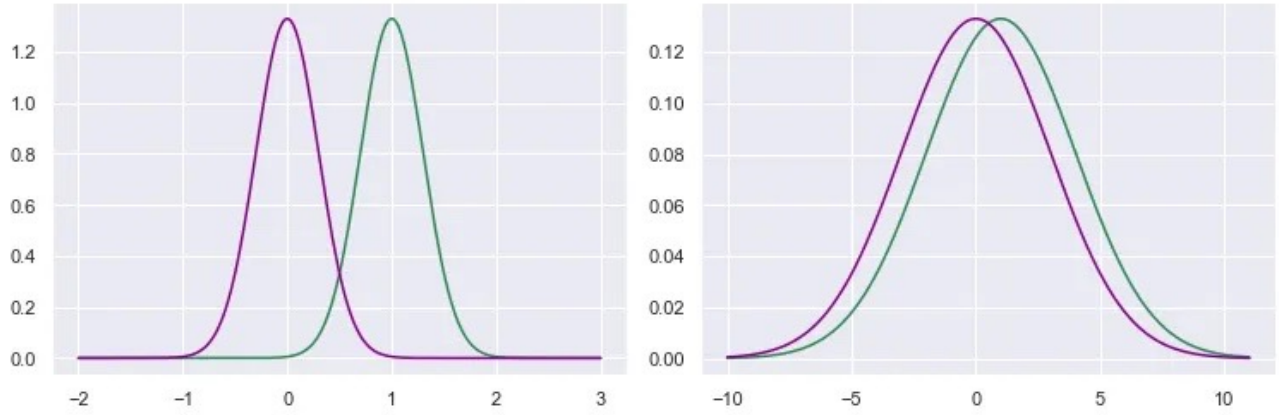
$$\theta \leftarrow \theta + \alpha \nabla_\theta J(\theta)$$

Policy Gradient (PG) - проблемы

Хотя у нас есть вознаграждение и направление обновления, мы не знаем, на какую величину обновлять политику. Выбор правильной скорости обучения α представляет собой гиперпараметр задачи.

Мы можем вычислить только евклидово расстояние для обновления.

Пример – одинаковое евклидово расстояние между моделями, но изменения в политике – разные.



Шаг изменения политики

Движение на одинаковую величину по горизонтали может привести к кардинальному изменению.

Если рельеф воспринимать как функцию политики, то необходимо оценивать насколько изменится политика, т.е фактически вычислять все возможные изменения политики и после этого делать шаг.

Фактически нам нужно двигаться следуя за максимумом функции политики. Это достигается методами оптимизации второго порядка.

Natural Policy Gradient (NPG)

KL-расстояние (Куллбека-Лейбнера):

$$\mathcal{D}_{KL}(\pi_\theta||\pi_{\theta+\Delta\theta}) = \sum_x \pi_\theta(x) \log\left(\frac{\pi_\theta(x)}{\pi_{\theta+\Delta\theta}(x)}\right) \Rightarrow$$

$$\Delta\theta^* = \arg \max_{\mathcal{D}_{KL}(\pi_\theta||\pi_{\theta+\Delta\theta})\leq\varepsilon} J(\theta + \Delta\theta)$$

С учетом этого ограничения мы можем гарантировать, что изменяем параметры таким образом, что политика изменится не слишком сильно.

Вычисление KL- расстояния требует оценки всех пар "состояние - действие", поэтому для работы с реалистичными задачами RL необходимы упрощения.

Natural Policy Gradient (NPG) - ограничения

- Чтобы не допустить слишком большого отклонения политики, накладываются ограничения на KL- расстояние между новой и старой политикой.
- Используя метод разложения Лагранжа, это ограничение преобразуется в разность.
- Поскольку мы не можем напрямую вычислить KL-расстояние, используется разложение Тейлора в качестве аппроксимации для схемы обновления весов.
- Для малых изменений параметров KL- расстояние аппроксимируется с помощью информационной матрицы Фишера, для которой есть легко вычислимое выражение.
- Вся аппроксимация является результатом, предполагающим $\theta \cong \theta_{old}$. Таким образом, все обоснование справедливо только для небольших изменений политики.

Natural Policy Gradient (NPG)

$$\Delta\theta = \sqrt{\frac{2\varepsilon}{\nabla J(\theta)^T F(\theta)^{-1} \nabla J(\theta)}} F(\theta)^{-1} \nabla J(\theta)$$

- Градиент "корректируется" обратной матрицей Фишера, учитывающей чувствительность политики к локальным изменениям. Матрица Фишера учитывает Риманову кривизну.
- Размер скорости обучения имеет динамическое выражение, которое адаптируется к градиентам и обеспечивая изменение политики на величину ε независимо от параметризации.

Natural Policy Gradient (NPG) - проблемы

- Разложение Тейлора дает локальную аппроксимацию до второго порядка. В связи с этим оцениваемый гессиан может не быть положительно определенным. На практике естественные градиентные методы являются численно хрупкими и не всегда дают стабильные результаты.
- Информационная матрица Фишера занимает пространство $|\theta| \cdot |\theta|$. Рассмотрим нейронную сеть со 100 000 параметрами – можно ожидать, что 10-миллиардная матрица на ноутбуке не вычислится. Кроме того, вычисление обратной матрицы – операция сложности $O(N^3)$, что довольно сложно для реальных задач. Таким образом, для глубоких методов RL NPG обычно выходят за пределы как памяти, так и вычислительных возможностей.
- NPG не проверяет, действительно ли обновление весов дает улучшение политики, а дает оценку новой политики и направление изменения весов.

Trust Region Policy Optimization (TRPO)

На каждом шаге мы оцениваем область «незначительного» изменения политики и для каждой области вычисляем оптимальное изменение политики.

Для вычисления области «незначительного» изменения политики нам все равно нужно вычислить расстояние Куллбека-Лейбнера, т.е. вычислять матрицу Фишера.

НО нам уже не нужно вычислять обратную к ней!

Trust Region Policy Optimization (TRPO)

Введем суррогатное преимущество:

$$J(\pi_{\theta+\Delta\theta}) - J(\pi_\theta) \approx \mathbb{E}_{s\sim\rho_{\pi_\theta}}\left[\frac{\pi_{\theta+\Delta\theta}(a|s)}{\pi_\theta(a|s)} A^{\pi_\theta}(s,a)\right] = \mathcal{L}_{\pi_\theta}(\pi_{\theta+\Delta\theta})$$

Ограничим ошибку аппроксимации суррогатного преимущества с помощью наихудшего KL-расстояния между политиками до и после обновления:

$$\Delta\theta^* = \arg\max_{\Delta\theta} \mathcal{L}_{\pi_\theta}(\pi_{\theta+\Delta\theta}) - \beta\mathcal{D}_{KL}^{\max}\ (\pi_\theta||\pi_{\theta+\Delta\theta})$$

Trust Region Policy Optimization (TRPO) - проблемы

- TRPO все равно не справляется с большими матрицами Фишера, даже если их не нужно инвертировать.
- Оптимизация второго порядка выполняется медленно - практическая реализация TRPO основана на ограничениях и требует вычисления матрицы Фишера.
- Мы не можем воспользоваться преимуществами стохастических градиентных оптимизаторов (первого порядка), таких как ADAM.
- TRPO достаточно трудно объяснить, реализовать и отладить. Когда обучение не дает желаемых результатов, бывает сложно определить в чем проблема.

Proximal Policy Optimization (PPO)

$$\Delta\theta^* = \arg\max_{\Delta\theta} \mathcal{L}_{\pi_\theta}(\pi_{\theta+\Delta\theta}) - \beta \mathcal{D}_{KL}^{\max}\ (\pi_\theta || \pi_{\theta+\Delta\theta})$$

Не будем вычислять теоретическое значение β. Зададим "целевое расхождение" δ, такое, чтобы наши обновления находились где-то в окрестности этого расхождения.

$$\frac{2}{3}\delta \leq \mathcal{D}_{KL}(\pi_\theta || \pi_{\theta+\Delta\theta}) \leq \frac{3}{2}\delta$$

Целевое расхождение должно быть достаточно большим, чтобы существенно изменить политику, но достаточно малым, чтобы обновления были стабильными.

После каждого обновления политики PPO проверяет β. Если реализованное расхождение превышает целевое расхождение более чем на 1,5, то на следующей итерации мы устанавливаем $\beta \to 2\beta$. В обратном случае, если обновления слишком малы, мы уменьшаем вдвое $\beta \to \frac{1}{2}\beta$.

$$\left(\beta \to \frac{1}{2}\beta\right):\ \frac{2}{3}\delta \leq \mathcal{D}_{KL}(\pi_\theta || \pi_{\theta+\Delta\theta}) \leq \frac{3}{2}\delta:\ (\beta \to 2\beta)$$

Практика

Практика

1. Реализовать алгоритм PPO
2. Посмотреть на применение алгоритма PPO из фреймворка StableBaseline3

Список материалов для изучения

1. PPO algorithm
2. PPO from scratch with PyTorch
3. Reinforcement Learning frameworks
4. Список ресурсов
5. Лю Ю. (Х.) Обучение с подкреплением на PyTorch: сборник рецептов / пер. с англ. А. А. Слинкина. – М.: ДМК Пресс, 2020. – 282 с.
6. Недостатки/неудачи обучения с подкреплением
7. Обзор алгоритмов и их недостатков
8. Особенности основных алгоритмов
9. Эволюционные стратегии в RL

Вопросы?

Ставим “+”,
если вопросы есть

Ставим “–”,
если вопросов нет

Рефлексия

Цели вебинара

1. Понимать пути улучшения алгоритма Policy Gradient
2. Понять сложности прямого вычисления расстояния Куллбека-Лейбнера
3. Понять схему алгоритма Trust Region Policy Optimization (TRPO)
4. Понять схему алгоритма Proximal Policy Optimization (PPO)
5. Реализовать алгоритм Proximal Policy Optimization в коде на Python
6. Сравнить собственную реализацию с работой фреймворка StableBaselain3

Заполните, пожалуйста, опрос о занятии по ссылке в чате

Спасибо за внимание!

Следующие вебинары

DDPG -> TD3 -> LSTM-TD3

Игорь Стурейко

Руководитель курсов: Reinforcement Learning, ML Professional, ML Basic, MLOps, FinML

Teamlead, главный инженер проекта,
Физический факультет МГУ, PhD теоретическая физика

Опыт:
Более 15 лет занимался прикладной математикой и мат моделированием (Data Scientist) (Python, C++) в НИИ ПАО Газпром

@stureiko (TG)

LinkedIn: igor-stureiko

@rl_fintech (Мой канал посвященный финансовым моделям)