BlackBox Challenge

5vision

5 июня 2016 г.

1 Теория RL

Поиск оптимальной стратегии был сделан с использованием классического подхода в RL - итерацией по стратегиям (policy iteration) [1]. Этот подход состоит из двух шагов: оценивание текущей стратегии и улучшение стратегии. Оценивание трактуется как задача прогнозирования для функции ценности, в данном случае, функции ценности действий. Прогнозирование осуществляется на основе накопленного при следовании текущей стратегии опыта. Совершенствование стратегии осуществляется за счет выбора жадного действия в каждом состоянии. На этапе оценивания стратегии, также была использована т.н. самонастройка (bootstrap) - корректировка оценок ценностей на основе других оценок ценностей.

Для решения задачи прогнозирования (значений функции ценности действий) нами была использована аппроксимация функции. Причем был протестирован как линейный метод, так и многослойная сеть. Все наши эксперименты сводились к проверке различных взаимодействий между аппроксимацией функции, самонастройкой и использование интегрированной/раздельной (on-policy/off-policy) оценкой ценности стратегии.

2 Практика BB

Для применения описанного подхода к BB были использованы следующие возможности:

a) для обучения в большинстве случаев брались только те действия, которые приводили к изменению 35 фичи (нумерация с 0)

b) в момент тестирования при значении 35 фичи > 0.5 делалось действие 2, а при значении $<$ -0.5 делалось действие 1

c) с использованием checkpoint на каждом шаге определялись награды для всех 4-х действий

d) было сделано online-обучение на проверяемом уровне.

Для описания итогового подхода введем следующие обозначения:

- a - номер действия (0,1,2,3)
- $S(t)$ - состояние в момент времени t (номера фич с 0 до 35)
- $Q(S(t), a)$ - функция ценности действий
- $Y(S(t), a)$ - целевое значение для Q-функции
- $Score(t)$ - очки в момент времени t
- N - количество шагов для раскручивания (rollout) текущей политики
- p_c - вероятность добавления любого действия в обучающую выборку (c - curriculum)
- эпоха - один проход тренировочного уровня.

Далее будет описано создание обучающей выборки и обучение нескольких моделей. Каждая из моделей прогнозирует Q-значения, и итоговое действие a в момент времени t определяется из суммы выходов всех моделей.

2.1 Формирование обучающей выборки

Здесь приводится описание алгоритма по созданию обучающей выборки в момент прохождения эпохи. В первую эпоху действия выполняются случайно, начиная со второй эпохи действия выбираются в соответствии с жадной политикой $\text{argmax}_a\, Q(S(t), a)$. На каждом шаге t для каждого действия a выполняются следующие шаги:

1) Выполняется действие a. Если произошло изменение 35 фичи ($S(t)[35]! = S(t+1)[35]$) или numpy.random.rand() $< p_c$, то выполняются следующие шаги 2 - 5.
2) Делается rollout на $N-1$ шагов с помощью текущей политики, которая равна $\text{argmax}_a\, Q(S(t), a)$ (либо с помощью 3 действия, если первая эпоха)
3) Вычисляется награда полученная за N шагов $Score(t+N+1) - Score(t)$
4) Если эпоха не первая, то к вычисленной на предыдущем шаге награде прибавляется bootstrap значение, которое равно $\max_a Q(S(t+N), a)$
5) Сохраняется обучающий пример - состояние $S(t)$ и целевое значение $Y(S(t), a)$.

2.2 Обучение линейной регрессии

Для обучения регрессии были заданы гиперпараметры: $N = 50$ и $p_c = 0.7$. Более того, было обучено три различные версии регрессии, в зависимости от дополнительных используемых фич (квадраты фич и перемножение с 35 фичей):

1) квадраты для [1,2] и перемножение для [1,4,11,12,13]

2) квадраты для [1, 2, 14, 16] и без перемножения с [35] фичей

3) квадраты для [0,1,2, 9, 10] и перемножение для [4, 11, 12, 13, 15, 18, 20]

Все три модели регрессии дообучались на трейне и тесте с $p_c = 1$ и для разных значений 35 фичи. 1-я и 3-я регрессия дообучались для трех значений 35 фичи (==0, >0, <0), а 2-я регрессия дообучалась для 5 диапазонов 35 фичи (==0, ==0.1, >0.1, == -0.1, < -0.1). Другими словами, у каждой модели для каждого значения 35 фичи был свой набор весов.

2.3 Обучение нейронной сети

В итоговом решении использовались две сети с 16 и 100 нейронами в одном скрытом слое и функцией активации $tanh$. Для сети со 100 нейронами использовались гиперпараметры: $N = 100$, p_c линейно увеличивался от 0.3 до 1 за 10 эпох. Эта сеть обучалась 50 эпох, и веса брались как средние веса начиная с 30 эпохи. Для сети с 16 нейронами: $N = 20$ и p_c от 0.6 и брались лучшие веса.

2.4 Обучение на валидационном уровне

Для обучения во время тестирования (online-обучение) были подгружены данные из train и test уровня (от прохождения 1-ой регрессии, и брались только те действия, которые меняли 35 фичу). Каждые 200 тыс шагов собирались данные с валидационного уровня и на этом обучались новые веса для 1-ой регрессии. Обновление делалось постепенно: $W = 0.02 * W_{new} + 0.98 * W$. Дополнительно вес, с которым подмешивалась 1-я регрессия в итоговую модель, рос на 0.1.

3 Заключение

Результаты для пяти моделей (три регрессии и две сети) и финального подхода представлены в следующей таблице.

Уровень	reg1	reg2	reg3	nn1	nn2	final
train	3081	3642	3547	3325	2735	3536
test	3536	3641	4196	3186	3030	3867
valid	2757	540	1322	2512	2060	3097

Таким образом, наше решение основывается на трех основных вещах: итерация по стратегиям (policy iteration), постепенная фильтрация обучающих примеров (curriculum learning) и online-обучение.

Список литературы

[1] Р.С. Саттон и Э.Г. Барто. *Обучение с подкреплением.* БИНОМ. Лаборатория знаний, 2015.