НИТУ МИСИС

Институт ИТАСУ

Кафедра инженерной кибернетики

Квалификация (степень): **магистр**

Направление подготовки: 09.04.03 «Прикладная информатика»

Группа: МПИ-20-4-2

Курсовая работа по курсу

«Облачные технологии»

III семестр 2021–2022 у. г.

Распределённое асинхронное обучение нейронных сетей. Проверка эффективности обучения на эталонных и реальных датасетах

Преподаватель _____ / доцент, к.т.н. Курочкин И. И. /

Подпись должность, уч. степ., Фамилия И.О.

Учащиеся:

Андреев Савелий

Дзюба Ирина

Ефремова Марьяна

Новицкий Дмитрий

Оглавление

Введение

1. Аналитический обзор литературы

1.1. РАСПРЕДЕЛЕННОЕ ОБУЧЕНИЕ СЕТИ

1.1.1. Подходы к распределенной реализации SGD

С развитием глубокого обучения и увеличения тренировочных данных перед разработчиками моделей нейронных сетей встала проблема обучаемости нейронных сетей. Обучение таких сетей требовало огромных ресурсов хранения данных, а также занимало большое количество времени.

Для того, чтобы преодолеть это «узкое место» глубоких нейронных сетей и добиться значительного снижения времени обучения, сам процесс обучения должен быть распределён между множеством процессоров или графических ускорителей с целью строгого масштабирования, то есть с увеличением количества вычислительных узлов время обучения сети в теории должно уменьшаться пропорционально.

Существуют два основных подхода к распределенному стохастическому градиентному спуску (SGD) для обучения глубоких нейронных сетей: синхронный all-reduce SGD, который основывается на быстром коллективном общении узлов системы [1][2], и асинхронный SGD, который использует параметрический сервер [3][4].

Оба метода имеют свои недостатки при масштабировании. Синхронный SGD теряет производительность при замедлении работы хотя бы одного узла, использует вычислительные ресурсы не в полном объёме и не толерантен к выходу из строя вычислительных процессоров или целых узлов. В свою очередь асинхронный подход использует параметрические сервера, создавая тем самым коммуникационную проблему «узкого горлышка» и неиспользуемых сетевых ресурсов, замедляя тем самым сходимость градиента.

Так как для нашей задачи необходимо решить задачу обучения глубокой нейронной сети на грид-системе, то рассмотрим более детально асинхронные методы распределения обучения сети, так как в грид-системах используются узлы различной вычислительной мощности, и к тому же получение результата со всех узлов системы не гарантировано.

1.1.2. Параллелизм данных и модели

При распределении обучения модели глубокой нейронной сети может быть применено два подхода: разделение обучающего датасета на разные части для различных

устройств (параллелизм данных) или разделение самой модели между различными устройствами (параллелизм модели).

При параллелизме данных всё исходное обучающее множество делится на N частей, которые отправляются на N узлов распределенной системы (рисунок 1). Каждый из этих узлов содержит копию модели нейронной сети и параметры этой модели Wn. Процесс обучения проходит следующим образом:

- Каждый узел считывает небольшую выборку из тренировочных данных и вычисляет на её основе локальные градиенты ∇W_n.
- Каждый узел посылает вычисленные значения локальных градиентов «мастер» - узлу. После получения градиентов со всех узлов «мастер» - узел агрегирует все градиенты и обновляет модель новым вычисленным значением.
- «Мастер» - узел распространяет модель на все другие узлы.
- Операция повторяется до завершения обучения модели.

Кроме параллелизма данных также существует параллелизм самой модели нейронной сети, при котором модель распределяется между разными узлами. Данный подход применяется в том случае, если модель слишком велика для расположения на одном узле.

Схемы параллелизма распределенного обучения представлены на рисунке 1.

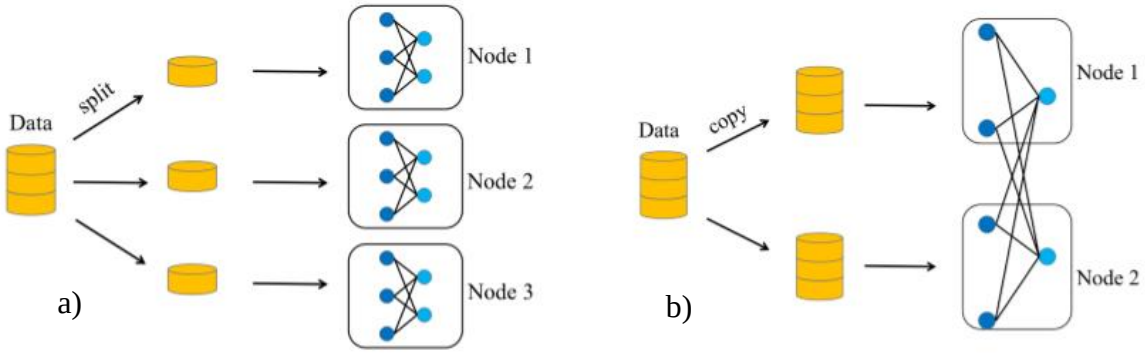

Рисунок 1 – Схемы параллелизма нейронных сетей (a – параллелизм данных, b – параллелизм модели)

1.1.3. Централизованные и децентрализованные архитектуры

Логическое построение архитектуры распределенной системы влияет на режимы коммуникации узлов и производительность сети. Использование параметрического сервера [5][6] является наиболее популярной централизованной архитектурой в распределенном обучении глубоких сетей. Такая архитектура обычно включает в себя серверный узел и «рабочие» узлы. На сервере находятся глобальные параметры модели, а на «рабочих» узлах находятся локальные копии глобальной модели. Если серверный узел представлен несколькими вычислительными машинами, то каждая такая машина содержит свою часть всех параметров модели.

Что касается «рабочих» узлов, то каждый из них содержит полную копию глобальной модели при параллелизме данных или часть модели при параллелизме модели. «Рабочие» узлы общаются с сервером посредством push/pull операций, тогда как между «рабочими» узлами нет никакой связи вообще.

На этапе операции push рабочие узлы сообщают параметрическому серверу градиенты ΔW, вычисленные локально. Далее, в зависимости от алгоритма обновления глобальной модели, параметрический сервер «с ходу» обновляет глобальную модель полученным значением градиента по формуле 1.

$$W' = W - \alpha * \Delta W, \quad (1)$$

где W' – обновленная модель;

W – значение параметров глобальной модели до обновления;

a – коэффициент обновления;

ΔW – градиенты, вычисленные локально.

Такая формула подходит для асинхронного обновления, которое будет рассмотрено далее. Затем на операции pull «рабочий» узел получает уже обновленную модель.

Для синхронного обновления характерно то, что операции pull и push совершаются одновременно всеми «рабочими» узлами. В связи с этим необходимо первоначально агрегировать результаты вычислений всех узлов и только затем обновить модель. В общем виде это представлено в формуле 2.

$$W' = W - \alpha * \sum \Delta W, \quad (2)$$

где W' – обновленная модель;

W – значение параметров глобальной модели до обновления;

a – коэффициент обновления;

$\sum \Delta W$ – агрегированные градиенты, вычисленные локально.

На рисунке 2 показана традиционная архитектура параметрического сервера, в которой сервер отвечает за хранение и обновление параметров глобальной модели. Она подвержена узкому месту в виде полосы пропускания сети со стороны сервера, особенно при наличии большого количества «рабочих» узлов. Древовидные параметрические серверы [7][8][9] в некоторой степени смягчают такие узкие места.

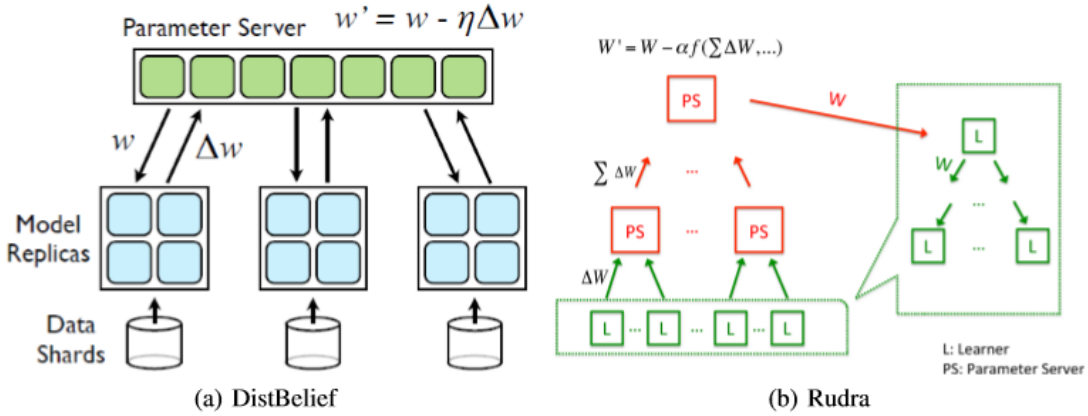

Рисунок 2 – Архитектуры параметрических серверов. (a – традиционная архитектура, b – древовидая архитектура)

Децентрализованная архитектура лишена этого недостатка. Также, как и в архитектуре параметрического сервера с параллелизмом данных, каждый узел содержит копию полной модели сети, но отличие заключается в том, что общение происходит между всеми узлами с помощью операции all-reduce, вместо push/pull. В ходе этого общения происходит обмен градиентами и различными параметрами. Тем не менее, операция all-reduce имеет множество реализаций с существенно отличной производительностью, что означает, что она может влиять на накладные расходы на коммуникацию.

1.1.4. Обновление глобальной модели

Исходя из различий в ширине каналов связи и вычислительных мощностей, некоторые узлы могут вычислять градиенты быстрее, чем другие. Основная проблема данного обстоятельства заключается в вопросе синхронизации градиентов на различных вычислительных узлах. Существует три различных метода обновления градиентов: синхронное, асинхронное и обновление с задержкой.

1.1.4.1. Синхронное обновление

При синхронном обновлении сервер не обновляет глобальную модель сети до тех пор, пока не получит градиенты от всех «рабочих» узлов на каждой итерации. Из этого вытекает описанная выше проблема – быстрые узлы ждут медленных. Одной из известных реализаций синхронного обновления является bulk synchronous parallel (BSP) [10]. Характерной особенностью синхронного режима является то, что сервер всегда будет получать последние градиенты всех узлов, которые не влияют на сходимость модели. Однако, быстрые узлы ничего не выполняют при ожидании медленных узлов, что приводит к пустой трате ресурсов. Кроме того, это также замедлит общее время обучения.

1.1.4.2. Асинхронное обновление

Асинхронные алгоритмы, такие как Hogwild [11], преодолевают вышеперечисленные проблемы. При асинхронных обновлениях быстрые «рабочие» узлы не ждут медленных. Один «рабочий» узел может посылать свои локальные градиенты на сервер, в то время как другие вычисляют свои градиенты, как показано на рисунке 3.

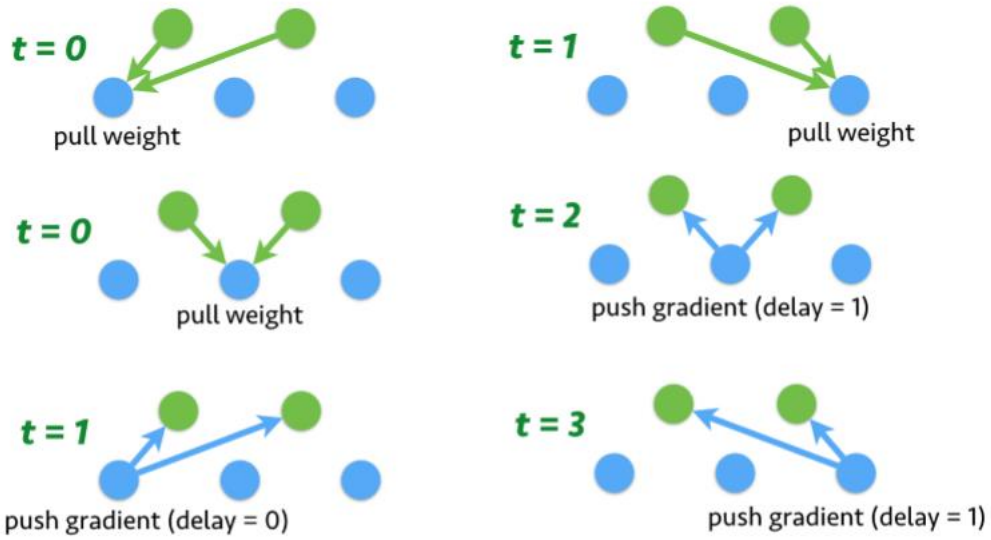

Рисунок 3 – Асинхронное обновление параметров модели. (зеленые – серверы, синие – «рабочие» узлы)

Основная проблема, возникающая при асинхронных обновлениях – это неактуальность данных, потому что быстрые «рабочие» узлы всегда могут использовать устаревшие параметры, которые ставят под угрозу сходимость модели. Кроме того, самый

быстрый «рабочий» узел обновляет свои локальные параметры только по своему обучающему подмножеству, что заставляет локальную модель отклоняться от глобальной модели.

Для преодоления недостатков асинхронных обновлений исследователи попытались ограничить неактуальность параметров. При таком обновлении быстрые «рабочие» узлы будут использовать неактуальные параметры, но эта неактуальность (как и задержка на рисунке 3) будет ограничена [12][13]. Такое ограничение в некоторой степени смягчает проблему неактуальности и увеличивает скорость обучение модели.

Однако, выбор ограничения неактуальности тоже является вопросом, ведь если оно будет слишком большим, то это означает полную асинхронность, а если слишком маленьким, то это соответствует синхронному обновлению модели.

1.1.4.3. Периодичность обновления

При обучении ГНС с использованием традиционного распределенного SGD, каждый работник на каждой итерации выполняет один шаг SGD и вычисляет локальные градиенты, затем обменивается градиентами и параметрами с другими узлами. Связь между узлами происходит в конце каждой итерации.

Предположим, что мы распараллеливаем задачу обучения ГНС на K узлах с помощью T итераций, сложность раундов идеального параллельного SGD равна O(T). В связи с такой высокой сложностью раундов связи, многие работы в распределении обучения моделей предлагают уменьшить частоту обмена градиентами и параметрами между узлами.

При усреднении модели отдельные параметры модели, обученные на локальных узлах, усредняются периодически. Она происходит в большинстве τ итераций, где τ является фактором периода. Усреднение происходит в каждой итерации, т. е. $\tau = 1$ будет идентична идеальному параллельному SGD, а если усреднение происходит только в конце тренировки, т. е. $\tau = T$, то это равнозначно усреднению в конце обучения («one-shot averaging»). Также есть другой вариант усреднения, при $1 < \tau < T$. На рисунке 4 изображены все 3 варианта, упоминавшихся ранее.

Рисунок 4 – Сравнение периодичности общения в разных режимах (зеленый – вычислительные операции, желтый – операции общения)

Экспериментальные работы [14][15][16][17] подтвердили, что усреднение модели может снизить накладные расходы на связь во время обучения до тех пор, пока этот период является приемлемым. Кроме того, некоторые теоретические исследования [18][19][20][21] дали анализ того, почему усреднение модели может обеспечить хорошую скорость сходимости.

Однократное усреднение («one-shot averaging»), которое требует коммуникации только в конце обучения, является экстремальным случаем усреднения модели. Однократное усреднение имеет достаточную производительность как по выпуклым [14], так и по некоторым невыпуклым [15] задачам оптимизации. Однако согласно работе Чжана [16] некоторые задачи невыпуклой оптимизации не могут быть решены с помощью однократного усреднения. Решением данной проблемы является более частое усреднение.

Список использованных источников

1. Dipankar Das, Sasikanth Avancha, Dheevatsa Mudigere, Karthikeyan Vaidynathan, Srinivas Sridharan, Dhiraj Kalamkar, Bharat Kaul, and Pradeep Dubey. Distributed Deep Learning Using Synchronous Stochastic Gradient Descent.– [Электронный ресурс]: arXiv preprint arXiv:1602.06709.– 2016.– URL: https://arxiv.org/pdf/1602.06709.pdf (дата обращения: 17.04.2019).
2. Jianmin Chen, Rajat Monga, Samy Bengio, and Rafal Jozefowicz. Revisiting Distributed Synchronous SGD.– [Электронный ресурс]: arXiv preprint arXiv:1604.00981.– 2016.– URL: https://arxiv.org/pdf/1604.00981.pdf (дата обращения: 17.04.2019).
3. Jeffrey Dean, Greg Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Mark Mao, Andrew Senior, Paul Tucker, Ke Yang, Quoc V. Le, and Andrew Y. Ng. Large Scale Distributed Deep Networks. // In Advances in Neural Information Processing Systems 25.– 2012.– С. 1223–1231.
4. Trishul Chilimbi, Yutaka Suzue, Johnson Apacible, and Karthik Kalyanaraman. Project Adam: Building an Efficient and Scalable Deep Learning Training System. // In 11th USENIX Symposium on Operating Systems Design and Implementation.– 2014.– С. 571–582.
5. Mu Li, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. // In 11th Symposium on Operating Systems Design and Implementation.– 2014.– С. 583–598.
6. Mu Li, David G Andersen, Alexander J Smola, and Kai Yu. Communication efficient distributed machine learning with the parameter server. // In Advances in Neural Information Processing Systems.– 2014.– С. 19–27.
7. Luo Mai, Chuntao Hong, and Paolo Costa. Optimizing network performance in distributed machine learning. // In 7th Workshop on Hot Topics in Cloud Computing (HotCloud 15).– 2015.
8. Suyog Gupta, Wei Zhang, and Fei Wang. Model accuracy and runtime tradeoff in distributed deep learning: A systematic study. // In 2016 IEEE 16th International Conference on Data Mining (ICDM) .–IEEE.– 2016.– С. 171–180.
9. Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. Gaia: Geo-distributed machine learning approaching LAN speeds. // In 14th Symposium on Networked Systems Design and Implementation.– 2017.– С. 629–647.

10. Leslie G Valiant. A bridging model for parallel computation. // Communications of the ACM.– 33(8).– 1990.– С. 103–111.
11. BenjaminRecht, ChristopherRe, StephenWright, and FengNiu. Hogwild: A lock-freeapproach toparallelizing stochastic gradient descent. // In Advances in neural information processing systems.– 2011.– С. 693–701.
12. James Cipar, Qirong Ho, Jin Kyu Kim, Seunghak Lee, Gregory R Ganger, Garth Gibson, Kimberly Keeton, and Eric Xing. Solving the straggler problem with bounded staleness. // In Presented as part of the 14th Workshop on Hot Topics in Operating Systems.– 2013.
13. Qirong Ho, James Cipar, Henggang Cui, Seunghak Lee, Jin Kyu Kim, Phillip B Gibbons, Garth A Gibson, Greg Ganger, and Eric P Xing. More effective distributed ml via a stale synchronous parallel parameter server. // In Advances in neural information processing systems.– 2013.– С. 1223–1231.
14. Martin Zinkevich, Markus Weimer, Lihong Li, and Alex J Smola. Parallelized stochastic gradient descent. // In Advances in neural information processing systems.– 2010.– С. 2595–2603.
15. Ryan McDonald, Keith Hall, and Gideon Mann. Distributed training strategies for the structured perceptron. // In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics.– Association for Computational Linguistics.– 2010.– С. 456–464.
16. Jian Zhang, Christopher De Sa, Ioannis Mitliagkas, and Christopher Ré. Parallel sgd: When does averaging help?.– [Электронный ресурс]: arXiv preprint arXiv:1606.07365.– 2016.– URL: https://arxiv.org/pdf/1606.07365.pdf (дата обращения: 20.04.2019).
17. Hang Su and Haoyu Chen. Experiments on parallel training of deep neural network using model averaging.– [Электронный ресурс]: arXiv preprint arXiv:1507.01239.– 2015.– URL: https://arxiv.org/pdf/1507.01239.pdf (дата обращения: 20.04.2019).
18. Hao Yu, Sen Yang, and Shenghuo Zhu. Parallel restarted sgd with faster convergence and less communication: Demystifying why model averaging works for deep learning. // In Proceedings of the AAAI Conference on Artificial Intelligence.– volume 33.– 2019.– С. 5693–5700.
19. Sebastian Urban Stich. Local sgd converges fast and communicates little. // In ICLR 2019 International Conference on Learning Representations.– 2019.
20. Yossi Arjevani and Ohad Shamir. Communication complexity of distributed convex learning and optimization. // In Advances in neural information processing systems.– 2015.– С. 1756–1764.

21. Fan Zhou and Guojing Cong. On the convergence properties of a k-step averaging stochastic gradient descent algorithm for nonconvex optimization. // In Proceedings of the 27th International Joint Conference on Artificial Intelligence.– AAAI Press.– 2018.– C. 3219–3227.