Iterated Piecewise Affine (IPA) Approximation for Language Modeling

Davood Shamsi
davood@axiomsix.com

Wen-yu Hua
wenyu_hua@apple.com

Brian Williams
brian@vartia.ai

Abstract

In this work, we demonstrate the application of a first-order Taylor expansion to approximate a generic function $F : R^{n\times m} \rightarrow R^{n\times m}$ and utilize it in language modeling. To enhance the basic Taylor expansion, we introduce iteration and piecewise modeling, leading us to name the algorithm the Iterative Piecewise Affine (IPA) approximation. The final algorithm exhibits interesting resemblances to the Transformers decoder architecture. By comparing parameter arrangements in IPA and Transformers, we observe a strikingly similar performance, with IPA outperforming Transformers by 1.5% in the next token prediction task with cross-entropy loss for smaller sequence lengths.

1 Introduction and Problem Description

Transformers [1] and their variations [2–16] have been the driver of the recent development in AI. However, the model architecture appears to be more the result of craftsmanship than formal function approximation methodology. In this paper, we demonstrate how a similar yet fundamentally different model can be developed by using Taylor expansion and piecewise function estimation techniques. [17, 18, 10, 19–21]. In a language model [17, 18, 10, 19–21], as shown in Fig. 1, first there is an embedding layer that maps each token to a vector in R^n. If the input sequence is of length m, the output of the embedding layer is a matrix $\mathbf{X} \in R^{n\times m}$. Next, the matrix $\mathbf{X}$ is passed through a function F. And finally, there is an affine head (with softmax) that maps the output of $F(\mathbf{X})$ to probability distribution of the next word. While there may be an embedding layer and final prediction head, at its core, a language model approximates a function $F : R^{n\times m} \rightarrow R^{n\times m}$ that maps a matrix space to itself. Once the language modeling task is mapped to the function approximation in the matrix domain, it is natural to ask how effective is a first-order Taylor expansion? Here is the Taylor expansion around a given point x_0 (in one dimension for simplicity):

$$F(x) \approx L(x) = F(x_0) + F'(x_0)(x - x_0). \tag{1}$$

The first-order Taylor expansion can be a good approximation around the center point x_0, but not globally. To improve the accuracy of the approximation, we can write the first-order Taylor expansion around P center points and combine them using a set of kernel functions:

$$F(x) \approx \sum_{p=1}^{P} K^p(x)L^p(x). \tag{2}$$

$K^p(x)$ and $L^p(x)$ are kernel functions (e.g. exponential) and affine approximations Eq. (1) for p-th center point. For visual illustration, the three red lines in Fig. 2 are Taylor expansions around 3 center points and we used kernels (dashed-green line) to combine them. Upon closer inspection, we can observe that estimating through multiple center points exhibits a very similar, though not

37th Conference on Neural Information Processing Systems (NeurIPS 2023) Workshop on Self-Supervised Learning.

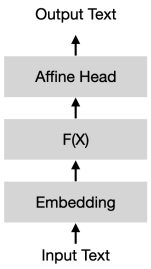

Figure 1: Stages in language modeling: tokens are embedded into vectors, the resulting matrix is passed through a function $F(\mathbf{X})$ and the next token is predicted with an affine head (typically a feedforward layer).

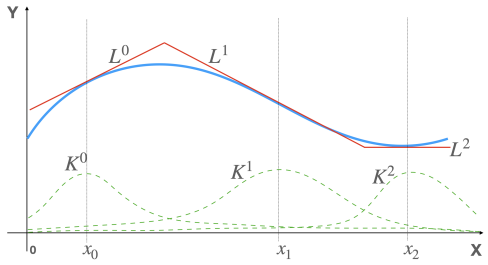

Figure 2: Piecewise Affine Function Estimation: The blue curve is estimated using first-order Taylor expansions at points x_0, x_1, and x_2. The final estimate is a combination of these lines using kernel functions K_0, K_1, and K_2.

identical, relationship to the multi-head architecture found in Transformers. Finally, we apply this approximation iteratively by creating layers to arrive at the final estimator. The parameters of the Tylor expansions are calculated using gradient descent on a training dataset. Our main contribution is the introduction of the Iterative Piecewise Affine (IPA) algorithm, which is a straightforward and mathematically intuitive method for approximating a function in the matrix space. The IPA algorithm is competitive to Transformers, but it does not use any heuristics and is easy to understand.

This paper is organized as follows. First, in Section 2, we extend Eq. (1) and (2) from one-dimensional functions to functions in the matrix domain. Then, in Section 3, we modify the basic IPA to make it suitable for language modeling (e.g., causality constraint). Next, in Section 4, we demonstrate the relationship between IPA and Transformers. Finally, in Section 5, we compare the performance of IPA to Transformers and conclude in Section 6.

2 Iterative Piecewise Affine Estimator (IPA) Approximation

Our goal is to estimate a function $F : R^{n\times m} \rightarrow R^{n\times m}$ that maps a matrix space. We use affine function estimators based on first-order Taylor expansion, piecewise affine estimation using kernel functions, and improve the estimator through iteration.

2.1 Affine Estimation

Lets consider writing Taylor expansion for the rows and columns of the function F separately. In what follows, as shown in Fig. 3, we use a dot before the index of a variable to denote that it represents a column and a dot after the index to show that it represents a row.

Column Representation Let $\mathbf{f}_{\cdot j}(\mathbf{X})$ be the j-th column of $F(\mathbf{X})$ and $\hat{\mathbf{f}}_{\cdot j}(\mathbf{X})$ be its first-order Taylor approximation. Then,

$$\mathbf{f}_{\cdot j}(\mathbf{X}) \approx \hat{\mathbf{f}}_{\cdot j}(\mathbf{X}) = \mathbf{a}_j + \sum_{l=1}^{m} \mathbf{S}_{j,l}\mathbf{x}_{\cdot l}, \tag{3}$$

where, $\mathbf{a}_j \in R^n$ and $\mathbf{S}_{j,l} \in R^{n\times n}$ are the approximation coefficients and $\mathbf{x}_{\cdot l}$ is the l-th column of $\mathbf{X}$.

Row Representation Similarly, we can write the Taylor expansion for rows. If $\mathbf{f}_{i\cdot}(\mathbf{X})$ is the i-th row of $F(\mathbf{X})$, and $\mathbf{x}_{r\cdot}$ is the r-th row of $\mathbf{X}$, then

$$\mathbf{f}_{i\cdot}(\mathbf{X}) \approx \hat{\mathbf{f}}_{i\cdot}(\mathbf{X}) = \mathbf{b}_i + \sum_{r=1}^{n} \mathbf{T}_{i,r}\mathbf{x}_{r\cdot}, \tag{4}$$

where, $\mathbf{b}_i \in R^m$ and $\mathbf{T}_{i,r} \in R^{m\times m}$ are the row approximation coefficients.

2.2 Piecewise Affine Estimation

In the previous section, we used first-order Taylor expansion to estimate F. However, Taylor expansion around one point might not be an accurate estimator over the general domain of the

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_{\cdot 1} & \mathbf{x}_{\cdot 2} & \dots & \mathbf{x}_{\cdot m} \end{pmatrix}, \mathbf{X} = \begin{pmatrix} \mathbf{x}_{1\cdot} \\ \mathbf{x}_{2\cdot} \\ \vdots \\ \mathbf{x}_{n\cdot} \end{pmatrix}$$

$$F(\mathbf{X}) = \begin{pmatrix} \mathbf{f}_{\cdot 1}(\mathbf{X}) & \mathbf{f}_{\cdot 2}(\mathbf{X}) & \dots & \mathbf{f}_{\cdot m}(\mathbf{X}) \end{pmatrix}, F(\mathbf{X}) = \begin{pmatrix} \mathbf{f}_{1\cdot}(\mathbf{X}) \\ \mathbf{f}_{2\cdot}(\mathbf{X}) \\ \vdots \\ \mathbf{f}_{n\cdot}(\mathbf{X}) \end{pmatrix}$$

Figure 3: Column and row representation of $\mathbf{X}$ and $F(\mathbf{X})$. These representations can result in different approximations.

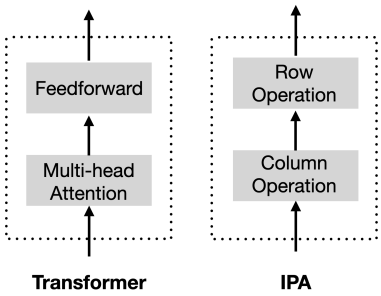

Figure 4: Comparing the IPA algorithm to Transformers. Both methods apply column and row operations consecutively.

function. To address this issue, we can write the expansion around multiple points and combine them using some kernel functions (e.g. see [8] page 172). Eq. (3), $\mathbf{S}_{j,l}$ can be estimated as:

$$\mathbf{S}_{j,l}(\mathbf{X}) \approx \sum_{p=1}^{P} K_{j,l}^{p}(\mathbf{X}) \mathbf{S}_{j,l}^{p}, \tag{5}$$

where $\mathbf{S}_{j,l}^{p}$ are coefficients for p-th Taylor expansions. We will discuss the choice of kernels $K_{j,l}^{p}(\mathbf{X})$ later in Section 3.2. Please note the concept of kernel used here is different from the one used in Transformers, e.g. [2, 3]. The same estimation can be applied to $\mathbf{T}_{i,r}$ in Eq. (4),

2.3 Estimating through Iterated Functions

As it is common in the deep learning literature [22, 23], we can use iteration to model higher levels of non-linearities:

$$F(\mathbf{X}) \approx \hat{F}_{\gamma_1} \circ \hat{F}_{\gamma_2} \circ \hat{F}_{\gamma_3} \circ \cdots \circ \hat{F}_{\gamma_n}(\mathbf{X}), \tag{6}$$

where symbol $\circ$ represents "function of function" and $\hat{F}_{\gamma_i}$ are, alternatively, piecewise affine estimations from Eq. (3) and (4) with different estimation parameters.

3 IPA Approximation for Language Modeling

As is commonly found in literature, we formulate language modeling by estimating a shifted sequence of tokens from its original form. Fig. 1 illustrates the end-to-end language modeling task, with the main objective being to approximate $F(\mathbf{X})$. The input text is first tokenized and embedded into vectors, which are then fed into the IPA to approximate $F(\mathbf{X})$. Finally, a commonly-used affine head predicts the output (next tokens). We need to make some modifications to the original IPA formulation to make it compatible with language modeling, which will be discussed below.

3.1 Adding Causality Constraints to the IPA Algorithm

To ensure that we only utilize past tokens for predicting the next token in both the affine functions and kernels, we set all coefficients from future tokens to zero.

Column Operation: To mask the future tokens, in Eq. (3), the sum should be from $l = 1$ to j.

Row Operation: Enforcing the causality constraint for row approximation can be more challenging. For simplicity, we make the constraint stronger by enforcing the matrix $\mathbf{T}_{i,r}$ in Eq. (4) to be diagonal.

3.2 Kernel Function

As has been proven effective in natural language processing literature, we use attention-style kernel functions for column representation: $K_{j,l}^{p}(\mathbf{X}) = \lambda^{-1} e^{\mathbf{x}_{\cdot l}^{T} \mathbf{W}^{p} \mathbf{x}_{\cdot j}}$, where $\mathbf{W}^{p} \in R^{n \times n}$ are parameters of the kernel, and λ is the normalization factor, and chosen such that $\sum_p K_{j,l}^{p}(\mathbf{X}) = 1$. For the row representation, we use Gaussian radial kernels (Section 5.8.2 in [8]).

3.3 Position Independent Mapping

In language modeling, each column of matrix X represents a token in the input sequence. Therefore, it is natural to assume that the position of the token should not affect the mapping: $\mathbf{S}_{j,l} = \mathbf{S}_{j',l} \quad \forall j, j'$,; $\mathbf{T}_{i,r} = \mathbf{T}_{i',r} \quad \forall i, i'$. This assumption is mainly for reducing number of parameters, and position of the token is still important in the IPA formulation.

3.4 Reducing Parameters with Low Rank Matrices

In the previous formulation, there were no restrictions on $\mathbf{S}_{i,l}$ and $\mathbf{W}^p$, so they could be full-rank matrices. To lower the number of parameters, we can assume that they have a lower rank of k. For example: $\mathbf{W}^p = \mathbf{W}^p_l \times \mathbf{W}^p_r$, where $\mathbf{W}^p_l \in R^{n \times k}$ and $\mathbf{W}^p_l \in R^{k \times n}$.

4 Relationship with Transformers

Although there are fundamental differences between the IPA algorithm and Transformers, as illustrated in Fig. 4, their architectures share some intriguing similarities. Specifically, the multi-head attention mechanism in Transformers can be viewed as a column operation and the subsequent feedforward layer as a row operation. Additionally, upon closer analysis, it can be observed that the kernels in the piecewise affine operation of the IPA algorithm have similar roles as attention heads in Transformers.

5 Experimental Results

In this section, we compare performance of the IPA algorithm with GPT architecture [24] (stack of Transformers decoder) on the WikiText103 dataset [25]. In order to make a meaningful comparison, we closely matched the internal parameters of IPA and Transformer. For all experiments, the embedding size was set to $n = 120$ and there were 4 layers. For the column operation, we set the number of affine functions equal to the number of heads in the Transformer model ($P_{\text{column}} = 8$) and the rank of matrices (k in Section 3.4) equal to the embedding size of each head in the Transformer model ($k = 15$). For the row operation, we set the number of affine functions equal to the ratio of the feedforward's inner dimension to the embedding size. Specifically, we set Transformer feedforward's inner dimension equal to 4 and thus, $P_{\text{row}} = 4$. We use Byte Pair Encoding [26, 27] to tokenize the input text, and no dropout was used for either model.

Table 1: Train and test loss on WikiText103 dataset. Loss is the cross-Entropy for the next word prediction, and m is the sequence length. Time per iteration is measured in milliseconds.

Model, (m)	# Parameters	Train Loss	Test Loss	Time per Iteration
GPT, (100)	4.45M	4.51	4.45	28.4
IPA, (100)	4.49M	4.49	4.38	30.7
GPT, (250)	4.47M	4.13	4.09	65.0
IPA, (250)	4.56M	4.17	4.07	66.5
GPT, (500)	4.50M	3.91	3.90	148.8
IPA, (500)	4.68M	3.98	3.89	147.0

Table 1 displays the train and test loss of the IPA algorithm compared to the GPT architecture (Transformer decoders) for three different sequence lengths on the WikiText103 dataset. As reminder, variable m represent length of the sequence. The loss is calculated as cross-entropy for the next token prediction. All experiments were run until convergence based on the test loss (≈ 10 million steps with a learning rate of 2e-5). As shown in Table 1, with a similar configuration, the IPA algorithm has better performance than GPT for small sequence lengths (1.5% for $m = 100$) but they have very similar performance for longer sequences ($m = 500$). From Table 1, you can observe that the training time ($\approx$ computation cost) is very similar in both models.

6 Conclusion

In this paper, we introduced IPA algorithm for estimating a general function $F : R^{n \times m} \to R^{n \times m}$ and applied it to language modeling. The IPA algorithm is straightforward, intuitive, and shows comparable performance to Transformers.

References

[1] Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; Polosukhin, I. Attention Is All You Need. *CoRR* **2017**, *abs/1706.03762*.

[2] Choromanski, K.; Likhosherstov, V.; Dohan, D.; Song, X.; Gane, A.; Sarlós, T.; Hawkins, P.; Davis, J.; Mohiuddin, A.; Kaiser, L.; Belanger, D.; Colwell, L. J.; Weller, A. Rethinking Attention with Performers. *CoRR* **2020**, *abs/2009.14794*.

[3] Beltagy, I.; Peters, M. E.; Cohan, A. Longformer: The Long-Document Transformer. *CoRR* **2020**, *abs/2004.05150*.

[4] Bello, I.; Zoph, B.; Vaswani, A.; Shlens, J.; Le, Q. V. Attention Augmented Convolutional Networks. *CoRR* **2019**, *abs/1904.09925*.

[5] Child, R.; Gray, S.; Radford, A.; Sutskever, I. Generating Long Sequences with Sparse Transformers. *CoRR* **2019**, *abs/1904.10509*.

[6] Britz, D.; Guan, M. Y.; Luong, M. Efficient Attention using a Fixed-Size Memory Representation. *CoRR* **2017**, *abs/1707.00110*.

[7] El-Nouby, A.; Touvron, H.; Caron, M.; Bojanowski, P.; Douze, M.; Joulin, A.; Laptev, I.; Neverova, N.; Synnaeve, G.; Verbeek, J.; Jégou, H. XCiT: Cross-Covariance Image Transformers. *CoRR* **2021**, *abs/2106.09681*.

[8] Hastie, T.; Tibshirani, R.; Friedman, J. H. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*; Springer Series in Statistics; Springer, 2009.

[9] Chen, C.; Fan, Q.; Panda, R. CrossViT: Cross-Attention Multi-Scale Vision Transformer for Image Classification. *CoRR* **2021**, *abs/2103.14899*.

[10] Devlin, J.; Chang, M.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers). 2019; pp 4171–4186.

[11] Mazzia, V.; Angarano, S.; Salvetti, F.; Angelini, F.; Chiaberge, M. Action Transformer: A Self-Attention Model for Short-Time Human Action Recognition. *CoRR* **2021**, *abs/2107.00606*.

[12] He, R.; Ravula, A.; Kanagal, B.; Ainslie, J. RealFormer: Transformer Likes Residual Attention. *CoRR* **2020**, *abs/2012.11747*.

[13] Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; Joulin, A. Emerging Properties in Self-Supervised Vision Transformers. *CoRR* **2021**, *abs/2104.14294*.

[14] Shvetsova, N.; Chen, B.; Rouditchenko, A.; Thomas, S.; Kingsbury, B.; Feris, R.; Harwath, D.; Glass, J. R.; Kuehne, H. Everything at Once - Multi-modal Fusion Transformer for Video Retrieval. *CoRR* **2021**, *abs/2112.04446*.

[15] Han, T.; Xie, W.; Zisserman, A. Self-supervised Co-training for Video Representation Learning. *CoRR* **2020**, *abs/2010.09709*.

[16] Fang, H.; Xie, P. CERT: Contrastive Self-supervised Learning for Language Understanding. *CoRR* **2020**, *abs/2005.12766*.

[17] Brown, T. B. et al. Language Models are Few-Shot Learners. *CoRR* **2020**, *abs/2005.14165*.

[18] Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *CoRR* **2019**, *abs/1907.11692*.

[19] Shoeybi, M.; Patwary, M.; Puri, R.; LeGresley, P.; Casper, J.; Catanzaro, B. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. *CoRR* **2019**, *abs/1909.08053*.

[20] Chowdhery, A.; et al. PaLM: Scaling Language Modeling with Pathways. 2022; `https://arxiv.org/abs/2204.02311`.

[21] Thoppilan, R. et al. LaMDA: Language Models for Dialog Applications. *CoRR* **2022**, *abs/2201.08239*.

[22] He, K.; Zhang, X.; Ren, S.; Sun, J. Deep Residual Learning for Image Recognition. *CoRR* **2015**, *abs/1512.03385*.

[23] Huang, F.; Ash, J. T.; Langford, J.; Schapire, R. E. Learning Deep ResNet Blocks Sequentially using Boosting Theory. *CoRR* **2017**, *abs/1706.04964*.

[24] Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. LZP: A New Data Compression Algorithm. Improving language understanding by generative pre-training. 2018.

[25] Merity, S.; Xiong, C.; Bradbury, J.; Socher, R. Pointer Sentinel Mixture Models. *CoRR* **2016**, *abs/1609.07843*.

[26] Bloom, C. LZP: A New Data Compression Algorithm. Proceedings of the 6th Data Compression Conference (DCC '96), Snowbird, Utah, USA, March 31 - April 3, 1996. 1996; p 425.

[27] Sennrich, R.; Haddow, B.; Birch, A. Neural Machine Translation of Rare Words with Subword Units. *CoRR* **2015**, *abs/1508.07909*.