Iterative Self Correction Through Dynamic Quadratic Loss Functions

Ian O. Omung'a I.OMUNGA @ ALUSTUDENT.COM
Department of Computing, African Leadership College

Abstract

Conventional error correction is based on implementations of dedicated error handling sub-systems that serialize all the ways the developers and designers of the main system envision runtime error could occur. In best-case scenarios where these error handling subsystems need not be utilized at all, they serve a vestigial purpose, existing solely for the off chance that they may be needed in very specific scenarios. When necessary, the capacity of these error handling sub-systems to sufficiently handle errors is limited by whatever scenario was envisioned during the main system's creation. Implementing quadratic loss functions as embedded, recursive elements of error handling that can be dynamically updated at runtime within robust self-correction algorithms serves as a progressively improving, automated framework for error handling that not only need not be constantly maintained or tweaked based on inferred problematic scenarios but also as one that gets better over time. Foundational formalizations of this algorithm show the logical structure of the self-correction framework and one such implementation pathway to achieve it. The modularity of the algorithm to incorporate varied loss functions is also exhibited through its alternative implementation using the Mean Absolute Error loss function.
Keywords: recursive self correction, quadratic loss function, algorithmic optimization

1. Introduction

Error handling as it is implemented now is heavily dependent on hypothetical scenarios envisioned during the design and creation of information processing systems and as such, is not able to handle data sources prone to high error occurrence in widely varied ways that are hard to predict. Iterative self-correction algorithms that work through dynamic quadratic loss functions are seen to be a source of constant improvement in a data-driven process where the human practitioners who calibrate it can do so from an informed standpoint. In artificial intelligence systems, the dynamic updating of the parameters of the self-correction algorithm can be automated by programming self-attention mechanisms within the artificial intelligence systems to update the loss functions at runtime whenever predicted values are divergent from the ground truth values on which training is being performed.

Quadratic loss functions' suitability for optimizing the distance of displacement away from the erroneous points in the vector space of solutions for information processing systems emerges from the heavy penalization they effect when such errors occur. Since the factors in the function are only evaluated up to square factors [by a power of 2], the mathematical penalizations and subsequent transformations away from erroneous vector regions are fairly straightforward to interpret and replicate if need be. Furthermore, the versatility of the quadratic loss function through scaling up by scalar factors means that error corrections of higher magnitudes can be executed all at once without the need for multiple iterations within the information processing system, thereby lessening processor time consumed on error correction in aggregate. Iterating through multiple generations of automated error correction therefore functions in the same manner as an evolutionary algorithmic process. This is effected through selecting for solutions with the least error and eliminating ones with errors from the vector space to create successively better information processing standards.

The quadratic loss function within the iterative self-correction algorithm is therefore defined as follows:

Quadratic Loss Formula

$$\frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{n}$$

Where
n = number of training examples
y_i = ground truth label for ith training example
$\hat{y}_i$ = model's prediction for ith training example
The algorithm is implemented in Python with the NumPy[6], a Python mathematical library.

2. Relation to Past Work

2.1 Test Mean Square Errors: Calculating A Model's Prediction Quality

The quadratic loss formula has been employed in past mathematical attempts to compute it's absolute value as a heuristic for a prediction/estimation system's quality and efficacy (James, Witten, Hastie and Tibshirani, 2021). This is due to the fact that for a stochastically randomized set of values, where **n** predictions are computed from a sample of **n** data points on all variables, the quadratic loss function can be proven to be equal to the variance in the non-biased dataset as follows(Bengio, Goodfellow and Courville, 2016):

$$MSE = \mathbb{E}\left[(\hat{\theta}_S - \theta)^2\right] = Bias^2(\hat{\theta}_S, \theta) + Var(\hat{\theta}_S)$$

$$\mathbb{E}\left[(\hat{\theta}_S - \theta)^2\right] = \mathbb{E}[\hat{\theta}_S^2] + \theta^2 - 2\mathbb{E}[\hat{\theta}_S]\theta$$

$$\begin{aligned} Bias^2(\hat{\theta}_S, \theta) &= \left(\mathbb{E}[\hat{\theta}_S] - \theta\right)^2 \\ &= \mathbb{E}^2[\hat{\theta}_S] + \theta^2 - 2\mathbb{E}[\hat{\theta}_S]\theta \\ Var(\hat{\theta}_S) &= \mathbb{E}[\hat{\theta}_S^2] - \mathbb{E}^2[\hat{\theta}_S] \end{aligned}$$

This variance can be represented as the transformation matrix as shown below:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(e_i)^2 = \frac{1}{n}\mathbf{e}^\mathsf{T}\mathbf{e}$$

Where
e_i = (y_i - $\hat{y}_i$)
e = *n*x1 matrix

In estimation or prediction scenarios where a section, *q* of the data is withheld during training so as to test the quality of the weights within the model, the quadratic loss function will no longer be based solely on the number of values analysed in the dataset but will also include the set of values priorly unknown to the model and is computed as follows:

$$MSE = \frac{1}{q}\sum_{i=n+1}^{n+q}\left(Y_i - \hat{Y}_i\right)^2.$$

and is referred to as ***testMSE***(James, Witten, Hastie and Tibshirani, 2021).

2.2 Gaussian Distributions' Correlation with Undesired Tmse Increase

This set of predicition and estimation systems now includes transformer-based models of non-sparse parametres in which an aggregation of marginal gains holds the potential to compound into substantial improvement.

The quadratic Loss function has been used a test value for comparing the variance between what models predict to be accurate data versus the actual labels being used to train them against how these models afterwards perform on new data that they have not analyzed before. In this framework, the quadratic loss formula's results are observed to increase unreliably at crucial points of analyses, thereby indicating increasing variance between griund truth values and what the model is predicting mistakenly. This is of increasing consequence since it occurs when working on data it was not trained on; the very case the model is supposed to excel at.

At this stage in the analysis where more and more data is being included in the model, the measure of randomness for this dataset as an aggregate whole keeps increasing. This is due to Gaussian phenomena occuring in accordance with the central limit theorem, such that the increasing number of samples encountered in these characeristically vast training datasets tend more and more towards a normal [Gaussian] distribution. It follows therefore that the sudden increase being observed in the tendency of the test MSE strongly correlates with the bounds of a Gaussian distribution, rising towards the maxima of the bell curve in either direction; positive or negative. Furthermore, it follows that subsequent computations that keep on utilising the testMSE sustained by the availability of a near-constant data stream [such as those used in training non-sparse parametre neural networks] will experience a negative trend of displacement within the vector space of the distribution and tend towards the tails of the curve on a timescale that is proportional to how much of the data that has been analysed can be described by a Gaussian distribution.

In this way, sustained error corrections automatically iterated throughout the lifecycle of the guassian curve as it traverses the normal distribution will lead to a smaller and smaller available solution space from where a prediction or estimation can be derived. The aggregation of these marginal reductions in the search space will substantially lower compute processing periods and lead to enhanced performance.

3. Main Contributions

3.1 Faster improvement timelines for models with non-sparse parameter clusters

Large scale parameter models today possess an immense number of parameters, ranging from 1.5 Billion[1] parameters to 1.75 Trillion[2] parameters. These herculean model sizes mean that error correction within the neural networks can only be effected after source data has been ingested through its data pipelines and processed across the neural networks' various parallel layers. Incorporating dynamic self-correction algorithms into the infrastructure of these neural networks at the parameter level cuts enables the identification and necessary correction of errors in real-time during processing. This further means that the timelines within which such large-scale neural networks can be developed, tested, and validated will be much shorter in comparison to neural networks functioning without the iterative self-correcting algorithm.

3.2 High efficiency error-correction without human expert intervention

Current error correction processes rely on human practitioners to monitor and test information processing systems like artificial intelligence models during data processing to attempt to determine the causal agents of errors within the models' parameters. In some cases, the causal agents of the errors within the neural networks are completely unknown to the human practitioners who design and operate them[3]. Using this iterative self-correction algorithm as an automated algorithm for error correction embedded within the hidden layers of the neural networks means that errors created by the neural networks' yet unknown mathematical operations can be probed, understood, and corrected before interfering with the neural network's functionality.

3.3 A FORMALIZED SELF-CORRECTING ALGORITHM THAT CAN BE CHANGED AND ADAPTED FOR VARIANT USE-CASES

Current error correction processes rely on human practitioners to monitor and test information processing systems like artificial intelligence models during data processing to attempt to determine the causal agents of errors within the models' parameters. In some cases, the causal agents of the errors within the neural networks are completely unknown to the human practitioners who design and operate them3. Using this iterative self-correction algorithm as an automated algorithm for error correction embedded within the hidden layers of the neural networks means that errors created by the neural networks' yet unknown mathematical operations can be probed, understood, and corrected before interfering with the neural network's functionality.

3.4 A VERSATILE LOSS FUNCTION THAT CAN BE VARIED USING SCALAR OPERATIONS OR BE REPLACED ENTIRELY WITHOUT INCAPACITATING THE ALGORITHM

The self-correcting algorithm proposed herein is versatile in the mode of its usage since the quadratic loss function can be alternated entirely for functions that pursue different approaches to the task of optimizing the error correction process. In place of the quadratic loss function, alternative optimization functions like the Mean Absolute Error function can be reliably used within the iterative self-correction algorithm for error correction.

3. Conclusions

- The algorithm is adaptable for inclusion within large scale projects working on highly variant and unlabelled corpora like texts scrapped from the entirety of the internet where errors occur frequently and need to be corrected quickly.
- Error correction proceeds automatically after errors are detected afterwhich the faulty values are recursed into the algorithm for subsequent iterations of error correction. In this way, the correctional action to be effected by the quadratic loss function is dynamically adapted to match the range of errors that occur in high frequency. Since the algorithm's action actively selects against these classes of errors, the evolution of the model to become less and less error prone is recursed to a higher and higher degree.

4. Future Work

- Subsequent work will be done to formalize the algorithm as an infrastructurally embedded component within the Self-Attention Transformer architecture (Vaswani et. al., 2017) so that attentiveness can be coupled with swift error correction at each juncture that is both automated and recursive
- Quanitfying a complete description of the trends of the test Mean Square Error and investigating the existence of a constant scalar of decline correlated to the corresponding changes in the Gaussian Distribution it follows

5. Acknowledgments

- A substantial amount of the mathematical knowledge used in the design of this algorithm has been adapted from the reference book, 'Mathematics for Machine Learning' [5]. I ackowledge with thanks the role that obtaining this knowledge from it in a succint manner has played in the development of the recursive self-correcting algorithm.

5. References

- Solaiman, I., Clark, J. and Brundage, M., 2019. *GPT-2: 1.5B Release*. [online] OpenAI. Available at: <https://openai.com/blog/gpt-2-1-5b-release/> [Accessed 13 January 2022].
- WeChat public platform. 2022. Facing cognition, *Zhiyuan Research Institute* jointly released a new super-large-scale pre-training model "Wu Dao·Wenhui". [online] Available at: <https://mp.weixin.qq.com /s/BUQWZ5EdR19i40GuFofpBg> [Accessed 14 January 2022].
- Bathaee, Y., 2018. THE ARTIFICIAL INTELLIGENCE BLACK BOX AND THE FAILURE OF INTENT AND CAUSATION. [online] Jolt.law.harvard.edu. Available at: <https://jolt.law.harvard.edu/assets/articlePDFs/v31/*The-Artificial-Intelligence-Black-Box-and-the-Failure-of-Intent-and-Causation*-Yavar-Bathaee.pdf> [Accessed 14 January 2022]
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Gomez, A., Kaiser, L. and Polosukhin, I., 2017. *Attention Is All You Need.* Arxiv.org, [online] arXiv:1706.03762, p.Abstract. Available at: <https://arxiv.org/abs/1706.03762> [Accessed 13 January 2022].
- Deisenroth, M., Faisal, A. and Ong, C., 2020. *Mathematics for Machine Learning. 2nd ed*. Cambridge: Cambridge University Press, pp.159-163.
- Harris, C.R., Millman, K.J., van der Walt, S.J. et al. *Array programming with NumPy*. Nature 585, 357–362 (2020).