БИБЛИОТЕКА ДЛЯ ПАРАЛЛЕЛЬНОГО ОБУЧЕНИЯ НЕЙРОННЫХ СЕТЕЙ В МУЛЬТИПРОЦЕССОРНЫХ СИСТЕМАХ С ПОМОЩЬЮ МЕТОДА РОЯ ЧАСТИЦ

Рудикова Л. В., Колосов А. А.
Кафедра современных технологий программирования, Гродненский государственный университет имени Янки Купалы
Гродно, Республика Беларусь
E-mail: rudikowa@gmail.com, arseniy.kolosov@gmail.com

В работе представлен подход к построению и обучению слоистых нейронных сетей в мультипроцессорных системах с мощью метода роя частиц. Предполагается, что представленный алгоритм значительно ускорит время обучения небольших нейронных сетей.

Введение

В настоящее время тема нейронных сетей является актуальной в связи с увеличением объёмов данных, появлением более мощных вычислительных систем и спросом на нейронные сети в науке и рынки информационных технологий. Но прежде чем применить нейронную сеть на практике необходимо ее обучить, что является крайне требовательным к вычислительным мощностям процессом. Большие нейронные сети могут состоять из десятков миллионов синапсов, представляемых, как правило, в виде матриц, а обучение с помощью самого распространенного алгоритма – обратного распространения ошибки требует перемножения этих матриц. Перемножение больших матриц может в разы ускориться с помощью графических процессоров, но скорости обучения более малых нейронных сетей графические процессоры могут только навредить. Так же не редко для обучения нейронных сетей применяются вычислительные устройства, включающие в себя несколько процессоров и видеокарт, использование которых одновременно накладывает множество ограничений на алгоритм обучения. Таким образом, для эффективного обучения нейронных сетей на мультипроцессорных системах необходимо создать новый алгоритм обучения.

I. Проектирование библиотеки. Выбор программных средств и особенности реализации

C++ - язык программирования, являющийся компилируемым и имеющий статическую типизацию. Основной причиной выбора языка программирования в пользу C++ является возможность управлением каждого аспекта кода, экономить память и вычислительные ресурсы, поддержка OpenCL – фреймворка для написания подпрограмм, использующих параллельные вычисления на CPU и GPU, позволяя программировать на мультипроцессорных системах [1].

Многослойные нейронные сети являются самой распространенной архитектурой нейронных сетей, обучение и применений которой требующее, как правило, умножения матриц.

Стадия обучения нейронной сети может проводиться на всех чипах CPU и GPU внутри одного устройства. Таким образом, задача должна разбиться на подзадачи, решаемые каждым процессором по отдельности с учетом дорогостоящего обмена информацией между графическим процессором и центральным процессором (см. рис. 1), а также отсутствие обмена данными между чипами. Алгоритм обучения должен исключать возможность обмена данными между процессорами, а также минимизировать поток обмена данных между ЦП и остальными процессорами. Более того, задача должна обладать свойством массового параллелизма, необходимое для эффективных вычислений на видеокарте.

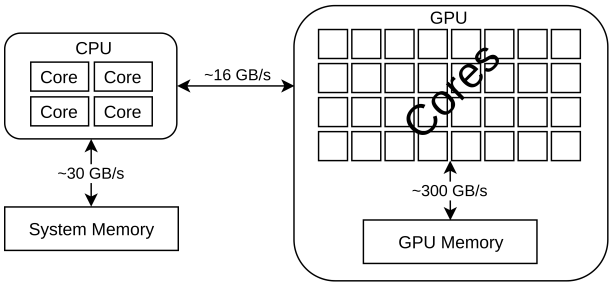

Рис. 1 – Схема скорости обмена данными между чипами

Метод роя частиц – метод численной оптимизации, который можно адаптировать под обучение нейронной сети [2]. Формально, алгоритм представляет из себя агентную систему, где каждый агент (частица) является отдельным решением исходной задачи. Частицы перемещаются в пространстве решений, передвигаясь по направлению к лучшему решению среди всех агентов и собственных лучших результатов с определенной долей вероятности. Более подробное описание алгоритма:

Для работы метода роя частиц необходимо задать некую целевую функцию – функция многих переменных $f : \mathbb{R}^n \to \mathbb{R}$, экстремум которой необходимо найти. Необходимо так же задать пространство решений - множество $\mathbb{S} \subseteq \mathbb{R}$.

Требуется задать K частиц, i-ая из которых представлена координатой из множества решений $x_i \in \mathbb{R}$ и начальной скоростью $v_i \in \mathbb{R}$. Пусть $loc_i \in \mathbb{R}$ – лучшее состояние, принимаемое когда-либо частицей i, а $glob \in \mathbb{R}$ – лучшее состояние, принимаемое всеми частицами. Общий вид метода роя частиц:

- Для каждой частицы $i \in \overline{1, \mathrm{K}}$
  - Сгенерировать случайно равновероятно начальное состояние $x_i \in \mathbb{S}$.
  - Присвоить это значение лучшему локальному состоянию $loc_i = x_i$.
  - Сгенерировать случайным равновероятным образом начальное состояние $v_i \in \mathbb{S}$.
  - Если $f(loc_i) < f(glob)$, то обновить наилучшее глобальное состояние $glob = loc_i$.
- До тех пор, пока не выполнено нужное число итераций или значение $f(glob)$ не будет меньше некоторого заданного значения:
  - Для каждой частицы $x_i \in \mathbb{S}$.
  - Сгенерировать случайные векторы $r_{loc}, r_{glob} \in [0..1]^n$.
  - Обновить скорость частицы $v_i = v_i s + k_{loc} r_{loc}(loc_i - x_i) + k_{glob} r_{glob}(glob - x_i)$.
  - Обновить координату частицы $x_i = v_i + x_i$.
  - Если $f(x_i) < f(loc_i)$, то $loc_i = x_i$ и
    - Если $f(loc_i) < f(glob)$, то $glob = loc_i$.
- $glob$ - лучшее найденное решение.

Коэффициенты $s, k_{loc}, k_{glob} \in [0..1]$ – константы, влияющие на замедление, значимость лучшего локального состояния и лучшего глобального соответственно.

Алгоритм обучения:

- Инициализировать все частицы, выборку и задачи.
- До тех пор, пока не выполнено нужное число итераций или нейронная сеть не обучится до приемлемого уровня:
  - Для каждой задачи посчитать значения частиц на каждом примере из выборки (параллельно для каждой задачи).
  - Для каждой задачи посчитать ошибку, полученную на каждом примере (параллельно для каждой задачи).
  - Для каждой задачи высчитать среднюю ошибку для каждой особи (параллельно).
  - Для каждой задачи найти лучшую частицу (параллельно).
  - Обновить лучшие локальные состояния в каждой частице для каждой задачи (параллельно).
  - Загрузить новое лучшее глобальное решение в каждую задачу (после инициализации - это единственное место обмена ощутимых объемов данных, хоть и оно – минимально).
  - Для каждой частицы каждой задачи обновить состояния (сделать смещение на вектор скорости, обновив вектор скорости).
- Лучшее глобальное решение – результат обучения.

Из-за особенностей фреймворка OpenCL накладывается ограничение на многомерные массивы данных, что усложняет разработку библиотеки (см. рис. 2).

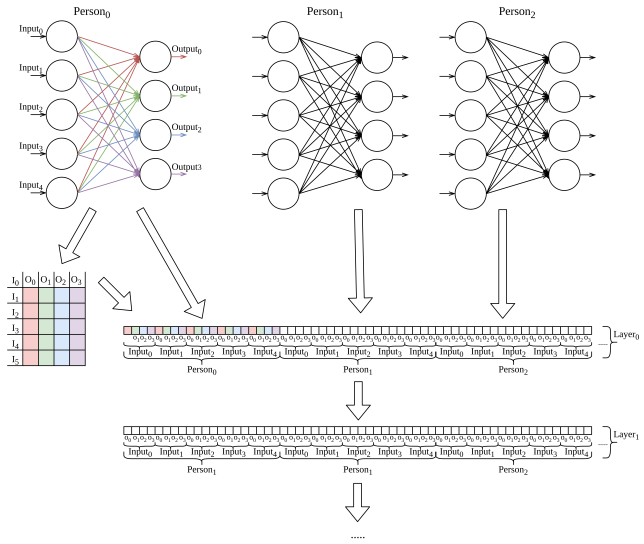

Рис. 2 – Схема хранения нейронных сетей в памяти чипов

II. Заключение

Таким образом, предполагаемых подход к обучению нейронных сетей может быть рассмотрен к применению. Предполагаемый алгоритм обучения может значительно ускорить процесс обучения небольших нейронных сетей с минимальными затратами к памяти, максимизируя преимущества вычислительного устройства с одними или несколькими процессорами и видеокартами, нивелируя слабые места таких вычислительных устройств, при этом не требуя наличия определенных операционных систем или специфичных аппаратных средств конкретных производителей.

Список литературы

1. Документация к OpenCL – [Электронный ресурс]. – Режим доступа: https://www.khronos.org/op.
2. Clerc M. (2012). Standard Particle Swarm Optimization.