МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н.Г. ЧЕРНЫШЕВСКОГО»

Кафедра теоретических основ компьютерной безопасности и криптографии

Обучение нейросетей. Методы глобальной оптимизации.

РЕФЕРАТ

студента 5 курса 531 группы
специальности 10.05.01 Компьютерная безопасность
факультета компьютерных наук и информационных технологий
Дусалиева Тахира Ахатовича

Проверил
доцент _____ И. И. Слеповичев
подпись, дата

Саратов 2024

СОДЕРЖАНИЕ

ВВЕДЕНИЕ 3

1 Методы глобальной оптимизации 4

1.1 Метод имитации обжига 6

1.2 Эволюционный метод 10

1.3 Метод роя частиц 17

1.4 Другие методы глобальной оптимизации 22

2 Недостатки и преимущества. Сравнение алгоритмов 24

ЗАКЛЮЧЕНИЕ 26

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 28

ВВЕДЕНИЕ

Современные нейронные сети являются мощным инструментом для решения широкого спектра задач, начиная от распознавания изображений и заканчивая обработкой естественного языка. Однако их эффективность во многом зависит от качества обучения, что представляет собой значительную исследовательскую проблему. Одним из ключевых аспектов в процессе обучения нейросетей является выбор оптимальных параметров, который значительно влияет на конечные результаты модели.

На данном курсе были рассмотрены несколько методов обучения, именуемые градиентными. Однако градиентные методы обучения нейронных сетей являются локальными: они ведут к одному из локальных минимумов целевой функции, лежащему в окрестности точки начала обучения. Только в ситуации, когда значение глобального минимума известно, удается оценить, находится ли найденный локальный минимум в достаточной близости от искомого решения. Если локальное решение признается неудовлетворительным, следует повторить процесс обучения при других начальных значениях весов и с другими управляющими параметрами. Можно либо проигнорировать полученное решение и начать обучение при новых (как правило, случайных) значениях весов, либо изменить случайным образом найденное локальное решение (встряхивание весов) и продолжить обучение сети.

При случайном приращении весов переход в новую точку связан с определенной вероятностью того, что возобновление процесса обучения выведет поиск из "сферы притяжения" локального минимума. При решении реальных задач в общем случае даже приблизительная оценка глобального минимума оказывается неизвестной. По этой причине возникает необходимость применения методов глобальной оптимизации.

1 Методы глобальной оптимизации

Градиентный спуск—это одна из важнейших идей в области машинного обучения. В этом методе алгоритм, учитывая функцию затрат, итеративно выполняет шаги с наибольшим уклоном вниз, и, спустя достаточное число циклов, теоретически завершается в точке минимума. Впервые открытый Коши в 1847 году и позднее расширенный Хаскеллом Карри в 1944 году для задач нелинейной оптимизации, градиентный спуск использовался для всех видов алгоритмов, начиная с линейной регрессии и заканчивая глубокими нейронными сетями.

Несмотря на то, что этот метод и его видоизмененная версия в виде обратного распространения ошибки стали величайшим прорывом в машинном обучении, оптимизация нейронных сетей остается неразрешенной проблемой. В интернете многие доходят до заявлений, что градиентный спуск изжил себя. Несмотря на то, что такие утверждения, конечно, преувеличены, у этой методики действительно есть ряд проблем:

- Оптимизаторы часто застревают в глубоких локальных минимумах. Существуют методы, которые иногда помогают обходить эти проблемы. Например, импульс может переносить оптимизаторы через значительные подъемы, а пакетная нормализация сглаживает пространство ошибок. Однако локальный минимум остается причиной многих проблем ветвления в нейронных сетях;
- Оптимизаторы могут надолго застревать в локальных минимумах, и даже если им удается выйти из них, это требует много времени. Градиентный спуск, из-за своего низкого показателя сходимости, обычно является длительным методом, даже с адаптациями для больших наборов данных, такими как пакетный градиентный спуск;
- Градиентный спуск очень чувствителен к инициализации. Например, производительность может значительно возрасти, если оптимизатор

инициализируется рядом со вторым локальным минимумом, а не первым, но это определяется случайным образом;

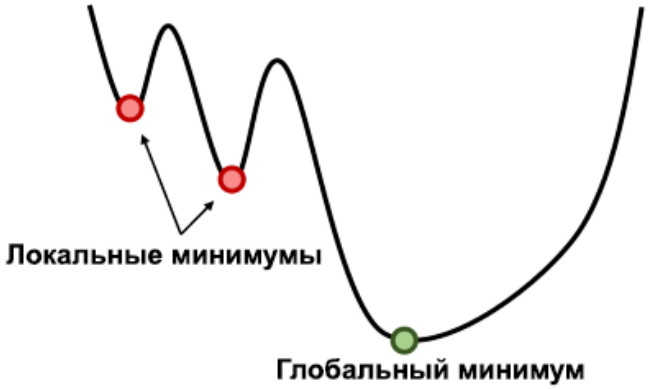

Рисунок 1 – Кривая функции потерь

- Темпы обучения определяют степень уверенности и рискованности оптимизатора. Завышенный темп обучения может привести к пропуску глобального минимума, а заниженная скорость слишком увеличивает время выполнения, что можно увидеть на рисунке 1. Чтобы решить эту проблему, темпы обучения уменьшаются постепенно, но выбрать правильную скорость уменьшения с учетом множества других переменных довольно сложно;
- Градиентный спуск требует градиентов, что делает его уязвимым к таким проблемам как:
    - взрывы градиентов;
    - затухания градиентов;
    - взрывы-затухания сигнала на прямом проходе;
    - плохие начальные инициализации, нестабильный старт обучения;
    - неспособность работать с не дифференцируемыми функциями.

Градиентный спуск был всесторонне изучен, и для его улучшения предложено множество решений. Некоторые из них являются его вариациями, другие основаны на архитектуре сети и работают лишь в определенных случаях.

Хотя градиентный спуск часто критикуется, это не означает, что он не является одним из лучших доступных методов.

Данная работа направлена на освещение некоторых методов глобальной оптимизации, не входящих в стандартный набор градиентных методов. Эти методы, как и любые другие способы повышения производительности нейронной сети, работают исключительно хорошо в одних сценариях и не так эффективно в других. Независимо от их эффективности в конкретных задачах, эти методы представляют собой увлекательную, креативную и многообещающую область для будущих исследований в машинном обучении.

В данной работе будут рассмотрены наиболее популярные методы глобальной оптимизации:

- Метод имитации обжига;
- Эволюционный метод;
- Метод роя частиц.

1.1 Метод имитации обжига

При обучении нейросетей, чтобы добавить «глобальности» в тот или иной метод оптимизации, можно применить метод имитации обжига на некотором этапе оптимизации.

Метод имитации отжига заимствует идеи из статистической механики и моделирует поведение расплавленного материала при отвердевании с управляемым охлаждением, когда температура постепенно снижается до нуля.

Во время медленного управляемого охлаждения, известного как отжиг, кристаллизация расплава сопровождается глобальным снижением его энергии. Допускаются кратковременные повышения энергии, например, при подогреве расплава для предотвращения слишком быстрого охлаждения. Эти временные повышения уровня энергии позволяют избежать ловушек локальных минимумов, которые могут возникнуть в процессе. Температура, сниженная до абсолютного нуля, делает дальнейшее повышение уровня энергии расплава невозможным. [1]

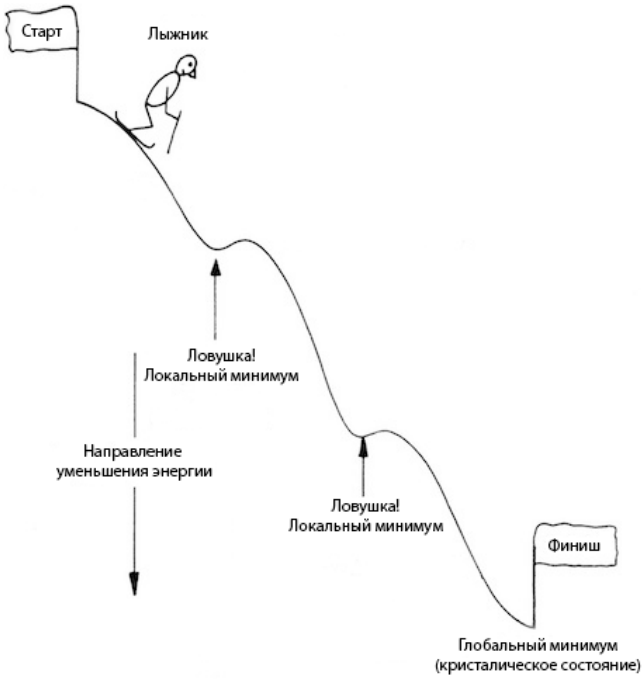

Рисунок 2 – Иллюстрация характеризирующая
метод имитации обжига

Для иллюстрации этого процесса можно представить лыжника, спускающегося по не совсем ровному склону, нашей целевой функции. Его цель – достичь самой низкой точки (глобального минимума). Однако склон неровный, и на нём есть низменности, которые могут показаться конечной точкой. Чтобы избежать ошибки, нам придётся проявить любопытство и подняться на небольшой холмик, чтобы продолжить спуск.

В этой метафоре склон представляет функцию измерения энергии вещества (целевую функцию), а вероятностный переход к состоянию с большей энергией – это карабканье на холм справа от низменности. [5]

Таким образом, этот алгоритм позволяет избежать застревания в локальном минимуме с высокой вероятностью.

Пусть $f: S \to R$ это целевая функция, где S – континуальное множество и $S \subseteq R^n$. Требуется найти $W^* \in S$, такое что $f(W^*) \leq f(W)$, $\forall W \in R^n$. Чтобы гарантировать существование W^*, будем также предполагать множество S компактным.

Впервые описанный в [2], алгоритм имитации отжига представляет собой термодинамический алгоритм нахождения глобального минимума целевой функции. В общем виде алгоритм может быть описан следующим образом:

Шаг 1. Запустить процесс из начальной точки W_0 при заданной начальной температуре $T = T_{\max}$;

Шаг 2. Выбрать новое решение W'_{l+1} из окрестности W_l;

Шаг 3. Выбирается случайная точка $r \in \{0,1\}$ согласно закону равномерного распределения. Следующее приближение выбирается по правилу:

$$W_{l+1} = \begin{cases} W'_{l+1}, & \text{если } r \le A(W_l, W'_{l+1}, T) \\ W_l, & \text{иначе} \end{cases};$$

Шаг 4. Уменьшить температуру $T = U(T_{max}, l)$. Если $T = 0$ перейти к шагу 5, иначе положить $l = l + 1$ вернуться к шагу 2;

Шаг 5. После снижения температуры до нуля провести обучение сети любым из *детерминированных методов* локальной оптимизации вплоть до достижения минимума целевой функции.

Наибольшего ускорения имитации отжига можно достичь путем замены случайных начальных значений весов w тщательно подобранными значениями с использованием любых доступных способов предварительной обработки исходных данных.

Благодаря функции выбора следующего приближения $A(W_l, W'_{l+1}, T)$, алгоритм на каждом шаге способен принимать решения, увеличивающие значение целевой функции, что позволяет избежать застревания в локальных минимумах. Одна из ключевых особенностей алгоритма, выгодно отличающих его от градиентных – отсутствие требований к дифференцируемости целевой функции f.

Заметим, что в большей части работ, посвящённых алгоритму имитации отжига, рассматривается применение алгоритма к решению задач дискретной

оптимизации. Несмотря на это, существует обоснованный интерес к возможности применения алгоритма в задачах непрерывной оптимизации [3, 4].

Так как функция выбора следующего приближения имеет особо важное место в алгоритме существует множество её вариантов. Например, для дискретного случая, в качестве функции, осуществляющей принятие решение о том, принимать ли новое значение в качестве решения, может быть выбрана функция Метрополиса [3]:

$$A(W, W', T) = \min\left[1, \exp\left(-\frac{f(W') - f(W)}{T}\right)\right].$$

Функция Метрополиса всегда принимает положительное решение о принятии очередного w' в качестве нового приближения в случае, если оно уменьшает значение f, т.е. $f(W') \leq f(W)$. Вероятность принятия положительного решения в случае увеличения значения целевой функции зависит от температуры T. Легко заметить, что в случае, когда T стремится к 0, то и $A(W, W', T) \to 0$ при $l \to \infty$. Таким образом вероятность принятия нового приближения, увеличивающего значение f уменьшается с течением времени. Другим возможным вариантом функции принятия нового приближения может являться, т. н. критерий Баркера [2]:

$$A(W, W', T) = \left[1 + \exp\left(\frac{f(W') - f(W)}{T}\right)\right]^{-1}.$$

Важным отличием критерия Баркера от функции Метрополиса является то, что первый может не принимать решения, уменьшающие значение целевой функции, особенно те, что незначительно улучшают его. Однако с уменьшением температуры T вероятность принятия лучших решений увеличивается.

Функция U служит для перерасчёта температуры и существует несколько подходов к её заданию. В самом простом варианте функция охлаждения зависит только от номера итерации и начального значения температуры. В некоторых источниках [2] предлагается определять температуру в каждой точке индивидуально, т.е. $T = U(T, l, W_{l+1})$.

В соответствии с определением функции $A(W, W', T)$, функция определения температуры должна быть монотонно убывающей. Исторически [3], первой схемой охлаждения была, т.н. схема Больцмана:

$$T = \frac{T_{max}}{\ln(1 + l)}.$$

Основной недостаток схемы Больцмана — медленное убывание температуры, что может приводить к медленной сходимости алгоритма. Альтернативной схемой охлаждения является схема Коши:

$$T = \frac{T_{max}}{1 + l}.$$

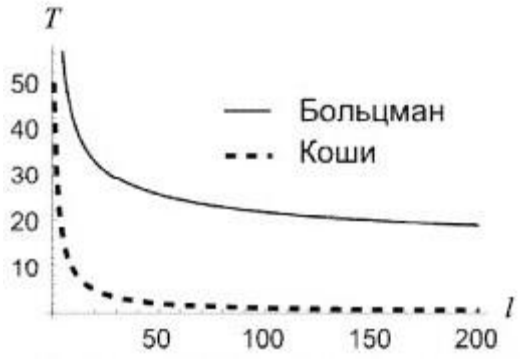

Рисунок 3 – Схема изменения температура
от 100 градусов при использовании
схем Больцмана и Коши

На рисунке 3 приводится график изменения температуры от $T_{max} = 100$ за первые 200 итераций.

1.2 Эволюционный метод

Прежде чем перейти к описанию нейроэволюционного подхода кратко рассмотрим эволюционные вычисления, методы которых лежат в его основе. Эволюционные принципы наследственности, изменчивости и естественного отбора используются для решения как задач моделирования (например, модели молекулярной эволюции и макроэволюционные модели в социологии и истории), так и прикладных задач оптимизации. Идея использования механизмов эволюции для машинного обучения прослеживается и в хрестоматийном труде

Алана Тьюринга «Могут ли машины мыслить?». Множество алгоритмов и методов, использующие для поиска решения эволюционные принципы, объединяют под общим названием эволюционные вычисления или эволюционные алгоритмы (ЭА). Выделяют следующие основные виды ЭА:

- генетический алгоритм (ГА);
- эволюционное программирование (ЭП);
- эволюционные стратегии (ЭС);
- генетическое программирование (ГП).

Здесь мы более подробно рассмотрим ГА, как один из самых распространенных ЭА.

Введём следующие определения:

- хромосома – последовательность генов вида $W = [w_1, \dots, w_n]$;
- особь – набор хромосом. Обычно особь состоит из одной хромосомы и обозначается также W;
- ген – атомарная (неделимая) часть хромосомы, отвечающая за конкретный признак;
- генотип – совокупность всех генов, находящихся в хромосомах индивидуума;
- популяция – совокупность особей $\{W_i\}_{i=0}^{M}$.

Идея ГА предложена Джоном Холландом в 60-х годах, а результаты первых исследований обобщены в его монографии «Адаптация в природных и искусственных системах», а также в диссертации его аспиранта Кеннета Де Йонга. В общем виде канонический ГА может быть представлен следующим образом:

Шаг 1. Формирование начальной популяции $g(M) \to (\{W_l\}_{l=1}^{M})_0$;

Шаг 2. Оценивание популяции. Вычисление функции приспособленности $f(W_l) = -e(W_l)$, где $e(W_l)$ – ее целевая функция, для всех особей из текущей популяции. Числовое значение этой функцию можно рассматривать, как критерий качества для конкретного решения;

Шаг 3. Применение оператора селекции, выбора родительских особей для дальнейшего скрещивания:

$$s((\{W_l\}_{l=1}^{M})_h) \to \{\widetilde{W}_q\}_{q=1}^{m},$$

где m – размер селекционной группы, $m < M$;

Шаг 4. Применение оператора скрещивания к родительским особям:

$$c(\widetilde{W}_{q_1}, \widetilde{W}_{q_2}) \to W_{q_1}^{*}, W_{q_2}^{*}$$

где $W_{q_1}^{*}, W_{q_2}^{*}$ – потомки;

Шаг 5. Применение оператора мутации к потомкам:

$$W_q' = \begin{cases} m(W_q^{*}), \text{если } r \le P_m \\ W_q^{*}, \text{иначе} \end{cases},$$

где P_m – вероятность выполнения мутации;

Шаг 6. Создание новой популяции из имеющейся популяции и получившихся потомков $o\left((\{W_l\}_{l=1}^{M})_h, \{W_q'\}_{q=1}^{m}\right) \to (\{W_l\}_{l=1}^{M})_{h+1}$ и положить $h = h + 1$;

Шаг 7. Если достигнуто условие остановки, выбрать из конечной популяции W_l особь с лучшим значением $f(W_l)$. Вернуть решение задачи и значение целевой функции $e(W_l)$, иначе перейти на шаг 2. [7]

Общая схема канонического ГА также представлена на рисунке 8. Необходимо отметить, что в настоящее время существует множество модификаций канонического ГА и не все из них могут быть описаны представленной схемой на рисунке 8.

Рисунок 8 – общая схема ГА

ГА часто используются для решения оптимизационных задач, но, в отличие от большинства других алгоритмов (например, градиентные алгоритмы), на каждом этапе работы алгоритма рассматривается не одно потенциальное решение, а множество решений – популяция. Каждая особь популяции кодирует информацию о некотором решении в хромосоме (генотип), а декодированное из хромосомы решение иногда называют фенотипом. [6, 7]

Формирование начальной популяции.

В начале работы ГА функцию $g(M)$ формирует популяцию хромосом $(\{W_l\}_{l=1}^{M})_0$, как правило, случайным образом. Размер популяции M, как правило, пропорционален количеству оптимизируемых параметров. Слишком малая популяция хромосом приводит к замыканию в неглубоких локальных минимумах. Слишком большое их количество чрезмерно удлиняет вычислительную процедуру и также может не привести к точке глобального минимума. [1, 7]

Селекция.

По результатам оценивания особей на шаге 2 наиболее приспособленные из них выбираются для скрещивания. Селекция может основываться на разных принципах. Одним из наиболее распространенных считается принцип элитарности, в соответствии с которым наиболее приспособленные (в смысле целевой функции) хромосомы сохраняются, а наихудшие отбраковываются и заменяются вновь созданным потомством, полученным в результате

скрещивания пар родителей. В простейшем случае, мы можем выбрать из популяции $m \leq M$ особей с наибольшими значениями функции приспособленности и заменить ими оставшиеся $(M - m)$ особей. [1]

$$s((\{W_l\}_{l=1}^{M})_h) \to \{\widetilde{W}_q\}_{q=1}^{m}.$$

Отбор может происходить разными способами, например, по принципу рулетки; при этом площадь сегмента колеса рулетки, сопоставленного конкретной хромосоме, пропорциональна величине ее функции приспособленности. Например, если $f(W_l)$ – функция приспособленности l-й особи, то условие выбора этой особи для скрещивание может быть следующим:

$$P_s(W_l) = \begin{cases} 1, \text{если } r \geq \dfrac{f(W_l)}{\sum_{j=1}^{M} f(W_j)}, \\ 0, \text{иначе} \end{cases}$$

где r случайное число из диапазона $\{0,1\}$ и $P_s(W_l)$ – вероятность, что l-й особь попадёт в селекционную группу.

Другим возможным способом отбора особи для скрещивания может быть представлена в виде турнира, где m случайных особей соревнуются за возможность попадания в селекционную группу $\{\widetilde{W}_q\}_{q=1}^{m}$. По итогам турнира в селекционную группу попадает лучшая особь W_l. Отбор турниром проводится пока в селекционной группе не появится m особей. Легко заметить, что чем больше значение m тем меньше шансов у «плохих» особей попасть в селекционную группу. [7]

Скрещивание.

В результате скрещивания выбранных особей посредством применения генетического оператора кроссовера создаются потомки, генетическая информация которых формируется в результате обмена хромосомной информацией между родительскими особями. [1]

$$c(\widetilde{W}_{q_1}, \widetilde{W}_{q_2}) \to W_{q_1}^{*}, W_{q_2}^{*}.$$

Существует огромное множество методов скрещивания. В простейшем случае выбирается случайный номер гена – точку скрещивания $1 < i < n$ в хромосомах $\widetilde{W}_{q_1} = \left[w_1^{q_1}, \dots, w_n^{q_1}\right]$, $\widetilde{W}_{q_2} = \left[w_1^{q_2}, \dots, w_n^{q_2}\right]$, после чего формируются их потомки путём обмена части генов из одной в другую и наоборот. Результатом такого скрещивания будет два потомка $W_{q_1}^* = \left[w_1^{q_1}, \dots, w_i^{q_1}, w_{i+1}^{q_2}, \dots, w_n^{q_2}\right]$ и $W_{q_2}^* = \left[w_1^{q_2}, \dots, w_i^{q_2}, w_{i+1}^{q_1}, \dots, w_n^{q_1}\right]$, в качестве модификации этого метода, может выступить увеличению количества точек скрещивания. [1, 7]

При выполнении этого метода родители могут породить потомков, два и более генов которых могут совпадать друг с другом. В случае если для работы алгоритма важна уникальность генов в хромосомах альтернативой этому методу может выступить упорядоченный метод скрещивания. Выбирается случайная точка скрещивания $1 < i < n$ и выполняются следующие действия:

- строится последовательность генов $W_{q_1}^*$ – копируется левый сегмент генов от точки скрещивания из хромосомы $\widetilde{W}_{q_1}$, а остальные гены берутся из хромосомы $\widetilde{W}_{q_2}$ в упорядоченном виде слева направо, исключая гены, которые уже попали в $W_{q_1}^*$;
- строится последовательность генов $W_{q_2}^*$ – копируется левый сегмент генов от точки скрещивания из хромосомы $\widetilde{W}_{q_2}$, а остальные гены берутся из хромосомы $\widetilde{W}_{q_1}$ в упорядоченном виде слева направо, исключая гены, которые уже попали в $W_{q_2}^*$. [7]

Мутация.

Созданные потомки формируют новую популяцию, причем часть потомков мутирует, что выражается в случайном изменении их генотипов.

$$W_q' = \begin{cases} m\left(W_q^*\right), \text{если } r \le P_m \\ W_q^*, \text{иначе} \end{cases}.$$

При двоичном кодировании мутация состоит в инверсии случайно выбранных битов или обмене двух случайных бит местами. При кодировании векторов десятичными числами мутация заключается в замене значения какого-

либо элемента вектора другим случайно выбранным значением или обмене двух случайных элементов местами. Мутация обеспечивает защиту как от слишком быстрого завершения алгоритма (в случае выравнивания значений всех хромосом и целевой функции), так и от представления в какой-либо конкретной позиции всех хромосом одного и того же значения. Однако необходимо иметь в виду, что случайные мутации приводят к повреждению уже частично приспособленных векторов. Обычно мутации подвергается не более 1-5% бит всей популяции хромосом. Элемент, подвергаемый мутации, отбирается случайным образом. [1, 6]

Создание новой популяции.

Формирование нового поколения — это оператор формирования нового поколения из массива родителей и получившихся потомков с использованием уже известных значений функции приспособленности как родителей, так и потомков.

$$o\left(\left(\{W_l\}_{l=1}^{M}\right)_h, \{W_q'\}_{q=1}^{m}\right) \rightarrow \left(\{W_l\}_{l=1}^{M}\right)_{h+1}.$$

Методы формирования нового поколения отличаются главным образом количеством получившихся потомков на каждой итерации. Возможны следующие случаи:

- $m = M$: все особи в популяции заменяются на потомков;
- $m < M$: все «плохие» особи заменяются на потомков;
- $m > M$: с помощью функции приспособленности выявляются лучшие потомки, которые остаются в популяции.

Этап, включающий последовательность операций, обозначенных блоками «Оценивание популяции» – «Селекция» – «Скрещивание» – «Мутация», называется поколением. Эволюция популяции состоит из последовательности таких поколений.

Окончание эволюции может определяться следующими факторами:

- нахождение решения в результате эволюционного поиска;

- ограничение на число поколений, количество вычислений целевой функции, общее время работы ГА;
- вырождение популяции, когда степень разнородности хромосом в популяции становится меньше установленного значения.

Упомянутые выше другие виды ЭА несколько отличаются от ГА. Так, основным отличием ЭС от ГА является значительно меньшая роль оператора кроссовера, а в ЭП кроссовер не используется совсем. Также ЭС и ЭП изначально разрабатывались не только для целочисленного, но и для вещественного кодирования. ГП, в отличие от ГА, ЭС и ЭП, которые часто используются для решения оптимизационных задач, предназначено для решения задач автоматизированного программирования и машинного обучения.

1.3 Метод роя частиц

Еще с незапамятных времен людей интересовало поведение животных в группе, так называемое роевое поведение, каким способом функционируют птицы, когда стая направляется в теплые страны, как обеспечивает себя пропитанием рой пчел, каким способом выживает колония муравьев, создавая сложные структуры, как ведут себя рыбы в косяке, ведь так синхронизировано их поведение. Организация особей в социуме, как бы в присутствии единого управляющего разума, несмотря на децентрализацию управления, некоторые закономерности слаженного целостного организма, подсмотренные взаимосвязи у природы побуждают на создание и развитие новых идей в области алгоритмической оптимизации.

Роевой интеллект (Swarm intelligence) описывает имитацию коллективного поведения самоорганизующейся системы. Существует достаточно большое число таких алгоритмов. В каноническом варианте, написанном в 1995г. Кеннеди и Эберхартом, модель, положенная в основу этого метода, была получена упрощением модели Рейнолдса. В результате этого упрощения отдельные особи популяции стали представляться отдельными объектами, не имеющими размера, но обладающими некоторой скоростью.

В силу крайней схожести с материальными частицами получившиеся простые объекты стали называться частицами, а их популяция — роем. В каждый момент времени (на каждой итерации) частицы имеют в пространстве некоторое положение и вектор скорости. Для каждого положения частицы вычисляется соответствующее значение целевой функции, и на этой основе по определённым правилам частица меняет своё положение и скорость в пространстве поиска, при определении следующего положения частицы учитывается информация о наилучшем положении из числа всех остальных соседних частиц, соответствующее задачам фитнесс-функции.

Примеры роевых алгоритмов:

- Метод роя частиц
- Муравьиный алгоритм
- Пчелиный алгоритм
- Искусственная иммунная система
- Алгоритм серых волков
- Алгоритм летучих мышей
- Алгоритм гравитационного поиска
- Алгоритм альтруизма
- И многие другие

Переход от моделирования коллективного поведения к коллективной оптимизации основан на следующей биологической идее: организмы объединяются в колонии для улучшения своих условий существования — каждый организм в колонии в среднем имеет больше шансов на выживание в борьбе с хищниками, колония может более эффективно производить поиск, обработку и хранение пищи по сравнению с отдельными особями и т.д. Другими словами любая колония организмов в течение всего времени своего существования с той или иной степенью эффективности решает различные оптимизационные задачи, например, максимизация количества пищи с одновременной минимизацией потерь от хищников. Указанные соображения и

легли в основу построения разнообразных математических методов оптимизации.

Рой частиц - один из самых известных и популярных алгоритмов оптимизации с самого момента его появления, многие авторы различных его реализаций утверждают, что алгоритм очень эффективен в оптимизации сложных со многими аргументами функций и даже подходит для обучения нейронных сетей.

Оптимизация методом роя частиц (МРЧ) – это основанный на популяции метод, который определяет набор ‘частиц’, исследующих область поиска для обнаружения минимума. МРЧ итеративно улучшил предполагаемое решение в отношении определенного показателя качества. Он решает задачу, используя популяцию потенциальных решений (“частиц”) и перемещая их согласно простым математическим правилам, таким как положение частицы и ее скорость. Каждое перемещение частицы определяется наилучшей, с ее точки зрения, локальной позицией, но также и лучшими позициями в пространстве поиска, найденными другими частицами. В теории рой через серию итераций перемещается по направлению к наилучшим решениям.

Метод виртуальных (случайных) частиц может надстраиваться почти над любым методом оптимизации. Он создан для:

- повышения устойчивости обученных сетей;
- вывода сетей из возникающих при обучении локальных минимумов оценки.

Основная идея метода – использование случайных сдвигов аргумента и суммирование полученных значений функции для усреднения. Ожидается, что в результате уменьшится влияние рельефа минимизируемой функции на процесс минимизации и откроется более прямой путь к её глобальному минимуму.

В основе МРЧ лежит представление о наборе частиц, перемещающихся в пространстве поиска. Каждая частица регулирует векторы позиции и скорости по своему опыту (когнитивная составляющая) и по информации, полученной от

других (социальная составляющая). Здесь опыт частицы понимается как её знание о лучшей позиции, через которую она сама пролетела, а социальное знание частицы понимается как её знание о лучшей позиции, через которую прошла одна из частиц в одной группе с ней. [1, 8, 9]

Частицы состоят из:

Трёх векторов:

- X – текущая позиция частицы в поисковом пространстве;
- P – лучшая позиция, найденная частицей;
- G – лучшая позиция, найденная всем роем;
- V – скорость движения частицы.

Двух целевых функций:

- f – значение целевой функции частицы;
- g – значение целевой функции всего роя.

Движение частицы задаётся следующими векторами:

$$X_i = X_i + V_i, \quad (1)$$

$$V_i = \omega V_i + \mathrm{c}_1 r_1 (G - X_i) + c_2 r_2 (P_i - X_i), \quad (2)$$

где i – номер частицы, V_i – вектор скорости, X_i – вектор текущей позиции, P_i – вектор лучшей позиции, найденный частицей, G – вектор лучшей позиции среди всех частиц:

$$G = g\left(\{P_i\}_{i=1}^{N}\right),$$

ω – инерциальный компонент, т. н. направлением полёта частицы (обычно равен $\omega = 1$), c_1, c_2 – константы скорости обучения, управляющие соответственно социальной и когнитивной компонентой, r_1, r_2 – случайные числа в диапазоне $\{0,1\}$.

Общая схема алгоритма канонического МРЧ выглядит так:

Шаг 1. Инициализация роя частиц. Создание N D-мерных частиц $X_i^0 = (x_1, \dots, x_D)$, где D – обозначает размерность пространства поиска или размера задачи. В контексте оптимизации нейросетей

D – размерность весов $W_i = (w_1, \dots, w_D)$. Создание N D-мерных скоростей $V_i^0 = (v_1, \dots, v_D)$. Положить, что $P_i^0 = X_i^0$ и вычислить $G^0 = g\left(\{P_i^0\}_{i=1}^N\right)$;

Шаг 2. Нахождение лучшего решения для каждой частицы. Если $f(X_i^l) < f(P_i^l)$, положить $P_i^l = X_i^l$;

Шаг 3. Нахождение лучшего решения среди всех частиц. Вычислить $G^l = g\left(\{P_i^l\}_{i=1}^N\right)$;

Шаг 4. Обновление скоростей. С помощью (2) обновить значения скоростей V_i^l;

Шаг 5. Перемещение каждой частицы. С помощью (1) переместить частицы X_i^l;

Шаг 6. Если достигнуто условие остановки вернуть G^l, $X_i^l = G^l$ и $f(X_i^l)$, иначе положить $l = l + 1$ и перейти на шаг 2. [8, 9]

Общая схема канонического МРЧ также представлена на рисунке 9. Соответственно каждая модификация этого алгоритма необязательно соответствует схеме на рисунке 9.

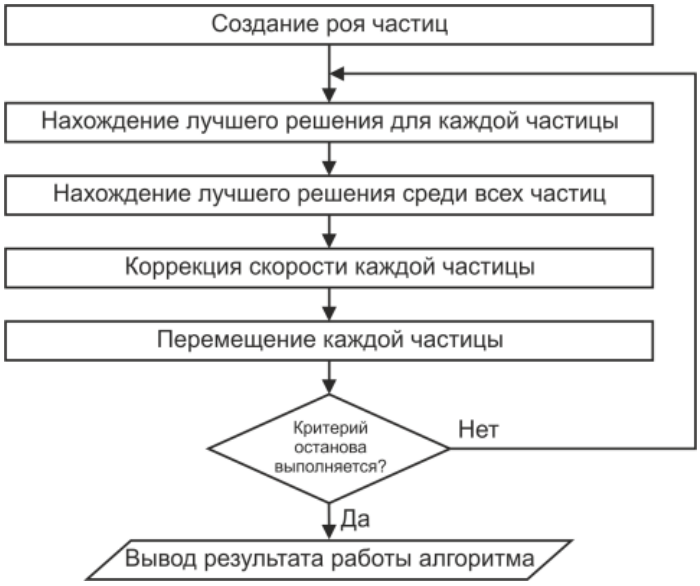

Рисунок 8 – общая схема ГА

1.4 Другие методы глобальной оптимизации

Стоит отметить, что различных методов глобальной оптимизации и их модификацией достаточно много, потому предложена следующая классификация:

Стохастические и термодинамические методы:

– Грубый случайный поиск (Метод Монте-Карло). Это самый простой и в то же время самый известный алгоритм случайного поиска, состоящий из равномерного случайного «бросания» точек в пространство поиска. Основное его достоинство – простота, и в теории глобальной оптимизации этот алгоритм применяется в основном в качестве эталона при теоретическом или численном сравнении алгоритмов и в качестве составной части некоторых алгоритмов глобального случайного поиска;

– Метод имитации обжига.

Детерминированные методы:

– Метод ветвей и границ. В методах ветвей и границ множество решений M разбивается на ряд подмножеств M_k (ветвление), и вместо перебора всех элементов этих подмножеств рассчитываются нижние границы $L(M_k)$ минимизируемой целевой функции $f(X)$ в подмножествах M_k. Сокращение перебора возможно в связи с тем, что далее ветвлению подвергается только то подмножество M_k, у которого нижняя граница оказалась наименьшей. Однако если у новых появившихся подмножеств нижние границы окажутся хуже, чем у какого-либо из ранее образованных подмножеств, то придется вернуться к шагу ветвления, на котором было образовано это более перспективное подмножество. При этом метод обеспечивает точное решение задачи, но в худшем случае из-за таких возвратов имеет место полный перебор. Для применения метода необходимо иметь алгоритм вычисления нижних границ. Если использовать, то или иное упрощение задачи (приближенное вычисление нижних границ, ограничение возвратов и т.п.), то метод становится приближенным и за счет потери точности гарантирует приемлемые затраты времени решения;

– Метод поиска с запретами. Он основан на аналогии с человеческим поведением, т.е. на присутствии в поведенческой схеме человека случайных элементов, которые означают, что в одной и той же ситуации человек может повести себя разным образом. Сохранение листа запретов, в котором, например, может храниться набор уже рассмотренных точек пространства поиска, является одной из основных характеристик данного подхода. Алгоритм заключается в выборе случайной точки в пространстве поиска, рассмотрении точек из окрестностей данной точки, и, при достижении определенного критерия, выборе новой точки в другом регионе поиска, который еще не рассматривался.

Эвристические и метаэвристические методы:

– Эволюционные методы;

– Роевые методы. [10]

2 Недостатки и преимущества. Сравнение алгоритмов

Отметим следующие преимущества и недостатки рассмотренных методов глобальной оптимизации.

Метод имитации обжига имеет несколько слабых сторон, которые могут ограничивать его эффективность при решении некоторых задач оптимизации. Некоторые из этих слабых сторон включают в себя:

- Зависимость от начального решения;
- Сложная настройка параметров;
- Вероятность застревания в локальном минимуме;
- Низкая скорость сходимости. Метод может быть медленным в сходимости к оптимальному решению, особенно для сложных задач оптимизации, из-за низкой скорости охлаждения;
- Неэффективность при большом количестве переменных.

Среди преимуществ можно выделить:

- Простота реализации;
- Высокая скорость работы;
- Небольшое количество внешних параметров. [11]

Эволюционные методы, в частности **генетические алгоритмы**:

Слабые стороны:

- Большое количество внешних параметров;
- Сложная реализация;
- Высокая вычислительная сложность.

Преимущества:

- Высокая эффективность при решении разнообразных задач;
- Высокая устойчивость к застреванию;
- Высокие результаты как на гладких, так и на сложных дискретных функциях;
- Высокая скорость сходимости. [12]

Метод роя частиц:

Слабые стороны:

- Низкое качество исследования оптимизируемой функции, алгоритм не может быть применим для решения задач, где требуется набор из нескольких уникальных решений;
- Низкая скорость и точность сходимости;
- Непригодность для оптимизации дискретных функций;
- Практически не масштабируем.

Преимущества:

- Алгоритм очень прост и нетребователен к вычислительным ресурсам, поскольку в канонической реализации нет даже сортировки массивов.
- Неплохо справляется с гладкими функциями. [13]

В статье [12] представлена таблица ранжирования методов глобальной оптимизации для гладких и дискретных проблем разной размерности. Однако, согласно теореме «бесплатного обеда не бывает», нельзя выделить самый лучший метод оптимизации, если усреднить результаты по всевозможным функциям качества на полном множестве задач. Практическое значение этой теоремы заключается в том, что не существует универсального метода оптимизации, который бы подходил для всех случаев. Успех определённого метода оптимизации в одной области знаний не гарантирует аналогичного успеха в другой. Это подчёркивает необходимость проведения исследований для каждой специфической области с целью нахождения подходящего метода оптимизации. Тем не менее, из таблицы видно, что ГА применяются с наибольшим успехом в очень широком круге задач.

ЗАКЛЮЧЕНИЕ

В данной работе были рассмотрены методы глобальной оптимизации, используемые для обучения нейросетей, а именно: метод имитации отжига, генетический алгоритм и метод роя частиц. Каждый из этих методов имеет свои уникальные особенности, преимущества и недостатки.

Метод имитации отжига показывает свою эффективность в задачах с высоким числом локальных минимумов, предоставляя возможность избежать застревания в них благодаря стохастической природе алгоритма. Тем не менее, данный метод может требовать значительного времени для сходимости, особенно при решении сложных задач.

Генетический алгоритм выделяется своей способностью находить оптимальные решения через механизмы эволюции и естественного отбора. Он эффективно справляется с задачами оптимизации, имеет широкий спектр применений и высокую адаптивность. Однако, настройка параметров генетического алгоритма, таких как размер популяции и вероятность мутации, является важным аспектом для обеспечения успешного решения задачи.

Метод роя частиц использует коллективное поведение и обмен информацией между частицами для нахождения глобального оптимума. Он обладает высокой скоростью сходимости и простотой реализации, однако, его эффективность может снижаться при увеличении размерности задачи.

В результате проведенного сравнительного анализа можно сделать вывод, что выбор метода глобальной оптимизации зависит от конкретной задачи и её характеристик. Каждому методу присущи свои сильные стороны, которые могут быть полезны в различных контекстах применения. Для достижения наилучших результатов рекомендуется проводить экспериментальные исследования и настройку параметров, учитывая особенности задачи и требования к качеству решения.

Эта работа подчеркивает важность грамотного выбора и применения методов глобальной оптимизации для обучения нейросетей, способствуя их успешному использованию в современных научных и практических задачах.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1 Искусственный интеллект : Нейрокомпьютерные системы : сборник из 17 лекций : Лекция 8: Методы глобальной оптимизации [Электронный ресурс] // НОУ ИНТУИТ [Электронный ресурс] : [сайт] : Новосибирский гос. ун-т. – Новосибирск, 2006. – URL: https://intuit.ru/studies/curriculums/19048/courses/61/lecture/20448 (Дата обращения 09.12.2024). – Загл. с экрана. – Яз. рус.

2 Kirkpatrick, S. Optimization by Simulated Annealing / S. Kirkpatrick, C.D. Gelatt Jr, M.P. Vecchi // Science. – 1983. – Vol. 220 (4598). – p. 671-680. (дата обращения 09.12.2024). – Яз. англ.

3 Gendreau, M. Handbook of Metaheuristics (International Series in Operations Research and Management Science) / M. Gendreau, J.-Y. Potvin. – Springer, 2010. – 675 p. (дата обращения 09.12.2024). – Яз. англ.

4 Ларионов, В. Н. Применение современных методов оптимизации для определения параметров математических моделей электроустановок / В. Н. Ларионов, Е. А. Акиньшин // Ползуновский вестник. – 2002. – № 1. – С. 17-20. (дата обращения 10.12.2024)

5 Елизаров, С. Введение в оптимизацию. Имитация отжига [Электронный ресурс] / С. Елизаров // Хабр [Электронный ресурс] : [сайт]. – 2014. – URL: https://habr.com/ru/articles/209610/ (Дата обращения 10.12.2024). – Загл. с экрана. – Яз. рус.

6 Цой, Ю. Р. Эволюционный подход к настройке и обучению искусственных нейронных сетей / Ю. Р. Цой, В. Г. Спицын, гл. ред. В. Д. Редько // Нейроинформатика. – 2006. – Т. 1, № 1. : сборник научных трудов / Томск. политех. ун-т. – Томск, 2006. – С. 34–61. (Дата обращения 13.12.2024).

7 Короткин, А. А. Генетические алгоритмы : учебно-методическое пособие / сост. А. А. Короткин, М. В. Краснов ; Ярославский гос. ун-т

им. П. Г. Демидова. – Ярославль : ЯрГУ, 2020. – Ярославль, 2020. – С. 40. (дата обращения 13.12.2024).

8 Казакова, Е. М. Краткий обзор методов оптимизации на основе роя частиц // Вестник КРАУНЦ. Физ.-мат. науки. – 2022. – Т. 39. № 2. / Ин-т прик. мат. и авт. КБНЦ РАН. – Нальчик, 2022. – С. 150-174. DOI: 10.26117/2079-6641-2022-39-2-150-174 (дата обращения 14.12.2024).

9 Ермаков, Б. С. Оптимизация роем частиц в обучении искусственных нейронных сетей. Системный анализ и логистика. – СПб ГУАП. – СПб, 2017. – С. 3-9. (Дата обращения 14.12.2024).

10 Газизов, Т. Т. Методы глобальной оптимизации : учебное пособие. / Т. Т. Газизов, верстка В. М. Бочкарёва. – Томск: В-Спектр, 2017. – 24 с. (Дата обращения 15.12.2024).

11 Andrey Dik. Популяционные алгоритмы оптимизации: Алгоритм имитации отжига (Simulated Annealing, SA). : в 2 ч. Ч. 1 [Электронный ресурс] // MQL5 [Электронный ресурс] / текст доступен по лицензии MetaQuotes Ltd. – Электрон. дан. – © MetaQuotes Ltd, 2000-2024. – URL: https://www.mql5.com/ru/articles/13851 (Дата обращения 15.12.2024). – Загл. с экрана. – Последнее изменение страницы: 16:39, 7 декабря 2023 года. – Яз. рус.

12 Andrey Dik. Популяционные алгоритмы оптимизации: Бинарный генетический алгоритм (Binary Genetic Algorithm, BGA). : в 2 ч. Ч. 2 [Электронный ресурс] // MQL5 [Электронный ресурс] / текст доступен по лицензии MetaQuotes Ltd. – Электрон. дан. – © MetaQuotes Ltd, 2000-2024. – URL: https://www.mql5.com/ru/articles/14040 (Дата обращения 15.12.2024). – Загл. с экрана. – Последнее изменение страницы: 16:13, 25 января 2024 года. – Яз. рус.

13 Andrey Dik. Популяционные алгоритмы оптимизации: Рой частиц (PSO) [Электронный ресурс] // MQL5 [Электронный ресурс] / текст доступен по лицензии MetaQuotes Ltd. – Электрон. дан. – © MetaQuotes Ltd, 2000-

2024. – URL: https://www.mql5.com/ru/articles/11386 (Дата обращения 15.12.2024). – Загл. с экрана. – Последнее изменение страницы: 13:34, 14 октября 2022 года. – Яз. рус.