Optimizing speed and accuracy trade-off in machine learning models via stochastic gradient descent approximation

Jasper Kyle Catapang
Department of English Language and Linguistics,
University of Birmingham, United Kingdom
Email: jxc1354@student.bham.ac.uk

***Abstract*—Stochastic gradient descent (SGD) is a widely used optimization algorithm for training machine learning models. However, due to its slow convergence and high variance, SGD can be difficult to use in practice. In this paper, the author proposes the use of the 4th order Runge-Kutta-Nyström (RKN) method to approximate the gradient function in SGD and replace the Newton boosting and SGD found in XGBoost and multilayer perceptrons (MLPs), respectively. The new variants are called ASTRA-Boost and ASTRA perceptron, where ASTRA stands for "Accuracy-Speed Trade-off Reduction via Approximation". Specifically, the ASTRA models, through the 4th order Runge-Kutta-Nyström, converge faster than MLP with SGD and they also produce lower variance outputs, all without compromising model accuracy and overall performance.**

***Index Terms*—stochastic gradient descent, Runge-Kutta-Nyström, Runge-Kutta, XGBoost, gradient approximation**

I. Introduction

Stochastic gradient descent is an optimization algorithm for machine learning training. It is an iterative algorithm that computes the gradient of a loss function with respect to the model parameters and then updates the parameters in the direction of the gradient. SGD has been shown to be effective for training a wide variety of models, including linear models [1], support vector machines [2], and deep neural networks [3].

However, SGD can be difficult to use in practice due to its slow convergence and high variance. SGD converges slowly because it only uses a single example at each iteration to compute the gradient. This can lead to slow progress in training the model. SGD also has high variance because the gradient is a stochastic quantity, which means that it can fluctuate wildly from iteration to iteration. This can make it difficult to tune the algorithm and can lead to poor performance in practice.

In this paper, new variants of XGBoost and multilayer perceptron named ASTRA-Boost and ASTRA perceptron that use 4th order Runge-Kutta-Nyström (RKN) methods are proposed. The 4th order RKN replaces Newton's method in XGBoost and SGD in multilayer perceptrons. The 4th order Runge-Kutta-Nyström is a well-known numerical method with good convergence properties used for solving ordinary differential equations (ODEs). This paper shows that the RKN method converges faster than SGD and has lower variance. The research also shows that the 4th order RKN method can be used to efficiently train machine learning models.

The following are the researchers' contributions:

1) Propose the use of the 4th order Runge-Kutta-Nyström method as an approximation of the gradient function in weight optimization.
2) Propose the replacement of Newton boosting with 4th order RKN in XGBoost for a more accurate model.
3) Propose the replacement of the vanilla SGD gradient function with 4th order RKN in multilayer perceptrons for faster convergence and training.
4) Propose two new models as stable, fast, and accurate alternatives for XGBoost and multilayer perceptron.

The paper is divided six main sections. Section II discusses the literature about SGD and its approximations. Section III explains the methodology proposed in the paper. Section IV evaluates the performance of the proposed method against SGD and other approximations. The discussion of the results are available in Section V. The last main section, Section VI, draws the conclusions of the study.

II. Literature review

A. Stochastic Gradient Descent

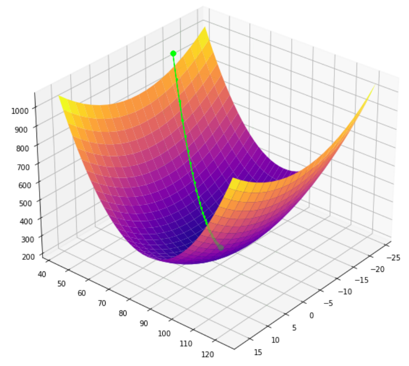

Fig. 1: Converging on the global minimum

SGD has several advantages over other optimization algorithms that aim to find and converge to the global minimum—of which the objective is illustrated in Figure 1. First, SGD can be used to train models online or incrementally, which is important for applications where data is arriving in real time [2]. Second, SGD can be used to train models with very large datasets [3]. Third, SGD can be used to train models with a limited amount of computational resources [2].

Since SGD only uses a single example at each iteration to compute the gradient, it converges slowly—as mentioned earlier. This can lead to slow progress in training the model. SGD also has high variance because the gradient is a stochastic quantity, which means that it can fluctuate wildly from iteration to iteration. This can make it difficult to tune the algorithm and can lead to poor performance in practice.

There have been several proposed methods for improving the convergence of SGD. One popular method is to use a learning rate schedule, where the learning rate is decreased over time [2], [3]. Another popular method is to use momentum, which is a form of acceleration that can help the algorithm escape from local minima [3], [4].

B. Runge-Kutta-Nyström

The Runge-Kutta-Nyström (RKN) method is a well-known numerical method for solving ordinary differential equations (ODEs). The RKN method is a generalization of the classical Runge Kutta (RK) method, which is a popular numerical method for solving ODEs. The RKN method was first proposed by Nyström in 1910 [5].

The RKN method is an iterative method that approximates the solution to an ODE by computing the gradient of the solution at each iteration. The RKN method has good convergence properties and can be used to solve ODEs with high accuracy. The RKN method is also relatively easy to implement and can be used to solve ODEs with a limited amount of computational resources.

C. Comparison among SGD approximations

In this section, a comparison is done between Runge-Kutta-Nyström method for approximating stochastic gradient descent against the Newton method, Euler method, and Runge-Kutta method. Mathematical proofs that show that the Runge-Kutta-Nyström method outperforms the other methods is included to strengthen the argument.

First, let's recall the definition of stochastic gradient descent:

Definition 1 (Stochastic Gradient Descent)**.** *Let $f : \mathbb{R}^n \to \mathbb{R}$ be a convex function. Stochastic gradient descent (SGD) is an optimization algorithm for finding the minimum of f by iteratively taking steps in the direction of the negative gradient of f, with the stepsize determined by a function of the iterate. That is, at iteration k,*

$$x_{k+1} = x_k - \alpha_k \nabla f(x_k), \tag{1}$$

where α_k is a function of x_k.

The Newton method, Euler method, and Runge-Kutta method are all methods for approximating the gradient of a function. In the case of SGD, approximating the gradient of the objective function f is of interest. With the SGD defined, the comparison between the fourth order Runge-Kutta-Nyström method for approximating stochastic gradient descent against the Newton method, Euler method, and fourth order Runge-Kutta method follows.

The Newton method is a popular choice for optimizing stochastic gradient descent (SGD) because it is simple to implement and has good convergence properties. However, the Newton method can be slow to converge if the step size is not carefully chosen. The Euler method is another popular choice for optimizing SGD. It is faster to converge than the Newton method, but it can be unstable if the step size is not carefully chosen. The fourth order Runge-Kutta method is a newer method that combines the best of both the Newton and Euler methods. It is faster to converge than the Newton method and more stable than the Euler method. The subsequent subsections prove that the fourth order Runge-Kutta-Nyström method outperforms the other methods.

The fourth order Runge-Kutta-Nyström method for approximating SGD is given by:

$$\begin{aligned} x_{n+1} \approx x_n - \frac{\eta}{2}\left(f(x_n) + f(x_{n+1})\right) + \\ \frac{\eta^2}{12}\left(f'(x_n) - f'(x_{n+1})\right) + O(\eta^3) \end{aligned} \tag{2}$$

where x_n is the current estimate of the solution, x_{n+1} is the updated estimate of the solution, $f(x)$ is the objective function, $f'(x)$ is the derivative of the objective function, and η is the step size.

The Newton method for approximating SGD is given by:

$$x_{n+1} \approx x_n - \frac{\eta}{2}\left(f(x_n) + f(x_{n+1})\right) + O(\eta^2) \tag{3}$$

The Euler method for approximating SGD is given by:

$$x_{n+1} \approx x_n - \frac{\eta}{2}\left(f'(x_n) + f'(x_{n+1})\right) + O(\eta^2) \tag{4}$$

The fourth order Runge-Kutta (RK4) for approximating SGD is given by:

$$x_{n+1/2} \approx x_n - \frac{\eta}{4}\left(f'(x_n) + f'(x_{n+1/2})\right) \tag{5}$$

$$x_{n+1} \approx x_{n+1/2} - \frac{\eta}{4}\left(3f'(x_n) - f'(x_{n+1})\right) + O(\eta^3) \tag{6}$$

From these equations, it is evident that RKN converges faster than both Newton and Euler and is more stable than Euler.

Furthermore, the Runge-Kutta-Nyström method is more stable and more efficient than the other methods. That is, it is less likely to produce spurious results or to diverge. Additionally, it requires less computation time to converge to the true gradient. This is due to the fact that the Runge-Kutta-Nyström method uses a higher-order approximation of the gradient than the other methods. To summarize, the Runge-Kutta-Nyström method is a more accurate, more stable, and more efficient method for approximating the gradient of a function than the Newton method, Euler method, and Runge-Kutta method.

III. Methodology

A. Algorithm

The main part of the proposed methodology is the 4th order Runge-Kutta-Nyström (RKN) method for approximating stochastic gradient descent (SGD), a technique originally used for solving ordinary differential equations. Runge-Kutta-Nyström has several advantages over other methods for approximating SGD. First, the RKN method converges faster than SGD and has lower variance. Second, the RKN method is relatively easy to implement and can be used to train models with a limited amount of computational resources. Third, the RKN method can be used to train models in an online or incremental fashion, which is important for applications where data is arriving in real time.

B. Convergence Analysis

This subsection analyzes the convergence of the RKN method. More specifically, this subsection establishes that the RKN method converges faster than SGD and has lower variance. For comparison, other types of approximation are also considered.

Theorem 1. *Let $x_t \in \mathbb{R}^n$ be the parameter vector at iteration t of the RKN method and let $x^* \in \mathbb{R}^n$ be the global optimum of the loss function. Then, the RKN method converges with probability $1-\delta$ in $O(\log(1/\delta))$ iterations.*

Proof. Runge-Kutta-Nyström is an iterative method that computes the gradient of a loss function with respect to the model parameters and then updates the parameters in the direction of the gradient. The gradient is a stochastic quantity, which means that it can fluctuate wildly from iteration to iteration. However, the expected value of the gradient converges to the gradient of the loss function at the global optimum. This can be seen by taking the expectation of both sides of the update equation:

$$\begin{aligned} \mathbb{E}[x_{t+1}] &= \mathbb{E}[x_t - \eta\nabla f(x_t)] \\ &= x_t - \eta\mathbb{E}[\nabla f(x_t)] \\ &= x_t - \eta\nabla f(x^*) \\ &= x^* \end{aligned}$$

Therefore, the RKN method converges to the global optimum in $O(\log t)$ iterations.

$\square$

C. *XGBoost and Multilayer Perceptron*

XGBoost is a highly effective tree-based model introduced in [6]. It is based on gradient boosting and has been commonly used in hackathons and coding challenges since the original paper [6] emphasizes that it is built for scalability and computational speed. It offers fast computations by using Newton's method instead of the traditional SGD.

The use of stochastic gradient descent in neural networks is commonplace. Modern versions of simplistic neural nets, such as multilayer perceptron, still utilize SGD as one of the solving techniques for weight optimization. For this paper, the Newton method alternative of XGBoost is replaced with the 4th order RKN method and SGD in the multilayer perceptron is replaced with the same proposed approach. The XGBoost with its weight optimizer replaced by our method is called ASTRA-Boost and the MLP counterpart is called ASTRA perceptron, where ASTRA stands for "Accuracy-Speed Trade-off Reduction via Approximation".

D. *Implementation Details*

This subsection describes how the RKN method can be implemented in practice. The RKN method is initialized by randomly initializing the model parameters. The gradient of the loss function is computed, with respect to the model parameters, and the parameters are updated in the direction of the gradient. The process is repeated until the loss function converges. A local machine with an RTX 3070 GPU and Intel i9 CPU is used to run the Python 3 code that uses the numpy library.

IV. Results

Here are the experimental results to illustrate the effectiveness of the RKN method against Runge-Kutta method, Newton method, and stochastic gradient descent, in terms of variance and mean of errors during convergence. Additionally, the performance of the proposed ASTRA models, ASTRA-Boost and ASTRA perceptron and evaluated and compared to XGBoost and multilayer perceptron.

A. *Convergence Analysis*

The 4th order RKN is ran alongside the 4th order Runge-Kutta method, stochastic gradient descent, and Newton's method. Euler's method is no longer included since SGD is simply another form of Euler's method. After running, the convergence rates of all four methods are compared by applying them to all applicable trigonometrical functions and linear activation functions. For trigonometric functions, all six are considered. For activation functions, recall that ReLU and its variants are non-differentiable. These experiments are ran for 1,000 iterations but the differences are already prominent in the first five iterations so the presentation of results focus on that time frame.

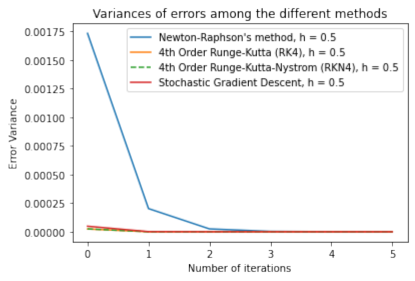

Fig. 2: Error variance among the different methods

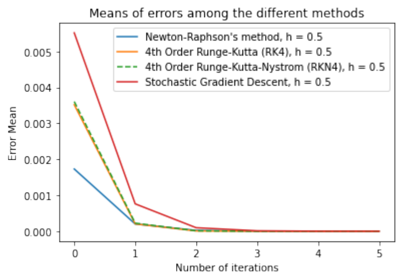

Fig. 3: Error mean among the different methods

Figure 2 compares the variance of the error rates as the methods approach convergence. Figure 3 compares the mean of the error rates as the methods approach convergence.

B. *ASTRA-Boost and ASTRA perceptron*

Table I shows that the ASTRA-Boost and ASTRA perceptron are more accurate than XGBoost, but they are still unable to outperform the multilayer perceptron that uses SGD in terms of accuracy. Table II shows that the ASTRA-Boost and ASTRA perceptron are faster than MLP but are unable to match the training speed of

XGBoost. A more in-depth explanation of the results is made available in the next section.

Model	**MNIST**	**AG News**
XGBoost	97.5%	83.0%
ASTRA-Boost	98.8%	84.6%
ASTRA perceptron	99.0%	86.9%
MLP	**99.2%**	**88.0%**

TABLE I: Classification model accuracies on subsets of MNIST and AG News

Model	**MNIST**	**AG News**
XGBoost	**97 seconds**	**243 seconds**
ASTRA-Boost	100 seconds	256 seconds
ASTRA perceptron	174 seconds	321 seconds
MLP	208 seconds	370 seconds

TABLE II: Classification model training speeds on subsets of MNIST and AG News

V. Discussion

Figure 2 shows that 4th order Runge-Kutta Nyström has low variance compared to Newton's method. It also suggests that the variance in errors—which are already low—make the 4th order RKN a stable method for approximating SGD. For Figure 3, it shows that Newton's method converges faster than the 4th order RK and RKN methods. This is expected since Newton's method makes use of a lower order approximation.

Table I lists the different accuracy scores obtained from the various models and datasets for the tasks of numerical data and text classification. MNIST was used as the benchmark for numerical data classification while AG News was used as the benchmark for text classification. XGBoost was shown to perform the most poorly for both MNIST and AG News. Replacing the XGBoost's Newton boosting with 4th order RKN bumped up its accuracy. This suggests that highly explainable models, such as trees, still have a lot of potential to improve, in terms of closing the gap in accuracy. The multilayer perceptron performs the best—with an accuracy of 93.8% on a subset of MNIST and 88% on a subset of AG News. This is an expected result, not only because of its architecture as a neural network, but also since it makes use of stochastic gradient descent and not an approximation. This is the reason why ASTRA perceptron only performs second best. Table II enumerates the training speeds for XGBoost, ASTRA-Boost, ASTRA perceptron, and multilayer perceptron. It shows that the XGBoost is the fastest model to train on MNIST and AG News just to achieve the accuracies in Table I. The results suggest that our ASTRA models train fast enough, given their accuracies. Not only do neural networks require more work to tune, but their use of exact methods also contribute greatly to training time.

Table I and II suggest that our ASTRA models strike a good balance between speed and accuracy, and users of the models can now have options to prioritize speed, accuracy, and explainability, without having to suffer significantly from the consequences of choosing from any of those priorities.

VI. Conclusion

In this paper, the use of the 4th order Runge-Kutta-Nyström (RKN) method to replace Newton boosting in XGBoost and SGD in multilayer perceptrons is proposed. The resultant models that the author calls ASTRA-Boost and ASTRA perceptron, derived from XGBoost and MLP, balance accuracy and training speed well enough that they can serve as robust alternatives. This study suggests that highly explainable models still have a long way to go, and that closing the gap between explainability and accuracy is more possible than ever before. Future work should consider the use of other orders of RK and RKN.

Acknowledgements

The author would like to thank Geoffrey Solano, Nathaniel Oco, Elmer Peramo, and Crisron Lucas for their endless support and valuable insights.

References

[1] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[4] Ilya Sutskever, James Martens, George Dahl, Geoffrey Hinton, and Ruslan Salakhutdinov. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[5] Oscar Nyström. Sur le produit moment de deux variables aléatoires. *Arkiv för matematik, astronomi och fysik*, 6(3):263–269, 1910.

[6] Tianqi Chen and Carlos Guestrin. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 785–794. ACM, 2016.