УДК 519.85:004.85

ПОРІВНЯННЯ КЛАСИЧНОГО ТА СТОХАСТИЧНОГО ГРАДІЄНТНОГО СПУСКУ В ЗАДАЧАХ ОПТИМІЗАЦІЇ

Єрьоменко М.І.
e-mail: maksym.ieromenko@nure.ua
Науковий керівник – канд. пед. наук, доц. Ситникова Ю.В.
Харківський національний університет радіоелектроніки,
кафедра прикладної математики
м. Харків, Україна

The relevance of this study lies in the need to improve the efficiency of training intelligent systems in Big Data environments. The object of research is the iterative processes of minimising the loss function in machine learning tasks. The paper provides a comparative analysis of Batch Gradient Descent and Stochastic Gradient Descent (SGD) algorithms, focusing on convergence speed, stability, and computational complexity. It is shown that while the classical approach provides high precision for small datasets, the stochastic method is economically feasible for large-scale systems due to its $O(1)$ iteration complexity and its ability to escape local minima under noise. The results of the study can be used to optimise the infrastructure of modern neural network training.

Стрімкий розвиток магістральних мереж та систем обробки Big Data вимагає постійного вдосконалення алгоритмів навчання інтелектуальних систем. Значна частина задач сучасного системного аналізу та машинного навчання зводяться до мінімізації цільової функції, що описує помилку моделі. Вибір оптимального методу оптимізації безпосередньо впливає на швидкість збіжності алгоритму та економічну доцільність використання обчислювальних ресурсів. Таким чином, актуальність дослідження зумовлена необхідністю пошуку балансу між математичною точністю класичних підходів та обчислювальною ефективністю стохастичних методів у задачах великої розмірності.

Задача оптимізації параметрів системи полягає у пошуку такого вектора параметрів θ, при якому значення функції втрат $J(\theta)$ буде мінімальним:

$$\theta^* = \arg\min J(\theta),$$

де θ – параметри моделі (ваги), а $J(\theta)$ – функція, що чисельно відображає відхилення прогнозів моделі від реальних даних. Як зазначається у роботі [3], градієнтні методи базуються на ітераційному процесі оновлення параметрів у напрямку, протилежному вектору градієнта $\nabla J(\theta)$, який вказує шлях найшвидшого зростання функції.

Основна ідея методу класичного градієнтного спуску (Batch GD) полягає в обчисленні градієнта цільової функції на основі всієї навчальної вибірки на кожному кроці ітерації:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J(\theta_t),$$

де η – швидкість навчання (learning rate), а t – номер поточної ітерації. Згідно з методологією, описаною Глибовцем М. М., такий підхід забезпечує стабільну траєкторію руху до мінімуму [1], як показано на рисунку 1 [6]. Важливою умовою зупинки за [3] є досягнення заданої точності $\left\|\nabla f(x_k)\right\| \le \varepsilon$. Проте обчислювальна складність $O(n)$ робить цей метод занадто повільним для сучасних датасетів.

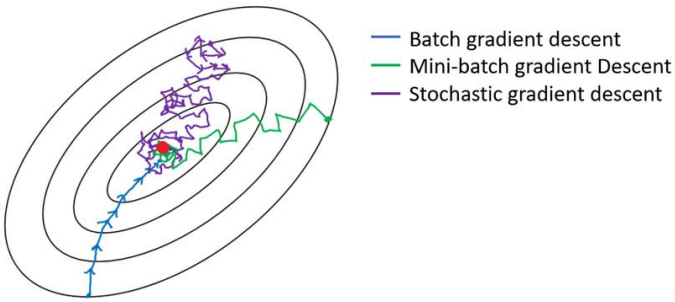

Рисунок 1 – Траєкторії збіжності градієнтних методів

Для подолання обмежень класичного підходу використовується стохастичний градієнтний спуск (SGD), де оновлення параметрів відбувається після обробки кожного окремого випадкового прикладу $\left(x^{(i)}, y^{(i)}\right)$:

$$\theta_{t+1} = \theta_t - \eta \cdot \nabla_\theta J\left(\theta_t; x^{(i)}; y^{(i)}\right).$$

Такий підхід, згідно Себастьяну Рудеру, вносить «стохастичний шум» у процес навчання [4]. Хоча це робить траєкторію зигзагоподібною, як показано на рисунку 2 [7], наявність коливань дозволяє алгоритму уникати пасток локальних мінімумів, що є критично важливим для нелінійних систем згідно з Субботіним С.О [2]. Обчислювальна складність однієї ітерації складає $O(1)$, що радикально прискорює процес на великих масивах даних.

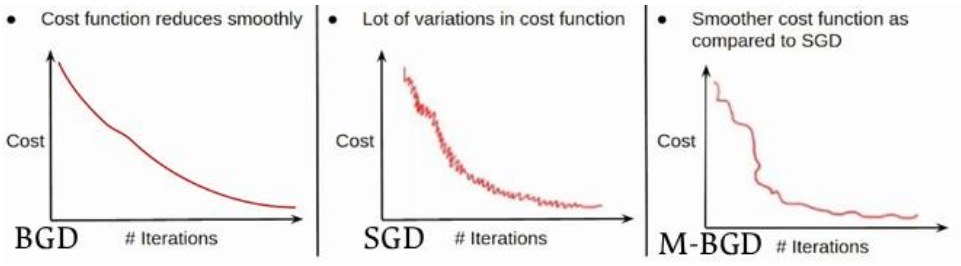

Рисунок 2 – Зміна функції втрат методів

На основі виконаного теоретичного аналізу робіт [4, 5] було сформовано порівняльну характеристику методів, що дозволяє визначити межі їх доцільного застосування. Результати представлено у таблиці 1.

Таблиця 1 – Порівняння характеристик алгоритмів оптимізації

Критерій порівняння	BGD	SGD
Обчислювальна складність	$O(n)$	$O(1)$
Стабільність траєкторії	Висока, детерміно-вана	Низька, стохасти-чна
Швидкість навчання	Низька на великих даних	Дуже висока
Ризик локальних мінімумів	Високий (може застрягати)	Низький (шум допомагає вийти)

Отже, проведене дослідження показує, що класичний градієнтний спуск залишається еталоном точності для задач малої та середньої розмірності. Водночас стохастичний підхід є безальтернативним для сучасних Big Data систем через свою економічну доцільність та здатність працювати в умовах обмежених обчислювальних ресурсів. Більш того розуміння математичної природи цих алгоритмів дозволяє ефективно проектувати цифрову інфраструктуру майбутнього.

Список використаних джерел:

1. Глибовець М. М., Олецький О. В. Системи штучного інтелекту : навч. посіб. Київ : Вид. дім «КМ Академія», 2002. 366 с. URL: http://kist.ntu.edu.ua/textPhD/ArtificIntell.pdf.

2. Субботін С. О. Подання й обробка знань у системах штучного інтелекту та підтримки прийняття рішень : навч. посіб. Запоріжжя : ЗНТУ, 2008. 341 с. URL: https://eir.zp.edu.ua/server/api/core/bitstreams/63742fdf-b5e4-46e2-83b4-561d97c2cffe/content.

3. Методи оптимізації та моделювання : навч. посіб. / С. В. Панченко, М. П. Медиченко, В. П. Лисечко та ін. Харків : УкрДУЗТ, 2016. 154 с. URL: http://lib.kart.edu.ua/bitstream/123456789/2371/1/Навчальний посібник.pdf.

4. Ruder S. An overview of gradient descent optimization algorithms. arXiv preprint, 2016. URL: https://www.ruder.io/optimizing-gradient-descent/.

5. Nocedal J., Wright S. J. Numerical Optimization. 2nd ed. Springer Science & Business Media, 2006. 664 p. URL: https://www.math.kent.edu/~reichel/courses/optimization/Numerical_Optimization.pdf.

6. Gandhi R. Batch, Mini-batch and Stochastic Gradient Descent. Data Science Collective. 2018. URL: https://medium.com/data-science-collective/batch-mini-batch-and-stochastic-gradient-descent-e9bc4cacd461.

7. ML | Mini-Batch Gradient Descent with Python. GeeksforGeeks. 2024. URL: https://www.geeksforgeeks.org/machine-learning/ml-mini-batch-gradient-descent-with-python/.