DOI 10.37882/2223–2966.2022.06.31

СТРУКТУРИРОВАНИЕ БОЛЬШИХ ДАННЫХ (BIG DATA) ДЛЯ ОБУЧЕНИЯ НЕЙРОСЕТИ CNN-ELM В ЗАДАЧАХ ПРОГРАММИРОВАНИЯ СИСТЕМ РАСПОЗНАВАНИЯ

STRUCTURING BIG DATA (BIG DATA) TO TRAIN THE CNN-ELM NEURAL NETWORK IN THE TASKS OF PROGRAMMING RECOGNITION SYSTEMS

M. Rasulov

Summary. Based on the theory of scalability of machine learning, within the framework of this study, the concept of neural network training in the tasks of programming recognition systems is proposed. The large-scale approach has some limitations. In this study, to solve the problems of programming recognition systems, an approach is proposed for integrating asynchronous ELM components into the CNN convolutional network, based on the MapReduce parallel computing platform as a classifier module. This approach can save a lot of training time than single CNN-ELM models trained alone. This approach can improve scalability efficiency in machine learning systems by combining convolution and scaling approaches.

Keywords: big data structuring, deep learning, convolutional, neural network, big data.

Расулов Мирзо Максудович
Аспирант, МИРЭА — Российский технологический университет
mirzorasulov@gmail.com

Аннотация. На основе теории масштабируемости машинного обучения в рамках данного исследования предложена концепция обучения нейросети в задачах программирования систем распознавания. Масштабный подход имеет некоторые ограничения. В данном исследовании, для решения задач программирования систем распознавания, предложен подход для интеграции в сверточную сеть CNN асинхронных компонентов ELM, основанный на платформе параллельных вычислений MapReduce в качестве модуля классификатора. Такой подход может сэкономить больше времени на обучение, чем одиночные модели CNN-ELM. Этот подход может улучшить эффективность масштабируемости в системах машинного обучения, объединив подходы свертки и масштабирования.

Ключевые слова: структурирование больших данных, глубокое обучение, сверточная, нейронная сеть, большие данные.

Введение

Сегодня наблюдается массовый рост объема информации и данных. Однако преимущества больших данных нивелируются, если ни одна из систем обработки данных не может адаптироваться к постоянно растущим данным достаточно быстро. Интеллектуальный глубокий анализ больших данных требует новых масштабных подходов машинного обучения в условиях ограниченного времени и ресурсов. Решение задач объема и скорости имеют важное значение для преодоления проблем больших данных [1]. Таким образом, требуется особый подход, чтобы максимизировать возможности использовать имеющееся обеспечение для эффективной работы на скорости, масштабируемости и простоте, представленных в реальном времени. Масштабирование должно обеспечить более быструю свертку больших данных за счет добавления различных размеров элементов в существующий пул.

Для повышения эффективности масштабируемости общепринятым подходом является параллельное выполнение процесса анализа больших данных. Параллельные вычисления должны осуществлять одновременное использование нескольких вычислительных ИС для решения масштабных задач программирования систем распознавания путем разделения процесса на более простые слои с учетом интеграции общем системы управления [2].

Для задач параллельного программирования, Google разработали платформу MapReduce [3], которая является фреймворком для распределенной обработки больших данных в кластере. Такое распределение задач на уровни является также характерным для сверточных моделей CNN, которое получает результат от параллельных вычислений и использует систему сверточных операций в множестве вычислительных ядер. Для решаемой задачи программирования систем распознавания

1: Define $P = \mathrm{b}m/k\mathrm{c}$
2: Randomly partition the training, giving P examples to each machine.
3: **for all** $i \in \{1,...,k\}$ **parallel do**
4: Randomly shuffle the data on machine i
5: Initialize $w_{i,0} = 0$
6: **for all** $p \in \{1,...,P\}$ **do**
7: Get the p^{th} training on the i^{th} machine $c^{i,p}$
8: $w_{i,p} \leftarrow w_{i,p-1} - \eta \delta c_i(w_{i,p-1})$
9: **end for**
10: **end for**
11: Aggregate from all machines $v = \frac{1}{k}\sum_{i=1}^{k} w_i$
12: **return** v

Algorithm 1: SimuParallelSGD(Training $\{x^1,...,x^m\}$; Learning Rate η; Machines k)

Рис. 1. Алгоритм параллельного SGD

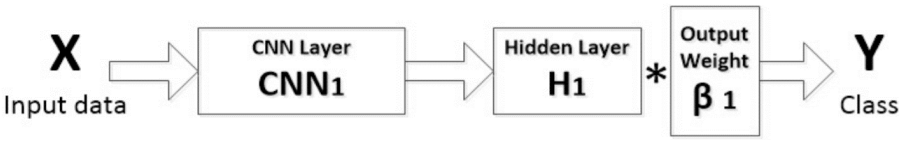

Рис. 2. Архитектура интеграции в сверточную сеть CNN компонентов ELM: выходные данные последнего слоя свертки представляются в виде скрытых узлов

для ускорения вычислений используется распараллеливание графических процессоров (GPU). Однако подход масштабирования при программировании систем распознавания по-прежнему имеет ограничения, вызванные главным образом объемом памяти, доступной на графических процессорах.

Изучая ограниченные возможности масштабирования, на базе MapReduce для распределения вычислений больших данных можно использовать алгоритмы CNN. Однако, при программировании систем распознавания важно ввести классификатор, такой как ELM. Для их соединения запустим процесс параллельного стохастического градиентного спуска (SGD)

Изложение основного материала

Для обоснования модели рассмотрим процесс ее реализации в параллельном SGD. В данном системе доступ к обучающим данным осуществляется отдельно каждой моделью и интегрируется только после ее завершения. Такой алгоритм параллельного SGD описан ниже (рис. 1).

После завершения оптимизации усредненный вес каждого слова заменяет обычный вес из SGD. То есть процесс основан на идее разбиения данных на слои, однако построенное на его основе приложение для программирования систем распознавания требует большого количества априорной графической информации. Так, если имеются неограниченные данные обучения в пределах того же распределения, то можно построить обобщающую функцию отображения $\hat{\beta}$. Но в системе по работе с большими данными при увеличении выборки обучающих данных или их массива (m) становится возможным использовать $\hat{\beta}(w)$ как целевую функцию, только если определить w — *базовые и классовые* параметры обучения. В связи с этим для программирования систем распознавания можно предположить, что модуль MapReduce на основе ELM будет более эффективен при условии использования параллельных вычислений на базе слоев CNN. Поэтому мы утверждаем, что сам алгоритм CNN на основе обратного распространения нуждается в итерациях, чтобы получить оптимальное решение.

Большинство реализаций CNN используют GPU для ускорения операции свертки, которая требовала сотни ядер процессоров. Однако количество многоядерных процессоров намного меньше, чем может обеспечить GPU.

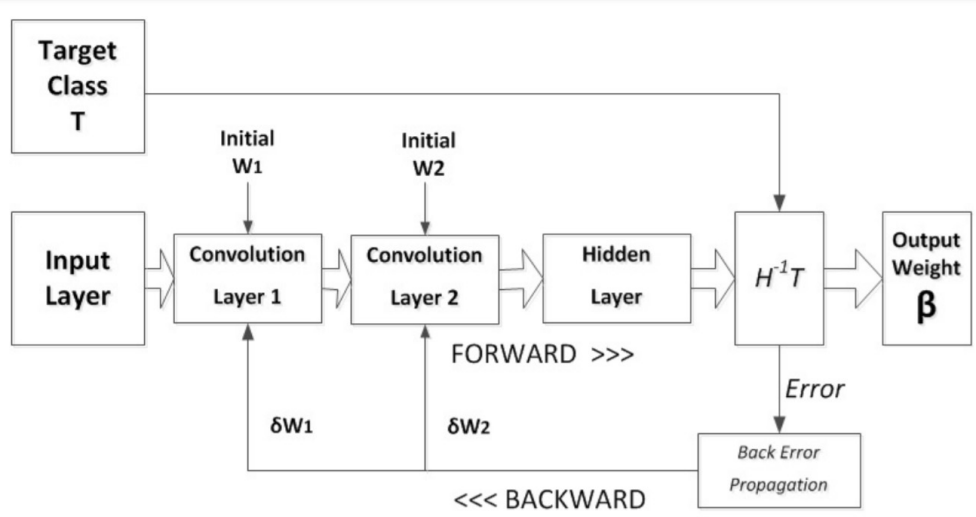

Рис. 3. Архитектура CNN

Define $P = \lfloor m/k \rfloor$
Randomly partition the training, giving P examples to each machine.
Initialize CNN weight parameters similar for k machines
for all $i \in \{1, ..., k\}$ **parallel do**
  Randomly shuffle the data on machine i
  for all $j \in \{1, ..., e\}$ **do**
    Reset $\Sigma U = 0; \Sigma V = 0$
    for all $p \in \{1, ..., P\}$ **do**
      Get $\mathbf{H}$ from p^{th} CNN training on the i^{th} machine $c^{i,p}$
      Compute $\Sigma U = \Sigma U + H^T H$
      Compute $\Sigma V = \Sigma U + H^T T$
      Compute β
      Propagate ELM Error back to CNN
      Update Kernel Weights $W_{ij}^{(l)}$ and bias $b_{ij}^{(l)}$
    end for
  end for
end for
Aggregate for each l^{th} layers $\hat{W}^{(l)} = \frac{1}{k} \sum_{i=1}^{k} W_i^{(l)}$
Aggregate for each l^{th} layers $\hat{b}^{(l)} = \frac{1}{k} \sum_{i=1}^{k} b_i^{(l)}$
Aggregate for each l^{th} layers $\hat{\beta} = \frac{1}{k} \sum_{i=1}^{k} \beta_i$
return $\hat{W}, \hat{b}, \hat{\beta}$

Рис. 4. Алгоритм расчета на основе набора данных MNIST

Мы использовали общую архитектуру интеграции в сверточную сеть CNN компонентов ELM, когда выход последнего слоя свертки подается в виде скрытых узлов (рис. 2).

Идея обратного сходна с плотно связанным методом ошибки обратного распространения CNN с функцией стоимости и распространялся обратно с помощью SGD для оптимизации весовых ядер слоев свертки (рис. 3).

Для определения эффективности работы метода интеграции в сверточную сеть CNN компонентов ELM на базе систем распознавания в основу будет положен общий набор данных MNIST. Мы расширили набор данных MNIST на 3× больше, добавив 3 типа шумов изображения (рис. 4).

Мы разделили набор на числовые и символьные данные, включая множество изображений с шумом.

a. Размер и количество набора данных

Концепции Набора данных	Входные	Выходы	Данные
MNIST	784	10 (0-9)	240,000
Not MNIST	784	20 (0-9,A-J)	900,000

b. Метод Оценки

Данные Набор	Оценка Метод	Обучение	Тестирование
MNIST	Несогласие (5× испытания на разных компьютерах)	240,000	40,000
Not MNIST	Перекрестная Проверка В 6 Раз	750,000	150,000

c. Измерение производительности

Мера	Спецификация
Точность	Точность классификации в % от *#Правильно Классифицированы* / *#Общее Количество Экземпляров*
Точность испытания	Точность измерения данных испытаний, которые не входят в состав обучающего набора.
Ошибка	Статистическое измерение взаимного согласия по категориальным позициям.

Рис. 5. Размерность, количество и метод оценки набора данных

Проблема с not-MNIST числовым и not-MNIST алфавитом заключается во многих сходствах между классом 1 с классом I, классом 4 с классом A и другими похожими изображениями (рис. 5).

Производительность CNN-ELM может быть улучшена с помощью алгоритма обратного распространения. Также нам необходимо выбрать соответствующий параметр скорости обучения, количество партий и количество итераций, которые могут повлиять на конечную производительность.

Мы сравнили точность модели интеграции в сверточную сеть CNN асинхронных компонентов ELM со средней моделью 2 разделов и средней моделью 5 разделов. К сожалению, производительность средней модели CNN-ELM снизилась, так как расширенный MNIST был построен из одного и того же дистрибутива на каждом размере раздела, а не на not-MNIST.

Вывод

Предложенный метод интеграции в сверточную сеть CNN компонентов ELM дает лучшую возможность масштабирования для параллельной обработки большого набора данных. Мы можем разделить большой набор данных, назначить классификатор для каждого слоя, а затем просто агрегировать результат, усредняя весовые параметры. В дальнейшем следует разработать эти методы на другой платформе CNN с вычислениями на GPU для более крупного сложного набора данных. Также станет возможным исследовать другие оптимальные параметры обучения на более сложной архитектуре CNN, то есть регуляризацию dropout и dropconnect и параметры распада.

ЛИТЕРАТУРА

1. Arif Budiman, Mohamad Ivan Fanany, Chan Basaruddin. Distributed Averaging CNN-ELM for Big Data.— 2016. URL: https://arxiv.org/abs/1610.02373v1
2. Minjung Ryu, Hong-Linh Truong, Matti Kannala. Understanding quality of analytics trade-ofs in an end-to-end machine learning-based classifcation system for building information modeling // Big Data 8:31.— 2021. URL: https://www.researchgate.net/publication/349337202_Understanding_quality_of_analytics_trade-offs_in_an_end-to-end_machine_learning-based_classification_system_for_building_information_modeling
3. Amarendra Mohanty, Ranjana. Usage of Predictive Research on further Business // International Journal of Innovative Technology and Exploring Engineering (IJITEE) 8: 11.— 2019.— URL: https://www.researchgate.net/publication/354052271_Usage_of_Predictive_Research_on_further_Business

Журнал «Современная наука: актуальные проблемы теории и практики»