УДК 004.032.26 **DOI 10.52928/2070-1624-2022-39-11-21-29**

ОБУЧЕНИЕ НЕЙРОННЫХ СЕТЕЙ НА ОСНОВЕ СЛУЧАЙНОГО ПОИСКА

В. В. МАЦКЕВИЧ
(Белорусский государственный университет, Минск)
ORCID: https://orcid.org/0000-0001-6964-542X

Рассматривается актуальная проблема, связанная с обучением нейронных сетей. Предлагается оригинальный алгоритм (со специальной процедурой распараллеливания), реализующий метод отжига. Эффективность обучения демонстрируется на примере архитектуры нейронной сети, ориентированной на параллельную обработку данных. Для задачи сжатия цветных изображений показано, что предложенный алгоритм существенно превосходит градиентные методы по эффективности. Полученные результаты позволяют повысить качество обучения нейронных сетей в целом и могут быть использованы для решения широкого класса прикладных задач.

Ключевые слова: *метод случайного поиска, метод градиентного спуска, метод отжига, обучение, ограниченная машина Больцмана, параллельные вычисления.*

Введение. В условиях развития цифрового общества стремительно растет количество информации, получаемой из различных источников. В связи с этим актуализируется проблема обработки больших объемов данных.

При решении данной проблемы часто используется нейросетевой подход, который обладает высоким уровнем универсальности. Настройка нейронной сети на предметную область происходит путем ее предварительного обучения. Благодаря этому нейронные сети обладают достаточной гибкостью с возможностью решения широкого класса прикладных задач. Однако по мере увеличения объема обрабатываемых данных усложняется архитектура используемой нейронной сети.

Процесс обучения (настройки) нейронной сети является типичной оптимизационной задачей [1]. Для ее решения, как правило, применяется метод градиентного спуска. Однако метод обладает рядом недостатков, что приводит к тому, что полученное решение может оказаться далеко не оптимальным. В работе рассматривается альтернативный подход к обучению, основанный на идее случайного поиска. Предлагается оригинальный алгоритм, реализующий метод отжига, и исследуется его эффективность.

Анализ проблемы. Нейронная сеть фиксированной архитектуры (как любая алгоритмическая модель) задается соответствующим набором настраиваемых параметров. Обучение заключается в нахождении таких их значений, которые гарантируют достижение наилучшего решения. Таким образом, обучение нейронной сети является типичной оптимизационной задачей, где целевая функция зависит от решаемой задачи, данных и значений параметров сети. Последовательно изменяя значения параметров, необходимо получить оптимум целевой функции.

Для решения оптимизационных задач существуют различные методы, среди которых можно выделить точные и приближенные. Приближенные методы могут быть как конечными, так и итерационными. В свою очередь, итерационные методы могут быть направленные и ненаправленные. К первому классу, в частности, относятся методы, основанные на вычислении градиентов. Это, например, методы простого градиентного спуска, секущих, Ньютона и т. д. Ненаправленные методы (случайного поиска) характеризуются случайностью выбора последующего приближения. К ним можно отнести методы дифференциальной эволюции, отжига, генетические алгоритмы и т. д.

Направленные методы ограничены строгими правилами выбора последующего решения, что сужает пространство поиска решения, в то время как методы случайного поиска имеют практически неограниченное пространство для поиска. Это дает интуитивное преимущество случайного поиска над направленными методами.

Для обучения нейронных сетей традиционно применяются различные вариации метода градиентного спуска. Данный метод получил широкое распространение за счет высокой скорости его сходимости. В условиях малых мощностей компьютеров (на начальном этапе развития нейронных сетей) это было очень важно. Однако градиентные методы обладают существенным недостатком: на процесс решения накладываются сильные ограничения по сходимости. Например, метод Ньютона имеет строгие ограничения на множество начальных приближений. Кроме того, любой градиентный метод сходится в точках, где производная от целевой функции может быть равна нулю, поэтому полученное решение может оказаться точкой перегиба или локального минимума, т. е. гораздо хуже оптимального.

В работе исследуется альтернативный подход к обучению нейронных сетей, основанный на идее случайного поиска (на примере метода отжига). Известно, что при определенных условиях данный метод сходится к оптимальному решению, причем из любого начального приближения [2].

Архитектура нейронной сети. Рассмотрим процесс обучения нейронных сетей на примере ограниченной машины Больцмана (ОМБ) для задачи сжатия цветных изображений. Нейронные сети данной архитектуры успешно применяются в задачах прогнозирования [3], информационного поиска [4], анализа данных [5–7], обработки речи [8] и т. п. ОМБ также лежат в основе глубоких доверительных сетей [9].

Известно, что основой архитектуры ограниченной машины Больцмана является стохастический нейрон (рисунок 1).

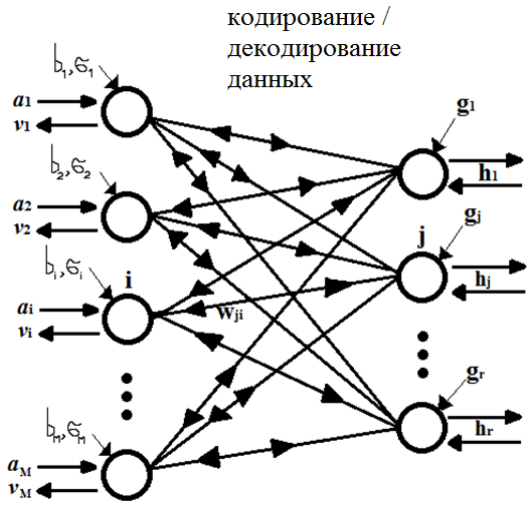

Рисунок 1. – Ограниченная машина Больцмана

Таким образом, формально архитектуру данной сети можно представить полносвязным двудольным графом $G=(X,\ U)$:

$$\begin{cases} X = X_1 \cup X_2,\ X_1 \cap X_2 = \varnothing; \\ U = \{u = (x_1,\ x_2) \mid \forall x_1 \in X_1,\ \forall x_2 \in X_2\}, \end{cases}$$

где X – множество вершин – стохастических нейронов;

U – множество ребер – синаптических связей, при этом вершины подмножества X_1 задают нейроны входного слоя, а X_2 – нейроны выходного слоя.

Число нейронов во входном слое определяется размером входного образа, а количество нейронов в выходном слое определяется исходя из требований к степени сжатия данных.

Выходные сигналы слоев ограниченной машины Больцмана реализуют некоторые законы вероятностного распределения. В зависимости от используемых законов распределения строят различные типы машин. В данной работе речь пойдет о машинах типах Гаусс – Бернулли и Бернулли – Бернулли, т. к. они являются наиболее распространенными.

Для ограниченной машины Больцмана типа Гаусс – Бернулли каждой вершине входного слоя поставим в соответствие множества параметров $VB=\{b\}$ – смещения и $\sigma=\{\sigma\}$ – дисперсии вершин, а вершинам выходного слоя – множество параметров $HB=\{g\}$ – смещение вершин. Размеры множеств равны соответственно $|VB|=|\sigma|=|X_1|$, $|HB|=|X_2|$. Каждому ребру, связывающему пару вершин входного и выходного слоев, поставим в соответствие множество параметров $W=\{w\}$ – весов ребер. Размер множества равен величине $|W|=|X_1||X_2|$.

Таким образом, описанное семейство нейронных сетей можно задать четырьмя типами параметров: $RBM=(W,\ VB,\ \sigma,\ HB)$. Стоит отметить, что у ограниченной машины Больцмана типа Бернулли – Бернулли отсутствует множество параметров σ.

Известно, что для качественного обучения нейронной сети объем обучающей выборки должен быть не меньше числа настраиваемых параметров сети. При этом вычислительная сложность обучения прямо пропорциональна произведению количества настраиваемых параметров сети на объем выборки, а количество настраиваемых параметров прямо пропорционально размерности входных данных. Таким образом, вычислительная сложность обучения прямо пропорциональна квадрату размерности входных данных.

Для снижения вычислительной сложности процесса обучения предлагается оригинальная архитектура ограниченной машины Больцмана, ориентированная на параллельную обработку данных.

Пусть входные данные имеют размерность N. Каждый элемент входных данных разбивается на k равных фрагментов размера m $(km = N)$. Значение k определяется на этапе проектирования архитектуры нейронной сети. После разбиения данных создаются k ограниченных машин Больцмана одинаковой архитектуры (по одной для каждого фрагмента данных).

Таким образом, исходные данные обрабатываются не одной, а ансамблем из k ограниченных машин Больцмана. Нетрудно видеть, что данный подход обладает следующими преимуществами:

– k ограниченных машин Больцмана работают независимо друг от друга, что позволяет производить их параллельное обучение;

– в силу того, что вычислительная сложность обучения прямо пропорциональна размерности входных данных, вместо исходной квадратичной получаем линейную зависимость;

– уменьшение числа настраиваемых параметров нейронной сети понижает сложность обучения и приводит к повышению качества полученного решения (при условии разбиения данных на не слишком малые фрагменты).

Задача обучения. В общем случае задача обучения в случае ее решения методами случайного поиска может быть сформулирована следующим образом.

Пусть на конечном множестве допустимых решений Ω определена целевая функция F, и для каждого элемента $x \in \Omega$ задано множество соседних элементов $N(x) \subset \Omega$. Задачу условной оптимизации в данном случае можно задать в виде тройки (Ω, F, N). Рассмотрим возможности ее решения с помощью случайного поиска, в частности, с помощью метода отжига.

При изложении алгоритма обучения необходимо учитывать особенности обучаемых сетей. Любая сеть состоит из одного или нескольких слоев. В зависимости от типа сети отдельные ее слои могут выполнять абсолютно разные преобразования входных данных. В зависимости от этого слой может состоять из различных фрагментов, выполняющих разные преобразования. Любой фрагмент нейронной сети задается набором параметров. Из-за того, что фрагменты выполняют разные преобразования, соответствующие им наборы параметров могут иметь разный диапазон значений. Таким образом, архитектура нейронной сети может быть задана объединением из m наборов параметров $NN = (x_1, x_2, \ldots, x_m)$. В результате конкретная сеть получается путем фиксации значений всех ее параметров.

Опишем теперь предлагаемый алгоритм, реализующий метод отжига.

Алгоритм обучения

Предварительный этап. Инициализация начального состояния нейронной сети.

Задание начальных значений параметров сети $NN_0 = (x_{10}, x_{20}, \ldots, x_{m0})$ и последовательности температур, связанных соотношением

$$T_k = T_0 / \ln(k+3),\ k > 0, \tag{1}$$

где T_0 – заранее заданное значение.

Общая k-я итерация

Шаг 1. Генерация случайных величин

Генерация m равномерно распределенных дискретных случайных величин a_1, a_2, …, a_m на отрезке от нуля до количества параметров в наборе.

Генерация m случайных перестановок длиной, равной количеству параметров в наборе. Первые a_1, a_2, …, a_m элементов перестановок задают индексы изменяемых параметров в каждом наборе параметров соответственно.

Шаг 2. Генерация нового решения.

Для каждого изменяемого параметра генерируются две равномерно распределенные случайные величины b, c на отрезках [0; 1], [0; l/2]. Величина l зависит от того, какому набору принадлежит изменяемый параметр, и равна l_1, l_2, …, l_m соответственно. Значения l для каждого набора задаются как параметры алгоритма.

Пусть x_i – изменяемый параметр, а его новое значение x'_i находим по формуле

$$x'_i = \begin{cases} x_i + c, & b \le 0{,}5; \\ x_i - c, & b > 0{,}5. \end{cases}$$

Шаг 3. Принятие решения о переходе.

Пусть x – текущее решение, y – новое решение, сгенерированное на шаге 2. Тогда решение x' на следующей итерации определяется следующим образом:

$$P(x' = y \mid x) = \min\left\{1, \exp\left((F(x) - F(y)) / T_k\right)\right\}. \tag{2}$$

Шаг 4. Критерий остановки.

Если время на обучение нейронной сети истекло, то алгоритм завершается. В противном случае производится переход на следующую итерацию.

Рассмотрим проблему сходимости описанного выше алгоритма.

Анализ сходимости. При доказательстве сходимости алгоритма будем использовать известные теоретические результаты, полученные для метода отжига.

Вначале введем необходимые для дальнейших рассуждений понятия и определения.

Верхняя граница значения целевой функции $h \in \Re$ называется высотой.

Путем, соединяющим решение $x \in \Omega$ *с решением* $y \in \Omega$, называется последовательность $x_1, x_2, \ldots, x_n$ такая, что

$$\begin{cases} x_1 = x,\ x_n = y; \\ x_i \in \Omega,\ i = \overline{1, n}; \\ x_i \in N(x_{i-1}),\ i = \overline{2, n}. \end{cases}$$

Решение $y \in \Omega$ является *достижимым* из $x \in \Omega,$ если существует путь из *x* в *y*. И оно является *достижимым из* $x \in \Omega$ *на высоте h*, если

$$x = y,\ F(y) \le h$$

или существует конечная последовательность решений $x = x_0, x_1, x_2, \ldots, x_p = y,\ p > 0$ такая, что

$$\begin{cases} x_{k+1} \in N(x_k),\ \forall k = \overline{0,\ p-1}; \\ x_k \in \Omega,\ k = \overline{0,\ p}; \\ F(x_k) \le h,\ \forall k = \overline{0,\ p}. \end{cases}$$

Пара множеств (Ω, N) называется *несократимой*, если для $\forall x, y \in \Omega$ – решение y достижимо из *x* тогда и только тогда когда *x* достижимо из *y*.

Задача $(\Omega,\ F,\ N)$ обладает *свойством слабой обратимости*, если для $\forall x, y \in \Omega$ *y* достижимо из *x* на высоте *h* тогда и только тогда, когда *x* достижимо из решения *y* на высоте *h*.

Решение *x* является локальным минимумом задачи $(\Omega,\ F,\ N),$ если $\neg\exists y : F(y) < F(x)$ и *y* достижимо из *x* на высоте *F(x)*.

Глубиной локального минимума x называется наименьшая величина *E*, для которой выполняется условие $\exists y \in \Omega : F(y) < F(x),$ *y* достижимо из *x* на высоте $F(x) + E.$ Для глобального минимума глубина определяется как плюс бесконечность.

Предположим теперь, что задача $(\Omega,\ F,\ N)$ несократима, обладает свойством слабой обратимости и задана числовая последовательность $T_k,$ для которой выполняется условие

$$\begin{cases} T_i \ge T_{i+1},\ \forall i \in N \cup 0 \\ \lim\limits_{i \to +\infty} T_i = 0. \end{cases} \tag{3}$$

Тогда для метода отжига справедлива следующая теорема.

Теорема 1 [2].

1. Для любого не локального минимума *x*

$$\lim_{k \to +\infty} P(x_k = x) = 0.$$

2. Если *B* – множество локальных минимумов глубины *d*, то для $\forall x \in B$

$$\lim_{k \to +\infty} P(x_k \in B) = 0$$

тогда и только тогда, когда

$$\sum_{k=1}^{+\infty} \exp(-d / T_k) = +\infty.$$

3. Пусть Ω^* – множество глобальных минимумов и d^* – максимум из глубин локальных минимумов, не совпадающих ни с одним из глобальных

$$\lim_{k\to+\infty} P(x_k \in \Omega^*) = 1$$

тогда и только тогда, когда

$$\sum_{k=1}^{+\infty} \exp(-d^*/T_k) = +\infty. \quad (4)$$

Покажем теперь, что описанный выше алгоритм сходится. Из приведенной теоремы следует, что для этого необходимо и достаточно выполнение следующих условий:

– новое решение должно выбираться с вероятностью (2);
– должны выполняться условия сходимости (критерий);
– задача (Ω, F, N) должна быть несократимой и обладать свойством слабой обратимости;
– последовательность температур должна удовлетворять ограничениям (3), (4).

Покажем выполнение для алгоритма всех условий в рамках критерия сходимости. Сформулируем их в виде отдельных утверждений.

Выполнение первого условия достаточно очевидно и не требует доказательства. Докажем выполнение последующих условий.

Утверждение 1. Задача (Ω, F, N) несократима и обладает свойством слабой обратимости.

Доказательство. Проверим выполнение условия слабой обратимости. Это свойство является более строгим требованием, чем условие несократимости. Поэтому из первого всегда следует второе.

Для каждого набора параметров (множителя исходного пространства поиска) задан параметр алгоритма l_i – длина отрезка для генерации равномерно распределенной случайной величины. Так как текущее значение изменяемого параметра (координаты) является осью симметрии данного отрезка, то это означает, что $x + \Delta x \in N(x)$ тогда и только тогда, когда $x - \Delta x \in N(x)$. На основе этого докажем утверждение от противного.

Пусть задана пара произвольных решений x, y из Ω. Предположим, что решение y достижимо из x на высоте h, но x не достижимо из y. Достижимость y из x означает, что существует некоторая последовательность $x = x_0, x_2, ..., x_n = y,$ для которой выполняется неравенство

$$\max_{i=\overline{0,n}} f(x_i) \le h.$$

Так как $x_0, x_1, \ldots, x_n$ – последовательность, то очевидно

$$x_i = x_{i-1} + \Delta x_i, \; i = \overline{1,n}.$$

Построим теперь последовательность решений $y = y_0, y_1, \ldots, y_n, y_0 = x_n$ такую, что

$$y_i = y_{i-1} - \Delta x_{n-i+1}, \; i = \overline{1,n}.$$

Тогда получим

$$x_i = y_{n-i}, \; i = \overline{1,n}.$$

Следовательно,

$$\begin{cases} f(x_i) = f(y_{n-i}), \; i = \overline{1,n} \\ \max\limits_{i=\overline{0,n}} f(x_i) \le h \Leftrightarrow \max\limits_{i=\overline{0,n}} f(y_i) \le h. \end{cases}$$

Однако это противоречит предположению о недостижимости решения x из y на высоте h, что и доказывает наше утверждение.

Утверждение 2. Последовательность температур (1) удовлетворяет ограничениям (3), (4).

Доказательство. Так как натуральный логарифм по своей природе является неограниченной и монотонно возрастающей функцией, то последовательность (1) строго убывает и сходится к нулю. Покажем теперь расходимость ряда (4). Для этого значения T_k заменим формулой

$$\sum_{k=1}^{+\infty} \exp\left(-\frac{d^*}{T_k}\right) = \sum_{k=1}^{+\infty} \exp\left(-\frac{d^*}{T_0}\ln(k+3)\right) \ge \left[T_0 \ge d^*\right] \ge \sum_{k=1}^{+\infty} \exp\left(-\frac{d^*}{d^*}\ln(k+3)\right) =$$

$$= \sum_{k=1}^{+\infty} \exp\left(-\ln(k+3)\right) = \sum_{k=1}^{+\infty} \exp \frac{1}{k+3} = +\infty.$$

Откуда и следует расходимость ряда.

Таким образом, построенный алгоритм удовлетворяет всем ограничениям и, следовательно, сходится к оптимальному решению, причем из любого начального приближения.

Процедура распараллеливания. Низкая скорость сходимости алгоритма обучения требует для получения результата совершения большого числа итераций. Из этого следует, что обучение данным алгоритмом требует большого объема вычислений и времени для сходимости. Время работы итерационного алгоритма определяется произведением числа итераций на время выполнения одной отдельной итерации. Для уменьшения времени выполнения одной отдельной итерации и, следовательно, времени работы всего алгоритма обучения предлагается следующая процедура распараллеливания.

Каждая итерация алгоритма обучения, как было указано выше, состоит из 4 этапов. Все этапы кроме вычисления значения функционала для нового решения выполняет процессор. Наиболее трудоемким этапом является вычисление значения функционала и, при малой архитектуре сети, генерация нового решения. Вычислительная мощность видеокарты выше мощности процессора в среднем в 20 раз. Это приводит к тому, что при обучении небольших нейронных сетей значительная часть времени расходуется на генерацию новых решений. Все этапы в отдельной итерации выполняются строго последовательно.

Пусть a времени требуется для обмена данными между видеокартой и процессором, b – время, необходимое для генерации нового решения, c – время, необходимое для вычисления значения функционала качества, g – время, необходимое для принятия или отклонения нового решения, проверки критерия останова. Таким образом, общее время одной итерации t в последовательном случае выполнения составит

$$t = a + b + c + g.$$

Генерация нового решения состоит из двух этапов: определение количества и выбор изменяемых параметров сети; изменение значений выбранных параметров сети. На определение количества и выбор изменяемых параметров требуется e времени, а для изменения их значений – f времени. Таким образом, получим

$$t = [b = e + f] = a + e + f + c + g.$$

Для снижения времени выполнения одной итерации можно воспользоваться тем фактом, что определение и выбор изменяемых параметров сети не зависят от этапа принятия нового решения. Это можно объяснить тем, что сгенерированное приращение к значениям изменяемых параметров при этом не изменится. Смена решения изменяет лишь исходные значения изменяемых параметров. В таком случае можно воспользоваться общеизвестным приемом сокрытия вычислений.

Процедура A1. Определять количество и выбирать изменяемые параметры можно параллельно на процессоре, когда видеокарта производит вычисление значения функционала качества. В таких условиях результат выбора параметров используется для генерации нового решения уже на следующей итерации. В таком случае время выполнения одной итерации составит

$$t = a + f + \max\{e,c\} + g \approx \left[g \ll a, f, \max\{e,c\} \right] \approx a + f + \max\{e,c\}.$$

Стоит отметить, что время необходимое для принятия или отклонения решения с последующей проверкой критерия останова требует на порядок меньшего времени по сравнению с другими действиями в отдельной итерации.

Процедура A2. Для последующей оптимизации алгоритма обучения рассмотрим подробнее этап вычисления значения функционала.

Пусть текущее решение равно x, а новое решение – y. Во время вычисления значения функционала на видеокарте для решения y процессор одновременно генерирует два новых решения $x_1 \in N(x)$ и $y_1 \in N(y)$. После вычисления значения функционала производится проверка необходимости перехода в новое решение. Если новое решение принято, то следующим проверяемым решением будет y_1, в противном случае x_1. Данная процедура позволяет замаскировать этап генерации нового решения. Таким образом, при переходе на следующую итерацию сразу будет производиться вычисление значения функционала для нового решения без его явной генерации. Однако при таком подходе объем вычислений на процессоре увеличивается практически вдвое, что может быть критично для небольшой по размеру сети. Время отдельной итерации при данном подходе составит

$$t = a + \max\{2e + 2f, c\}.$$

Оценим величину выигрыша времени r для отдельной итерации. Возможны три случая:

1) $c > 2e + 2f$. Такой случай возможен при обучении больших нейронных сетей либо при использовании слабой видеокарты, либо мощного процессора. В таком случае выигрыш составит

$$r = a + f + \max\{e, c\} - a - \max\{2e + 2f, c\} = f;$$

2) $e \le c \le 2e + 2f$. Наиболее распространенный случай на практике. Возникает при использовании видеокарты и процессора средней мощности и обучении нейронных сетей со средним количеством настраиваемых параметров. В таком случае величина выигрыша составит

$$r = a + f + \max\{e, c\} - a - \max\{2e + 2f, c\} = a + f + c - a - 2e - 2f = c - 2e - f.$$

В данном случае процедура А2 не всегда лучше процедуры А1. Если $c \ge 2e - f$, то процедура А2 имеет смысл. В противном случае процедура А1 лучше во всех отношениях: она вдвое снижает объем генерируемых случайных чисел, чем уменьшает их расход, вдвое сокращает объем вычислений на процессоре и тем самым оптимизирует расход электроэнергии процессором;

3) $c < e$. Данный случай возникает при обучении небольших нейронных сетей либо при использовании мощной видеокарты, либо слабого процессора. В таком случай выигрыш составит

$$r = a + f + \max\{e, c\} - a - \max\{2e + 2f, c\} = a + f + e - a - 2e - 2f = -(e + f).$$

В данном случае процедура А1 всегда лучше А2.

Замечание. Использование процедур А1 и А2 никак не влияет на качество обучения. Выбор между ними влияет только на конечное время обучения и, возможно, на энергоэффективность обучения сетей.

Для решения проблемы выбора оптимального алгоритма обучения сконструируем процедуру А3.

Процедура А3. Основная ее идея заключается в наиболее точной оценке скорости работы процедур А1 и А2 при решении конкретной прикладной задачи. Так как выполнение одной итерации алгоритма требует, как правило, менее одной тысячной секунды времени, а кэш процессора проявляется через некоторое время и таймер имеет ненулевую погрешность, то для точной оценки нужно использовать большое число итераций.

Шаг 1. Запуск процедуры А1 на двадцати тысячах итераций с замером времени работы.

Шаг 2. Запуск процедуры А2 на двадцати тысячах итераций с замером времени работы.

Шаг 3. На основе полученных результатов времени выполнения производится выбор процедуры обучения для оставшихся итераций с меньшим временем работы.

Эффективность работы описанного выше алгоритма обучения проверим на примере решения задачи сжатия цветных изображений.

Результаты экспериментов. Для экспериментов использовано два алгоритма обучения: разработанный выше алгоритм обучения с применением процедуры А3 и метод адаптивного момента как сильнейшего градиентного алгоритма обучения [10; 11].

В случае с градиентным алгоритмом обучения для повышения качества решения будет использоваться алгоритм CD-10. При большом количестве итераций (свыше десяти тысяч) он превосходит алгоритм PCD по качеству [12], а CD-100 требует на порядок большего объема вычислений, чем CD-10 [13]. В то же время алгоритм обучения CD-10 по эффективности примерно равен 10-PT5, однако требует гораздо меньшего объема вычислений [14].

Для сжатия использованы изображения выборок CIFAR-10[1] [15] и STL-10[2] [16]. В первом случае объекты на изображениях принадлежат одному из десяти определенных выборкой классов разрешением 32 на 32 пикселя. Во втором же случае изображения выборки разрешением 96 на 96 пикселей содержат абсолютно произвольные объекты, что существенно усложняет процедуру сжатия.

Эксперименты проводились для 16-кратной степени сжатия. Более высокая степень сжатия приводит к слишком большим потерям, более низкая – неэффективна по сравнению с классическими алгоритмами сжатия.

Для оценки качества сжатия были использованы распространенные функционалы оценки потерь: PSNR (Peak Signal to Noise Ratio), PSNR-HVS (Peak Signal to Noise Ratio Human Vision System), SSIM (Structure Similarity Image Measure), MSE (Mean Square Error). Оценка времени обучения производилась с помощью функции gettimeofday.

---

[1] Выборка CIFAR-10 [Электронный ресурс]. URL: https://www.cs.toronto.edu/~kriz/cifar.html.

[2] Выборка STL-10 [Электронный ресурс]. URL: https://web.archive.org/web/20110803194852/http://www.staford.edu/~acoates//stl10/.

Программная среда тестирования: Linux, 2x8 1600 Mhz DDR3 RAM, процессор intel i7-4770k (4 ядра), видеокарта nvidia rtx 3070 (5888 ядер). Для тестирования и обучения нейронных сетей разработан фреймворк на языке C++ с использованием библиотек OpenML, OpenCL.

Исходные изображения предварительно разбивались на блоки по 3 на 4 цветных пикселя. Более крупное разбиение существенно повышало сложность обучения, а разбиение на более мелкие – снижало качество обучения.

При использовании градиентного алгоритма обучения изображения разбивались на блоки по 3 на 2 цветных пикселя с использованием той же логики.

Также стоит отметить, что каждая отдельная ограниченная машина Больцмана обрабатывает одновременно все три цветовых канала изображений. Это повышает семантическую связность пикселей в пределах одного фрагмента изображения. За счет этого повышается качество обучения.

Из результатов экспериментов нетрудно заметить, что алгоритм, реализующий метод отжига, превосходит градиентные алгоритмы по качеству обучения более чем в 2,5 раза. При этом скорость обучения обоими алгоритмами примерно равная. Стоит также отметить скорости сходимости алгоритмов (таблица).

Таблица. – Результаты экспериментов

Алгоритм обучения	Алгоритм отжига		Алгоритм градиентного спуска	
Выборка	CIFAR-10	STL-10	CIFAR-10	STL-10
MSE	459	472	2940	2551
PSNR	21,6	21,5	13,7	14,4
PSNR-HVS	21,8	21,7	13,8	14,6
SSIM	0,749	0,618	0,227	0,346
время обучения, ч	1	6	1	6

В случае с алгоритмом отжига для достижения результата понадобилось порядка миллиона итераций. В случае с градиентным алгоритмом понадобилось порядка десятка тысяч итераций. Т. е. разница составляет около двух порядков. Однако за счет мощного оборудования метод отжига догоняет градиентные алгоритмы по скорости обучения.

Заключение. Таким образом, эксперименты показали, что алгоритм обучения, реализующий метод отжига, позволяет существенно повысить качество обучения нейронных сетей. Кроме того, скорость работы алгоритма повышена за счет специальных процедур распараллеливания. Экспериментально показано, что он не только в 2,5 раза превосходит градиентные методы по качеству получаемого решения, но практически не уступает им по скорости работы.

С учетом того, что компьютерная техника развивается быстрыми темпами, а нейронные сети применяются для решения широкого класса прикладных задач, можно сделать вывод, что полученные в работе результаты обладают хорошей перспективой для их использования на практике.

ЛИТЕРАТУРА

1. Мацкевич, В. В. Возможности метода отжига в задаче обучения нейронных сетей / В. В. Мацкевич // Технологии передачи и обработки информации : материалы междунар. науч.-техн. семинара, Минск, март – апр. 2022 г. / Белорус. гос. ун-т информатики и радиоэлектроники. – Минск, 2022. – С. 69–73.
2. Hajek, B. Cooling schedules for optimal annealing / B. Hajek // Mathematics of operations research. – 1988. – Vol. 13, iss. 2. – P. 311–329. – URL: http://www.jstor.org/stable/3689827.
3. Li, W. Recurrent restricted Boltzmann machine for chaotic time-series prediction / W. Li, M. Han, J. Wang // 12th Intern. conf. on advanced computational intelligence (ICACI 2020). – P. 439–445. – DOI: 10.1109/ICACI49185.2020.9177510.
4. Sharma, Bh. Extractive text summarization using F-RBM / Bh. Sharma, M. Tomer, Kr. Kriti // J. of statistics and management systems. – 2020. – Vol. 23, iss. 6. – P. 1093–1104. – DOI: 10.1080/09720510.2020.1808353.
5. A novel radar signal recognition method based on a deep restricted Boltzmann machine / D. Zhou [et al.] // Engineering Review. – 2017. – Vol. 37, iss. 2. – P. 165–171.
6. Experiment improvement of restricted Boltzmann machine methods for image classification / Ch. Devi [et al.] // Vietnam J. of Computer Science. – 2021. – Vol. 8, iss. 3. – P. 417–432. – DOI: 10.1142/S2196888821500184.
7. Ensemble RBM-based classifier using fuzzy integral for big data classification / J. Zhai [et al.] // International J. of machine learning and cybernetics. – Springer, 2019. – DOI: 10.1007/s13042-019-00960-3.
8. Nakashika, T. LSTBM: a novel sequence representation of speech spectra using restricted Boltzmann machine with long short-term memory / T. Nakashika // Proc. Interspeech 2018, 2–6 September 2018. – Hyderabad, 2018. – P. 2529–2533. – DOI: 10.21437/Interspeech.2018-1753.
9. Krasnoproshin, V. V. Neural Network Data Processing Based on Deep Belief Networks / V. V. Krasnoproshin, V. V. Matskevich // Communications in Computer and Information Science ; vol. 1282: Open Semantic Technologies for Intelligent System. – Springer, 2020. – P. 234–244. – DOI: 10.1007/978-3-030-60447-9_15.

10. Kingma, D. P. Adam: A Method for Stochastic Optimization / D. P. Kingma, J. L. Ba // Proc. of the 3rd Intern. Conf. on Learning Representations (ICLR 2015). – P. 1–15.
11. Hamis, S. Image Compression at Very Low Bitrate Based on Deep Learned Super-Resolution / S. Hamis, T. Zaharia, O. Rousseau // IEEE 23rd Intern. Symposium on Consumer Technologies (ISCT), Ancona, 19–21 June 2019. – Ancona, 2019. – P. 128–133. – DOI: 10.1109/ISCE.2019.8901038.
12. Oswin, K. Population-Contrastive-Divergence: Does consistency help with RBM training? / K. Oswin, A. Fischer, Ch. Igel // Pattern Recognition Letters. – 2018. – Vol. 102. – P. 1–7. – DOI: 10.48550/arXiv.1510.01624.
13. Li, X. A Novel Restricted Boltzmann Machine Training Algorithm With Dynamic Tempering Chains / X. Li, X. Gao, Ch. Wang // IEEE ACCESS. – 2021. – Vol. 9. – P. 21939–21950. – DOI: 10.1109/ACCESS.2020.3043599.
14. Brugge, K. The flip-the-state transition operator for restricted Boltzmann machines / K. Brugge, A. Fischer, Ch. Igel // Machine Learning. – 2013. – Vol. 93, iss. 1. – P. 53–69. – DOI: https://doi.org/10.1007/s10994-013-5390-3.

REFERENCES

1. Matskevich, V. V. (2022). Vozmozhnosti metoda otzhiga v zadache obucheniya neironnykh setei [Annealing method possibilities in the neural networks training problem]. In *Tekhnologii peredachi i obrabotki informatsii [Information transmission and processing technologies]* (69–73). Minsk: BSUIR (in Russ., abstr. in Engl.).
2. Hajek, B. (1988). Cooling Schedules for Optimal Annealing. *Mathematics of Operations Research*, *13*(2), 311–329. http://www.jstor.org/stable/3689827.
3. Li, W., Han, M., & Wang, J. (2020). Recurrent restricted Boltzmann machine for chaotic time-series prediction. In *12 th International Conference on Advanced Computational Intelligence (ICACI)* (439–445). DOI: 10.1109/ICACI49185.2020.9177510.
4. Sharma, Bh., Tomer, M., & Kriti, Kr. (2020). Extractive text summarization using F-RBM. *Journal of Statistics and Management Systems*, *23*(6), 1093–1104. DOI: 10.1080/09720510.2020.1808353.
5. Zhou, D., Wang, X., Tian, Y., & Wang, R. (2017). A novel radar signal recognition method based on a deep restricted Boltzmann machine. *Engineering Review*, *37*(2), (165–171).
6. Devi, Ch., Chen, R-Ch, Hendry, & Hung, H.-T. (2021). Experiment improvement of restricted Boltzmann machine methods for image classification. *Vietnam Journal of Computer Science*, *8*(3), (417–432). DOI: 10.1142/S2196888821500184.
7. Zhai, J., Zhou, X., Zhang, S., & Wang. T. (2019). Ensemble RBM-based classifier using fuzzy integral for big data classification. *International Journal of machine learning and cybernetics*, (10), 3327–3337. DOI: 10.1007/s13042-019-00960-3.
8. Nakashika, T. (2018) LSTBM: a novel sequence representation of speech spectra using restricted Boltzmann machine with long short-term memory. In *Proc. Interspeech 2018* (2529–2533). DOI: 10.21437/Interspeech.2018-1753.
9. Krasnoproshin, V. V., & Matskevich, V. V. (2020). Neural Network Data Processing Based on Deep Belief Networks. In *Communications in Computer and Information Science. Vol. 1282: "Open Semantic Technologies for Intelligent System"* (234–244). Springer. DOI: 10.1007/978-3-030-60447-9_15.
10. Kingma, D. P., & Ba, J. L. (2015). Adam: A Method for Stochastic Optimization. In *Proc. of the 3rd International Conference on Learning Representations* (1–15).
11. Hamis, S., Zaharia, T., & Rousseau, O. (2019). Image Compression at Very Low Bitrate Based on Deep Learned Super-Resolution. In *IEEE 23rd International Symposium on Consumer Technologies (ISCT)* (128–133). DOI: 10.1109/ISCE.2019.8901038.
12. Oswin, K., Fischer, A., & Igel, Ch. (2018). Population-Contrastive-Divergence: Does consistency help with RBM training? *Pattern Recognition Letters*, (102), 1–7. DOI: 10.48550/arXiv.1510.01624.
13. Li, X., Gao, X, & Wang, Ch. (2021). A Novel Restricted Boltzmann Machine Training Algorithm With Dynamic Tempering Chains. *IEEE ACCESS*, (9), 21939–21950. DOI: 10.1109/ACCESS.2020.3043599.
14. Brugge, K., Fischer, A., & Igel, Ch. (2013). The flip-the-state transition operator for restricted Boltzmann machines. *Machine Learning*, *93*(1), 53–69. DOI: https://doi.org/10.1007/s10994-013-5390-3.

Поступила 13.09.2022

NEURAL NETWORKS TRAINING BASED ON RANDOM SEARCH

V. MATSKEVICH
(Belarusian State University, Minsk)

The paper deals with a state-of-art problem, associated with neural networks training. Training algorithm (with special parallelization procedure) implementing the annealing method is proposed. The training efficiency is demonstrated by the example of a neural network architecture focused on parallel data processing. For the color image compression problem, it is shown that the proposed algorithm significantly outperforms gradient methods in terms of efficiency. The results obtained make it possible to improve the neural networks training quality in general, and can be used to solve a wide class of applied problems.

Keywords: *random search method, gradient descent method, annealing method, training, restricted Boltzmann machine, parallel computing.*