Київський національний університетімені Тараса Шевченка
Факультет комп'ютерних наук та кібернетики

Кафедра обчислювальної математики

Кваліфікаційна робота

на здобуття ступеня бакалавра

за спеціальністю 113 Прикладна

математикана тему:

Дослідження застосування розподілених методів для навчання нейромереж

Виконав студент IV курсу
Галиш Антон Володимирович

Науковий керівник:
асистент
Денисов Сергій Вікторович

100 (відмінно)

Засвідчую, що в цій роботі немає запозичень з праць інших авторів без відповідних посилань.
Студент

Роботу розглянуто й допущено до захисту на засіданні кафедри обчислювальної математики
«__29____» _травня__ 2023 р.
протокол № 8
Завідувач кафедри
С. І. Ляшко

Київ — 2023

Зміст

Вступ

У сучасному світі, де обсяги даних постійно зростають, а задачі аналізу цих даних, стають все більш складними, нейромережі здобули велику популярність як потужний інструмент машинного навчання. Їхня здатність виявляти складні залежності в даних є важливою складовою сучасного прогресу в багатьох галузях, включаючи комп'ютерний зір, обробка мови, рекомендаційні системи та багато інших.

Однак, все більші та більші обсяги даних стає складніше зберігати на одному фізичному носії та відповідно довше обробляти. Можливим рішенням цієї проблеми є розподілене навчання: підхід, який дозволяє прискорити процес оптимізації шляхом розподілу великого набору даних між кількома обчислюваними пристроями. Це дозволяє не створювати один потужний комп’ютер, а об’єднати в мережу декілька менш потужних. До того ж така мережа є більш стійкою до збоїв(проблемний пристрій можна виключити з мережі) та більш гнучка до масштабування(в мережу можна додавати нові пристрої).

Метою даної дипломної роботи є дослідження розподілених методів для навчання нейромережі, а саме як архітектура мережі обчислюваних пристроїв та розподіл даних по них впливає на кінцевий результат. У роботі будуть розглянуті три розподілених методи: DSGD, DSGD з моментом та DSGD з відстеженням моментів.

Скорочення та умовні позначення

I_n	одинична матриця розміру $n \times n$
1_n	вектор з одиниць розмірності n
$\mathbf{0}$	нульовий вектор довільної розмірності
$\deg(v)$	степінь вершини графа
$\langle x, y \rangle$	скалярний добуток
$\otimes$	добуток Кронекера
$\|x\|$	евклідова норма
E	математичне сподівання
$U[N]$	рівномірний дискретний розподіл для чисел від 1 до N

1 Розподілені методи

1.1 Консенсус

Нехай є задача, яку розв’язують декілька обчислюваних пристроїв(надалі агенти). Жоден з них не знає всю інформацію про поставлену задачу, наприклад функцію, яку потрібно мінімізувати, або граф, в якому треба знайти ейлеровий цикл, тощо. Натомість у кожного є своя частина задачі та можливість обмінюватись інформацією з іншими агентами(не обов’язково з усіма). Задача консенсусу полягає в наступному: в результаті обробки та обміну інформації, кожен агент повинен отримати розв’язок загальної задачі, причому всі повинні отримати однаковий розв’язок.

Трохи детальніше опишемо процес обміну інформацією. Є мережа з N агентів, кожен з яких отримує своє початкове наближення x_i^0. Агенти можуть обмінюватись між собою інформацією про поточне значення x_i, що задається ненаправленим графом $G=(V,E)$, де вершини відповідають агентам $V=\{v_1, v_2, ..., v_N\}$, а ребра – можливості обміну інформацією(агенти i та j можуть нею обмінюватись якщо $(v_i,v_j)\in E$). Також позначимо через $\aleph_i$ – множину "сусідів" агента i, тобто тих агентів, з якими можливий обмін інформацією: $\aleph_i := \{v_j \in V \mid (v_i,v_j)\in E\}$.

Озн. Нехай маємо граф $G=(V,E)$ з N вершинами. Лапласіаном цього графа будемо називати квадратну матрицю L розмірності $N\times N$, елементи якої визначаються наступним чином:

$$L_{i,j} := \begin{cases} \deg(v_i) & i=j \\ -1 & i\neq j,\ (v_i,v_j)\in E \\ 0 & i\neq j,\ (v_i,v_j)\notin E \end{cases}$$

Лапласіан використовується для опису дистрибутивних у неперервному випадку, найчастіше у вигляді дифереціального рівняння. Лапласіан L зв'язного графа має наступні властивості:

1. L – невід'ємновизначена матриця
2. власному значенню $\lambda=0$ відповідає один власний вектор $v=(1,1,...,1)$

Наведемо декілька прикладів:

- консенсус зі статичним зворотнім зв'язком:

$$\dot{x}_i = \sum_{j\in\aleph_i}(x_j - x_i)$$

- консенсус з інтегральним зворотнім зв'язком:

$$\dot{x}_i = \sum_{j\in\aleph_i}(x_j - x_i) + \int_0^t \sum_{j\in\aleph_i}(x_j - x_i)$$

В обох випадках $x_i(t) \underset{t\to\infty}{\longrightarrow} \frac{1}{N}\sum_{i=1}^{N} x_i^0$

Озн. Нехай маємо граф $G = (V, E)$ з N вершинами. Ваговою матрицею суміжності(mixing matrix) цього графа будемо називати квадратну матрицю W розмірності $N \times N$, яка задовольняє наступним умовам:

1. $W_{ij} > 0$ якщо $(i, j) \in E$, інакше $W_{ij} = 0$
2. $W_{ii} > 0$
3. $W1_N = W^T 1_N = 1_N$

Матрицю W частіше використовують для опису розподілених алгоритмів у дискретному випадку.

Приклади:

- $W = I_N - \eta L,$ де η – достатньо мала додатня стала
- $$W_{ij} := \begin{cases} 1 - \sum\limits_{\substack{k=1 \\ k\neq j}}^{N} W_{kj} & i = j \\ \dfrac{1}{1 + \max\left(\deg(v_i), \deg(v_j)\right)} & i \neq j,\ (v_i, v_j) \in E \\ 0 & i \neq j,\ (v_i, v_j) \notin E \end{cases}$$

Тоді наприклад консенсус зі статичним зворотнім зв’язком матиме вигляд:

$$x_i^{(t+1)} = \sum_{j\in\aleph_i} W_{ij} x_j^{(t)}$$

1.2 Розподілений градієнтний спуск

Вище були наведені алгоритми, які по факту не розв'язують якусь глобальну задачу, а просто усереднюють значення x_i усіх агентів. Тепер же розглянемо задачу мінімізації функції методом розподіленого градієнтного спуску.

Постановка задачі: знову є мережа з N агентів, кожен з яких знає свою функцію $f_i(x): \mathbb{R}^n \to \mathbb{R}$. Граф $G = (V, E)$ визначений так само, як було описано вище. Потрібно мінімізувати суму даних функцій, так щоб всі агенти отримали однакове значення x_i.

$$\begin{aligned} &\underset{x_i \in \mathbb{R}^n}{\text{minimize}} \sum_{i=1}^{N} f_i(x_i) \\ &\text{subject to } x_1 = x_2 = \ldots = x_N = \mathbf{x}^* \end{aligned}$$

Для опису самого алгоритму введемо деякі позначення. Нехай L – лапсасіан графа G, а W – вагова матриця суміжності. Тоді алгоритм задається наступною системою [1]:

$$\begin{aligned} &\dot{x}_i = -\nabla f_i(x_i) + \sum_{j \in \aleph_i} (x_j - x_i) \\ &x_i(0) = x_{i0} \end{aligned}$$

Або в дискретному випадку:

$$x_i^{(t+1)} = \sum_{j \in \aleph_i} W_{ij} \left(x_j^{(t)} - \eta \nabla f_j \left(x_j^{(t)} \right) \right)$$

Для зручності введемо наступні позначення:

$$\begin{aligned} &x \triangleq col\{x_1, x_2, ..., x_N\} \\ &\mathbf{L} \triangleq L \otimes I_n \\ &\mathbf{W} \triangleq W \otimes I_n \\ &\nabla f(x) \triangleq col\{\nabla f_1(x_1), \nabla f_2(x_2), ..., \nabla f_N(x_N)\} \end{aligned}$$

Тоді вищенаведені системи можна подати у такому вигляді:

$$\dot{x} = -\left(\nabla f(x) + \mathbf{L}x\right)$$
$$x(0) = x_0$$

та

$$x^{(t+1)} = \mathbf{W}\left(x^{(t)} - \eta \nabla f(x^{(t)})\right)$$

1.3 Доведення збіжності(загальна ідея)

У неперервному випадку намагаються довести асимптотичну збіжність і, як правило, доведення складається з двох частин [3]:

- доведення існування сідлової точки(переважно ще й єдність)
- нормування системи та побудова функції Ляпунова

Наприклад, для вище описаного рівняння нормована система має вигляд

$$\dot{\tilde{x}} = -\left(\nabla f(\tilde{x} + x^*) + \mathbf{L}(\tilde{x} + x^*)\right)$$
$$0 = -\left(\nabla f(x^*) + \mathbf{L}(x^*)\right)$$

де x^* – сідловка точка, а функція Ляпунова

$$V(\tilde{x}) = \frac{1}{2}\left\|\tilde{x}\right\|^2$$

Так як в подальшому ми будемо використовувати дискретний випадок, то зупинимось на ньому трохи детальніше. І почнемо з уточнення постановки задачі. Практично всі статті про дискретні розподілені методи оптимізації враховують той факт, що цільова функція $f(x)$, яку потрібно мінімізувати, є по суті критерієм апроксимації по деякому набору даних $\mathcal{D}$, який випадковим чином розподіляється серед агентів, іншими словами розглядають математичне сподівання випадкової функції. Тоді постановка задачі зводиться до наступної

$$\underset{x\in\mathbb{R}^n}{\text{minimize}}\ f(x) = \frac{1}{N}\sum_{i=1}^{N}\underbrace{E_{\xi\sim\mathcal{D}_i}F_i(x,\xi)}_{:=f_i(x)}$$

де $\mathcal{D}_i$ – це розподіл ймовірності для і-го агента, який пов'язаний з отриманням доступу до загальної набору даних. Тоді ξ – це частина початкового набору даних, яка була отримана відповідно до деякого закону розподілу.

Збіжність методу в такому випадку розуміють наступним чином: говорять про ε-апроксимацію, якщо

$$\frac{1}{K}\sum_{k=0}^{K-1} E\left\|\nabla f\left(\frac{1}{N}\sum_{i=1}^{N} x_i^{(k)}\right)\right\|^2 \le \varepsilon$$

На розподіли даних найчастіше накладають наступні обмеження

- усі $\mathcal{D}_i$ однаково розподілені
- $\forall i \forall x \;\; E_{\xi\sim\mathcal{D}_i}\left\|\nabla F_i(x,\xi) - \nabla f(x)\right\|^2 \le \sigma^2$
- $\forall x \;\; E_{i\sim U[N]}\left\|\nabla f_i(x) - \nabla f(x)\right\|^2 \le \varsigma^2$

Якщо додати ще доволі стандартні обмеження як опуклість функцій та умова Ліпшиця градієнтів, для вище описаного розподіленого градієнтного спуску отримана наступна оцінка [4]:

$$\frac{1}{K}\left(\frac{1-\eta L}{2}\sum_{k=0}^{K-1} E\left\|\frac{1}{N}\sum_{i=1}^{N}\nabla f_i\left(x_i^{(k)}\right)\right\|^2 + D_1\sum_{k=0}^{K-1} E\left\|\nabla f\left(\frac{1}{N}\sum_{i=1}^{N} x_i^{(k)}\right)\right\|^2\right) \le$$

$$\frac{f(0)-f^*}{\eta K} + \frac{\eta L}{2N}\sigma^2 + \frac{\eta^2 L^2 N\sigma^2}{(1-\rho)D_2} + \frac{9\eta^2 L^2 N\varsigma^2}{\left(1-\sqrt{\rho}\right)^2 D_2}$$

де K – номер ітерації, L – константа з умови Ліпшиця, ρ – друге найбільше власне число матриці W, η – крок методу, D_1 та D_2 – константи:

$$D_1 = \frac{1}{2} - \frac{9\eta^2 L^2 N}{\left(1-\sqrt{\rho}\right)^2 D_2} \qquad D_2 = 1 - \frac{18\eta^2 L^2 N}{\left(1-\sqrt{\rho}\right)^2}$$

Якщо вибрати фіксований крок $\eta = \frac{1}{2L + \sigma\sqrt{K/N}}$, що на практиці просто неможливо, оцінка буде наступною:

$$\frac{1}{K}\sum_{k=0}^{K-1} E\left\|\nabla f\left(\frac{1}{N}\sum_{i=1}^{N} x_i^{(k)}\right)\right\|^2 \le \frac{8\left(f(0)-f^*\right)L}{K} + \frac{\left(8f(0)-8f^*+4L\right)\sigma}{\sqrt{NK}}$$

Як видно, для дискретного випадку доволі важко отримати «гарну» оцінку. Саме ж доведення по суті є набором лем(чия кількість може сягати 20-ти) [6].

Основу самого доведення складають різного роду нерівності(від обмеження норм матриць та матсподівання до нерівностей пов’язаних з Гільбертовими просторами та опуклими функціями).

Також доволі часто використовується наслідок з ергодичної теореми: для вагової матриці суміжності W існує границя $\lim_{k\to\infty} W^k = \frac{1_N^T 1_N}{N}$

2. Нейромережі

2.1 Опис та постановка задачі

Нехай є задача: потрібно апроксимувати функцію $f : \mathbb{R}^n \to \mathbb{R}$ на основі значень y_i у точках x_i. Головна ідея нейромереж полягає у представленні розв'язку як композиція багатьох «простих» функцій, тобто:

$$f = f_N \circ f_{N-1} \circ \ldots \circ f_2 \circ f_1$$

Де $f_i(x, \theta_i)$ залежать від x – значення функції f_{i-1}, та θ_i – параметра функції, який власне і потрібно підібрати. Також кожну таку функцію називають шаром нейромережі, конкретно в нашому випадку маємо N-шарову нейромережу [7].

Наведеми приклад «простої» функції та її параметрів: для лінійних функцій $f_i(x, A_i, b_i) = A_i x + b_i$ параметрами будуть матриці A_i та вектори b_i (саме по них будемо рахувати градієнти та проводити оптимізацію). Іншим прикладом може бути операція дискретної згортки для тензорів.

Оскільки композиція лінійних функцій є лінійною функцією(те саме і для згортки), то для усунення цієї проблеми використовують нелінійні функції активації g_i, які, як правило, не мають власних параметрів. Тоді робиться наступне покращення: $\ldots \circ f_{i+1} \circ f_i \circ f_{i-1} \circ \ldots \to \ldots \circ f_{i+1} \circ g_i \circ f_i \circ g_{i-1} \circ f_{i-1} \circ \ldots$

Нижче наведено загальні вимоги для функцій активації, зауважимо що залежно від задачі деякі з них можуть не виконуватись:

- нелінійність, необхідність якої пояснена вище
- ін'єктивність, яка необхідна для здатності розрізняти різні вхідні дані
- диференційованість, адже в подальшому будуть використовуватись градієнтні методи оптимізації
- швидке обчислення значення функції та її градієнта, адже градієнтні методи є ітераційними, а тому вимагають частих обчислень
- відносно велике значення норми градієнта, що необхідно для зменшення кількості ітерацій при оптимізації

Приклади функцій активації:

- ReLU $g(x) = \max(0, x)$

- Sigmoid $g(x) = \dfrac{1}{1+e^{-x}}$
- tanh $g(x) = \tanh(x)$

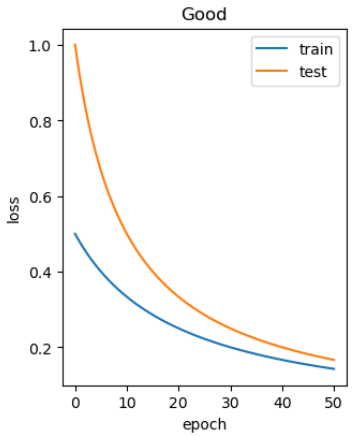
Також потрібно описати функцію втрат(loss function), яка буде показувати наскільки добре нейромережа апроксимує функцію f на основі даних $(x_i,\ y_i)$. Як правило її намагаються мінімізувати.

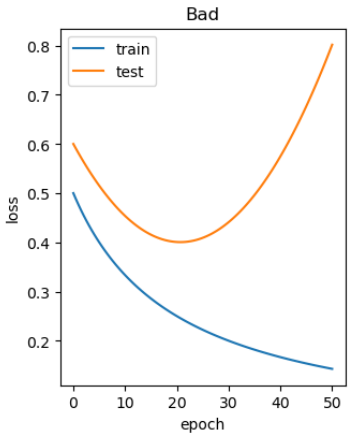
Приклад: $\ell(x,\ y) = \sum_{i=1}^{N} \left(y_i - f(x_i)\right)^2$

Тоді тренування нейромережі зведеться до задачі мінімізації функції втрат за параметрами θ_i

$$\underset{\theta}{\text{minimize}}\ \ell(x,\ y)$$

Зробимо невеличке зауваження: подібно до того, як існує безліч неперервних функцій, що проходять через задані точки, так само розв'язків вище описаної задачі є нескінченно багато [2]. Тому окрім набору даних власне для апроксимації, розглядають тестовий набір даних, по якому перевірятимуть якість апроксимації: якщо і на тренувальних, і на тестових даних апроксимація «хороша», то отриманий розв'язок приймають, якщо ж на тренувальних даних апроксимація «хороша», а на тестових даних – «погана», то розв'язок відхиляють і розглядають більш кращі методи оптимізації та моделі. Тут під «хорошим» та «поганим» розуміють поведінку функції втрат, а саме чи вона спадає, чи зростає. Нижче наведено приклад:





2.2 Пряме і зворотнє поширення

Практично всі методи тренування нейромереж є узагальненнями чи покращеннями градієнтного спуску. Тому детальніше зупинемось на обчисленні самих градієнтів.

З опису нейромережі видно, що записати саму модель в явному вигляді досить важко, тому розглянемо простіший приклад, а саме трьохшарову нейромережу:

$$f = g_3 \circ f_3 \circ g_2 \circ f_2 \circ g_1 \circ f_1$$

Нехай маємо вхідні дані $(x,\ y)$ та відповідну функцію втрат $\ell(x,\ y)$. Для зручності введемо наступні позначення, які відображатимуть порядок обчислення значення функції втрат:

$$\begin{aligned}
z_1 &= f_1(x, \theta_1) \\
a_1 &= g_1(z_1) \\
\\
z_2 &= f_2(a_1, \theta_2) \\
a_2 &= g_2(z_2) \\
\\
z_3 &= f_3(a_2, \theta_3) \\
a_3 &= g_3(z_3) \\
L &= \ell(a_3,\ y)
\end{aligned}$$

Так як мінімізація відбувається за параметрами $\theta_1, \theta_2, \theta_3$ то порахуємо відповідні градієнти за правилом композиції:

$$\frac{\partial L}{\partial \theta_1} = \frac{\partial L}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial a_1} \cdot \frac{\partial a_1}{\partial z_1} \cdot \frac{\partial z_1}{\partial \theta_1}$$

$$\frac{\partial L}{\partial \theta_2} = \frac{\partial L}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial a_2} \cdot \frac{\partial a_2}{\partial z_2} \cdot \frac{\partial z_2}{\partial \theta_2}$$

$$\frac{\partial L}{\partial \theta_3} = \frac{\partial L}{\partial a_3} \cdot \frac{\partial a_3}{\partial z_3} \cdot \frac{\partial z_3}{\partial \theta_3}$$

Кожний окремо взятий градієнт легко рахується, а саме виражається через градієнти $\nabla_x g_i, \nabla_x f_i$ та $\nabla_{\theta_i} f_i$. Для того щоб не нагромаджувати формули ми опустили точки, в яких обчислюються значення відповідних градієнтів, а тому наведемо уточнену формулу для градієнта по θ_3:

$$\frac{\partial L}{\partial \theta_3}(x) = \frac{\partial L}{\partial a_3}(a_3) \cdot \frac{\partial a_3}{\partial z_3}(z_3) \cdot \frac{\partial z_3}{\partial \theta_3}(a_2)$$

Отже, в загальному випадку обчислення градієнтів по θ_i відбувається в два етапи:

- пряме поширення – послідовно обчислюються значення z_i та a_i
- зворотнє поширення – послідовно обчислюються значення градієнтів $\frac{\partial L}{\partial a_N}(a_N)$, $\frac{\partial a_i}{\partial z_i}(z_i)$, $\frac{\partial z_i}{\partial a_{i-1}}(a_{i-1})$, $\frac{\partial z_i}{\partial \theta_i}(a_{i-1})$ і за їх допомогою обчислюються значення градієнтів $\frac{\partial L}{\partial \theta_i}(x)$

Зауваження №1: в загальному випадку використовується правило для обчислення композиції градієнтів тензорів, єдина відмінність якого полягає в наявності різноманітних добутків для тензорів(тензорний, Адамара, тощо)

Зауваження №2: при зворотньому поширенні при обчисленні градієнта $\frac{\partial L}{\partial \theta_i}(x)$ як правило одразу застосовують оптимізаційний метод для того щоби використовувати менше пам’яті обчислюваного пристрою.

2.3 Методи навчання нейромереж

Як було сказано вище, під навчанням нейромережі розуміють мінімізацію відповідної функції втрат за допомогою градієнтних ітераційних методів. Перед оглядом деяких з них потрібно дещо уточнити.

По-перше, через особливості обчислення градієнтів, як показано вище, зміна параметрів θ_i відбувається не одночасно, а поступово, за одним і тим же правилом, тому описувати методи будемо лише для одного параметра.

По-друге, зазвичай цього не вказують, але всі описані методи будуть стохастичними, тобто під градієнтом $\nabla_\theta f$ розумітимемо $\nabla_\theta f(\tilde{x})$, де $\tilde{x}$ – це невелика підмножина загальної сукупності даних(mini-batch), як правило вибрана за рівномірним розподілом. На практиці це дозволяє зменшити час виконання оптимізації, хоча ускладнює математичну постановку задачі.

По-третє, математично для застосування даних методів функція f повинна бути диференційованою, але найчастіше це не так, через використання функції активації RELU $g(x) = \max(0, x)$, яка очевидно не має похідної в точці нуль. Навіть така звична умова Ліпшиця втрачає сенс. Проте це не заважає інженерам по машинному навчанні використовувати і RELU, і градієнті методи та отримувати доволі непогані результати, математичне обґрунтування яких до кінця невідоме. Отож перейдімо до опису самих методів [7].

Стохастичний градієнтний спуск(SGD)

Формула зміни параметрів:

$$\theta^{(t+1)} = \theta^{(t)} - \eta \nabla_\theta f$$

Тут і надалі $\theta^{(t)}$ – параметр на t-ій ітерації, η – стала, яку називають кроком методу. Даний метод є найпростішим в реалізації, але має й декілька недоліків. По-перше, він знаходить локальні, а не глобальні мінімуми. По-друге, при потраплянні в деякий окіл точки мінімуму, діаметр якого значно менший за крок η, наступною ітерацією параметр θ «перескочить» точку мінімуму і опиниться за межами даного околу. Але чим менший крок η обрати, тим все менше відрізнятиметься $\theta^{(t+1)}$ від $\theta^{(t)}$, що означає повільнішу збіжність. І по-третє, у випадку багатовимірної функції f є ризик піти «неправильним» шляхом через сідлові точки.

Про перше та третє поговоримо згодом, а от для другого рішенням є використання змінного кроку $\eta^{(t)}$, який повинен прямувати до нуля при $t \to \infty$. Найпоширенішими є два:

$$\eta^{(t+1)} = \alpha \eta^{(t)} \qquad \text{та} \qquad \eta^{(t)} = \frac{\eta^{(0)}}{1 + \alpha t}$$

SGD з моментом

Тепер щодо локальних мінімумів, рішенням є використання моментів. Ідея прийшла з фізики, а точніше кінематики: якщо розглядати градієнт як «швидкість», то формула SGD є по суті рухом матеріальної точки. Пропонується додати ще й «прискорення» або точніше експоненційне згладження градієнта, яке й називають моментом. Формула зміни параметрів наступна:

$$m^{(t+1)} = \beta m^{(t)} + \nabla_\theta f$$
$$\theta^{(t+1)} = \theta^{(t)} - \eta m^{(t+1)}$$
$$m^{(0)} = 0$$

де $\beta \leq 1$ – коефіцієнт, що визначає ступінь згладження(обирають близьким до одиниці)

Таким чином навіть якщо метод досягне точки локального мінімуму, де градієнт рівний нулю, параметр θ продовжить змінюватись за рахунок ненульового моменту і, можливо, зможе потрапити в окіл точки глобального мінімуму. Тобто збіжність саме до глобального мінімуму не гарантується, але гарантується результат не гірший ніж в SGD(адже даний метод ідентичний йому при $\beta = 0$).

Метод ADAM

Тепер шодо сідлових точок. Проблема полягає у тому, що частинна похідна за однією змінною може бути близькою до нуля, поки інша є доволі великою, що сповільнює процес оптимізації. Для вирішення цієї проблеми градієнт нормують, так само використовуючи експоненційне згладження.

Також у SGD з моментами стандартною ініціалізацією є $m^{(0)} = 0$, що трохи спотворює моменти, змушуючи їх тяжіти до нуля. Тому додатково застосовують процедуру корекції. Формула зміни параметрів наступна:

$$m^{(t+1)} = \beta_1 m^{(t)} + (1 - \beta_1)\nabla_\theta f$$
$$g^{(t+1)} = \beta_2 g^{(t)} + (1 - \beta_2)\left\|\nabla_\theta f\right\|^2$$
$$\hat{m}^{(t+1)} = \frac{m^{(t+1)}}{1 - \beta_1^{t+1}}$$
$$\hat{g}^{(t+1)} = \frac{g^{(t+1)}}{1 - \beta_2^{t+1}}$$
$$\theta^{(t+1)} = \theta^{(t)} - \eta \frac{\hat{m}^{(t+1)}}{\sqrt{\hat{g}^{(t+1)} + \varepsilon}}$$

де β_1, β_2 – коефіцієнти згладження, $g^{(t)}$ – згладжений квадрат норми градієнта, $\hat{m}^{(t)}$, $\hat{g}^{(t)}$ – відповідні параметри після корекції, ε – мала стала, аби запобігти діленню на занадто мале число.

Як і раніше збіжність до глобального мінімуму не гарантується, лише до локального, але на практиці даний метод є одним з найкращих для тренування нейромереж, тому й широко використовується.

2.4 Розподілені методи тренування

Тепер перейдімо до розподілених методів навчання. Як було сказано в першому розділі, маємо мережу з N обчислюваних пристроїв. Кожен з них має доступ лише до частини даних, але при цьому потрібно натренувати нейромережу на повному наборі даних.

Конкретизуємо задачу: у кожного агента є своя нейромережа зі своїми параметрами θ_i, які належать одному простору $\mathbb{R}^d$, що можливо лише якщо всі нейромережі мають однакову структуру. Далі є розбиття повного набору даних $(x,\ y)$ згідно деяких законів розподілу $\mathcal{D}_i$ на підмножини $\xi_i := (x_i,\ y_i)$, що в свою чергу тягне представлення загальної функції втрат $\ell(x,\ y)$ як суму всіх $\ell(x_i,\ y_i)$, можливо з певними ваговими множниками, якщо підмножини містять різну кількість елементів. Остаточно потрібно знаючи $F_i(\theta_i, \xi_i) := \ell(x_i,\ y_i)$ мінімізувати $\ell(x,\ y)$ за параметрами θ_i з дотриманням консенсусу. Стосовно мережі агентів, то вона описується ваговою матрицею суміжності W. Математично все зводиться до наступного:

$$\underset{\theta\in\mathbb{R}^d}{\text{minimize}}\ f(\theta) = \frac{1}{N}\sum_{i=1}^{N} \underbrace{E_{\xi_i\sim\mathcal{D}_i} F_i(\theta,\xi_i)}_{:=f_i(x)}$$

Після постановки задачі можемо перейти до розгляду відповідних розподілених методів оптимізації, але спочатку наведемо деякі загальні припущення, які є достатніми для збіжності описуваних методів.

- усі $\mathcal{D}_i$ однаково розподілені
- Функція $f(\theta)$ обмежена знизу якимось значенням f^*
- $\forall x, y \quad \|\nabla f_i(x) - \nabla f_i(y)\| \le L\|x - y\|$
- $\forall i \forall \theta \ E_{\xi\sim\mathcal{D}_i} \|\nabla F_i(\theta,\xi) - \nabla f(\theta)\|^2 \le \sigma^2$
- $\forall \theta \ E_{i\sim U[N]} \|\nabla f_i(\theta) - \nabla f(\theta)\|^2 \le \varsigma^2$

Останні дві умови є по суті умовами обмеженої дисперсії як по усім можливим параметрам так і по усіх агентах мережі. Іншими словами, для їхнього виконання достатньо, щоб усі дані були рівномірно розподіленими по агентах в однаковій кількості.

Розподілений стохастичний градієнтний спуск(D-SGD)

$$\theta_i^{(t+1)} = \sum_{j \in \aleph_i} W_{ij} \left(\theta_j^{(t)} - \eta \nabla F_j \left(\theta_j^{(t)}, \xi_j^{(t)} \right) \right)$$

Відрізняється від звичайного SGD лише операцією усереднення з ваговими коефіцієнтами, що є елементами матриці W. Як було сказано вище, швидкість збіжності даного методу є $O\left(\frac{1}{K} + \frac{1}{\sqrt{NK}}\right)$, де K – кількість ітерацій. Дану оцінку намагаються покращити з двох причин: по-перше, коли кількість ітерацій значно перевищить кількість агентів, то вона стане $O\left(\frac{1}{\sqrt{NK}}\right)$, що повільніше звичайного SGD, хоча кількість операцій зросла в N разів; по-друге, коли набір даних розбивають на N частин, які потім одночасно обробляються агентами, очікується прискорення теж в N разів, але з оцінки видно, що лише в $\sqrt{N}$. До того ж на практиці дана оцінка досягається лише при досить великій кількості ітерацій, на що не завжди є достатньо часу [4].

До того ж є й ті самі проблеми, що й в SGD, а саме застрягання в локальних мінімумах та сідлові точки.

D-SGD з моментом

$$m_i^{(t+1)} = \beta m_i^{(t)} + \nabla f_i \left(\theta_i^{(t)} \right)$$
$$\theta_i^{(t+1)} = \sum_{j \in \aleph_i} W_{ij} \left(\theta_j^{(t)} - \eta m_j^{(t+1)} \right)$$
$$m_i^{(0)} = \mathbf{0}$$

Що ж, даний метод теж відрізняється від класичного аналога лише операцією усереднення. Оцінка збіжності $O\left(\frac{1}{\sqrt{NK}}\right)$, хоча показує набагато кращий результат ніж D-SGD за рахунок сталих коефіцієнтів, що сховані в $O(.)$ [6].

D-SGD з відстеженням моментів

Нагадаємо, що операція усереднення разом з властивостями вагової матриці суміжності потрібні для досягнення консенсусу. Проблема попереднього методу полягає в тому, що моменти ніяк не усереднюються, а тому можуть дуже сильно відрізнятися, що в свою чергу ускладнює досягнення консенсусу для параметрів. Рішення цього є введення доданка c_i, який уточнює момент m_i до усередненого $\frac{1}{N}\sum_{j=1}^{N} m_j$, що повинно пришвидшити досягнення консенсусу параметрів. Оцінка збіжності $O\left(\frac{1}{\sqrt{NK}}\right)$ [6].

$$m_i^{(t+1)} = \beta m_i^{(t)} + \nabla f_i\left(\theta_i^{(t)}\right)$$
$$\theta_i^{(t+1)} = \sum_{j\in\aleph_i} W_{ij}\theta_j^{(t)} - \eta\left(m_i^{(t+1)} - c_i^{(t)}\right)$$
$$c_i^{(t+1)} = \sum_{j\in\aleph_i} W_{ij}\left(c_j^{(t)} - m_j^{(t+1)}\right) + m_i^{(t+1)}$$
$$m_i^{(0)} = c_i^{(0)} = \frac{1}{1-\beta}\nabla f_i\left(\theta_i^{(0)}\right) - \frac{1}{N}\sum_{j\in\aleph_i} W_{ij}\nabla f_j\left(\theta_j^{(0)}\right)$$

Не дивлячись на те, що існує розподілений метод ADAM, але поки він розроблений для умовної мінімізації по компакту $X \subset \mathbb{R}^d$ а не по всьому просторі. До того ж оцінка збіжності така ж сама, як і в попередніх методах $O\left(\frac{1}{\sqrt{NK}}\right)$, через що було вирішено не розглядати його в даній роботі [5].

3. Експериментальна частина

3.1 постановка задачі

Розглянемо задачу класифікації зображень, а саме маючи якесь зображення віднести його до одного з наперед визначених класів. Зображення задаються трьома матрицями із значеннями елементів від 0 до 255, що відображають інтенсивність пікселів по трьом кольорах: червоному, зеленому та синьому. Найчастіше ці матриці об'єднують в трьохвимірний тензор. Про те як задаватимуться класи поговоримо пізніше.

Для навчання будемо використовувати дані з cifar-10, що містять 60 тис. зображень 32x32, які належать десятьом класам: *літак, автомобіль, птах, кіт, олень, пес, жаба, кінь, корабель та вантажівка.*

Тепер щодо того, що повинна видавати модель. Очевидно з властивостей нейромереж, що на виході буде якийсь тензор з дійсними значеннями: число, вектор, матриця тощо. Тому пропонують використовувати числовий вектор, з десятьма значеннями, який буде по суті розподілом ймовірностей того, що зображення належить певному класу. Відповідно «справжня» інформація про клас так само буде подана у вигляді розподілу ймовірності, а саме вектором з дев’ятьма нулями та однією одиницею, що відповідатиме класу зображення.

Отже, остаточно маємо задачу:

Позначимо через $Y := \left\{ u \in \mathbb{R}^{10} \mid u_i \geq 0 \wedge \sum_{i=1}^{10} u_i = 1 \right\}$ множину описаних вище розподілів та через X множину усіх зображень 32x32x3. Тоді маючи відповідний набір даних $(x_i,\ y_i)$ потрібно апроксимувати функцію $f : X \to Y$

Як функцію втрат візьмемо функцію перехресної ентропії(яку наведено для одного прикладу даних, відповідна функція по всьому наборі береться як середнє арифметичне):

$$\ell(x_i,\ y_i) = -\langle y_i, \ln(f(x_i)) \rangle$$

Розглядати будемо розподілений варіант цієї задачі, тобто доступу до повного набору даних немає, натомість є мережа агентів, кожен з яких має доступ лише до частини даних. Як і раніше, мережа задається графом G з відповідною ваговою матрицею суміжності W.

3.2 Результати

Опис моделі

Для проведення експериментів будемо використовувати мережу з 5-ма агентами, вагову матрицю суміжності визначимо наступним чином:

$$W_{ij} := \begin{cases} 1 - \sum\limits_{\substack{k=1 \\ k \neq j}}^{N} W_{kj} & i = j \\ \dfrac{1}{1 + \max\left(\deg(v_i), \deg(v_j)\right)} & i \neq j,\ \left(v_i, v_j\right) \in E \\ 0 & i \neq j,\ \left(v_i, v_j\right) \notin E \end{cases}$$

Кожен з агентів має власну згорткову нейромережу з наступною архітектурою:

- Conv2d(32 фільтра 3x3, relu)
- BatchNormalization
- Pool(2x2, max)
- Conv2d (64 фільтра 3x3, relu)
- Pool (2x2, max)
- Conv2d (64 фільтра 3x3, relu)
- BatchNormalization
- Flatten
- Dropout(50%)
- Dense(64 нейрона, relu)
- Dense(10 нейронів, sofmax)

Усього в даній нейромережі 122 974 числових параметрів. Якщо не сказано іншого, то тренувальні дані розподіляються між агентами рівномірно.

Під час кожного експерименту використовуватимемо 3 розподілених метода: розподілений стохастичний градієнтний спуск, DSGD з моментом та DSGD з відстеженням моментів. До кожного з них виводитимемо значення функції втрат для тренувального та тестового наборів даних на кожній ітерації, всього ітерацій буде по 50. Також для перевірки виконання консенсусу виводитимемо максимальну норму різниці параметрів моделей різних агентів(для простоти норма рівномірна), надалі дану велечину

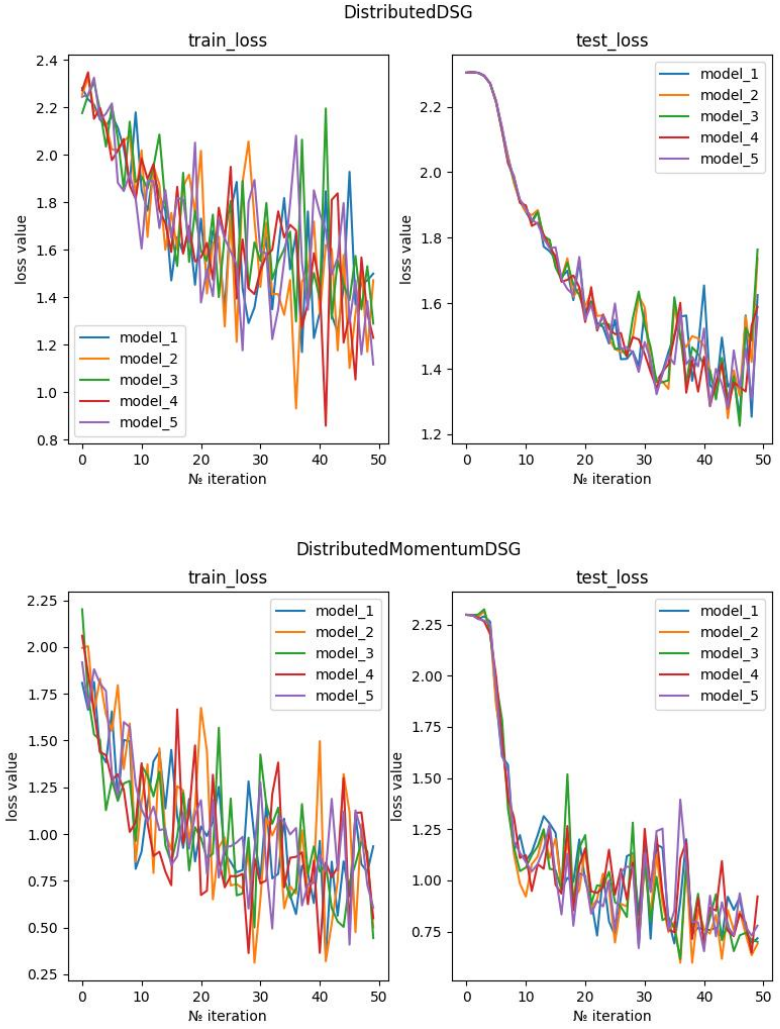
називатимемо параметричним радіусом. Очевидно, що для виконання консенсусу даний радіус має прямувати до нуля.

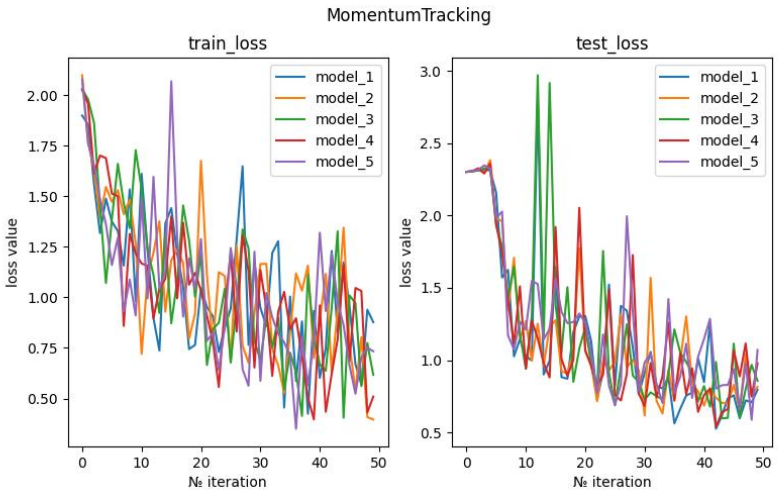
Оскільки розглядається задача класифікації, то також виводитимемо остаточну точність класифікації для всіх п’ятьох моделей, тобто частку зображень, які нейромережа віднесла до правильного класу.

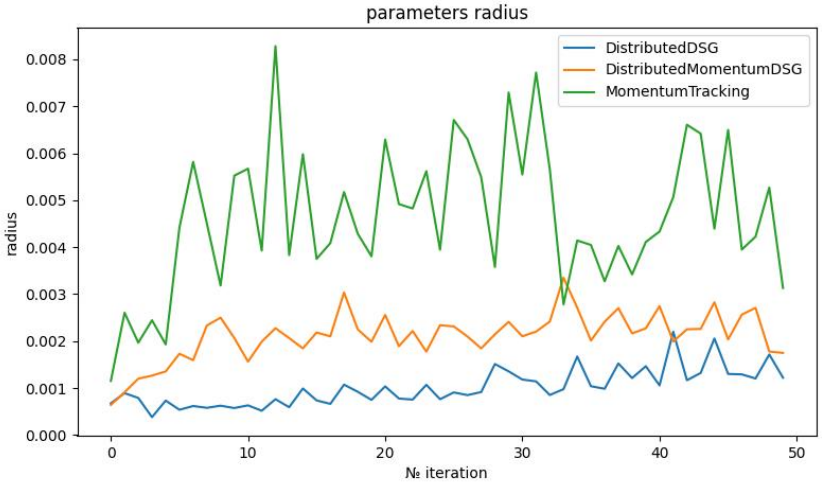
Отже тепер можна перейти до результатів експериментів.

Загальна поведінка

Для початку розглянемо загальну поведінку розподілених оптимізаційних методів. Візьмемо найбільш розповсюджену мережу агентів – кільце (див. додаток).







	DSGD	DSGD з моментом	DSGD з відстеженням моментів
модель 1	45.15%	66.80%	69.10%
модель 2	43.83%	67.05%	70.02%
модель 3	43.51%	67.28%	70.46%
модель 4	44.72%	67.66%	68.97%
модель 5	45.94%	67.78%	68.47%

Отже, що видно: судячи з поведінки функції втрат для тренувальних та тестових даних тренування нейромережі проходить успішно. Враховуючи, що точність при випадковій класифікації зображень з 10-ма класами складає 10%, то отримано доволі непогані результати. Відносно точності методи з моментом виявились кращими за простий градієнтний спуск, проте стосовно консенсусу він виявився найкращим. Можливо поясненням цьому є менша кількість змінних у розподіленому градієнтному спускові, що пришвидшує настання консенсусу. Зауважимо, що параметричні радіуси зафіксувались на

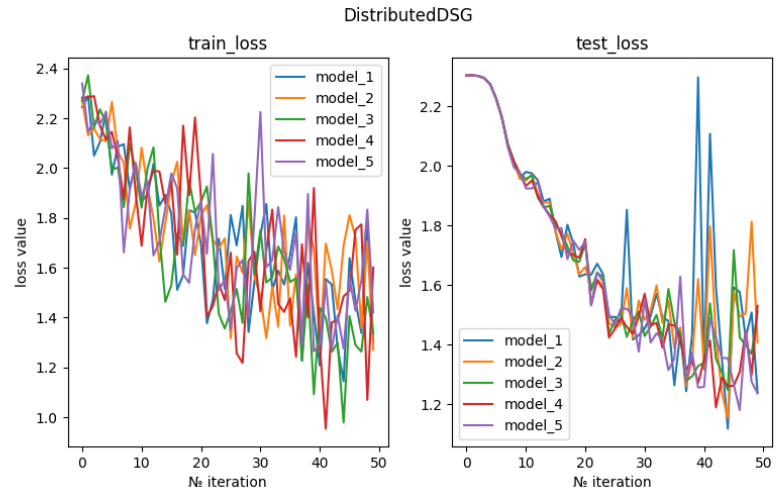
певному ненульовому рівні, але його можна вважати допустимим оскільки різниця в точності для різних моделей не перевищує 2%, що часто не є критичним.

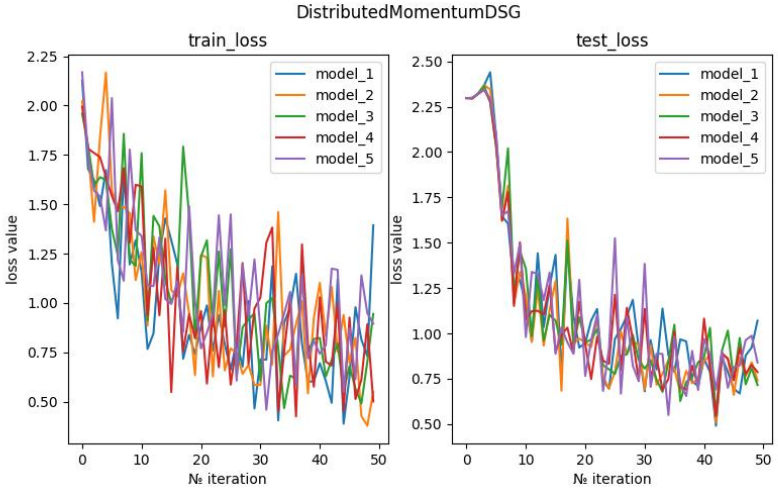
Також варто зауважити, що не дивлячись на те, що найвища точність вийшла в DSGD з відстеженням моментів, по-перше, вона не дуже сильно відрізняється від точності DSGD з моментом, а по-друге, даний метод має найвищий параметричний радіус та й поведінка функції втрат на тестових даних гірше ніж в інших методів, адже наявні доволі помітні піки. Власне це узгоджується з результатами авторів статті, які наголошували, що DSGD з відстеженням моментів буде кращим за DSGD з моментам якщо кількість агентів перевищуватиме кількість класів при класифікації, а в нас агентів 5, а класів 10 [6].

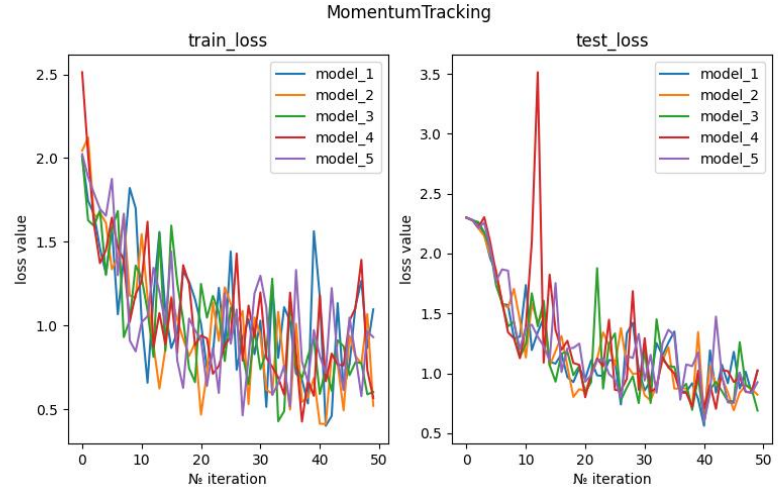
Мережа великого діаметру

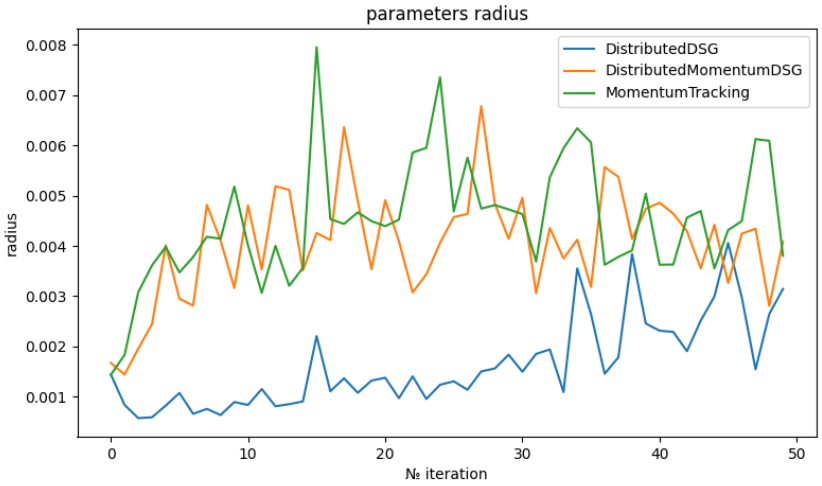
Тепер розглянемо як структура мережі агентів впливає на результат класифікації. Зауважимо, що кожне ребро в графа G відповідає можливості обміну інформації між агентами. В реальному житті це відповідає свого роду кабелю між двома обчислюваними машинами. Цей кабель має бути доволі якісний, можливо навіть доволі довгий, а тому дорогим. Через що постає питання: чи сильно впливає наявність ребер в графі на результат? Інтуїтивно чим менше ребер, тим більші ланцюги між окремими вузлами графа, через що градієнти в процесі усереднення частіше множаться на вагові множники(які менші одиниці), а тому стають близькими до нуля, не впливаючи на навчання. Іншими словами є ризик «затухання» градієнтів між агентами з великою відстанню один від одного.

Логічно, що граф має бути зв’язним, а тому розглянемо граф з найменшою можливою кількістю ребер – лінійний або простий ланцюг. Звернемо увагу на агентів, що розміщенні в кінцях ланцюга(model1 та model5).


DistributedDSG
train_loss
test_loss
model_1
model_2
model_3
model_4
model_5
loss value
№ iteration


DistributedMomentumDSG
train_loss
test_loss
model_1
model_2
model_3
model_4
model_5
loss value
№ iteration


MomentumTracking
train_loss
test_loss
model_1
model_2
model_3
model_4
model_5
loss value
№ iteration



	DSGD	DSGD з моментом	DSGD з відстеженням моментів
модель 1	49.86%	60.86%	70.42%
модель 2	50.85%	65.89%	68.96%
модель 3	50.55%	68.01%	69.88%
модель 4	50.08%	68.54%	69.99%
модель 5	50.46%	68.75%	68.68%

Як бачимо результати практично не відрізняються від попереднього експерименту, де було кільце. Більше того, практичної різниці між агентами на кінцях(model1 та model5) та іншими агентами немає, звідки випливає що достатньо лише зв’язності графа, коли діаметр графа на впливає на кінцевий результат. Також параметричний радіус зафіксувався на певному рівні.

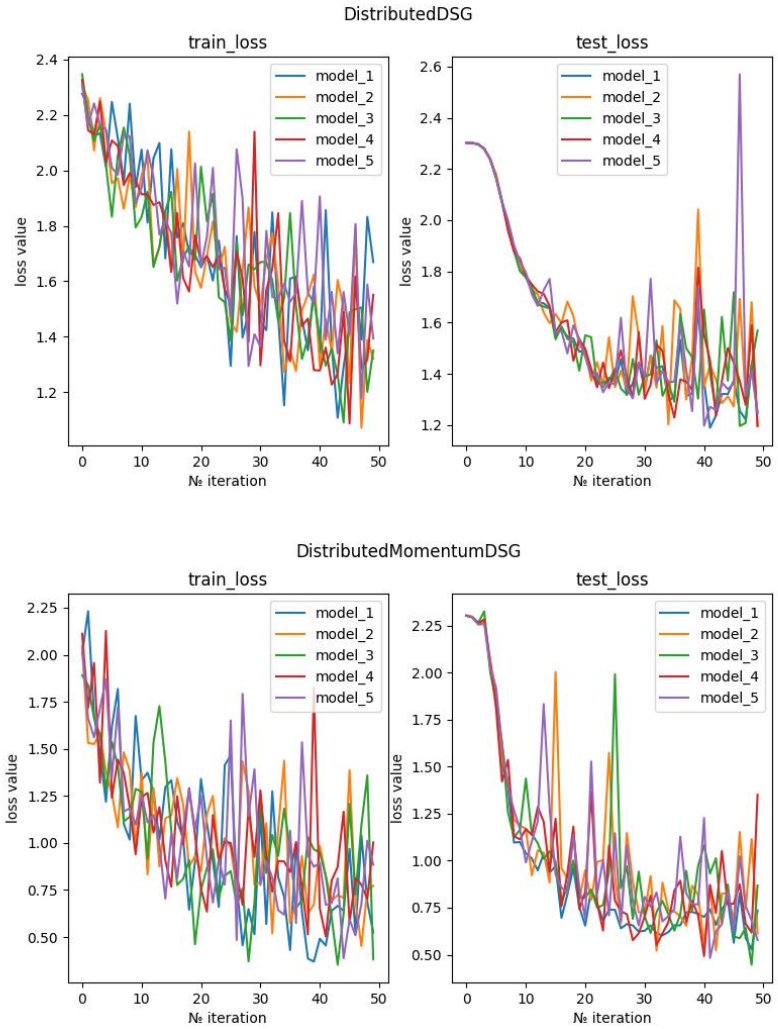
Єдине у випадку DSGD з моментом точність моделі 1(60.86%) значно відрізняється від решти, де точність близько 68%. Якщо поглянути на графік функції втрат для тестових даних, то побачимо наявність періодичних піків та черговий пік для моделі 1 на останній ітерації, чим пояснюється низька точність. Рішенням буде просто провести більшу кількість ітерацій.

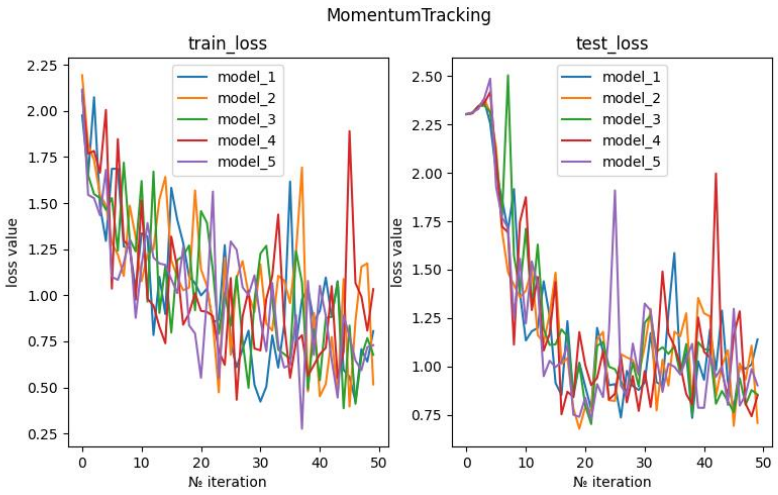
Радіальна мережа

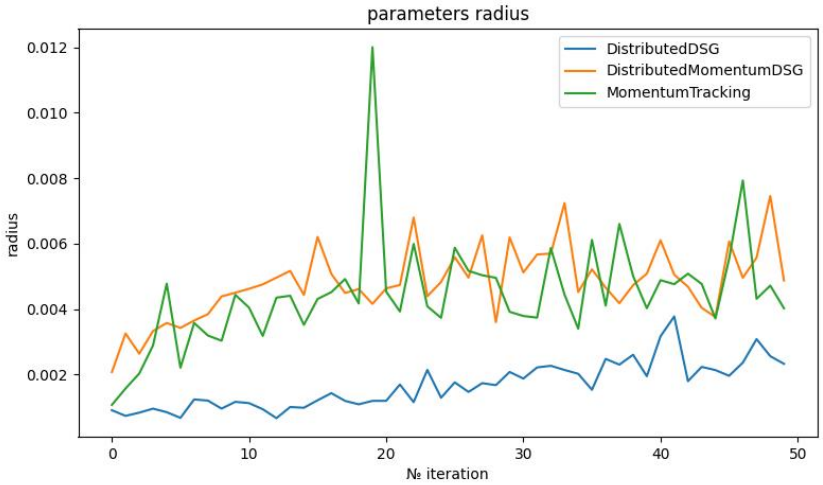
Як було показано вище, можна зекономити ресурси на побудові мережі агентів, зменшивши кількість ребер. Але ще дешевше і логічніше використовувати вже існуючі мережі. Проте в них є одна проблема: частіше

за все вони радіальні. Тобто є певний центральний вузол який поєднаний з іншими «дочірніми» агентами, які в свою чергу між собою не поєднані. Прикладом може бути центральний офіс компанії то дочірні філії, або мережі між столицею держави та регіональними центрами. Відповідно виникає питання: а чи будуть всі агенти в рівних умовах по доступу до даних? Іншими словами чи матиме перевагу центральний агент над іншими в силу структури даної мережі, адже відомим є порядок апроксимації, але не сталий коефіцієнт, яким нехтують при прямуванні кількості ітерацій до нескінченності, але у випадку скінченної кількості він може вплинути на результат.

Оберемо граф «зірка» – один агент(model1) з'єднаний з усіма іншими агентами.







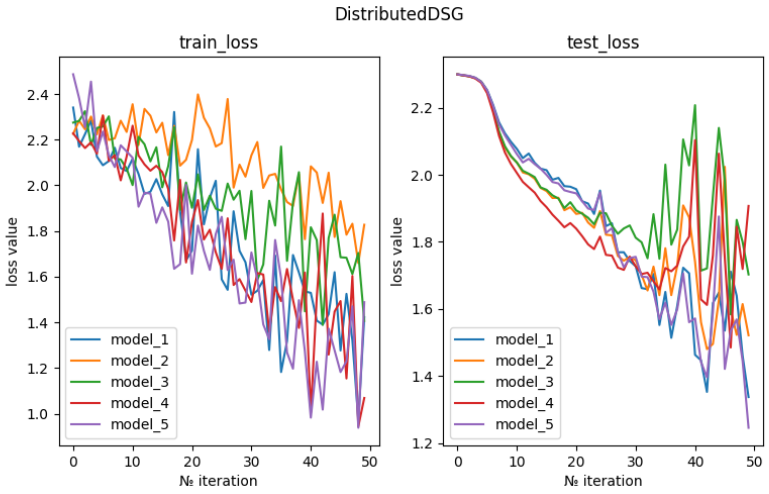
	DSGD	DSGD з моментом	DSGD з відстеженням моментів
модель 1	51.17%	71.79%	67.69%
модель 2	49.02%	70.01%	68.11%
модель 3	46.57%	69.08%	68.28%
модель 4	47.61%	69.83%	68.30%
модель 5	48.40%	69.91%	67.92%

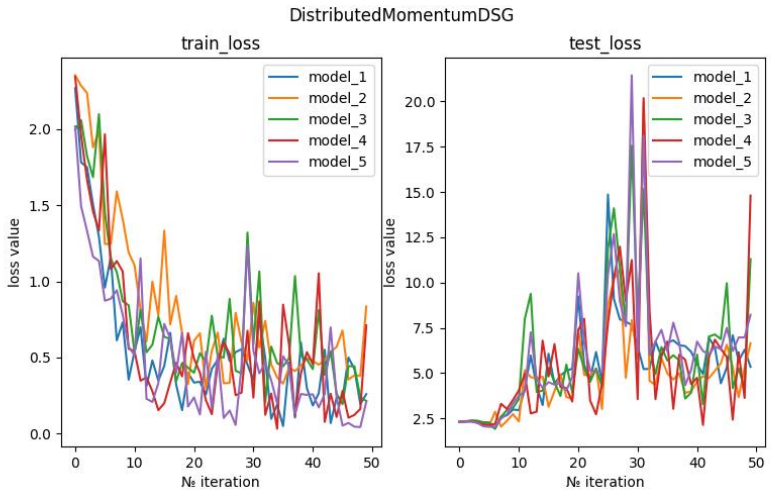
Як видно з результатів, різниця в точності більш помітна в методі DSGD і менш помітна в методах з моментами. У випадку з DSGD з відстеженням моментів центральний агент навіть отримав не найвищу точність, через що можна гарантувати рівність усіх агентів під час розподіленого навчання.

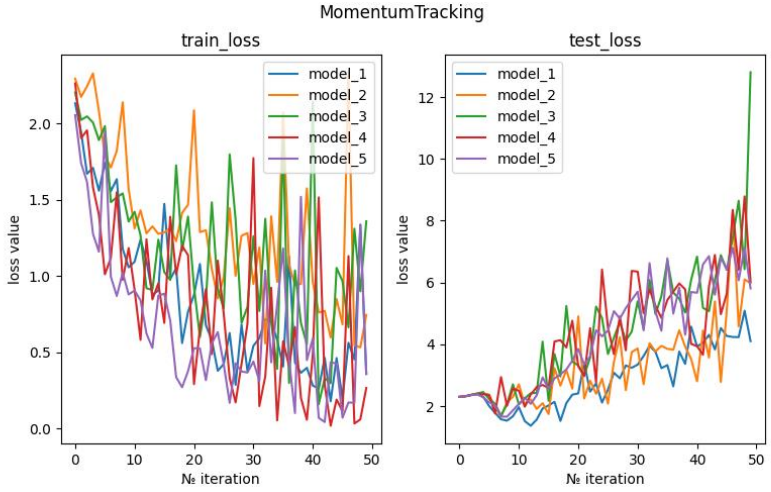
Нерівномірно розподілені дані

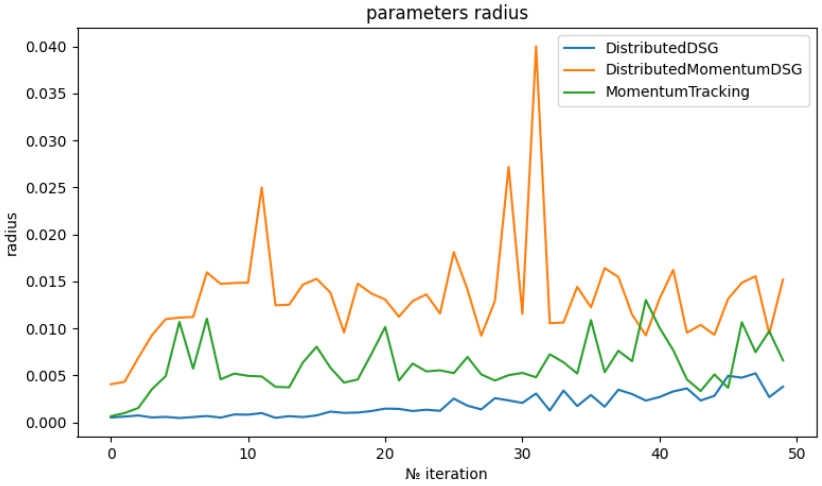
Тепер перевіримо як розподіл даних по агентах впливає на навчання нейромережі. В попередніх експериментах дані були розподілені рівномірно, себто були однорідними, серед усіх агентів. Проте на практиці це може бути не так, наприклад для навчання якісного автопілота потрібно дуже багато зображень доріг, знаків, тротуарів тощо. Якщо агенти незалежно збирали ці дані самостійно в різних містах чи навіть країнах, то зображення відрізнятимуться через різний стиль архітектури та підхід до містобудування. Витрачати час аби поділитись даними з усіма іншими агентами, враховуючи, що даних багато, ніхто не хоче. Тому виникає питання: чи впливає неоднорідність даних на процес навчання? Зауважимо, що однорідність даних є однією з умов збіжності для розподілених методів(усі $\mathcal{D}_i$ однаково розподілені), проте в нашій нейромережі функцією активації є ReLU $g(x) = \max(0, x)$, для якої не виконується інша умова – гладкість. Тим більше що згадані вище умови є достатніми а не необхідними, тому поставлене питання має сенс.

Отже, мережа – кільце. Кожен агент має доступ до даних лише певного класу: перший (літак, автомобіль), другий (птах, кіт), третій (олень, пес), четвертий (жаба, кінь), п’ятий (корабель, вантажівка). Перевірку точності робитимемо як на усіх клас, так і для «знайомих» класів.




DistributedMomentumDSG
train_loss
test_loss
model_1
model_2
model_3
model_4
model_5
loss value
№ iteration


MomentumTracking
train_loss
test_loss
model_1
model_2
model_3
model_4
model_5
loss value
№ iteration


parameters radius
DistributedDSG
DistributedMomentumDSG
MomentumTracking
radius
№ iteration

Точність на тестових даних, усі класи

	DSGD	DSGD з моментом	DSGD з відстеженням моментів
модель 1	29.83%	12.68%	23.08%
модель 2	33.35%	17.12%	30.47%
модель 3	33.06%	17.72%	28.95%
модель 4	33.82%	16.79%	29.75%
модель 5	35.48%	15.13%	30.80%

Точність на тестових даних, «знайомі» класи

	DSGD	DSGD з моментом	DSGD з відстеженням моментів
модель 1	34.32%	19.78%	34.77%
модель 2	31.55%	19.63%	33.48%
модель 3	33.69%	18.57%	30.75%
модель 4	35.31%	16.18%	32.29%
модель 5	36.30%	14.77%	32.77%

По-перше, поведінка функції втрат на тестових даних вказує на неточну апроксимацію, хоча на тренувальних даних вона спадає, тобто маємо явище «перенавчання». По-друге, точність на усіх класах і на «знайомих» є однаково низькою, іншими словами ефективніше було кожному агенту окремо натренувати свою власну модель, аніж використовувати розподілене навчання. По-третє, значно виріс параметричний радіус, що погіршує консенсус.

Висновок

У даній роботі було успішно змодельовано розподілене навчання нейромережі наступними методами: DSGD, DSGD з моментом та DSGD з відстеженням моментів. Також було досліджено як архітектура мережі обчислюваних пристроїв та розподіл даних по них впливає на кінцевий результат.

Найкращим розподіленим методом виявились DSGD з моментом та DSGD з відстеженням моментів залежно від задачі.

Архітектура мережі обчислюваних пристроїв, а саме великий діаметр графа та радіальна архітектура, не впливає на кінцевий результат і показує доволі непоганий результат.

Дослідження показало, що розподіл даних впливає на кінцевий результат навчання. Дані мають бути однорідними у всіх агентах мережі, інакше по-перше, набагато зменшиться точність моделі і по-друге, нейромережа буде «перенавчатись» – з кожною ітерацією точність на тестових даних буде зменшуватись.

Отримані результати свідчать про ефективність розподіленого методів навчання нейромереж, адже для нього можна використовувати мережі обчислюваних пристроїв різної архітектури, яку можна легко масштабувати, доєднавши додаткові пристрої, або навпаки від’єднавши пошкоджені пристрої.

Основних проблем пов’язаних з розподіленого навчання є декілька: необхідна однорідність даних по усіх пристроях в мережі, теоретичне прискорення навчання є повільнішим за лінійним, що не дозволяє використовувати занадто великі мережі і також ще не для всіх класичних методів навчання нейромереж розроблені розподілені аналоги.

Усе вище наведене свідчить про перспективність розробки та дослідження розподілених методів навчання нейромереж.

Бібліографія

1. Youbang Sun and Shahin Shahrampour «Distributed Mirror Descent with Integral Feedback: Asymptotic Convergence Analysis of Continuous-time Dynamics» – arXiv:2009.06747v1, 2020

2. Chaoyue Liu, Libin Zhu and Mikhail Belkin «Loss landscapes and optimization in over-parameterized non-linear systems and neural networks» – arXiv:2003.00307v2, 2021

3. Martin Andreasson, Dimos V. Dimarogonas, Henrik Sandberg and Karl H. Johansson «Distributed Control of Networked Dynamical Systems: Static Feedback, Integral Action and Consensus» – arXiv:1310.8620v2, 2014

4. Xiangru Lian, Ce Zhang, Huan Zhang, Cho-Jui Hsieh, Wei Zhang, Ji Liu «Can Decentralized Algorithms Outperform Centralized Algorithms? A Case Study for Decentralized Parallel Stochastic Gradient Descent» – arXiv:1705.09056, 2017

5. Parvin Nazari, Davoud Ataee Tarzanagh, George Michailidis «DADAM: A consensus-based distributed adaptive gradient method for online optimization» – arXiv:1901.09109, 2019

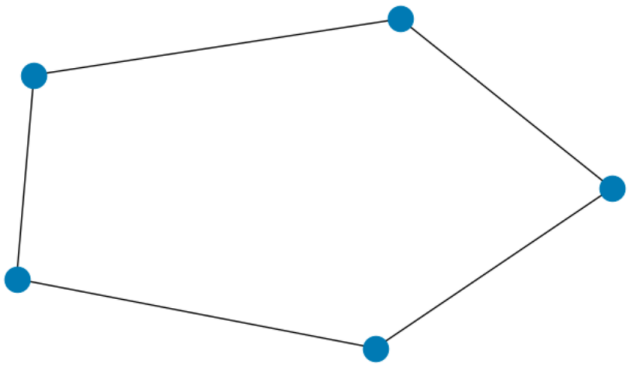
6. Yuki Takezawa, Han Bao, Kenta Niwa3, Ryoma Sato, Makoto Yamada «Momentum tracking: momentum acceleration for decentralized deep learning on heterogeneous data» – arXiv:2209.15505, 2022

7. Francois Chollet «Deep learning with Python» – New York, NY: Manning Publications, 2017

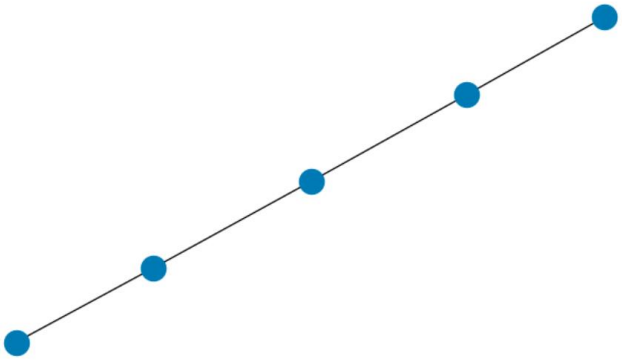
Додаток

Основні класи графів

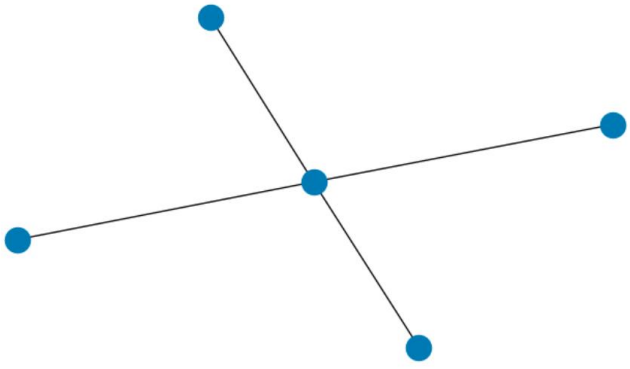
Кільце



Ланцюг



Зірка



Посилання на код:
https://colab.research.google.com/drive/1jvoSlwqN_86B6WIwhMTnxuHv-6c2CIpl?usp=share_link

Відгук на кваліфікаційну роботу бакалавра на тему:
«Дослідження застосування розподілених методів для навчання нейромереж»
студента 4-го курсу факультету комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Галиша Антона Володимировича

У сучасному світі, де обробка та аналіз великих обсягів даних стають все більш важливими, нейромережі виявились потужним інструментом для розв'язання складних завдань у багатьох сферах, таких як медичне діагностування, аналіз зображень, обробка тексту тощо. Однак, навчання нейромереж залишається складною задачею, оскільки воно вимагає великої кількості обчислювальних ресурсів та часу.

Розподілені методи стали об'єктом активних досліджень, оскільки вони дозволяють ефективно використовувати обчислювальні ресурси та прискорювати процес навчання нейромереж, шляхом використання мережі обчислюваних пристроїв.

В дипломній роботі студент Галиш А. В. зробив теоретичний огляд розподілених методів для навчання нейромереж. На прикладі декількох мереж було досліджено поведінку таких методів, зокрема швидкість збіжності та точність для задачі класифікації. Також було вивчено поведінку розподілених методів при неоднорідності розподілу даних в мережі.

Під час роботи над дипломом студент продемонстрував здатність самостійно проводити дослідження, працювати з математичною теорією та застосовувати теоретичний матеріал в практичних задачах. Вважаю, що кваліфікаційна робота студента Галиша Антона Володимировича відповідає всім вимогам, які висуваються до бакалаврських робіт, і заслуговує на оцінку «відмінно», а її автор заслуговує на присвоєння кваліфікації бакалавра.

Асистент кафедри обчислювальної математики
Факультету комп'ютерних наук та кібернетики
Київського національного університету
імені Тараса Шевченка

Сергій ДЕНИСОВ

Рецензія
на кваліфікаційну роботу бакалавра на тему:
«Дослідження застосування розподілених методів
для навчання нейромереж»
студента 4-го курсу факультету комп'ютерних наук та кібернетики
Київського національного університету імені Тараса Шевченка
Галиша Антона Володимировича

Сьогодні чи не кожен аспект життя людини залежить від аналізу даних. І чим більше цих даних, тим точнішими можна створити відповідні моделі, але тим довше триватиме сам аналіз. У даній роботі студент Галиш Антон досліджує особливості використання розподілених методів аналізу даних, а саме проводить дослідження з використання нейромереж.

Задача зводиться до побудови мережі комп'ютерів, між якими розподіляються дані для подальшого аналізу. У першій частині роботи автор проводить теоретичне дослідження кількох розподілених алгоритмів для навчання нейромереж, а в другій — демонструє та порівнює їх поведінку, проводячи обчислювальні експерименти на різних мережах пристроїв та демонструючи результати досліджень за допомогою таблиць і графіків збіжності.

За змістом роботи можна зробити деякі зауваження, що не знижують загальну позитивну оцінку роботи. Так, не достатньо описано усі використані шари нейромережі з практичної частини. Також хотілося б побачити дослідження поведінки алгоритмів для супутніх задач.

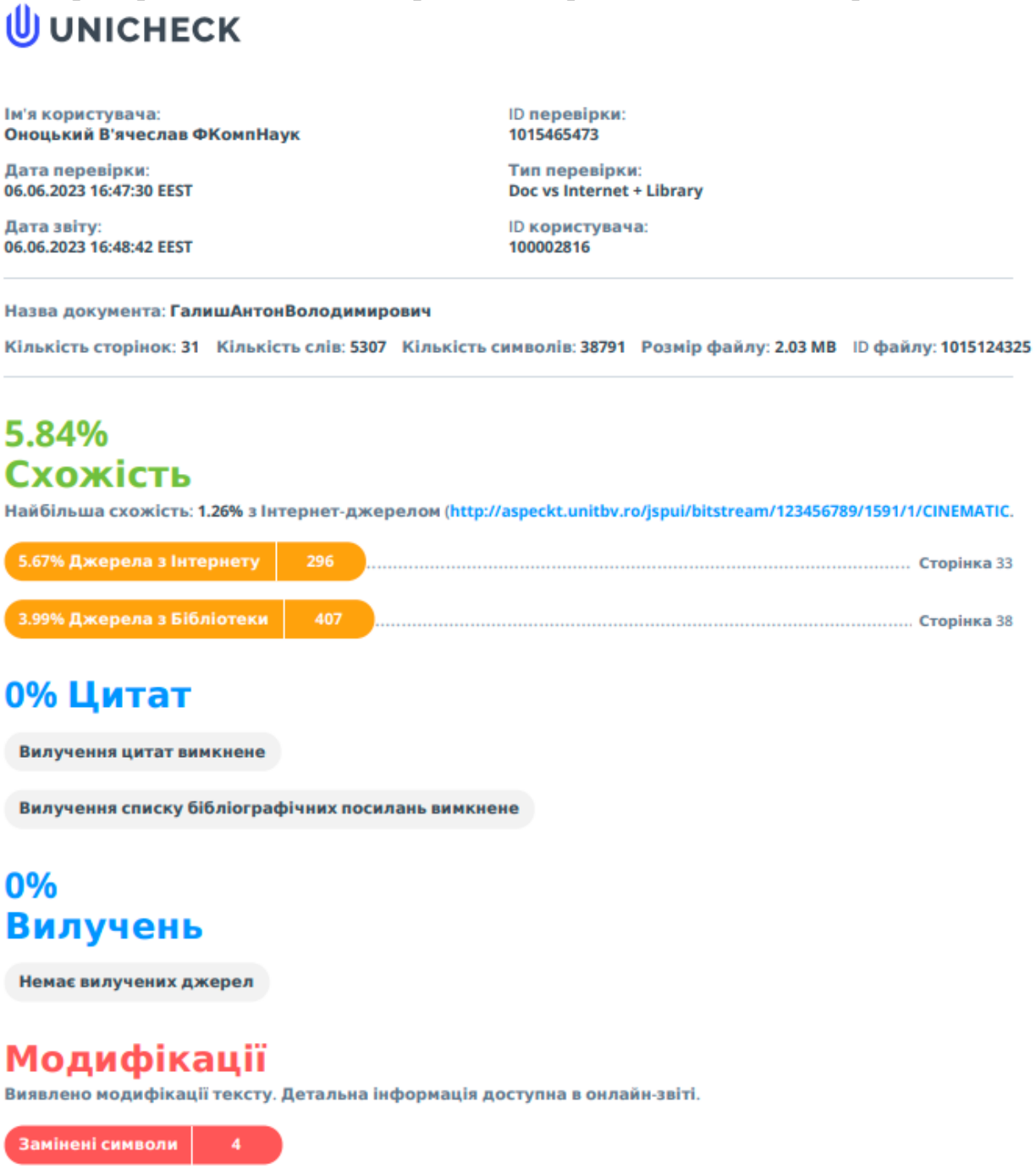
Вважаю, що кваліфікаційна робота студента Галиша Антона Володимировича відповідає всім вимогам, які висуваються до бакалаврських робіт, і заслуговує на оцінку «відмінно», а її автор заслуговує на присвоєння кваліфікації бакалавра.

Рецензент:
доктор фізико-математичних наук,
професор Дмитро Номіровський

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

СИСТЕМА ЗАПОБІГАННЯ ТА ВИЯВЛЕННЯ АКАДЕМІЧНОГО ПЛАГІАТУ

Довідка про оригінальність кваліфікаційної роботи за освітнім рівнем бакалавра



Експертна оцінка роботи науковим керівником:

Робота виконана самостійно та не містить відомостей без посилання на джерела. Найбільш схоже джерело – робота румунських авторів «Кінематика і динаміка. Збірник задач» з використаними математичними символами.

Науковий керівник: _____ Денисов С. В.

Оператор: _____ Оноцький В. В.