# 2. ЛЕКЦИЯ. Мета-эвристики

***Бэйзлайны***

Бэйзлайн – принято считать, что это самая простая модель, результат работы которой используют как базовую линию, и все остальные модели должны работать лучше или, по крайней мере, не хуже.

*Задача безградиентной оптимизации*

Определение 1: Мета-эвристикой называется метод черного ящика (black-box) оптимизации

$$J(\theta) \to max$$

со стохастическим предсказателем (оракулом) нулевого порядка, то есть возможностью для каждой точки $\theta \in \Theta$ получить несмещённую оценку $\hat{J} \approx J(\theta)$.

Такие методы также называются безградиентными (gradient-free), поскольку не используют градиент функции и в принципе не предполагают её дифференцируемости. Понятно, что такие методы — «универсальный» инструмент, который можно использовать для любой задачи оптимизации. В первую очередь, этот инструмент полезен, если пространство аргументов $\Theta$ нетривиально (например, графы) или если оптимизируемая функция принципиально недифференцируема, состоит из седловых точек или есть другие препятствия для градиентной оптимизации.

Заметим, что если аргументы $\Theta$ — конечное множество (о градиентной оптимизации речи идти не может), задача сводится к следующей: надо найти тот аргумент $\theta \in \Theta$, для которого настоящее значение $J(\theta)$ максимально, при этом используя как можно меньше стохастичных оценок $\hat{J}$ оракула. В теории мета-эвристик опция «вызвать оракул в одной и той же точке несколько раз» в ходе алгоритма обычно не рассматривается; предполагается достаточно богатое пространство $\Theta$, для которого более прагматичной альтернативой кажется запросить значение условно в «соседней» точке вместо уточнения значения оракула для одной и той же.

В контексте обучения с подкреплением, чтобы свести задачу к черного ящика (black-box) оптимизации, достаточно представить стратегию $\pi_\theta(a|s)$ в параметрическом семействе с параметрами $\theta \in \Theta$. В качестве $J$, конечно, выступает оптимизируемый функционал

$$J(\theta) := \mathbb{E}_{\mathcal{T} \sim \pi_\theta} R(\mathcal{T}) \to max$$

а в качестве $\hat{J}$ — его Монте-Карло оценка:

$$\hat{J}(\theta) := \frac{1}{B} \sum_{i=1}^{B} R(\mathcal{T}_i) \ , \mathcal{T}_i \sim \pi_\theta, i \in \{1 \dots B\}$$

Если дисперсия оценки достаточно высока (число сэмплов $B$ недостаточно велико), почти все далее рассматриваемые алгоритмы сломаются (будут выживать «везучие», а не «сильнейшие»). Поэтому может оказаться крайне существенным использовать $B > 1$, даже если $\pi_\theta$ – семейство детерминированных стратегий.

Определение 2: Точку $\theta$, в которой алгоритм оптимизации запрашивает значение оракула, будем называть особью, а само значение $\hat{J}(\theta)$ для данной особи — её оценкой или приспособленностью (fitness).

Однако необходимо учитывать силу гигантского разнообразия мета-эвристик (методы популяционных алгоритмов которые мы проходили в системном анализе). Поэтому попробуем структурировать мотивации некоторых из основных идей. В частности, нас в первую очередь будут интересовать алгоритмы, в той или иной степени успешно применявшиеся в RL.

*Случайный поиск*

Случайный поиск – метод оптимизации и самый простой пример мета-эвристики.

Определение 3: Распределение $q(\theta)$ в пространстве $\Theta$ будем называть стратегией перебора.

Случайный поиск сводится к сэмплированию из стратегии перебора особей $\theta_k \sim q(\theta)(k \in \{0,1,2\dots\})$, после чего в качестве результата выдаётся особь с наилучшей оценкой.

Забавно, что случайный поиск — метод глобальной оптимизации: если $\forall\theta \in \Theta: q(\theta) > 0$, после достаточного числа итераций метод найдёт сколь угодно близкое к глобальному оптимуму решение. Есть ещё один парадокс грубого перебора: если в наличии есть неограниченное число серверов, то возможно запустить на каждом вычисление приспособленности одной особи, и за время одного вычисления провести «глобальную» оптимизацию.

Идея случайного поиска, на самом деле, вводит основные понятия мета-эвристик. Так или иначе необходимо запросить данные приспособленности некоторого набора особей и итоге отобрать лучший. Для имитации умности происходящего введём следующую нотацию.

Определение 4: Набор особей $P := (\theta i | i \in 1,2\dots N\})$ называется популяцией (population) размера $N$.

Определение 5: Запрос оракула для всех особей популяции называется оцениванием (evaluation) популяции:

$$\hat{J}(\mathcal{P}) := (\hat{J}(\theta_i) \;\; i \in \{1, 2 \dots N\})$$

Определение 6: Процедурой отбора называется выбор (возможно,

случайный, возможно, с повторами) $M$ особей из популяции. Формально, это распределение $select(\mathcal{P}^+\ \mathcal{P}, \hat{J}(\mathcal{P}))$, такое что $\forall\theta \in \mathcal{P}^+: \theta \in \mathcal{P}$ с вероятностью 1.

Определение 7: Жадный отбор $select_M^{top}$ – выбор топ-$M$ самых приспособленных особей.

Жадный отбор плох тем, что у нас нет гарантий, что мы на самом деле выбираем наилучшую точку из рассмотренных — наши оценки $\hat{J}$ могут быть неточны, и наилучшей на самом деле может оказаться особь с не самой высокой приспособленностью. В частности, поэтому могут понадобиться альтернативы жадного отбора. Например — Турнирный отбор: Для отбора $M$ особей $M$ раз повторяется следующая процедура: случайно выбираются $K$ особей популяции (из равномерного распределения) и отбирается та из них, чья приспособленность выше. Число $K$ называется размером турнира и регулирует вероятность плохо приспособленной особи быть отобранной: чем больше $K$, тем меньше у слабенькой особи шансов выжить

*Hill Climbing (Скалолазание)*

Процедура отбора позволяет только «сокращать» разнообразие популяции. Хочется как-то обусловить процесс генерации новых кандидатов на уже имеющуюся информацию (которая состоит только из особей и их приспособленностей). Мотивация введения мутаций в том, что даже в сложных пространствах $\Theta$ зачастую можно что-то «поделать» с точкой $\theta$ так, чтобы она превратилась в другую точку $\hat{\theta}$.

Определение 8: Мутацией называется распределение $m(\hat{\theta}|\theta)$, где $\theta$ называется родителем, $\hat{\theta}$ – потомком.

Например: Пусть $\Theta$ — множество путей обхода вершин некоторого графа. Такое пространство аргументов возникает во многих комбинаторных задачах (таких как задача коммивояжёра). Пусть $\theta \in \Theta$ — некоторый путь обхода, то есть упорядоченное множество вершин графа. Мутацией может выступать выбор случайных двух вершин и смена их местами в порядке обхода. В исходном обходе пройдено пять вершин графа в порядке (4, 3, 1, 5, 2), а после мутации - в порядке (4, 2, 1, 5, 3).

Рассмотрим простейший способ использования мутации. На $k$-ом шаге алгоритма будем генерировать $N$ потомков особи $\theta_k$ при помощи мутации и отбирать из них жадно особь $\theta_{k+1}$. Очень похоже на градиентный подъём: мы сэмплим (формируя выборки) несколько точек вокруг себя и идём туда, где значение функции максимально. Поэтому отчасти можно считать, что Hill Climbing (Скалолазание) с большим $N$ – локальная оптимизация: можно взять

наилучшее направление изменения $\theta$ из, например, градиентов, но можем поискать хорошее направление, условно, случайным перебором.

Что получается: если мутация такова, что $\forall\theta, \hat{\theta}: m(\hat{\theta}|\theta) > 0$, остаются гарантии оказаться в любой точке пространства, и алгоритм остаётся методом глобальной оптимизации. При этом, мутация может быть устроена так, что вероятность оказаться «неподалёку» от родителя выше, чем в остальной области пространства.

Понятно, если мутация генерирует очень непохожие на родителя особи, алгоритм схлопывается примерно в случайный поиск. И понятно, что если мутация, наоборот, с огромной вероятностью генерирует очень близкие к родителю особи, алгоритм, помимо того, что будет сходиться медленно, будет сильно надолго застревать в локальных оптимумах. Возникает компромисс между использованием и исследованием: баланс между выбором уже известных хороших точек и поиском новых «вдали»; изучением окрестностей найденных локальных оптимумов и поиском новых.

*Имитация отжига*

Имитация отжига решает «проблему исследования» при помощи более умной процедуры отбора: вероятность выбрать потомка $\theta'_{k+1} \sim m(\theta'_{k+1}\theta|\theta_k)$, а не остаться в родительской точке $\theta_k$, вводится так:

$$select\ (\theta_{k+1} = \theta'_{k+1}) := \ min\left(1, exp\frac{\hat{J}(\theta'_{k+1}) - \hat{J}(\theta_k)}{\tau_k}\right)$$

где $\tau_k > 0$ – температура, гиперпараметр, зависящий от номера итерации. Иными словами, если новая точка более приспособлена, то мы «принимаем» новую точку $\theta'_{k+1}$ с вероятностью 1; если же новая точка менее приспособлена, мы не выкидываем её, а переходим в неё с некоторой вероятностью. Эта вероятность тем ближе к единице, чем «похожее» значения оракула (предсказателя), и температура регулирует понятие похожести между скалярами относительно масштаба оптимизируемой функции $J$.

Наша цепочка $\theta_0, \theta_1, \theta_2 \ldots$ задаёт марковскую цепь: мы генерируем каждую следующую особь на основе только предыдущей, используя некоторое стохастичное правило перехода. Теория марковских цепей говорит, что может существовать стационарное распределение: распределение, из которого приходят $\theta_k$, при стремлении $k \to \infty$ всё ближе к некоторому распределению $p(\theta)$, которое определяется лишь функцией переходов и не зависит от инициализации $\theta_0$.

Теорема – Алгоритм Метрополиса-Гастингса: Пусть в пространстве $\Theta$

задано распределение $p(\theta)$ и распределение $q(\hat{\theta}|\theta)$, удовлетворяющее $\forall\hat{\theta},\theta: q(\hat{\theta}|\theta) > 0$. Пусть строится цепочка $\theta_0, \theta_1, \theta_2 \ldots$ по следующему правилу: генерируется $\theta'_{k+1} \sim q(\theta'_{k+1}|\theta_k)$, после чего с вероятностью

$$min\left(1, \frac{p(\theta'_{k+1})}{p(\theta_k)} \frac{q(\theta_k|\theta'_{k+1})}{q(\theta'_{k+1}|\theta_k)}\right)$$

$\theta_{k+1}$ полагается равным $\theta'_{k+1}$, а иначе $\theta_{k+1} := \theta_k$. Тогда для любого $\theta_0$:

$$\lim_{k\to\infty} p(\theta_k) = p(\theta)$$

Отсюда следует утверждение: Пусть оракул точный, то есть $\hat{J}(\theta) \equiv J(\theta)$, а мутация удовлетворяет

$$\forall\theta,\hat{\theta}: m(\hat{\theta}|\theta) = m(\theta|\hat{\theta}) > 0$$

Тогда, если температура $\tau$ не зависит от итерации, для любой инициализации $\theta_0$ алгоритм имитации отжига строит марковскую цепь со следующим стационарным распределением:

$$\lim_{k\to\infty} p(\theta_k) \propto exp\frac{J(\theta_k)}{\tau}$$

Алгоритм Метрополиса даёт, вообще говоря, «гарантии сходимости» для имитации отжига: то, что через достаточно большое количество итераций мы получим сэмпл (данные) из распределения $exp\frac{J(\theta_k)}{\tau}$, нас, в общем-то, устраивает. Если температура достаточно маленькая, это распределение очень похоже на вырожденное в точке максимального значения оптимизируемой функции. Одновременно, правда, маленькая температура означает малую долю исследований в алгоритме, и тогда процедура вырождается в наивный поиск при помощи мутации. Поэтому на практике температуру снижают постепенно; подобный отжиг часто применяется для увеличения доли исследований в начале работы алгоритма и уменьшения случайных блужданий в конце.

*Эволюционные алгоритмы*

Hill Climbing (Скалолазание) – эвристика с «одним состоянием»: есть какая-то одна текущая основная особь-кандидат, на основе которой и только которой составляется следующая особь-кандидат. Нам, вообще говоря, на очередном шаге доступна вся история проверенных точек. Первый вариант — построить суррогат-приближение $\hat{J}(\theta)$, которую легко можно прооптимизировать и найти так следующего кандидата (к нему относятся, например, алгоритмы на основе гауссовских процессов), но этот вариант не сработает в сложных пространствах $\Theta$. Второй вариант – перейти к «эвристикам с $N$ состояниями», то есть использовать последние $N$ проверенных особей для порождения новых кандидатов.

Определение 9: Алгоритм называется эволюционным, если он строит последовательность популяций $\mathcal{P}_1, \mathcal{P}_2 \ldots$ , на $k$-ом шаге строя очередное поколение $\mathcal{P}_k$ на основе предыдущего.

Практически все мета-эвристики сводятся к эволюционным алгоритмам. При этом заметим, что, пока единственным инструментом «генерации» новых особей выступает мутация, алгоритмы различаются только процедурой отбора. Таким образом, в алгоритме всегда поддерживается текущая популяция из $N$ особей (первая популяция генерируется из некоторой стратегии перебора $q(\theta)$), из них отбирается $N$ особей (отбор может выбирать одну особь несколько раз, поэтому этот шаг нетривиален), и дальше каждая отобранная особь мутируется. Эвристика элитизма предлагает некоторые из отобранных особей не мутировать, и оставить для следующей популяции; это позволяет уменьшить шансы популяции «потерять» найденную область хороших значений функции, но увеличивает шансы застревания в локальном оптимуме. При элитизме для каждой особи хранится её возраст— число популяций, через которые особь прошла без мутаций; далее этот возраст влияет на отбор, например, выкидывая все особи старше определённого возраста.

Рассмотрим простейший эволюционный алгоритм. Любой алгоритм локальной оптимизации (градиентный спуск или Hill Climbing (Скалолазание)), результат работы которого зависит от начального приближения $\theta_0 \sim q(\theta)$, можно «заменить» на метод глобальной оптимизации, запустив на каждом условном сервере по «потоку» со своим начальным $\theta_0$. Время работы алгоритма не изменится (считая, конечно, что сервера работают параллельно), а обнаружение неограниченного числа локальных оптимумов с ненулевым шансом найти любой гарантирует нахождение глобального. Набор из текущих состояний всех потоков можно считать текущей популяцией.

При этом, любая процедура отбора позволит потокам «обмениваться информацией между собой». Для примера рассмотрим распространённую схему $(M, K)$-эволюционной стратегии, в которой процедура отбора заключается в том, чтобы топ-$M$ особей отобрать по $K$ раз каждую (дать каждой особи из топа породить $K$ детей). Иными словами, параллельно ведём M скалолазов, в каждом из которых для одного шага генерируется $K$ потомков; если число потоков $M = 1$, алгоритм вырождается в обычный Hill Climbing (Скалолазание). Порождённые $N = MK$ особей образуют текущую популяцию алгоритма. Среди них отбирается $M$ лучших особей, которые могут как угодно распределиться по потокам. Получится, что некоторые потоки, которые не нашли хороших областей $\theta$, будут прерваны, а хорошие получат возможность сгенерировать больше потомков и

как бы «размножаться» на несколько процессов.

Алгоритм: (M, K)-эволюционная стратегия

Вход: оракул $\hat{J}(\theta)$

Гиперпараметры: $m(\hat{\theta}|\theta)$ – мутация, $q(\theta)$ – стратегия перебора, $M$ – число потоков, $K$ – число сэмплов на поток

Инициализируем $\mathcal{P}_0 := (\theta_i \sim q(\theta)|i \in \{1,2,\dots,MK\})$

На $k$-ом шаге:

1. проводим жадный отбор: $\mathcal{P}_k^+ := select_M^{top}(\mathcal{P}_k, \hat{J}(\mathcal{P}_k))$

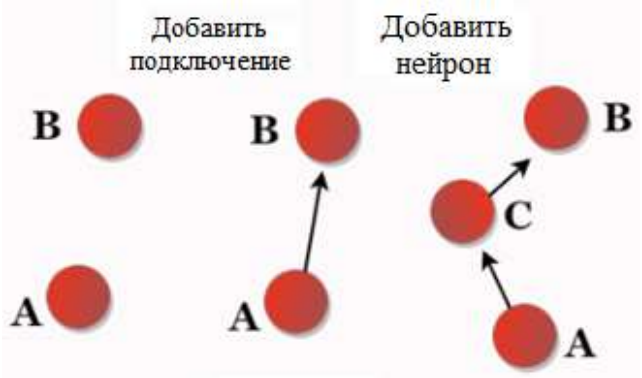
2. размножаем: $\mathcal{P}_{k+1} := \left(\hat{\theta}_i \sim m(\hat{\theta}|\theta)|\theta \in \mathcal{P}_k^+, i \in \{1,2,\dots,K\}\right)$

*Weight Agnostic Neural Networks (WANN) – Нейронные сети, не зависящие от веса*

Поскольку мета-эвристики — универсальный метод оптимизации, они могут применяться к нейронным сетям. Такой подход даёт такие преимущества, как возможность использовать дискретные веса или недопустимые для градиентной оптимизации функции активации вроде пороговой функции Хевисайда (пороговая функция).

Особенностью нейроэволюции является возможность искать топологию сети вместе со значениями самих весов: для этого достаточно предложить некоторый оператор мутации. Рассмотрим распространённый пример: пусть дана некоторая нейросеть с произвольной топологией и некоторыми весами. Для весов процедуру мутирования можно взять стандартную (добавление шума с заданной дисперсией; дополнительно можно для каждого веса сэмплировать бинарную величину, будет ли данный вес мутировать). Далее случайно выбирается, будет ли проходить мутация архитектуры (топологии) сети, и если да, то какого типа (обычно рассматривается несколько типов мутации).

Например – топологические мутации нейросети. Распространён набор из трёх видов топологических мутаций: добавление связи, добавление нейрона и смена функции активации.




Добавление связи означает, что случайно выбираются два ранее не

соединённых связью нейрона, и выход одного добавляется ко входу в другой (вес инициализируется, например, случайно). Какой из двух нейронов является входом, а какой — выходом, однозначно определяется требованием ацикличности к вычислительному графу.

Под добавлением нейрона понимается именно разбиение уже имеющейся связи: имевшаяся связь A–B, где A, B — нейроны, выключается, и появляются связи A–C и C–B, где C — новый нейрон. Новые нейроны необходимо добавлять именно так, чтобы они сразу же участвовали в вычислительном процессе.

Смена функции активации меняет функцию активацию в случайном нейроне на произвольную из предопределённого набора.

*Видовая специализация*

Возникает простой вопрос к $(M, K)$-эволюционной стратегии: а не случится ли такого, что он схлопнется к Hill Climbing (скалалазанию), если в очередной популяции среди топ-$M$ останутся только дети одного и того же родителя? Получится, что, хоть мы и исходили из идеи исследовать параллельно много локальных оптимумов, жадный отбор может убить все потоки, кроме одного, и «область пространства аргументов», покрываемая текущей популяцией, «схлопнется».

Для борьбы с этим эффектом в мета-эвристиках рассматривают методы защиты инноваций. Если для получения новых хороших свойств необходимо сделать «несколько» шагов эволюции, то мы каким-то образом помогаем выживать особям, оказавшихся в не исследуемых местах пространства $\Theta$. Самый простой способ – использование более «мягких» процедур отбора, когда у неприспособленной особи есть небольшой шанс выжить. Более интеллектуально было бы как-то оценить, насколько «новой» является область пространства аргументов, в которой оказалась особь, и дать ей больше шансов выжить, если алгоритм эту область ещё не рассматривал: ведь проблема мягких процедур отбора, очевидно, в том, что слабые особи выживают в том числе там, где функция уже в достаточной степени исследована.

Рассмотрим общую идею видов. Допустим, мы сможем в $\Theta$ придумать метрику (или хоть функцию близости) $\rho(\theta_1, \theta_2)$ и на её основе разбить все особи популяции $\mathcal{P}$ на непересекающиеся множества – «виды». Процесс разбиения, что важно, не обязан удовлетворять каким-то особым свойствам и может быть стохастичным, в том числе чтобы быть вычислительно дешёвым.

Разделение на виды позволяет делать, например, explicit fitness sharing (явный обмен фитнесом): особи соревнуются только внутри своих видов. Пусть $\tilde{\mathcal{P}} \subseteq \mathcal{P}$ – вид, а $\hat{J}_{mean}(\tilde{\mathcal{P}})$ – среднее значение приспособленности в данном виде.

Тогда на основе этих средних значений между видами проводится (в некоторой «мягкой» форме – слабые виды должны выживать с достаточно высокой вероятностью) некоторый мягкий отбор; например, виду $\tilde{\mathcal{P}}$ позволяется сгенерировать потомков пропорционально $\exp \hat{J}_{mean}(\tilde{\mathcal{P}})$ с учётом того, что в сумме все виды должны породить заданное гиперпараметром число особей. Генерация необходимого числа потомков внутри каждого вида происходит уже, например, стандартным, «агрессивным» образом: отбирается некоторая доля топ-особей, к которым и применяется по несколько раз мутация.

Виды защищены мягким отбором, и поэтому застрявшие вдали особи, образующие новый вид, будут умирать реже; при этом в скоплениях слабых особей в одном месте пройдёт жёсткий внутривидовой отбор, а сам вид получит не так много «слотов потомства», и число точек сократится.

*Генетические алгоритмы*

До сих пор в наших лекциях создавались новые особи только при помощи мутации. В генетических алгоритмах дополнительно вводится этап рекомбинации, когда новые особи можно строить на основе сразу нескольких особей, как-то «совмещая» свойства тех и других в надежде получить «лучшее от двух миров»; найти хороший оптимум между двумя локальными оптимумами. В ванильной версии генетических алгоритмов у детей по два родителя, хотя можно рассматривать и скрещивание большего числа особей:

Определение: Кроссинговером называется распределение $c(\hat{\theta}|\theta_1, \theta_2)$, где $\theta_1, \theta_2$ называются родителями (parent), $\hat{\theta}$ – потомком (child).

Чтобы сохранить свойство «глобальности» оптимизации, чтобы мы могли при помощи такого инструмента порождения оказаться в любой точке пространства, т.е. $\forall \hat{\theta}, \theta_1, \theta_2 \in \Theta: c(\hat{\theta}|\theta_1, \theta_2) > 0$. Однако, для практически любых примеров кроссинговера это не так. Поэтому считается, что это требование НЕ выполняется: процедура рекомбинации, возможно, стохастична, но всегда приводит к точке «между» $\theta_1$ и $\theta_2$. Это означает, что, используя только кроссинговер, область, покрываемая потомством, будет уже области, покрываемой родителями: теряется исследование. Чтобы «вылечить» это, в генетических алгоритмах всё равно остаётся этап применения мутации.

Алгоритм: Генетический поиск

Дано: оракул $\hat{J}(\theta)$

Гиперпараметры: $c(\hat{\theta}|\theta_1, \theta_2)$ – кроссинговер, $m(\hat{\theta}|\theta)$ – мутация, $select(\mathcal{P}^+|\mathcal{P}, \hat{J}(\mathcal{P}))$ – процедура отбора, $q(\theta)$ – стратегия перебора, $N$ – размер популяции.

Инициализируем $P_0 := (\theta_i \sim q(\theta) | i \in \{1, 2, \ldots, N\})$

На $k$-ом шаге:

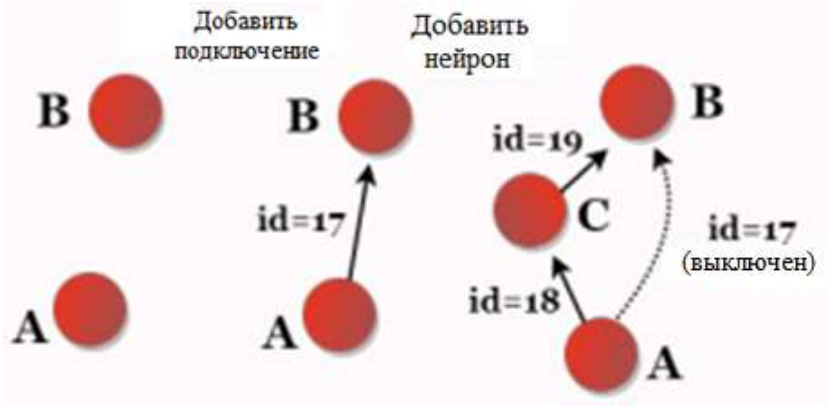
1. проводим отбор: $\mathcal{P}_k^+ \sim select(\mathcal{P}_k^+ | \mathcal{P}_k, \hat{J}(\mathcal{P}_k))$

2. проводим размножение: $\mathcal{P}_{k+1} := (\theta_i \sim c(\theta | \theta_l, \theta_r) | \theta_l, \theta_r \in \mathcal{P}_k^+)$

3. проводим мутирование: $\mathcal{P}_{k+1} \leftarrow \left(\hat{\theta} \sim m(\hat{\theta} | \theta) | \theta \in \mathcal{P}_{k+1}\right)$

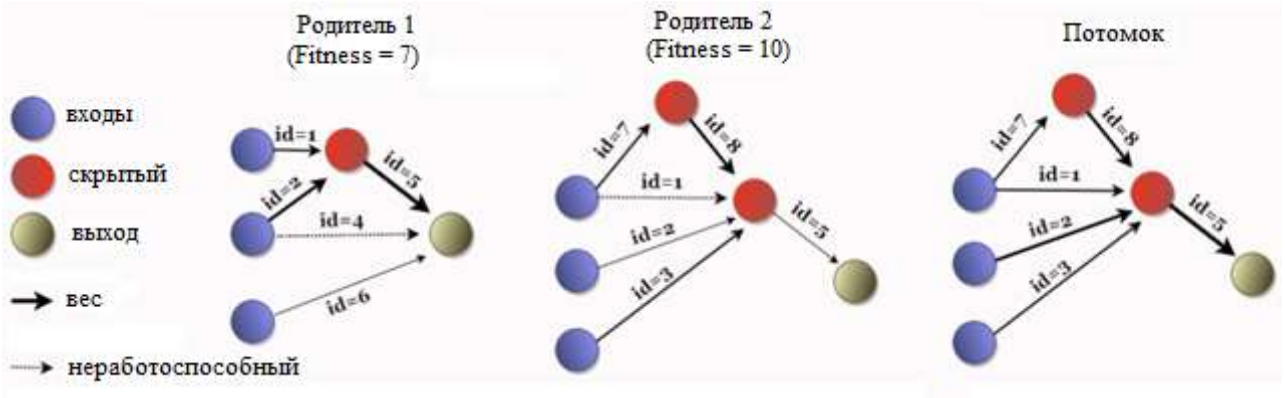
Пример – Neuroevolution of Augmented Topologies – Нейроэволюция расширенных топологий

Рассмотрим в качестве примера набор эвристик нейроэволюционного алгоритма NEAT, который был основан на генетическом поиске, то есть дополнительно вводил оператор кроссинговера для нейросетей (двух произвольных топологий).




В NEAT все связи всех особей, помимо веса, имеют статус «включена-выключена» и уникальный идентификационный номер (id), или исторический маркер (historical marker). Связи могут появляться только в ходе мутаций (используется оператор мутации из предыдущего примера); в момент создания связи ей присваивается уникальный (в рамках всего алгоритма) id и статус «включена». Наличие у двух особей связи с одним id будет означать наличие общего предка. У изначальной «пустой» топологии связей нет вообще. Связь может попасть в статус «выключена» только во время мутации вида «добавление нейрона», когда имевшаяся связь A–B «исчезает»: то есть, соответствующий «ген» не удаляется из генома особи, а переходит в статус «выключена». Выключенная связь означает, что у особи был предок, у которого связь была включена.

Как введение статусов и id связей позволяет устраивать между разными топологиями кроссинговер? Если связь с данным id имеется у обоих скрещиваемых особей, связь сохраняется и у потомка, с весом и статусом случайного родителя. Связи, имеющиеся только у одного из родителей, будем называть непарными; они копируются из того родителя, чья приспособленность выше (или из обоих сразу, если приспособленности одинаковые).




NEAT также использует видовую специализацию для процесса отбора, для чего на таких генотипах необходимо задать метрику $\rho$. Пусть $\theta_1$, $\theta_2$ – две особи, $G_1$, $G_2$ – количество связей у этих особей, $D$ – число непарных связей, $w_1$, $w_2$ — веса особей в парных связях. Понятно, что веса можно сравнить только для парных связях, и понятно, что чем больше непарных связей, тем больше должно быть расстояние. В NEAT предлагается просто объединить эти два критерия:

$$\rho(\theta_1, \theta_2) := \alpha_1 \frac{D}{max(G_1, G_2)} + \alpha_2 \|w_1 - w_2\|$$

где $\alpha_1$, $\alpha_2$ — гиперпараметры. Внутри самих видов отбор жадный.

NEAT — исторически один из первых алгоритмов, которые можно с каким-то результатом применить для RL задач без подготовленного удобного признакового описания состояний, например, изображений. Можно найти много интересных примеров применения алгоритма к разным задачам, с визуализацией получающейся сети

**Эволюционные стратегии**

*Идея эволюционных стратегий*

Когда пытаются генерировать $k+1$-ое поколение, используя особей $k$-го поколения в качестве исходного материала, единственная зависимость от всей истории заложена в составе $k$-ой популяции. Можно рассмотреть распределение, из которого появляются особи очередной популяции: весь смысл нашей процедуры отбора в том, чтобы это распределение для очередной итерации поменялось так, что вероятность появления более хороших особей стала выше. Давайте обобщим эту идею: будем в явном виде хранить распределение для порождения особей новой популяции и по информации со всей популяции аккумулировать всю информацию внутри его параметров — «скрещивать все особи»

Определение: Распределение $q(\theta|\lambda_k)$, из которого генерируются особи $k$-ой популяции, называется эволюционной стратегией (evolutionary strategy, $ES$):

$$\mathcal{P}_k := \{\theta_i \sim q(\theta|\lambda_k) | i \in \{1,2 \dots N\}\}$$

где $\lambda_k$ – параметры эволюционной стратегии.

Из каких соображений подбирать $\lambda_k$ на очередном шаге? В принципе, хочется найти такой генератор особей, что их оценки как можно больше, то есть нащупать область $\Theta$ с высоким значением $J(\theta)$. Эти соображения можно формализовать довольно по-разному и таким образом оправдывать разные метаэвристики. В частности, можно сказать, что $\lambda$ есть особь или несколько особей, но можно отойти от пространства Θ и учить модель-генератор особей с какой-то хорошей параметризацией λ. Дальше рассмотрим две основные идеи, как это можно делать.

*Оценка вероятности редкого события*

Первую идею возьмём немного сбоку. Допустим, стоит задача оценки вероятности редкого события:

$$l = P(f(x) \geq \gamma) = \mathbb{E}_{x \sim p(x)}[f(x) \geq \gamma] — ? \qquad (2.1)$$

где $p(x)$ — некоторое распределение, $f: X \to \mathbb{R}$ – функционал, $\gamma$ — некоторый порог.

Под словами «редкое событие» подразумевается, что условие внутри индикатора $f(x) \geq \gamma$ выполняется с вероятностью, крайне близкой к нулю. Это означает, что Монте-Карло оценка с разумным на практике числом сэмплов $N$ выдаст или ноль или $1/N$, если один раз повезёт; короче, лобовой подход не годится.

Хочется сэмплировать $x$ не из того распределения, которое нам дали — $p(x)$, — а из чего-нибудь получше. Для этого применим выборку по важности (importance sampling) с некоторым распределением $q(x)$, которое будем выбирать сами:

$$l = \mathbb{E}_{x \sim p(x)} \frac{p(x)}{q(x)} [f(x) \geq \gamma] \qquad (2.2)$$

Нам хочется выбрать такое $q(x)$, чтобы дисперсия Монте-Карло оценки такого интеграла была как можно меньше. Желание может быть исполнено:

Утверждение: Дисперсия Монте-Карло оценки (2.2) минимальна при

$$q(x) \propto p(x)[f(x) \geq \gamma] \qquad (2.3)$$

Доказательство. Искомое значение $l$ (2.1) является нормировочной константой такого распределения. Подставим данное $q(x)$ в подынтегральную функцию

$$\frac{p(x)}{q(x)} [f(x) \geq \gamma] = l \frac{p(x)[f(x) \geq \gamma]}{p(x)[f(x) \geq \gamma]} = l$$

Поскольку всё сократилось, для любых сэмплов $x \sim q(x)$ значение Монте-

Карло оценки будет равно l; то есть, дисперсия равна нулю.

Посчитать такое $q(x)$ мы не можем, однако можем пытаться приблизить в параметрическом семействе $q(x|\lambda)$, минимизируя, например, такую $KL$-дивергенцию (расхождение в ходе эволюции признаков, возникших от общего предка.):

$$KL(q(x) \parallel q(x|\lambda)) = const(\lambda) - \mathbb{E}_{q(x)} log q(x|\lambda) \to min$$

Единственное зависящее от параметров $\lambda$ слагаемое называется кросс-энтропией (cross entropy) и даёт название методу.

Для такой оптимизации всё равно нужно уметь сэмплировать из $q(x)$. Однако от задачи оценки числа стал переход к поиску распределения. Поскольку это распределение строили так, чтобы оно помогало сэмплировать нам точки из редкого события, можно воспользоваться им же с прошлой итерации, чтобы помочь решать ту же задачу лучше. А то есть: строим последовательность $q(x|\lambda_k)$, $\lambda_0$ — любое, на очередной итерации:

$$\lambda_{k+1} = \operatorname*{argmin}_{\lambda} -\mathbb{E}_{q(x)} \log q(x \mid \lambda) =$$

$$\{\text{подставляем вид оптимального } q(x) \text{ из (2.3)}\} = \operatorname*{argmin}_{\lambda} -\mathbb{E}_{p(x)} [f(x) \geq \gamma] \log q(x \mid \lambda) =$$

$$\{\text{важность выборки через } q(x \mid \lambda_k)\} = \operatorname*{argmin}_{\lambda} -\mathbb{E}_{q(x|\lambda_k)} \frac{p(x)}{q(x \mid \lambda_k)} [f(x) \geq \gamma] \log q(x \mid \lambda)$$

Каждая задача нахождения $\lambda_k$ всё ещё тяжела в связи с тем, что подынтегральное выражение всё ещё почти всегда ноль. Ключевая идея: поскольку мы теперь строим целую последовательность, мы можем поначалу решать сильно более простую задачу, разогревая $\gamma$. Будем на $k$-ом шаге брать $\gamma$ не из условия задачи, а поменьше, так, чтобы с итерациями $\gamma$ увеличивалась (и мы решали бы задачу, всё более похожую на ту, что требовалось решить исходно), и одновременно достаточное число сэмплов значения подынтегральной функции были отличны от нуля.

Важно, что мы можем не задавать заранее последовательность $\gamma_k$, а определять очередное значение прямо на ходу, например, исходя из сэмплов $x_1 \dots x_N \sim q(x|\lambda_k)$ и значений $f(x)$ в них.

Алгоритм: Метод Кросс-Энтропии для оценки вероятности редкого события

**Вход:** распределение $p(x)$, функция $f(x)$, порог $\gamma$
**Гиперпараметры:** $q(x \mid \lambda)$ — параметрическое семейство, $N$ — число сэмплов, $M$ — порог отбора

Инициализируем $\lambda_0$ произвольно.
**На $k$-ом шаге:**

1. сэмплируем $x_1 \dots x_N \sim q(x \mid \lambda_k)$
2. сортируем значения $f(x_i)$: $f_{(1)} \le f_{(2)} \le \dots \le f_{(N)}$
3. полагаем $\gamma_k := \min(\gamma, f_{(M)})$
4. решаем задачу оптимизации:

$$\lambda_{k+1} \leftarrow \underset{\lambda}{\operatorname{argmax}} \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}[f(x_j) \ge \gamma_k] \frac{p(x_j)}{q(x_j \mid \lambda_k)} \log q(x_j \mid \lambda) \tag{2.4}$$

5. **критерий останова:** $\gamma_k = \gamma$

**Получение итоговой оценки:**

1. сэмплируем $x_1 \dots x_N \sim q(x \mid \lambda_k)$
2. возвращаем

$$l \approx \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}[f(x_j) \ge \gamma_k] \frac{p(x_j)}{q(x_j \mid \lambda_k)}$$

Метод Кросс-Энтропии для стохастической оптимизации

Из рассуждений было видно, что мы практически учим q(x|λ) нащупывать область с высоким значением заданной функции без использования какой-либо информации о ней. Поэтому можно адаптировать метод, чтобы он стал мета-эвристикой. Для этого вернёмся к нашей задаче безградиентной оптимизации:

$$J(\theta) \to max$$

и перепишем предыдущий алгоритм в условиях, когда порог γ «не ограничен», ну или что тоже самое, $\gamma := maxJ(\theta)$.

Формально можем выбирать любое $p(x)$; положим $p(x) := q(x|\lambda_k)$, просто чтобы в задаче (2.4) сократилась коррекция важность выборки (importance sampling). Получим очень простой на вид алгоритм, в котором фактически на очередном шаге минимизируется такое расстояние:

$$\mathbf{KL}([J(x) \ge \gamma_k] q(x \mid \lambda_{k-1}) \parallel q(x \mid \lambda_k)) \to \min_{\lambda_k},$$

где первое распределение задано с точностью до нормировочной константы.

Алгоритм: Метод Кросс-Энтропии для оптимизации с оракулом нулевого порядка

**Вход:** оракул $\hat{J}(\theta)$
**Гиперпараметры:** $q(\theta \mid \lambda)$ — параметрическое семейство, $N$ — число сэмплов, $M$ — порог отбора

Инициализируем $\lambda_0$ произвольно.
**На $k$-ом шаге:**

1. сэмплируем $\mathscr{P}_k := (\theta_i \sim q(\theta \mid \lambda_k) \mid i \in \{1, 2, \dots, N\})$
2. проводим отбор $\mathscr{P}_k^+ := \mathbf{select}_M^{top}(\mathscr{P}_k)$
3. решаем задачу оптимизации:

$$\lambda_{k+1} \leftarrow \underset{\lambda}{\operatorname{argmax}} \sum_{\theta \in \mathscr{P}_k^+} \log q(\theta \mid \lambda)$$

Видно, что по сути действуем эволюционно: хотим генерировать при помощи распределения $q$ точки из области, где значение функции велико; берём и сэмплируем несколько точек из текущего приближения; из сгенерированных отбираем те, где значение функции было наибольшим и учим методом максимального правдоподобия повторять эти точки. Поскольку некоторая доля плохих точек была выкинута из выборки, распределение, которое учит очередное $q(x|\lambda_k)$, лучше предыдущего. Это первый способ обучения эволюционных стратегий.

Метод Кросс-Энтропии для обучения с подкреплением (CEM)

В обучении с подкреплением в кросс-энтропийном методе можно сделать ещё один очень интересный шаг. В отличие от всех остальных рассматриваемых мета-эвристик, мы можем проводить эволюционный отбор не в пространстве возможных стратегий (в пространстве $\Theta$), а в пространстве траекторий. У нас будет одна текущая стратегия, из которой мы сгенерируем несколько траекторий, и в силу стохастичности некоторые из этих траекторий выдадут лучший результат, чем другие. Мы отберём лучшие и будем методом максимального правдоподобия (по сути, имитационным обучением) учиться повторять действия из лучших траекторий.

Алгоритм: Cross Entropy Method (Метод перекрестной энтропии)

**Гиперпараметры:** $\pi(a \mid s, \theta)$ — стратегия с параметрами $\theta$, $N$ — число сэмплов, $M$ — порог отбора

Инициализируем $\theta_0$ произвольно.

**На $k$-ом шаге:**

1. сэмплируем $N$ траекторий $\mathcal{T}_1 \dots \mathcal{T}_N$ игр при помощи стратегии $\pi(a \mid s, \theta_k)$
2. считаем кумулятивные награды $R(\mathcal{T}_i)$
3. сортируем значения: $R_{(1)} \le R_{(2)} \le \dots \le R_{(N)}$
4. полагаем $\gamma_k := R_{(M)}$
5. решаем задачу оптимизации:

$$\theta_{k+1} \leftarrow \operatorname*{argmax}_{\theta} \frac{1}{N} \sum_{j=1}^{N} \mathbb{I}[R(\mathcal{T}_j) \ge \gamma_k] \sum_{s,a \in \mathcal{T}_j} \log \pi(a \mid s, \theta)$$

Натуральные эволюционные стратегии (NES)

Рассмотрим альтернативный вариант подбора параметров эволюционной стратегии q(θ|λ). Будем подбирать λ, исходя из следующего функционала:

$$g(\lambda) := \mathbb{E}_{\theta \sim q(\theta|\lambda)} J(\theta) \to \max_{\lambda} \tag{2.5}$$

Будем оптимизировать этот функционал градиентно по $\lambda$.

*Теорема*

$$\nabla_\lambda g(\lambda) = \mathbb{E}_{\theta \sim q(\theta|\lambda)} \nabla_\lambda \log q(\theta|\lambda) J(\theta) \tag{2.6}$$

Итак, есть следующая идея: сгенерируем популяцию $\mathcal{P}_k$ при помощи $q(\theta|\lambda_k)$, после чего воспользуемся особями как сэмплами для несмещённой оценки градиента функционала (2.5), чтобы улучшить параметры $\lambda$ и впоследствии сгенерировать следующее поколение $\mathcal{P}_{k+1}$ из более хорошего распределения.

Адаптация матрицы ковариации (CMA-ES)

Более глубокомысленно было бы для $\Theta \equiv \mathbb{R}^h$ (матрица ковариации размера $\mathbb{R}^{h\times h}$, где $h$ — количество параметров нашей стратегии) адаптировать не только среднее, но и матрицу ковариации, которая имеет смысл «разброса» очередной популяции. Covariance Matrix Adaptation Evolution Strategy (CMAES - Ковариационная матрица Адаптация Эволюция Стратегия) — алгоритм, включающий довольной большой набор эвристик, в основе которого лежит эволюционная стратегия, адаптирующая не только среднее, но и матрицу ковариации:

$$q(\theta|\lambda) := N(\mu, \Sigma)$$

где $\lambda := (\mu, \Sigma)$.

Рассмотрим только основную часть формул алгоритма, касающихся формулы обновления $\Sigma$. Посмотрим на формулу для обновления среднего (с точностью до скорость обучения):

$$\mu_{k+1} = \mu_k + \alpha \sum_{\theta \in \mathcal{P}_k} \underbrace{\hat{J}(\theta)}_{\text{вес}} \underbrace{(\theta - \mu_k)}_{\substack{\text{«предлагаемое»} \\ \text{особью изменение}}} \tag{2.7}$$

Для обновления матрицы ковариации будем рассуждать также: каждая особь популяции $\theta$ «указывает» на некоторую ковариацию $(\theta - \mu_k)(\theta - \mu_k)^{\mathrm{T}}$ и таким образом «предлагает» следующее изменение:

$$(\theta - \mu_k)(\theta - \mu_k)^{\mathrm{T}} - \Sigma_k$$

где $\Sigma_k$ — матрица ковариации на текущей итерации. Усредним эти «предложения изменения» по имеющейся популяции, взвесив их на $\hat{J}(\theta)$, и получим «градиент» для обновления матрицы:

$$\Sigma_{k+1} := \Sigma_k + \alpha \sum_{\theta \in \mathcal{P}_k} \hat{J}(\theta) \left( (\theta - \mu_k)(\theta - \mu_k)^{\mathrm{T}} - \Sigma_k \right) \tag{2.8}$$

Здесь нужно оговориться, что мы используем оценку ковариации как бы «методом максимального правдоподобия при условии известного среднего $\mu_k$». Исторически к этим формулам пришли эвристически, но позже у формулы появилось теоретическое обоснование.

Теорема: Формулы (2.7) и (2.8) для обновления $\lambda = (\mu, \Sigma)$ являются формулами натурального градиентного спуска для (2.5).

Именно поэтому данный вид алгоритмов для обучения эволюционных стратегий называется «натуральными»: считается, что функционал (2.5) корректнее оптимизировать именно при помощи натурального градиентного спуска, инвариантного к параметризации $q(\theta|\lambda)$, а не обычного.

Если мы попробуем проделать с данным подходом (оптимизацией (2.5)) тот же трюк, что и с кросс-энтропийным методом, и попытаемся считать градиент «в пространстве траекторий», а не в пространстве стратегий, то получим методы оптимизации $J(\theta)$ уже первого порядка — «policy gradient – градиент политики» методы.