DRL Course Домашнее задание 1

Алексей Матушевский

В этой практической работы мы изучаем работу алгоритма Cross Entropy Method (CEM) совмещенного с нейронными сетями для поиска оптимальных параметров в среде с непрерывным пространством с, не обязательными, непрерывными действиями. Непрерывные значения для действий и состояний не позволяют использовать предыдущие подходы и их улучшения, так как предыдущие подходы опирались на матрицы стратегий. Каждая строка такой матрицы представляло собой конкретное состояние системы, а столбец - действия. Непрерывные значения — не позволяют работать с матрицами, или попросту создают огромной матрице.

В таких случая лучше альтернатива - нейронные сети. Они подходят для работы с непрерывными значениями.

Задание 1

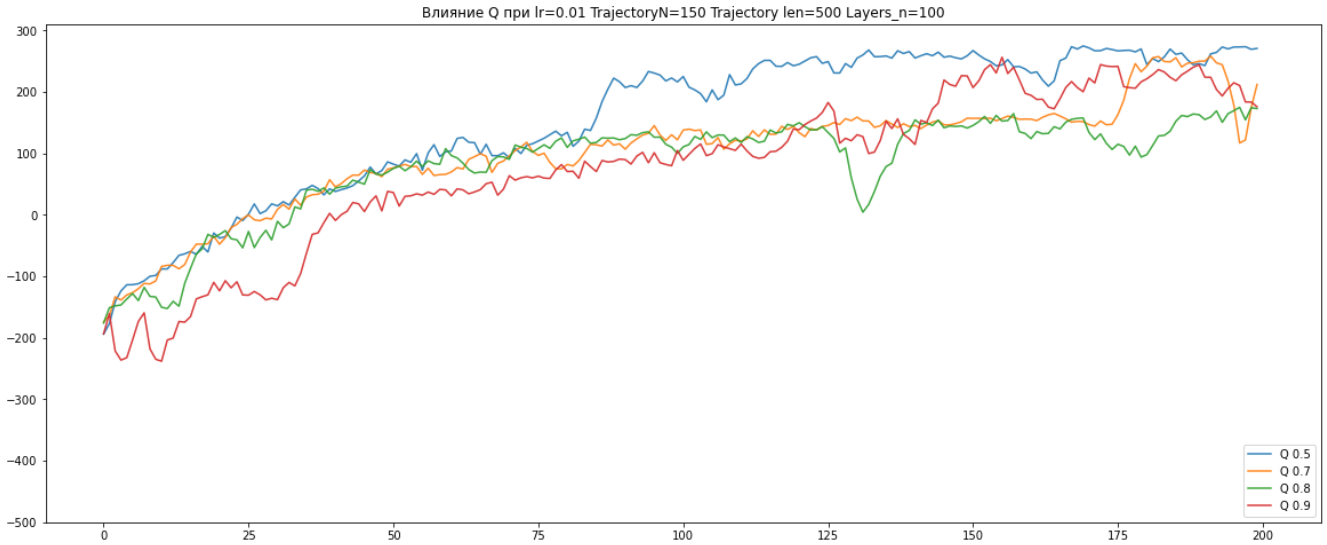
Работа в среде Lunar Lander.

В этой среде мы управляем посадочным модулем 4мя дискретными действиями. Среда описывается 8мью непрерывными состояниями.

Для решения этой задачи я использовал CEM +NN с разными наборами параметров

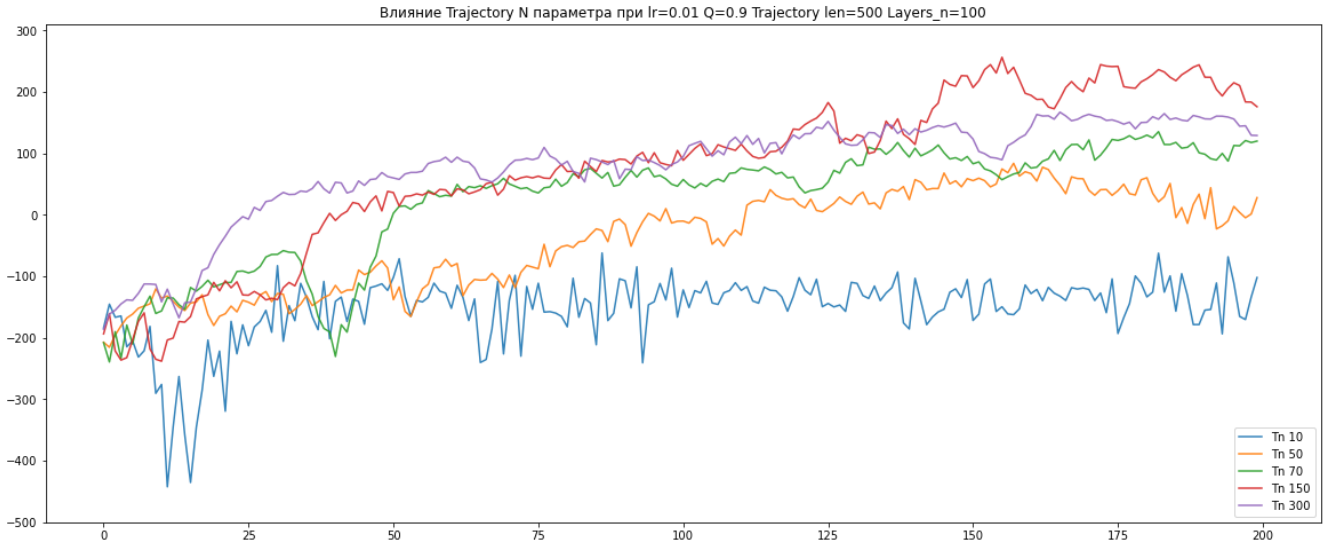
Episode Len — количество эпизодов тренировки
Q-param — квантиль отбираемые
Trajectory N — количество траекторий в одном эпизоде
Trajectory Len — длинна одной траектории
Layers N — количество нейронов в каждом слое
Learning Rate — темп изменения параметров оптимизатором

Layers N при разных Q
Проанализируем влияния Q на достижения большой награды. Как видно с увеличением Q достижение высокой награды занимает большее количество эпизодов тренировки.

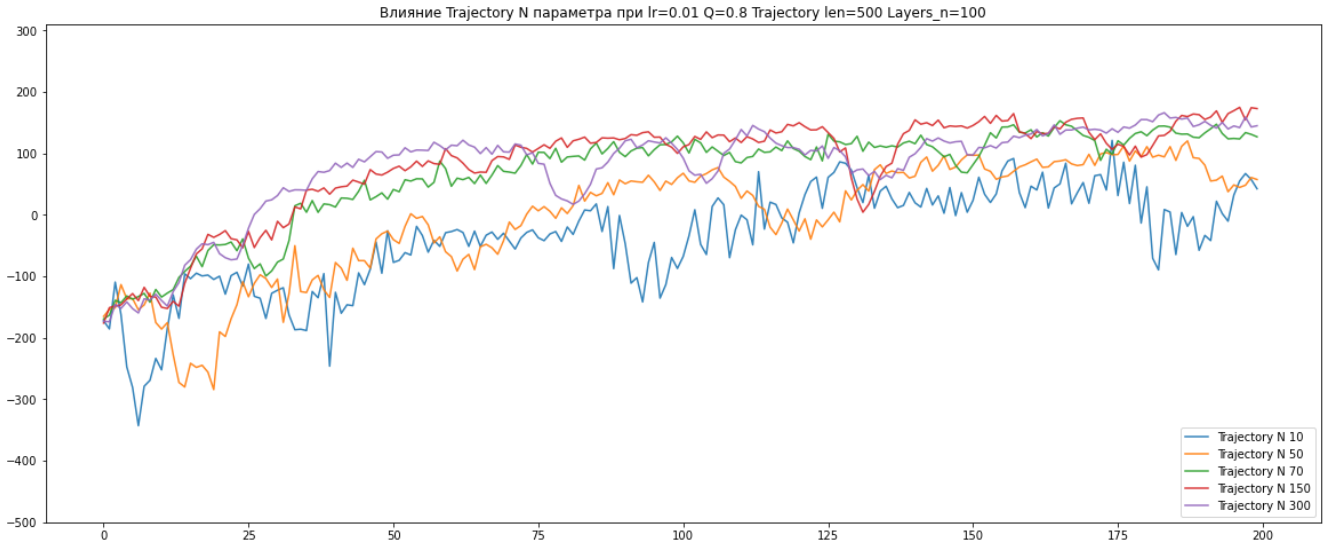


Trajectory N

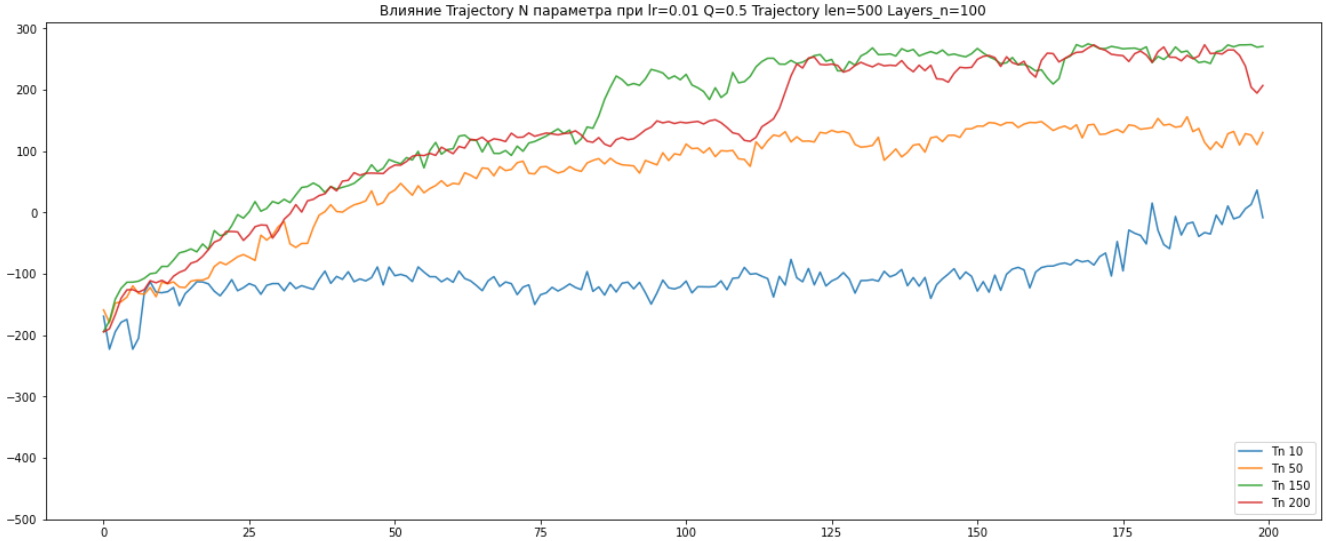
Увеличение Trajectory N при **q=0.9** Trajectory len 500. с увеличением - уменьшатся дисперсия в росте оценки.



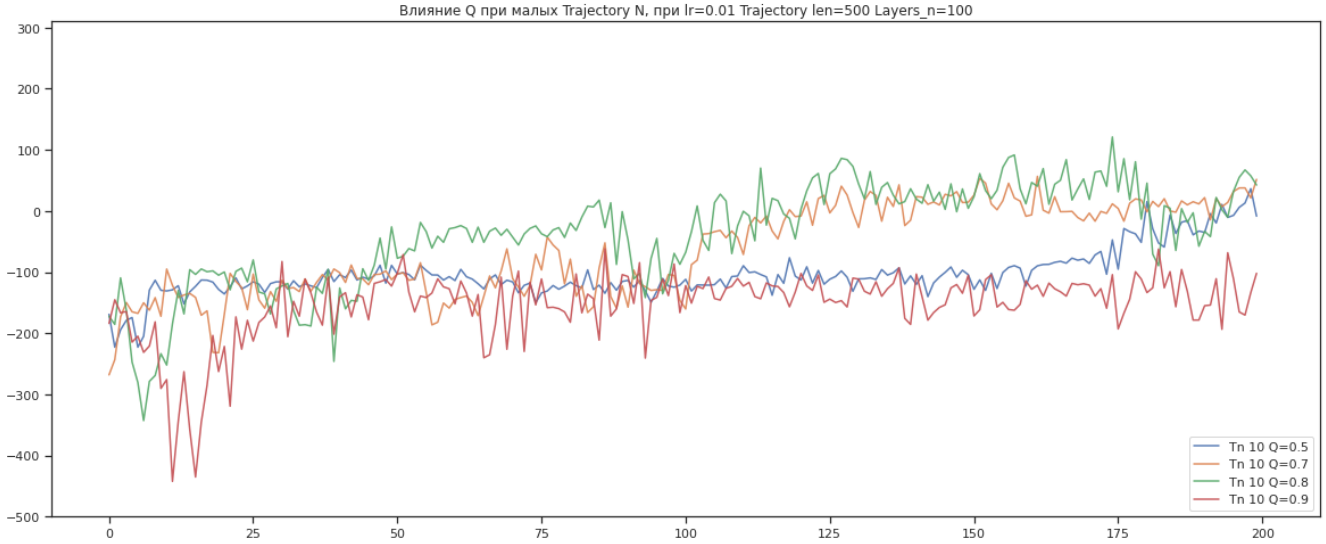
Увеличение Trajectory N при **q=0.8** Trajectory len 500. Так-же с увеличением - уменьшатся дисперсия в росте награды.



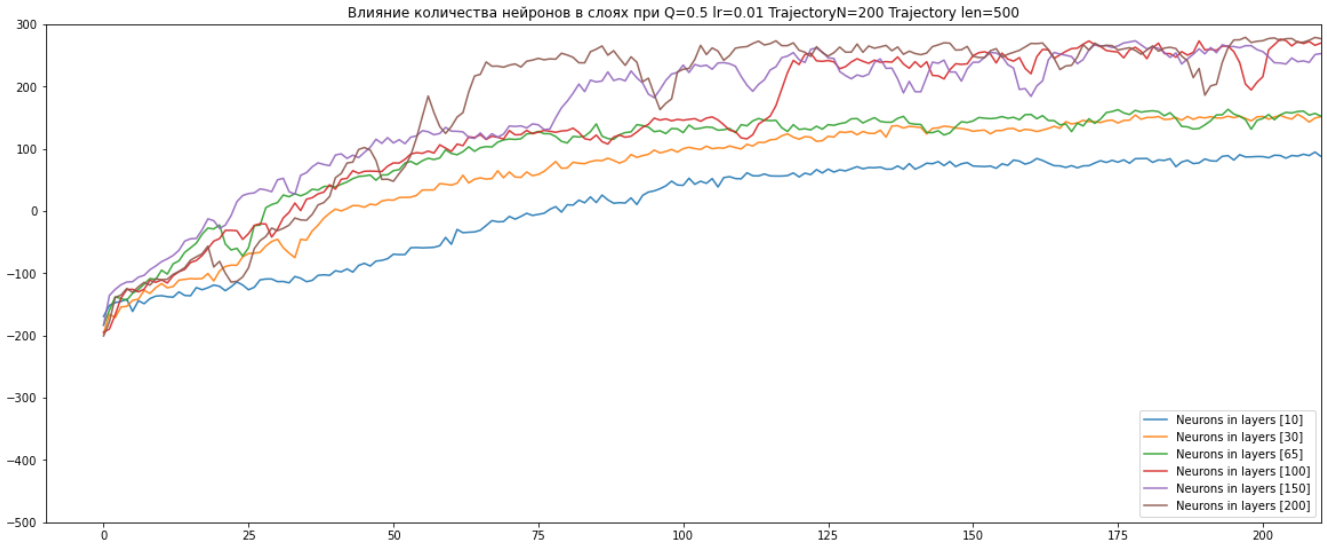
Аналогичное поведение при Q=**0.5**



При очень низких Trajectory N алгоритм не приходит к стабильному состоянию в сравнении с более высокими значениями параметра.



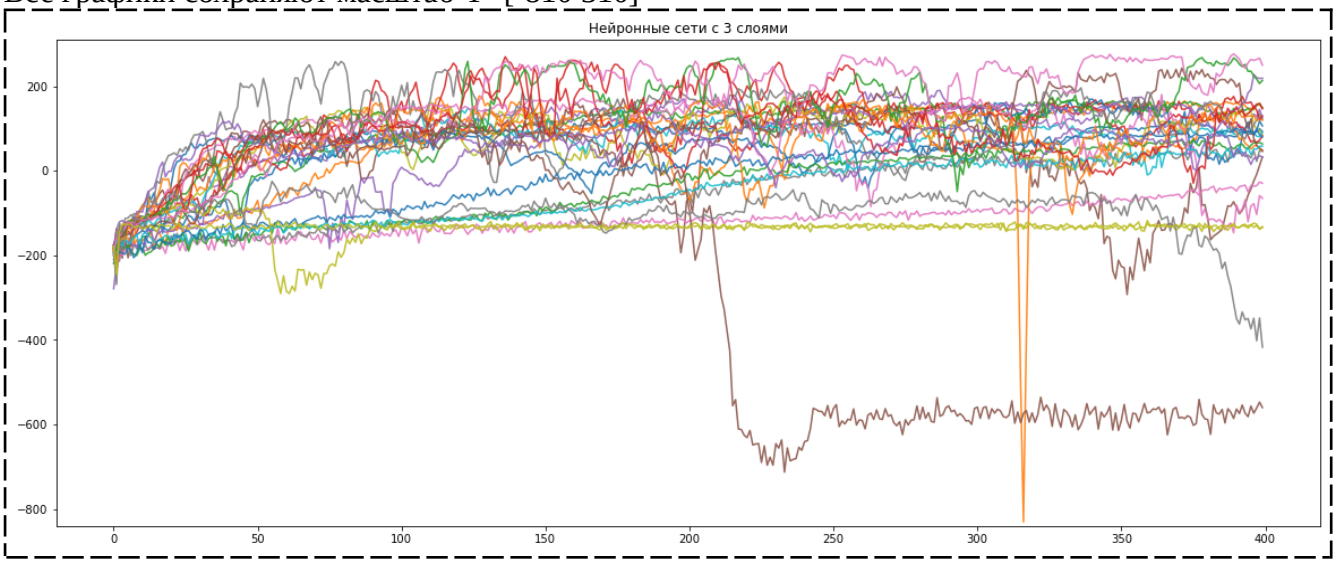
Влияние количества нейронов Layers N



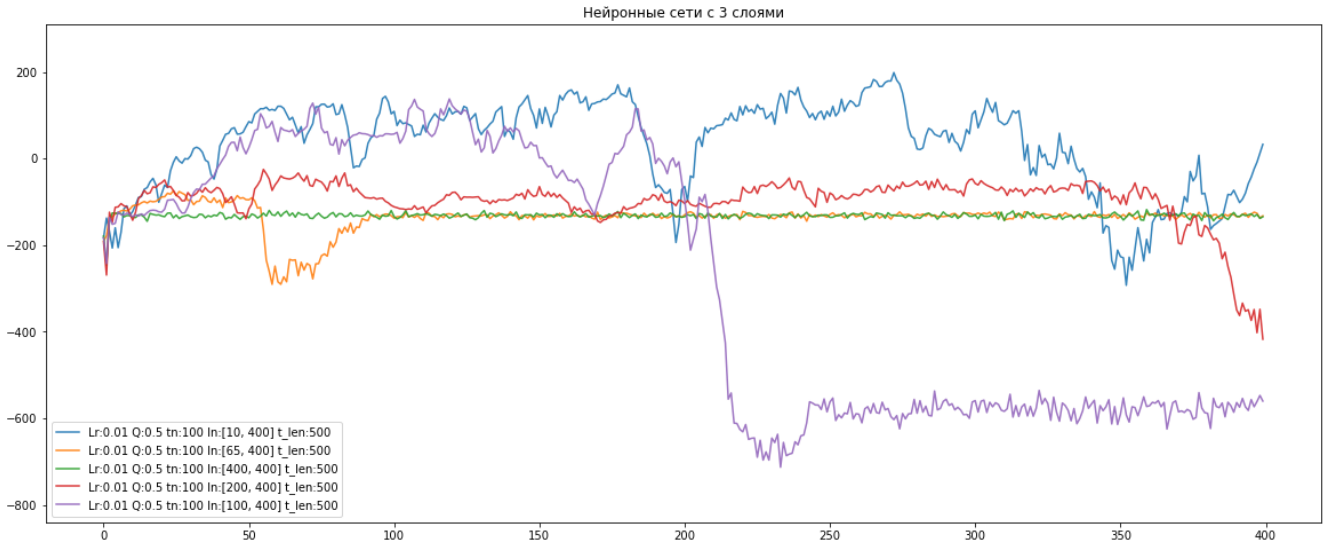
При одних и тех же параметрах Q=0.5, Trajectory N=200, и Trajectory Len=500 — создали сети с разным количеством нейронов. Как видно из графика с увеличением количества нейронов алгоритм быстрее достигает высокой награды, но при этом увеличивается дисперсия.

Увеличение количества слоев в сети.

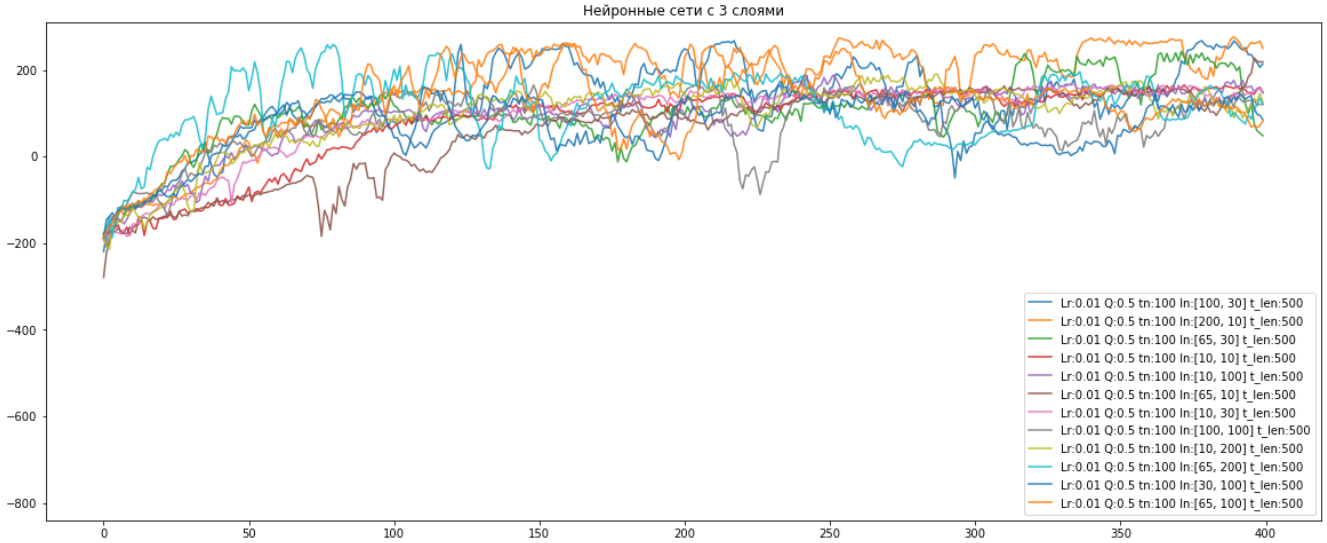
Оставляем те же параметры Q=0.5, Trajectory N = 100, Trajectory Len=500 и **400** эпизодов
Все графики сохраняют масштаб Y=[-810 310]



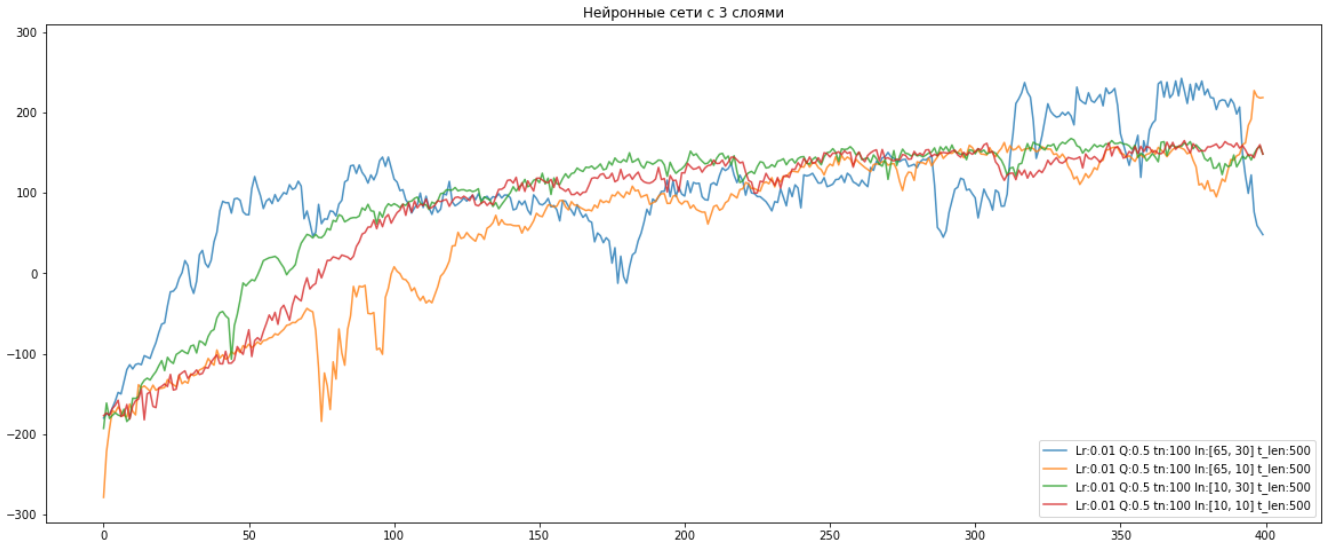
При очень большом количество нейронов, сеть не может удержаться на высокой награде или вообще не достигает положительной награды.



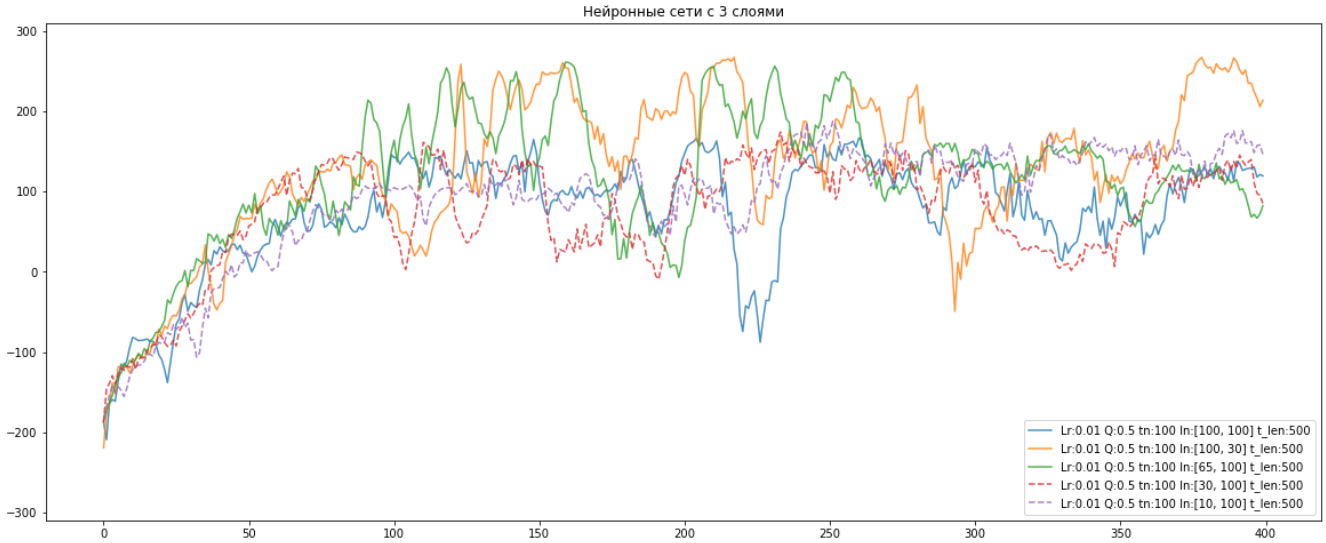
Сети которые удерживаются выше нуля, имеем слои не больше 200 нейронов в одном из слоев



Сети с меньшим в сумме количеством нейронов дают меньшую дисперсию в росте награды



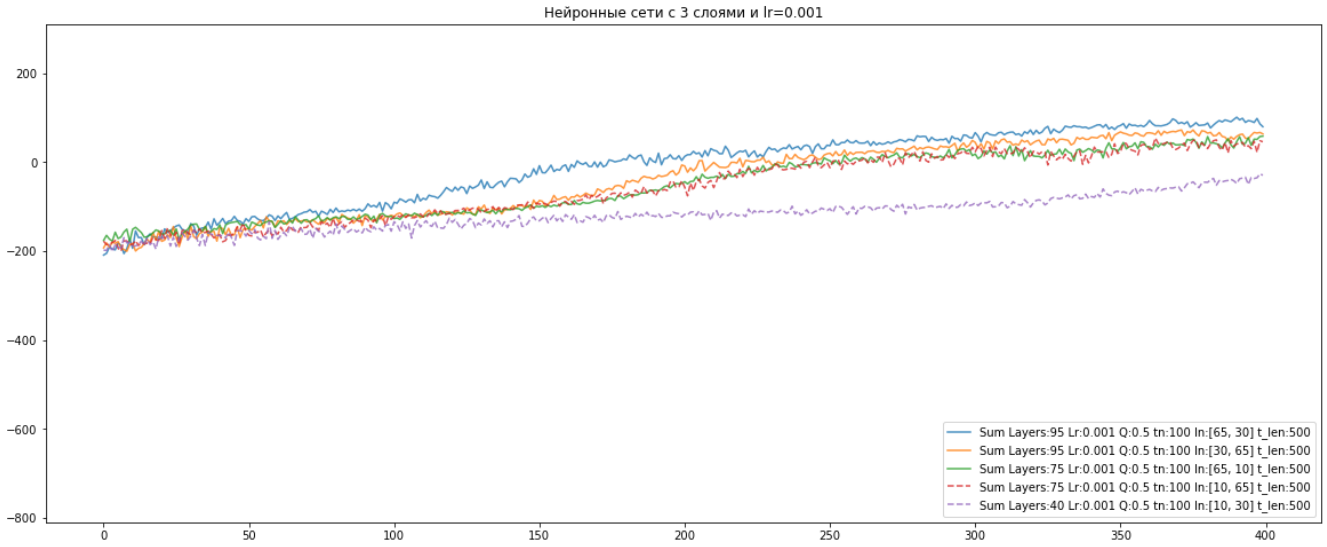
(масштаб изменен на [-310 300])



(масштаб изменен на [-310 300])

при изменении Learning Rate — скорость схождения сети уменьшается пропорционально изменению lr, сеть с меньшим количеством нейронов дает наименьшую скорость схождения

(масштаб изменен на [-810 300])

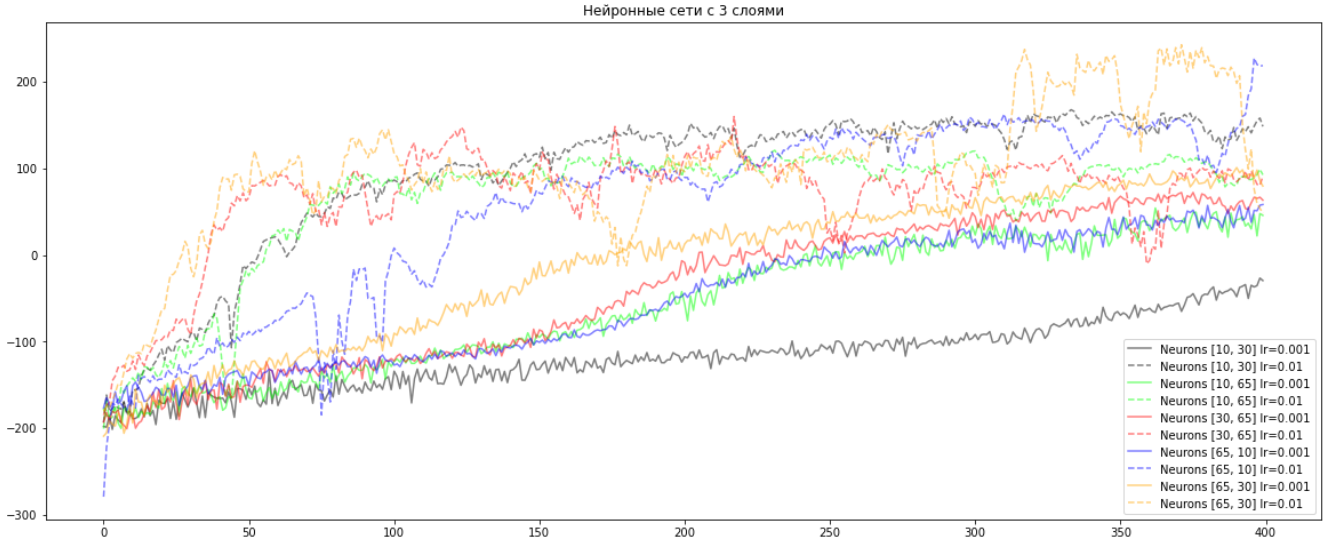


Четырехслойные сети

При увеличении количества слоев до 4 — общее поведение остается тем же. Высокое количество нейронов — дает
- больший разброс в росте награды
- новый минимум в награде

Learning Rate

Влияние Learning Rate на разные сети. Как и в случай двухслойных чем ниже LR тем дольше сходится алгоритм

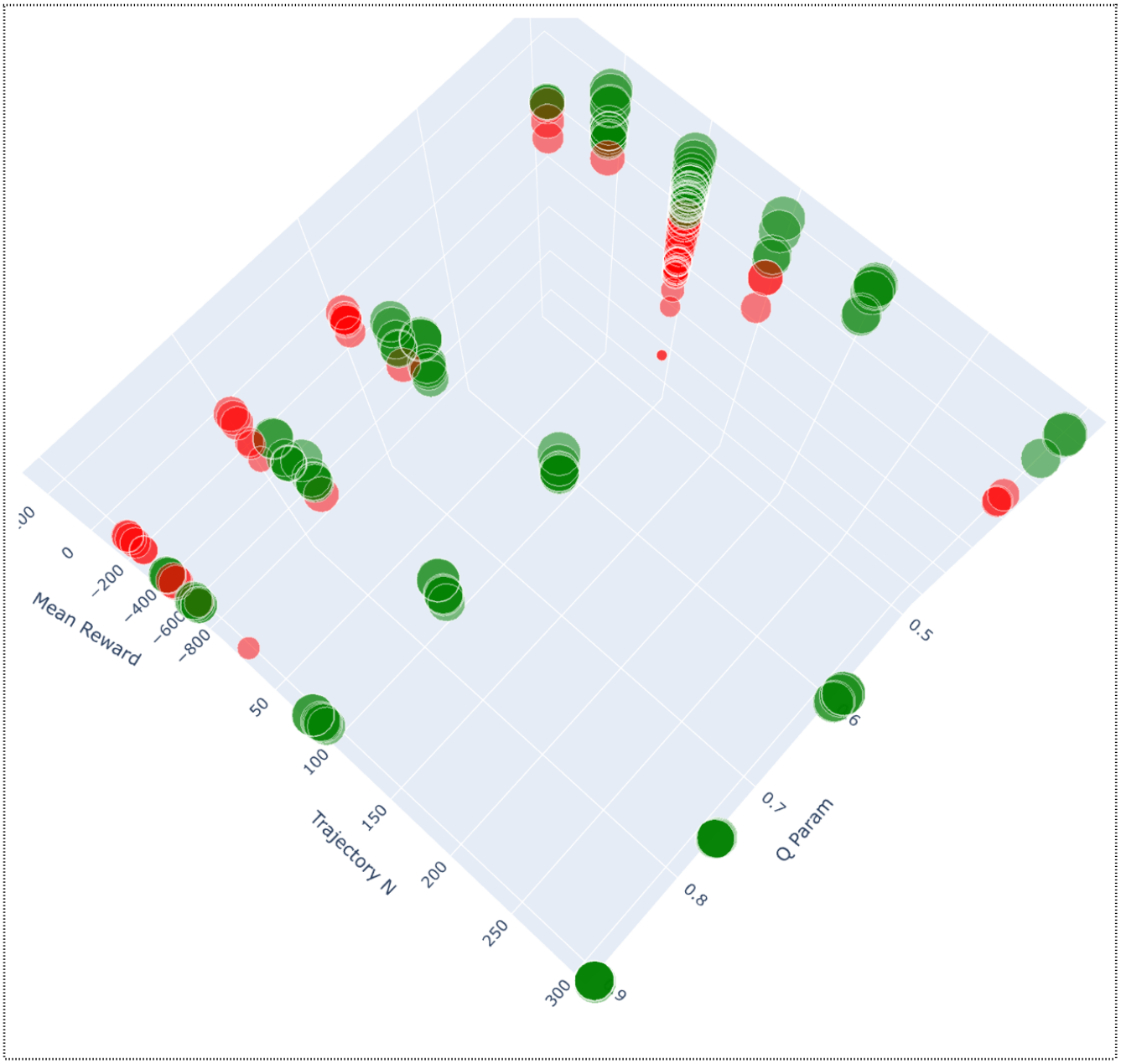


Лучшие результаты

Если выложить все параметры на 3D график,
X — Q-param
Y — Trajectory N
Z — Mean Reward

Можно увидеть что в обласит Q от 0.6-0.9 и Trajectory N [75-300] нет моделей со средними отрицательными значениями награды.

http://alexeimatusevski.com/odsrl/report_2-1-2.html
http://alexeimatusevski.com/odsrl/report_2-1-1.html

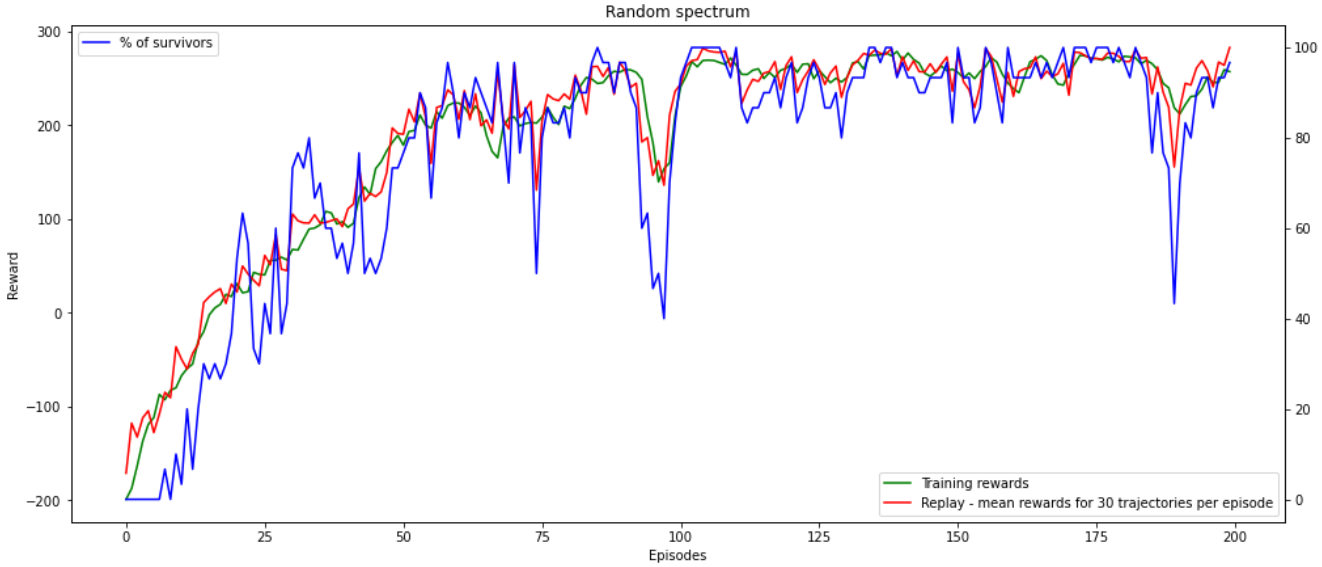


Эксперимент — это запуск алгоритма с определенными параметрами. Критерии выбора лучших результатов основываются на полученных данных из 190 экспериментов.

Для каждого эксперимента мы сохранили:
- параметра эксперимента
- состояние нейросети в каждом его эпизоде
- средние награды в течении эксперимента

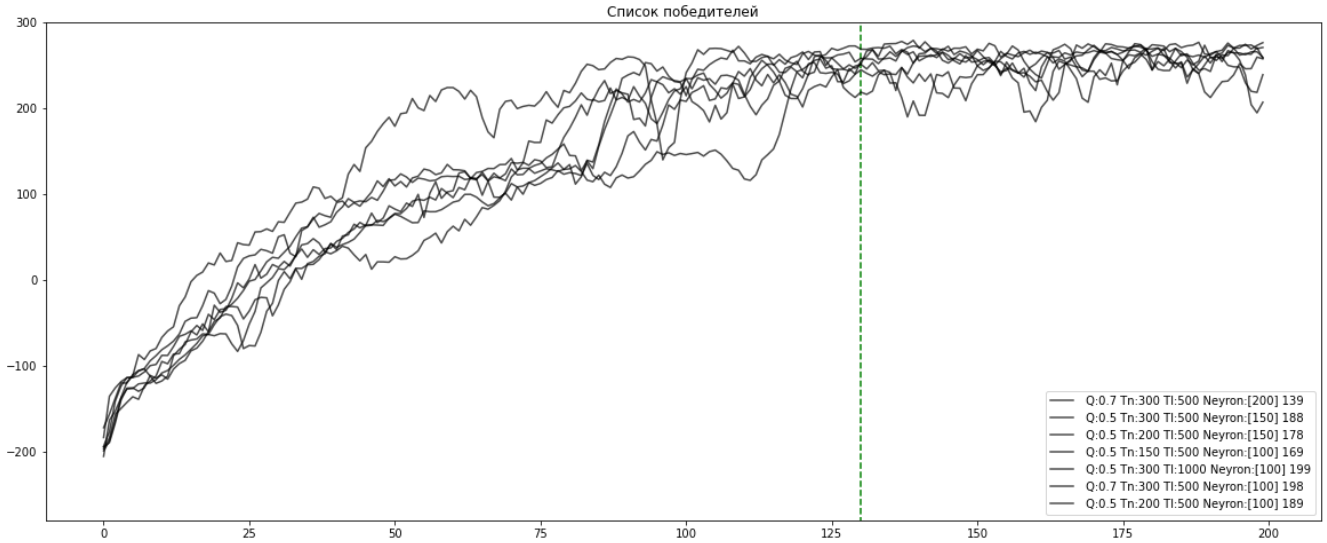
Можно выбирать лучшие параметры по средней оценки во время тренировки. Графики тренировок могут содержать «провалы» в награде. Если повторить это эксперимент используя сохранные состояния весов нейросети, можно получить выборки по средним наградам а также оценить количество «выживших». Выжившие — это те траектории которые не получали награду -100 за конкретный шаг.

Из графика видно что 100% выживаемости (из 30 запусков) достигается только у нейросетей с самыми высокими средними наградами.



Минимальная награда которая дает 100% выживших (из выборки с 30ю запусками) 256. Поэтому можно предположить что эпизоды дающие такие и выше средние награды, будут подходить для лучших результатов.

Reward	Survivers
262.951434	1
269.111621	1
270.265116	1
282.396529	1
279.517435	1
278.370167	1
277.963013	1
277.992915	1
274.999818	1
280.340668	1
275.9605	1
281.941145	1
276.330687	1
282.151603	1
256.508145	1
266.308203	1
278.3295	1
277.618083	1
274.333956	1
271.442862	1
270.33336	1
277.222227	1
273.670604	1
280.847168	1

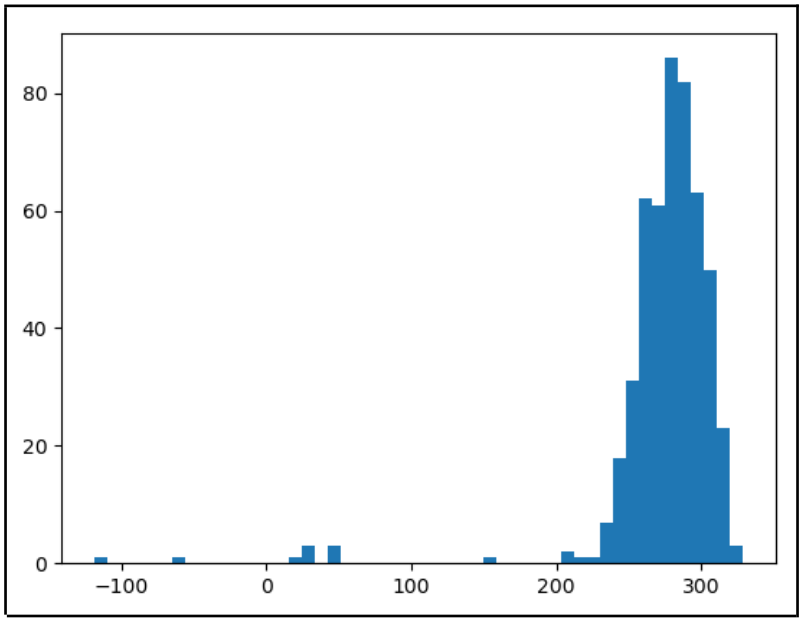


Итого список топ 7 параметров:

	lr	episode_n	trajectory_n	trajectory_len	q_param	layers_cnt	layers_n1	layers_n2	layers_n3
0	0.01	200	300	500	0.7	1	200	-	-
1	0.01	200	150	500	0.5	1	100	-	-
2	0.01	200	300	500	0.5	1	150	-	-
3	0.01	200	300	500	0.7	1	100	-	-
4	0.01	200	300	1000	0.5	1	100	-	-
5	0.01	400	200	500	0.5	1	100	-	-
6	0.01	400	200	500	0.5	1	150	-	-

в качестве EpisodeN — 130 выглядит приемлемым.

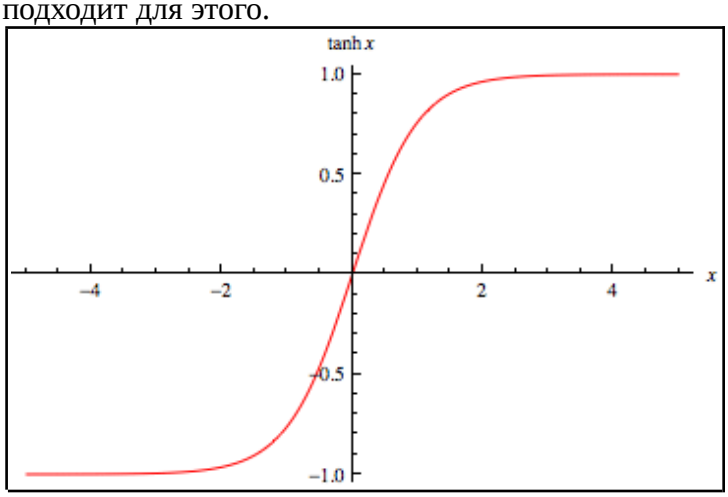
layers_n=[200]
episode_n=200
trajectory_len=100
trajectory_n=300
q_param=0.7
Средняя оценка 275.49 из 500 траекторий
9 разбившихся экипажей из 500



Задание 2 — Вагонетка

В этом задании, мы управляет тележкой, которая находится в долине. Раскачивая, мы должны выгнать тележку на верх, к флажку. Сила с которой мы толкаем тележку, варьирует от [-1, 1].

Для решения задачи мы должны модифицировать алгоритм.
1. параметр силы должен находится в пределах [-1, 1]. Для это можно применить функцию тангенс. Она отлично подходит для этого.



2. Для того чтобы алгоритм работал, нам нужно чтобы действия выдавались из какого-то распределения. Для этого может подойти нормальное распределение с параметрами Mu, Sigma. Тогда мы будет учить нейросеть подбирать параметры для нормального распределения таким образом, чтобы в зависимости от состояния, она выбирала подходящие Mu, Sigma.

Полученные значение из нормального распределения, мы преобразуем функцией tnah в диапазон [-1, 1].
Таким образом мы получим вектор действия размером 1, подходящий для среды Вагонетка.

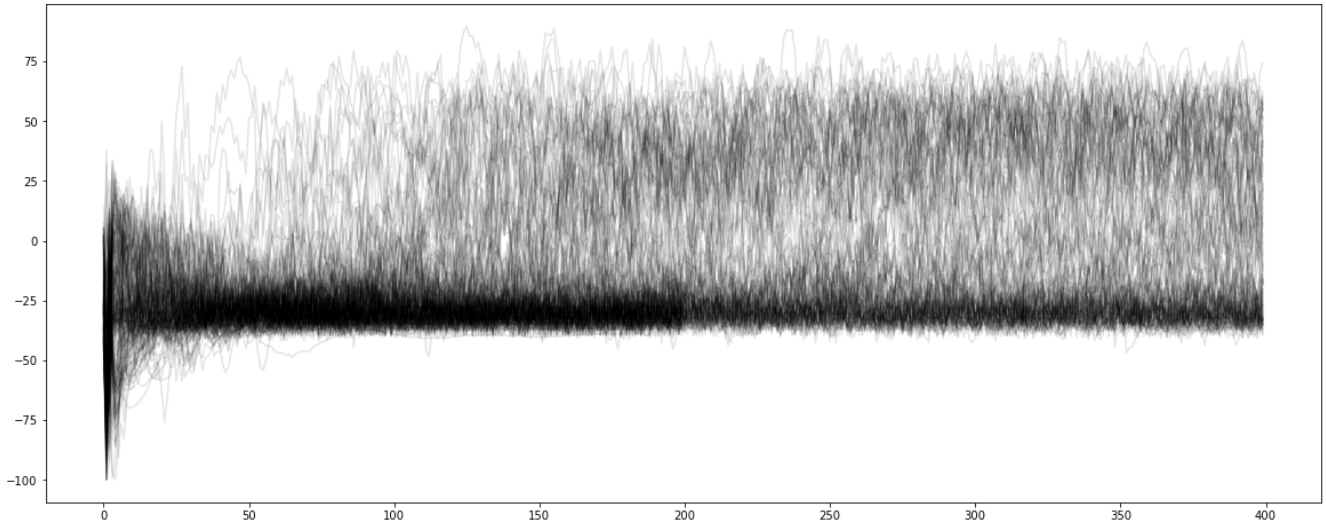
Полученный алгоритм, мы будет изучать с разными наборами параметров.

Изучаемые параметры:

Learning Rate:	{0.001, 0.005, 0.01, 0.1, 0.15, 0.2, 0.3, 0.4}
Episode N:	{200 400}
Trajectory N:	{50}
Trajectory Len:	{1000}
Q:	{0.5, 0.6, 0.65, 0.7, 0.75, 0.8, 0.825, 0.85, 0.875, 0.9, 0.925, 0.95, 0.975}
Layers N:	{50, 100}

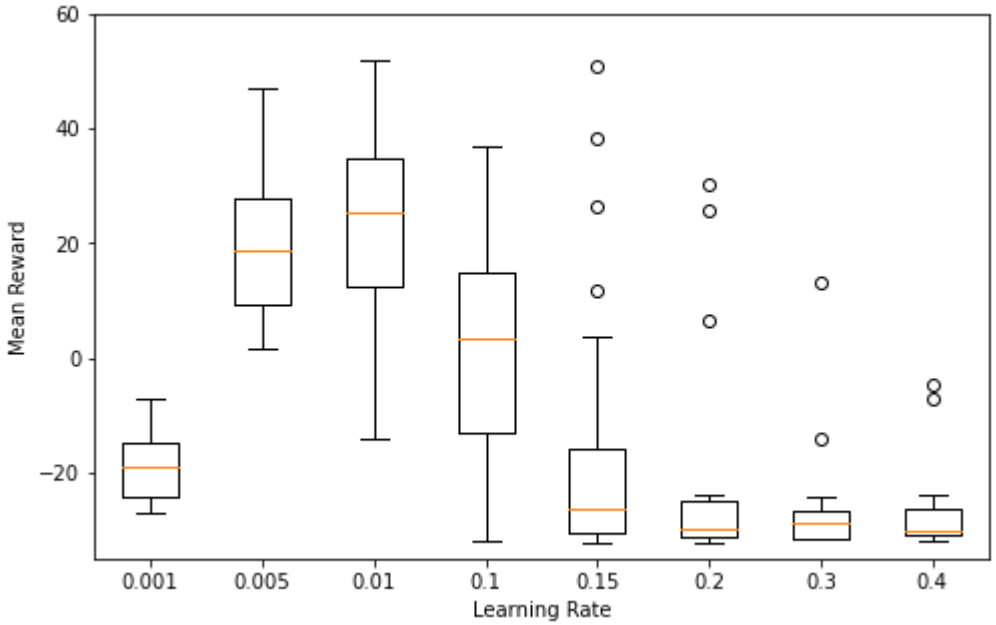
Такой набор параметров был выбран специально, с учетом полученного опыта в первом задании второй домашней работы. Я не использовал многослойные нейросети, т. к. оборудование не позволяет запускать такое количество экспериментов. Они помогут легче сравнивать полученные траектории.

Итого было проведено 175 экспериментов

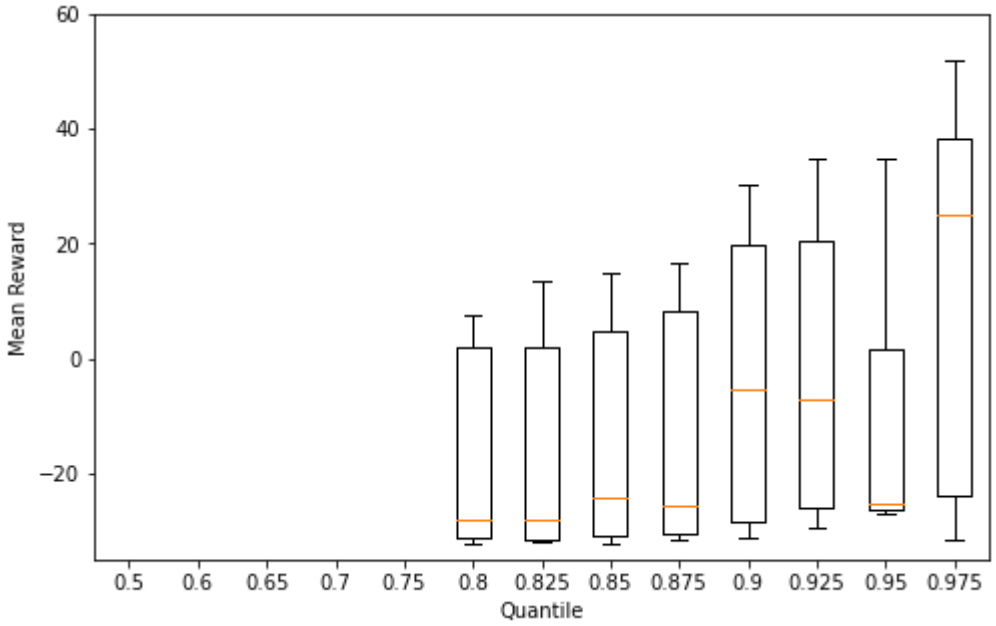


Далее все графики представлены в масштабе [-35, 60] по Y

Параметры Q и Learning Rate в большей степени влияют на форму кривой наград.



Сравнение влияние Lerning Rate на среднюю оценку по всем траекториям из экспериментов, при прочих равных параметрах. Высокая средняя оценка находится в области Lr [0.005 и 0.1]



Сравнение средних в зависимости от параметра Q.

Из этого графике мы видим, что большие значения приходятся к более высоким значениям Q

Посмотрим более внимательно на эти же данные, в нескольких разрезах по Learning Rate

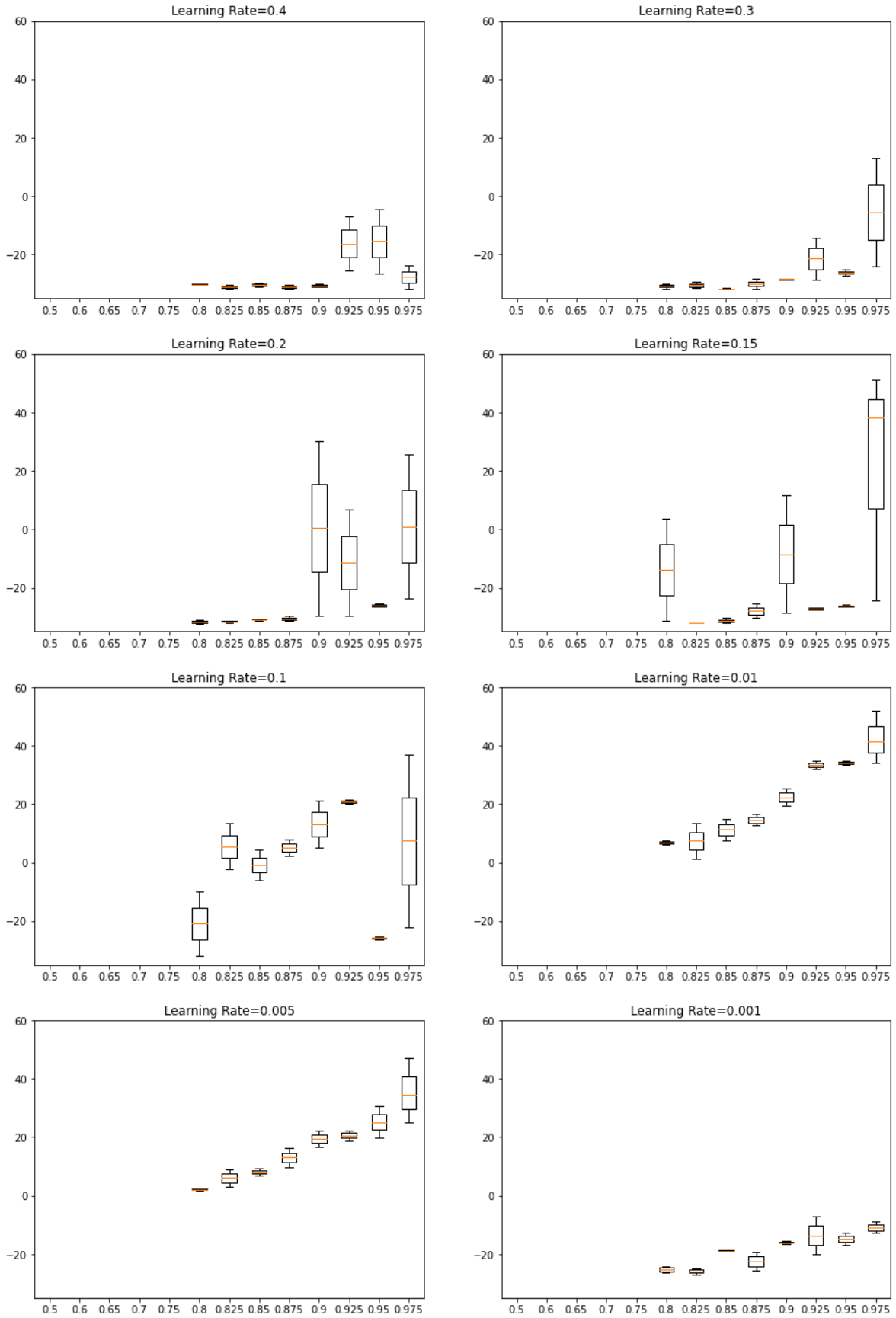
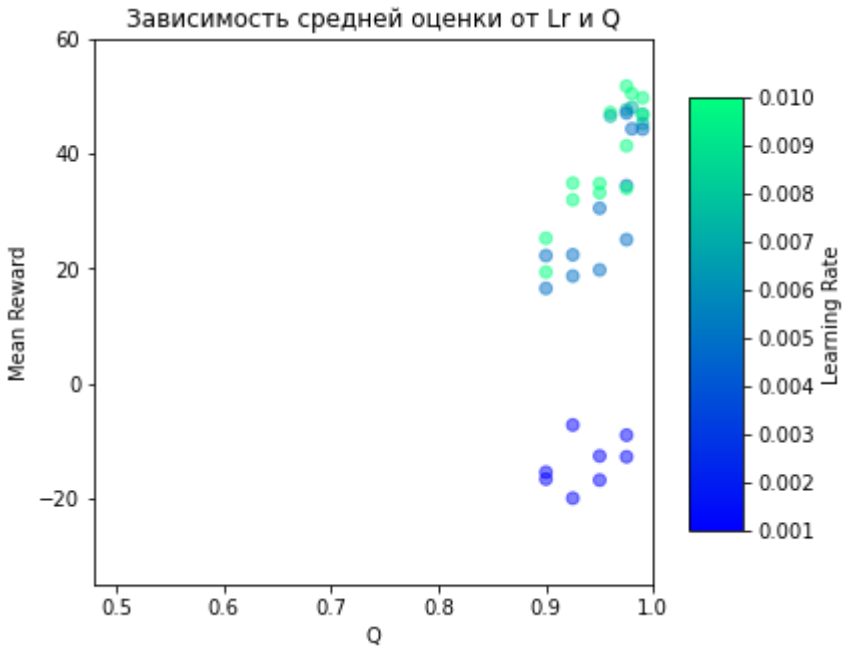


График зависимости средней награды от Q при разных Learning Rates

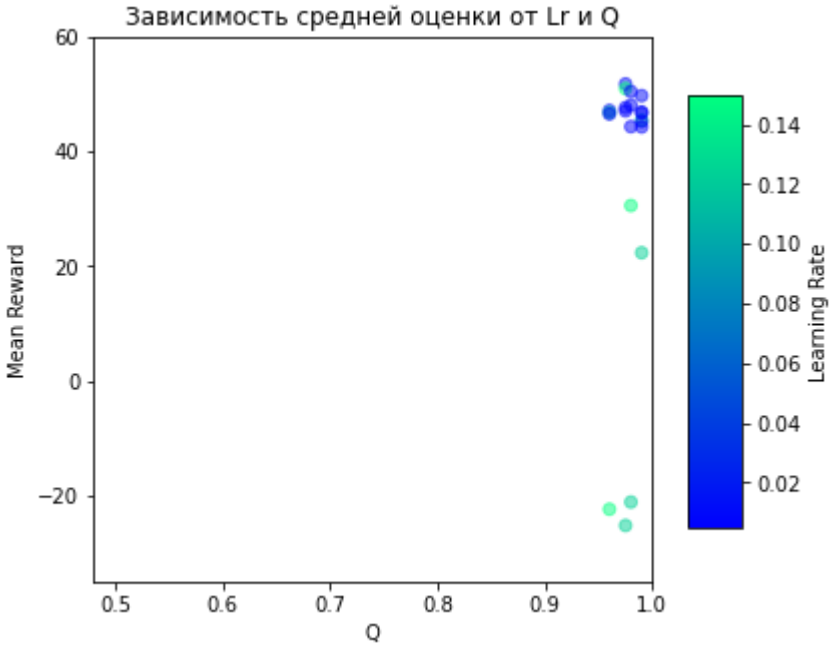
Из графиков видно что хорошие модели получаются при низких Learning Rates {0.01, 0.005} и высоких Q [0.925, 0.975]

Кандидаты на лучшие модели, средние значения по траекториям



Learning Rates `{0.0075, 0.005, 0.001, 0.01,}`

Если провести еще 21 уточняющих экспериментов:



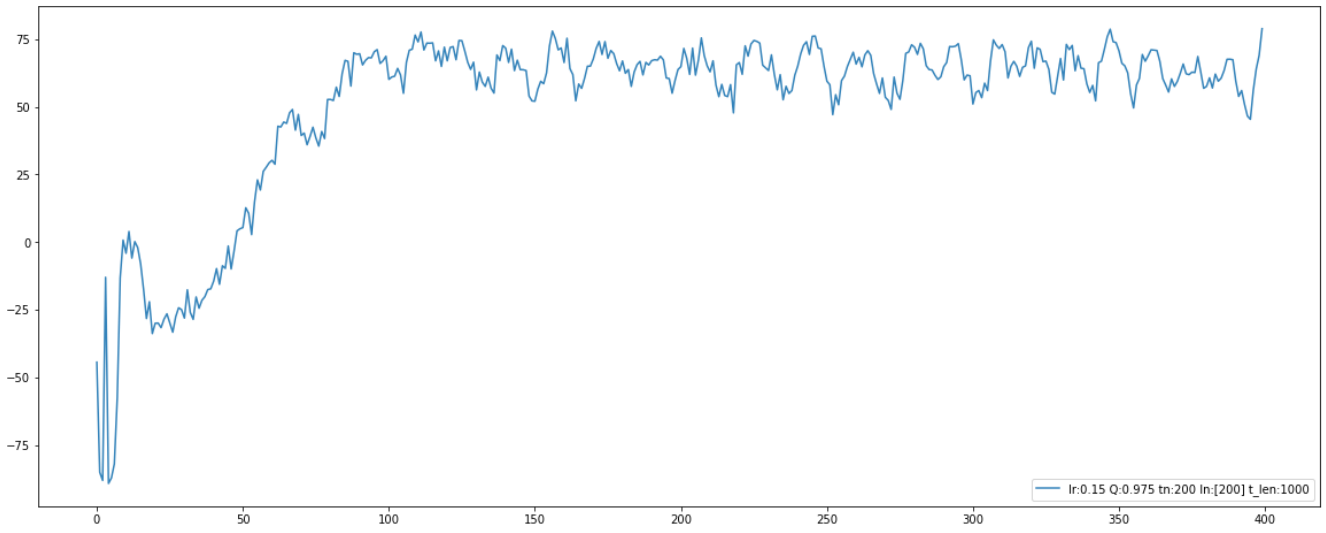
Learning Rates `{0.001, 0.005, 0.0075, 0.01}`

По этой таблице мы может выбрать лучшие параметры:

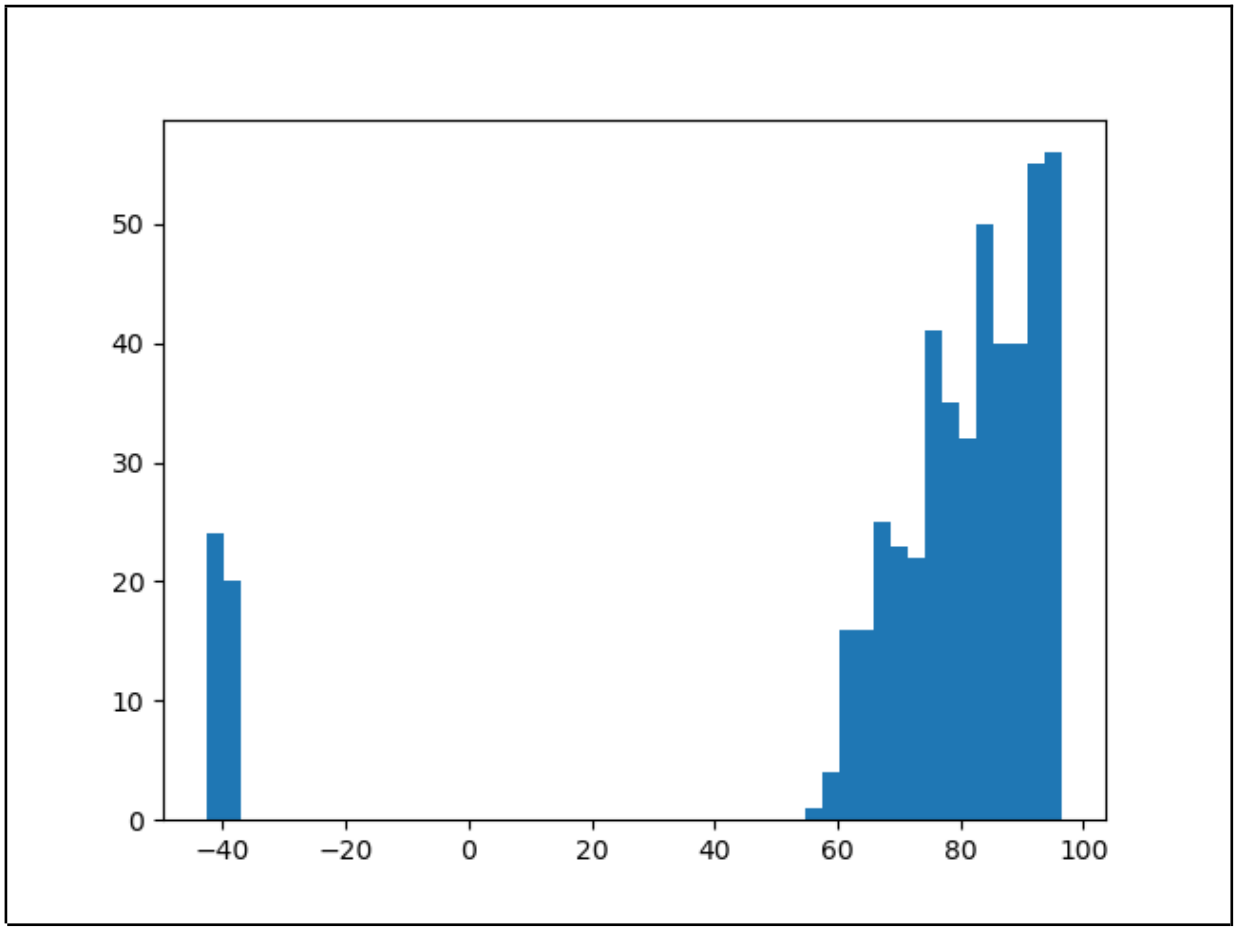
TarjectoryN=200, TrajectoryLen=1000, Q=0.975 и Learning Rate =0.15 потому что у этого эксперимента дисперсия между первым и третьим экспериментами.

	lr	episode_n	trajectory_n	trajectory_len	layers_n1	q_param	mean	max	std
1	0.01	400	200	1000	200	0.975	51.81	79.20	7.44
2	**0.15**	**400**	**200**	**1000**	**200**	**0.975**	**51.01**	**78.92**	**6.57**
3	0.0075	400	200	1000	200	0.98	48.11	79.03	5.48
4	0.0075	400	200	1000	200	0.975	47.69	77.38	5.52
5	0.01	400	200	1000	200	0.96	47.19	79.14	7.34
6	0.005	400	200	1000	200	0.975	47.11	75.09	5.34
7	0.125	400	200	1000	200	0.96	46.80	77.50	8.39
8	0.0075	400	200	1000	200	0.96	46.59	74.96	7.66
9	0.01	400	200	1000	200	0.98	45.85	77.19	5.92
10	0.005	400	200	1000	200	0.98	44.38	77.39	7.55
11	0.005	400	200	1000	200	0.99	44.32	71.95	5.56
12	0.01	400	200	1000	200	0.99	41.60	76.78	14.66
13	0.0075	400	200	1000	200	0.99	37.19	76.00	4.18
14	0.15	400	200	1000	200	0.99	26.35	76.96	17.95
15	0.01	400	200	1000	200	0.99	-13.96	7.74	14.53
16	0.0075	400	200	1000	200	0.99	-20.57	0.23	16.36
17	0.15	400	200	1000	200	0.96	-22.39	-2.21	3.89
18	0.125	400	200	1000	200	0.98	-22.95	2.78	4.30
19	0.125	400	200	1000	200	0.975	-25.23	-9.74	3.34
20	0.15	400	200	1000	200	0.98	-26.45	62.07	21.90
21	0.125	400	200	1000	200	0.99	-27.92	12.86	5.09

И по графику траекторий можно сказать, что **Eposode_N = 125** будет приемлемым.



Для более точных результатов — нужно увеличивать TrajectoryN — это даст более плавные графики и новые увлекательнейшие результаты для анализа.



Средняя оценка 71.21 из 500 траекторией

lr=0.15
layers_n=[200]
episode_n=125
trajectory_len=1000
trajectory_n=200
q_param=0.975

Выводы

В этой домашней работы, мы получили опыт работы в средах с непрерывными состояниями и действиями. Работы в таких средах значительно осложняется, так как природа среды а также стахостичность среды, накладывают многократное усложненение в процесс исследования гипер-параметров алгоритма.

Время выполнения некоторых экспериментов занимали десятки часов(!), а какие-то пары часов. Не смотря на мощное оборудование, главный успех исследование, это умение навигироваться в пространстве гипер-параметров, в ограниченное время на домашнее задание.

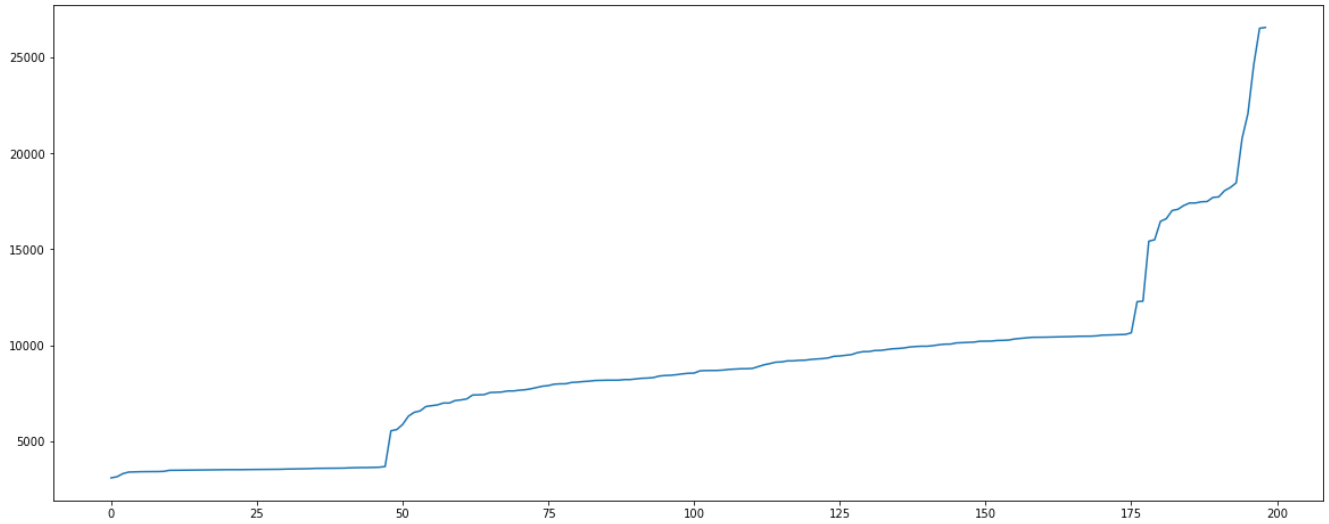
Важно учитывать, что для сравнения результатов, нужно чтобы эксперименты легко делились на группы, и четко различались одним признаком и были одинаковыми по другим признакам, чтобы была возможность учитывать фактор этого признака. Такой подход экономит время на сравнения, и помогает делать более мотивированные изменения в гипер-параметрах, для новой ветки экспериментов.

Количество слоев в конечной модели сильно влияют на полученную модель. Во втором задании было замечено что «умная» сеть требует больше нейронов. Проводить исследовательскую работу по количеству слоев — очень сложно, и требует больше навыков и умения, но я точно не знаю каких.

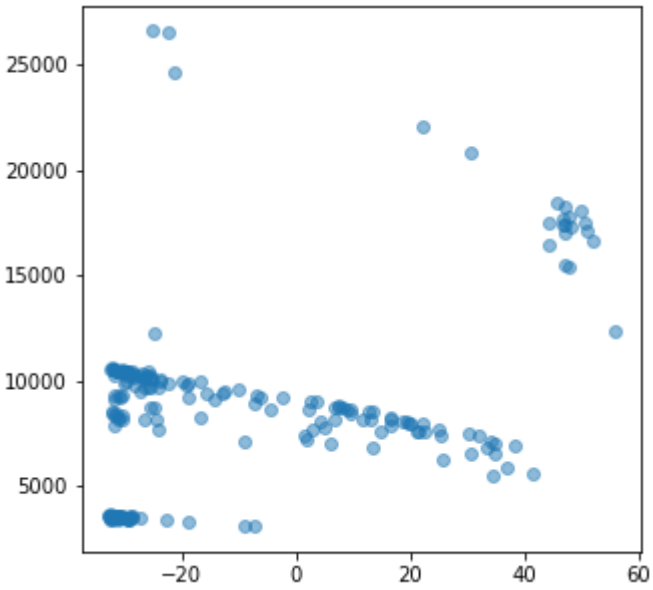
Спасибо за увлекательнейшее домашнее задание!!! <3

Аппендикс.

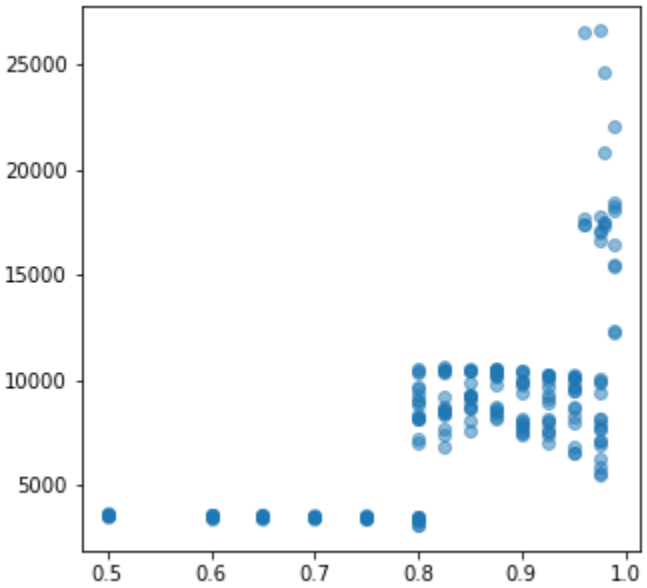
Время работы алгоритма второго задания.



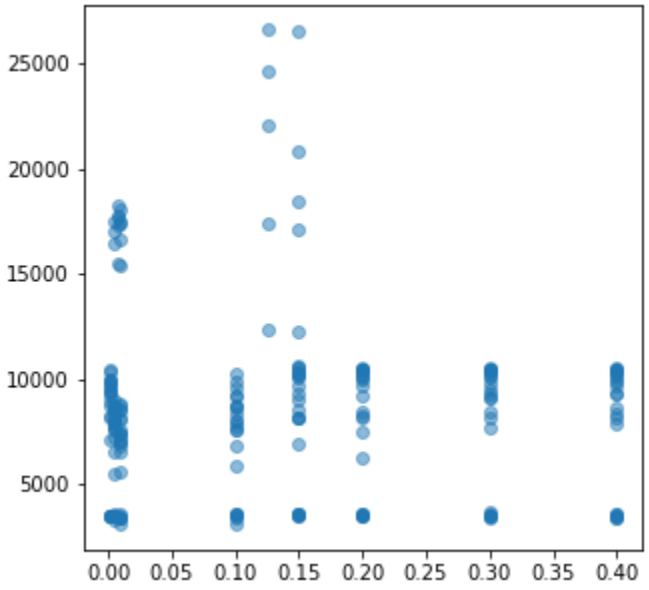
Ранк эксперимента по времени:



Средняя награда на время выполнения



Параметр Q на время выполнения



Параметр Learning Rate на времени выполнения

Забавный момент из Lunar Lender
https://www.youtube.com/watch?v=go3thYum5xU