Breaking the Softmax Bottleneck via Learnable Monotonic Pointwise Non-linearities

Octavian-Eugen Ganea [1 *] **Sylvain Gelly** [2] **Gary Bécigneul** [1] **Aliaksei Severyn** [3]

Abstract

The Softmax function on top of a final linear layer is the de facto method to output probability distributions in neural networks. In many applications such as language models or text generation, this model has to produce distributions over large output vocabularies. Recently, this has been shown to have limited representational capacity due to its connection with the rank bottleneck in matrix factorization. However, little is known about the limitations of Linear-Softmax for quantities of practical interest such as cross entropy or mode estimation, a direction that we explore here. As an efficient and effective solution to alleviate this issue, we propose to learn parametric monotonic functions on top of the logits. We theoretically investigate the rank increasing capabilities of such monotonic functions. Empirically, our method improves in two different quality metrics over the traditional Linear-Softmax layer in synthetic and real language model experiments, adding little time or memory overhead, while being comparable to the more computationally expensive mixture of Softmaxes.

1. Introduction

Most of nowadays deep learning architectures produce a low dimensional data representation that is important both from a computational (parameter reduction) and generalization (less overfitting) perspective. The underlying assumption is that data lies on a small dimensional manifold. These compressed representations are then used for classification or generation. In the discrete case, they are usually fed to a linear layer to produce the so-called "logits", followed by a Softmax function to output a probability distribution over the desired class labels. We refer to this setup as the **Linear-Softmax** layer.

However, there are situations when the output vocabulary or class label set is (much) larger than the dimension of the data embedding. Typical examples are text models such as neural language models (Zaremba et al., 2014) or sequence to sequence generative models (Sutskever et al., 2014; Graves, 2013; Pascanu et al., 2013) for problems such as machine translation (Bahdanau et al., 2015; Cho et al., 2014), text summarization (Chopra et al., 2016; Rush et al., 2015) or conversational agents (Vinyals & Le, 2015). These models need to approximate different distributions over the full large vocabulary of words generally of size $\Theta(10^5)$. Recent work (Yang et al., 2017; Kanai et al., 2018) has revealed that, in these cases, the Linear-Softmax layer has limited representational power. They show the connection between this problem and the classic low-rank matrix factorization framework, concluding that the rank deficiency prevents Linear-Softmax from exactly matching in representation almost all[1] probability distributions.

To address the Softmax bottleneck issue, (Yang et al., 2017) propose to use a mixture of Softmax distributions (MoS) which achieves state-of-the-art language model perplexity on PennTreeBank (PTB) and WikiText2 (WT2) datasets. However, this method has no theoretical guarantees, being also several orders of magnitude more computationally expensive than Linear-Softmax as we show in section 5.

A different model was proposed by (Kanai et al., 2018) that replace the exponential in Softmax by a product between exponential and sigmoid. This model called *Sigsoftmax* can be reformulated as applying the pointwise nonlinearity $ss(x) := 2x - \log(1 + \exp(x))$ to the logits before they are fed to the Softmax function. Unfortunately, there is no theoretical guarantee that Sigsoftmax can convert low-rank to full-rank matrices. In addition, this model raises a few questions that we seek to explore here: i) what other non-linearities are suitable for breaking the Softmax bottleneck? ii) can we theoretically understand and guarantee which pointwise functions will break the Softmax bottleneck by

[*]Work done in an internship at Google Brain. [1]Department of Computer Science, ETH Zürich, Switzerland [2]Google Brain [3]Google Research. Correspondence to: Octavian-Eugen Ganea <octavian.ganea@inf.ethz.ch>, Sylvain Gelly <sylvain-gelly@google.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

[1]Except a subset of measure 0.

increasing the matrix rank? iii) can we efficiently learn a good non-linearity for the task of interest jointly with the rest of the model?

To address all aforementioned issues, we here propose to learn continuous increasing pointwise functions that would beneficially distort the logits before being fed to the Softmax layer. Our model is called **Linear-Monotonic-Softmax (LMS)**. We constrain our functions to be increasing since we want this transformation to be rank preserving, but we theoretically show that if there exists any other pointwise non-linearity to make our matrix full rank, then there is also an increasing continuous and differentiable function with the same property.

We now summarize our contributions:

- We propose the novel **Linear-Monotonic-Softmax (LMS)** model to break the Softmax bottleneck. It generalizes the approach in (Kanai et al., 2018) by learning parametric pointwise increasing functions to optimally distort the logits before feeding them to the final Softmax. Theoretically, we investigate its power to alleviate the rank-deficiency causing the Softmax bottleneck.
- We show insights into the Linear-Softmax bottleneck by analyzing some metrics of practical utility such as cross-entropy or mode matching. Theoretically, we connect the cross entropy minimization of this model and the principle of maximum entropy with linear constraints.
- Empirically, we show that, in a synthetic setting, Linear-Softmax and MoS (Yang et al., 2017) are (sometimes significantly) worse than LMS for cross-entropy minimization or mode matching. In the real task of language modeling, LMS applied to state-of-the-art models improves the test perplexity over vanilla Linear-Softmax (Merity et al., 2017) and Sigsoftmax (Kanai et al., 2018) on standard benchmark datasets, with very little GPU memory or running time overhead, being comparable to the significantly more expensive MoS model.

2. Language Modeling

We first briefly explain a representative task for the Softmax bottleneck problem, namely language modeling (LM). However, this issue concerns any models that produce probability distributions over large output vocabularies.

Language models are the simplest fully unsupervised generative models for natural language text which are actively used to improve state-of-the-art results in various natural language processing tasks (Peters et al., 2018; Devlin et al., 2018). Formally, assume we are given a vocabulary of words in a language $\mathcal{V} = \{x_1, \dots, x_M\}$ and a text corpus represented as a sequence of words $\mathcal{X} = (x_{j_1}, \dots, x_{j_N})$, where typically $N >> M$. We assume that this corpus is generated sequentially from a true conditional next-token distribution $P^*(X_i|X_{i-1}, \dots, X_1) = P^*(X_i|C_i)$, where the context random variable is denoted by $C_i = X_{<i}$ and its outcome by $c_i = x_{j_{<i}}$. The chain rule formula then gives the full corpus likelihood $P^*(X_1, \dots, X_N) = \Pi_{i=1}^N P^*(X_i|C_i)$. Therefore, we view natural language as a set of conditional next-token distributions $\mathcal{S} = \{(c_1, P^*(X|c_1)), \dots, (c_N, P^*(X|c_N))\}$.

The goal of language models is to approximate the true P^* with a parametric distribution Q_θ. Popular and state of the art methods (Takase et al., 2018; Zolna et al., 2017; Yang et al., 2017; Merity et al., 2017; Melis et al., 2017; Krause et al., 2017; Merity et al., 2016; Grave et al., 2016) use recurrent neural networks (RNNs) such as stacked LSTMs (Hochreiter & Schmidhuber, 1997) to represent each context c_i as a vector of fixed dimension d denoted by $\mathbf{h}_i \in R^d$. Words are also embedded in the same continuous space, i.e. word x_j is mapped to vector $\mathbf{w}_j \in \mathbb{R}^d$. Typically $d << M$. The RNN cell is a function that allows to express the context vectors recursively: $\mathbf{h}_{i+1} = RNN(\mathbf{h}_i, \mathbf{w}_i)$. Finally, the conditional probability of the next word in a context is given by the **Linear-Softmax** model which is a standard Softmax function on top of the word-context dot-product logits:

$$Q_\Theta(x_i|c_j) = \frac{\exp(\mathbf{h}_j^\top \mathbf{w}_i)}{\sum_{s=1}^M \exp(\mathbf{h}_j^\top \mathbf{w}_s)} \tag{1}$$

Θ being the model's parameters. Training is done by minimizing the cross entropy (or its exponential, the *perplexity*)

$$\mathcal{L}(\Theta) = \frac{1}{N} \sum_{i=1}^{N} -\log Q_\Theta(x_i|c_j) \tag{2}$$

which is an approximation of the true expected cross entropy

$$\mathcal{L}(\Theta) \approx \mathbb{E}_C \mathbb{E}_X[-\log Q(X|C)] = \mathbb{E}_C[H(P^*, Q|C)] \tag{3}$$

Active LM research focuses on better context embedding models, optimization, long range dependencies or caching techniques. Inspired by (Yang et al., 2017), we here focus on investigating and alleviating the *bottleneck of the Linear-Softmax model*.

3. Softmax Bottleneck - Problem and Insights

Main questions. In the above model of eq. (1) we made the assumption that any (conditional) probability distribution over a large word vocabulary $\mathcal{V}$ can be "well" parameterized by a single low-dimensional vector $\mathbf{h}$ and an exponential family distribution (Linear-Softmax), while also having

access to a set of word embeddings $\{\mathbf{w}_i, i = 1, \dots, M\}$ shared across all data distributions in the real set $\mathcal{S}$:

$$Q_{\mathbf{h}}(x_i) := Q_\Theta(x_i|c) = \frac{\exp(\mathbf{h}^\top \mathbf{w}_i)}{\sum_{i'=1}^{M} \exp(\mathbf{h}^\top \mathbf{w}_{i'})} \quad (4)$$

One question we would like to theoretically and empirically investigate is:
Is one embedding vector $\mathbf{h}$ *enough to fully represent any distribution of interest, i.e. can we always find* Θ *s.t.* $Q_\Theta(X|c) = P^*(X|c)$ *for all distributions of interest* $P^*(\cdot|c) \in \mathcal{S}$ *? If not, how "close" can we get in terms of different interesting metrics (e.g. fitting the logits matrix, cross entropy, mode matching) ?*

We will see that Linear-Softmax is indeed limited. Next, to alleviate this bottleneck, we will introduce the Linear-Monotonic-Softmax (LMS) model and we will take steps in re-investigating the above question.

Connection with Matrix Factorization. We follow the formalism of (Yang et al., 2017) and define the *log-P matrix* associated with any family of conditional probability distributions P over all possible contexts:

$$\mathbf{A}_P \in \mathbb{R}^{M \times N}, \quad (A_P)_{ij} = \log P(x_i|c_j) \quad (5)$$

We further define the context and word matrices:

$$\mathbf{H}_\Theta = \begin{bmatrix} \mathbf{h}_1^\top \\ \mathbf{h}_2^\top \\ \dots \\ \mathbf{h}_N^\top \end{bmatrix} \in \mathbb{R}^{N \times d}, \quad \mathbf{W}_\Theta = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \dots \\ \mathbf{w}_M^\top \end{bmatrix} \in \mathbb{R}^{M \times d} \quad (6)$$

as well as the logits matrix $\mathbf{W}_\Theta \mathbf{H}_\Theta^\top$. Then, one derives that

$$\mathbf{A}_{Q_\Theta} = \mathbf{W}_\Theta \mathbf{H}_\Theta^\top - \mathbf{e}_M \cdot \mathbf{logZ}^\top \quad (7)$$

where $\mathbf{e}_M = \begin{bmatrix} 1 \\ 1 \\ \dots \\ 1 \end{bmatrix} \in \mathbb{R}^M$, and $\mathbf{logZ} = \begin{bmatrix} \log Z_1 \\ \log Z_2 \\ \dots \\ \log Z_N \end{bmatrix} \in \mathbb{R}^N$ is the vector of log-partition functions in eq. (1).

Denoting by $r(\cdot)$ the matrix rank function, one has

$$r(\mathbf{e}_M \cdot \mathbf{logZ}^\top) = 1, \quad r(\mathbf{W}_\Theta \mathbf{H}_\Theta^\top) \leq d, \quad r(\mathbf{A}_{Q_\Theta}) \leq d+1$$

Where the rightmost inequality is proved using a classic rank inequality[2]. Moreover, $r(\mathbf{A}_{Q_\Theta}) \geq d-1$ if $r(\mathbf{W}_\Theta \mathbf{H}_\Theta^\top) = d$, which shows that the log-partition functions cannot change the final rank by more than 1. Since $\mathbf{A}_{P^*}$ is likely of full rank M for real distributions, (Yang et al., 2017) note that $\mathbf{A}_{P^*} \neq \mathbf{A}_{Q_\Theta}$ when $d < M - 1$, meaning that the *Linear-Softmax* model has a representational bottleneck.

[2] $r(\mathbf{B} + \mathbf{C}) \leq r(\mathbf{B}) + r(\mathbf{C}), \forall \mathbf{B}, \mathbf{C}$ matrices of the same dimensions.

Quantifying the Error. The above exposure shows one face of the coin, but, in practice, we might also be interested to know how "bad" this bottleneck can be. This depends on the choice of the distance function between discrete probability distributions. Such functions may be explicitly minimized in order to learn the parametric distribution Q_Θ closest to the true (unknown) data distribution.

a) Mean Squared Error. Assuming we remain in the matrix factorization setting, one natural choice of such a distance is mean square error (MSE):

$$\mathcal{L}_{MSE}(\Theta) = \frac{1}{N} \|\mathbf{A}_{P^*} - \mathbf{A}_{Q_\Theta}\|_F^2 \quad (8)$$

A simple consequence of the Eckart-Young-Mirsky theorem is the following result:

Theorem 1. $\forall \Theta, \|\mathbf{A}_{P^*} - \mathbf{A}_{Q_\Theta}\|_F^2 \geq \sqrt{\sigma_{d+2}^2 + \dots + \sigma_M^2}$ *where* $\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_M$ *are the singular values of matrix* $\mathbf{A}_{P^*}$.

Proof is in appendix A. This result shows that, under the MSE distance, we cannot find a model arbitrary close to our true distribution if we have a rank deficiency on $\mathbf{A}_{Q_\Theta}$. In section 4, we will also investigate the rank deficiency for the Linear-Monotonic-Softmax (LMS) model.

However, the MSE error has a major drawback when used to quantify how well two distributions match: it puts emphasis on matching the tail of the distributions rather then their means or modes. To see this intuitively, we use the inequality:

$$\frac{1}{x+\epsilon} < \frac{\log(x+\epsilon) - \log(x)}{\epsilon} < \frac{1}{x}, \quad \forall x, \epsilon > 0 \quad (9)$$

which, since $\lim_{x \to 0} 1/x = \infty$, shows that mis-matching the small values of $\log P(x)$ incurs a much higher error compared to the large values. This behavior can be highly undesirable in practical settings such as prediction of the most likely next word or class, especially since a wide variety of real-world distributions exhibit a power law (e.g. Zipf's law for natural language (Manning & Schütze, 1999)).

b) Cross Entropy. The most used loss for discrete data is cross entropy, so it is natural to analyze the Softmax bottleneck in terms of its minimum value.

For a single (one context) true distribution $P^*(X)$ and a parametric model $Q_{\mathbf{h}}(x_i) \propto \exp(\langle \mathbf{w}_i, \mathbf{h} \rangle)$ with fixed word embeddings $\mathbf{W}$ and variable (learnable) context vector $\mathbf{h}$, this loss is:

$$H(P^*, Q_{\mathbf{h}}) = \mathbb{E}_{P^*}[-\log Q_{\mathbf{h}}] = -\langle \mathbb{E}_{P^*}[\mathbf{w}], \mathbf{h} \rangle + \log Z^{(\mathbf{h})}$$

where $\mathbb{E}_{P^*}[\mathbf{w}] = \sum_{j=1}^{M} P^*(x_j) \mathbf{w}_j \in \mathbb{R}^d$ and the partition function is $Z^{(\mathbf{h})} := \sum_{j=1}^{M} \exp(\langle \mathbf{w}_j, \mathbf{h} \rangle)$.

A question is: *What is the minimum achievable cross-entropy for a learnable vector* $\mathbf{h} \in \mathbb{R}^d$ *?*

Towards this direction, we make the connection with the Maximum Entropy Principle under Linear Constraints via the following theorem.

Theorem 2. *Let* $H(R) = -\sum_{i=1}^{M} R(x_i) \log R(x_i)$ *be the entropy of the discrete distribution* R*. Then:*

i) $H(P^*, Q) \geq H(P^*)$*, for any probability distribution* Q *(not necessarily from the Linear-Softmax/exponential family).*

ii) $min_{\mathbf{h} \in \mathbb{R}^d} H(P^*, Q_{\mathbf{h}}) = max_{R \in \mathcal{P}^*} H(R)$*, where* $\mathcal{P}^* := \{R |\ R \geq 0,\ \sum_{i=1}^{M} R(x_i) = 1,\ \mathbb{E}_R[\mathbf{w}] = \mathbb{E}_{P^*}[\mathbf{w}]\}$ *is a convex polytope defined by d+1 linear constraints.*

A proof is given in appendix B. One can see that increasing the word embedding dimension and assuming the word embedding matrix W has full rank (i.e. the new constraints cannot be derived from the previous constraints) implies that the polytope $\mathcal{P}^*$ "shrinks", i.e. the maximum entropy becomes lower. Thus, the following hold.

Corollary 2.1. *The minimum achievable cross-entropy* $H(P^*, Q_{\mathbf{h}})$ *becomes lower as the word embedding dimension increases, if W keeps having full rank.*

Corollary 2.2. *If* $d = M$ *and the word embedding matrix W has full rank, then* $\mathcal{P}^* = \{P^*\}$ *and the lowest possible cross entropy is achieved:* $min_{\mathbf{h}} H(P^*, Q_{\mathbf{h}}) = H(P^*)$*.*

c) Mode Matching. In classification or generative models for discrete data we are often interested in predicting the modes of the true data distributions, e.g. the most likely next word given a context. Thus, we hope that a parametric model trained with our loss of choice (e.g. cross entropy) also exhibits a high accuracy at matching the modes. In our setting, this is represented by the success percentage:

$$\frac{1}{N} \#\{j : \arg\max_i P^*(x_i|c_j) = \arg\max_i Q_\Theta(x_i|c_j)\} \tag{10}$$

We will empirically estimate this quantity for a synthetic experiment in section 5.

Breaking the Bottleneck via Mixture of Softmaxes (MoS). (Yang et al., 2017) propose to use a MoS to alleviate this bottleneck. Concretely, they move from single point context embeddings to K embeddings as

$$Q_\Theta^{MoS}(x_i|c_j) = \sum_{k=1}^{K} \pi_{j,k} \frac{\exp(\mathbf{g}_{j,k}^\top \mathbf{w}_i)}{\sum_{s=1}^{M} \exp(\mathbf{g}_{j,k}^\top \mathbf{w}_s)} \tag{11}$$

where $\pi_{j,k} = \frac{\exp(\mathbf{v}_k^\top \mathbf{h}_j)}{\sum_{k'=1}^{K} \exp(\mathbf{v}_{k'}^\top \mathbf{h}_j)}$ are mixture priors, and $\mathbf{g}_{j,k} = \tanh(\mathbf{U}_k \mathbf{h}_j)$ are the K embeddings representing the context j. Here, $\mathbf{v}_k$ and $\mathbf{U}_k$ are the model parameters, shared across all contexts.

While effective and achieving state-of-the-art LM perplexities, this model is several orders of magnitude more expensive than Linear-Softmax, having no theoretical guarantees to the best of our knowledge.

4. Monotonic Pointwise Functions

Our main contribution is to analyze and learn pointwise non-linearities $f(\cdot)$ that would alleviate the Softmax bottleneck. We are thus interested in the **Linear-Monotonic-Softmax (LMS)** layer defined as

$$Q(x_i) = \frac{\exp(f(\mathbf{h}^\top \mathbf{w}_i))}{\sum_{s=1}^{M} \exp(f(\mathbf{h}^\top \mathbf{w}_s))} \tag{12}$$

This model draws inspiration from non-metric multidimensional scaling (Kruskal, 1964a;b). We desire to restrict to pointwise functions that have the following properties:

- *non-linearity*: to break the Softmax bottleneck, i.e. to not limit the rank of $f(\mathbf{W}_\Theta \mathbf{H}_\Theta^\top)$ to d
- *increasing*: to preserve the ranking/order of logits
- *bijectivity on* $\mathbb{R}$: $\lim_{x \to \pm\infty} f(x) = \pm\infty$ to have no obvious limitation in modeling sparse or other distributions
- *continuous and (piecewise) differentiable*: to be learned using backpropagation
- *fast and memory efficient*: to add little overhead compared to Linear-Softmax and unlike MoS

We first note that our model is a generalization of vanilla linear Softmax, which can be recovered by taking the identity function in eq. (12). Another particular example of a function with the above properties is $2x - \log(1 + \exp(x))$. This is the main focus of (Kanai et al., 2018), but here we generalize their approach by investigating and learning generic parametric pointwise increasing functions.

We will show in theorem 3 that the above first 4 conditions are not limiting the expressiveness of our models in terms of matrix rank deficiency. Moreover, in theorem 7 we show that the class of continuous piecewise linear increasing functions is a universal approximator for all differentiable increasing functions with bounded derivative that are defined on a finite interval. Related to the last property, we will explain why these functions are fast and memory efficient.

Notations: For any matrix $\mathbf{A} \in \mathbb{R}^{M \times N}$ and pointwise function $f : \mathbb{R} \to \mathbb{R}$, we denote by $f(\mathbf{A})$ the matrix $\mathbf{B} \in \mathbb{R}^{M \times N}$ with $B_{ij} = f(A_{ij})$. In the case $f(x) = x^p$, we will follow (Amini et al., 2012) and use the notation $\mathbf{A}^{\odot p}$.

We list our main theoretical results for pointwise functions.

How powerful are monotonic pointwise non-linearities? We prove that the conditions imposed above on pointwise f's are not restrictive when concerned about matrix rank increase.

Theorem 3. *Let $\mathbf{A} \in \mathbb{R}^{M \times N}$ be any fixed real matrix of any rank. If there exists a pointwise function $f : \mathbb{R} \to \mathbb{R}$ s.t. $f(\mathbf{A})$ has rank at least K, then there also exists a bijective, continuous, piecewise differentiable and strictly increasing function $g : \mathbb{R} \to \mathbb{R}$ s.t. $g(\mathbf{A})$ has rank at least K.*

Proof is in appendix C.

Making a matrix full-rank via pointwise operators. Theorem 3 shows that we only need to characterize low-rank matrices for which there exists any pointwise operator that increases its rank. In the most useful case, we would like to know when such operators can make it full rank. But, first, we observe that not all matrices can be made full-rank no matter what pointwise function one uses, for example matrices that have the same column repeated, or those that have two columns with constant entries. Next, we state a simple, but practically useful result for our language model formalism. Proof is in appendix D.

Lemma 4. *Let $\mathbf{A} \in \mathbb{R}^{M \times N}, M \leq N$ be any fixed real matrix of rank at most d, i.e. $\mathbf{A} = \mathbf{W}\mathbf{H}^\top$ where $\mathbf{W} \in \mathbb{R}^{M \times d}, \mathbf{H} \in \mathbb{R}^{N \times d}$. Denote by $\mathbf{h}_i$ and $\mathbf{w}_i$ the i-th rows in $\mathbf{H}$ and $\mathbf{W}$. If one can find M distinct rows $j_1, \dots, j_M$ in $\mathbf{H}$ s.t. the values $\langle \mathbf{w}_i, \mathbf{h}_{j_i} \rangle$ are distinct from all the other entries of matrix $\mathbf{A}$, then there exists a pointwise function $f : \mathbb{R} \to \mathbb{R}$ s.t. $f(\mathbf{A})$ has full rank M.*

Next, we focus on simple power operators $\mathbf{A}^{\odot p}$ and cite a previous result that shows a limitation: small p values cannot make the matrix rank arbitrarily large.

Theorem 5. *(Amini et al., 2012) Let $\mathbf{A} \in \mathbb{R}^{N \times M}$ be a rank d matrix. Let p be any positive integer. Then*

$$r(\mathbf{A}^{\odot p}) \leq \min \left\{ N, M, \binom{d+p-1}{p} \right\} \tag{13}$$

However, $\lim_{p \to \infty} \binom{d+p-1}{p} = \infty, \forall d > 1$, so there is still hope we can find monomials that make a matrix full rank if we look at sufficiently large powers. The following novel result proved in appendix E confirms in a particular case that this is almost surely achieved. Let $O_k^N = \{\mathbf{A} \in \mathbb{R}^{N \times N} : r(\mathbf{A}) = k\}$ be the submanifold of $\mathbb{R}^{N \times N}$ consisting of rank k matrices.

Theorem 6. *For $N > 1$, the pointwise function $f(x) = x^2$ makes matrices in O_{N-1}^N to almost surely become full rank.*

Architecture(s) of Learnable Monotonic Functions. Even though some particular pointwise functions such as monomials/polynomials can make a low-rank matrix to be full-rank and, thus, remove the rank deficiency bottleneck, it is not guaranteed that this new matrix is close to the true data matrix. For this reason, we propose to learn parametric pointwise functions together with our model. For the reasons previously described, we design these functions to be bijective on full $\mathbb{R}$, increasing, continuous and (almost everywhere) differentiable. We note that the problem of learning parametric monotonic functions was analyzed by (Sill, 1998), but here we use the architecture proposed in (Daniels & Velikova, 2010), namely:

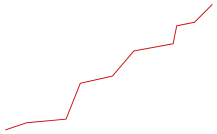

Figure 1. Example of an increasing continuous piecewise linear function.

$$f(x) = \sum_{i=1}^{K} v_i \sigma(u_i x + b_i) + b \tag{14}$$

where $u_i, v_i, b_i, b \in \mathbb{R}, u_i, v_i \geq 0, \forall i \in \{1, \dots, K\}$. This corresponds to an one hidden layer neural network with K hidden units and positively constraint weights (but not the biases). (Daniels & Velikova, 2010) prove that this class of functions is universal approximator for all continuous increasing functions. However, in practice, one has to use a large number of hidden units K in order to achieve a good approximation. While for our synthetic experiments in section 5 this was not an issue, for the real language modeling experiments this results in a heavy computational overhead. To understand why, in language modeling one has to process at a time large minibatches of contexts, meaning that matrices of size $N \times M$ [3] have to be stored in the GPU memory. If one uses the above architecture or the MoS architecture (Yang et al., 2017), one has to store in the GPU memory and process intermediate tensors of dimension $N \times M \times K$, which can result in a significant running time and memory overhead even for small values of K such as 10 or 15. This may lead to smaller batch sizes and thus impede the model's scalability.

To address the above computation problem, we propose to use an efficient class of parametric piecewise linear increasing functions called **PLIF = P**iecewise **L**inear **I**ncreasing **F**unctions. An example is shown in fig. 1. Formally, we fix a (large enough) interval $[-T, T]$ and K + 1 equally distanced knots in this interval: $l_i = -T + \frac{2Ti}{K}, \forall 0 \leq i \leq K$. We define our function to be piecewise linear, meaning that $f(x) = s_i x + b_i, \forall x \in [l_i, l_{i+1}]$, where s_i is the

[3] M is the vocabulary size, N is the number of contexts in a minibatch.

slope of the linear function on the interval $[l_i, l_{i+1}]$. To enforce monotonicity, we need $s_i > 0$ which is achieved using the parametric form $s_i = \log(1 + \exp(v_i))$, where v_i are unconstrained parameters. Moreover, we need the function to be continuous, meaning that f has the same value in each knot l_i. This is achieved iff $\forall i > 0, b_i = b_0 + s_0 l_0 - l_i s_i + \frac{2T}{K} \sum_{j=0}^{i-1} s_j$. Thus, this model has as (learnable) parameters the values s_i and the initial bias b_0.

Computational Efficiency of PLIF. The above formulation of the PLIF model is computationally efficient: i) a forward pass for an input x is done by converting x to the index $i_x := \lfloor (x + T)\frac{K}{2T} \rfloor$, doing two lookups for index i_x in the vectors $\mathbf{s} := \{s_i : 0 \leq i \leq K\}$ and in $cumsum(\mathbf{s})$, and then returning the value $s_{i_x} x + b_{i_x}$. ii) The backward pass only updates s_{i_x} and $cumsum(\mathbf{s})_{i_x}$, which have efficient implementations. Thus, the additional running time is negligible, while the additional memory only consists of two K-dimensional vectors and does not depend on the mini-batch size like MoS or the model in eq. (14) do. For these reasons, we are computationally able to use large values of K (e.g. $10^5 - 10^6$) which offers great flexibility in modeling highly non-linear functions.

Universal Approximation Property of PLIF. From a theoretical perspective, we state the following result and prove it in appendix F:

Theorem 7. *The PLIF model with large enough number of knots K can approximate arbitrarily well any differentiable increasing real function defined on [-T,T] that has bounded derivatives.*

5. Experiments

We empirically assess Linear-Softmax, Linear-Monotonic-Softmax (LMS) and Mixture of Softmaxes (MoS).

5.1. Synthetic Experiments

We first explore a synthetic experimental setting that has the following advantages:

- allows to separate the Softmax bottleneck from other bottlenecks, e.g. in the RNN context embedding layer.
- allows to understand how powerful these models are to represent very different distributions using single low dimensional vectors, i.e. we remove the dependency between context vectors that happens when embedding contexts with a shared neural network.
- allows to evaluate how well the modes of the true and the parametric distributions match, a metric of practical importance (e.g. for text generative models) that can be quantified only when given access to the true data distribution.

To this end, we repeatedly sample N different "true" discrete distributions over a fixed synthetic word vocabulary of size M. We use a Dirichlet prior with all concentration parameters equal to α:

$$P^*(\cdot|c_j) \sim \text{Dir}(\alpha), \text{ for } j = 1, \ldots, N \tag{15}$$

Larger α's result in close to uniform distributions, while low values result in sparse distributions. The effect of α is also shown for different values of M in fig. 5 from appendix G.

We learn parametric models $Q_\Theta(\cdot|c_j)$ to match the true P^* distributions. We learn a set of word embeddings shared across all contexts and a separate context embedding per each distribution $Q_\Theta(\cdot|c_j)$. All embeddings have dimension D. We use the Linear-Softmax model as defined by eq. (1), the Mixture of Softmaxes (MoS) model (Yang et al., 2017), and our LMS model given by eq. (12) with a pointwise monotonic function parameterized using the K hidden units architecture shown in eq. (14). Learning the models' parameters is done by minimizing the cross entropy which is equivalent to minimizing the divergence $KL(P^*||Q_\Theta)$ for each context c_j.

Results. We present the results for different Dirichlet parameter α, vocabulary sizes M, embedding sizes D and evaluation metrics (mode matching and cross entropy / KL divergence) in figs. 2 and 3, but also show additional results in appendix H in figs. 6 to 8. In all the settings, we used $N = 10^5$ contexts, where N is the number of different distributions $P^*(\cdot|c_j)$.

Discussion. We observe that, in most of the presented cases, LMS outperforms Linear-Softmax and MoS on both the task of mode matching and on the minimum achievable cross-entropy (KL divergence). Especially in the "low D - large M" setting, the difference is significantly large showcasing the existence of the Softmax bottleneck and the merits of our LMS model.

However, we note that there is still room for future work and improvements, for example mode matching still largely suffers for low D and large M.

5.2. Language Model Experiments

We move to the real setting of language modeling. Here, due to computational reasons discussed in section 4, we will use our PLIF architecture introduced in the same section.

Datasets. Following previous work (Mikolov; Inan et al., 2016; Kim et al., 2016; Zoph & Le, 2016), we use the two most popular LM datasets: Penn TreeBank (Mikolov et al.,

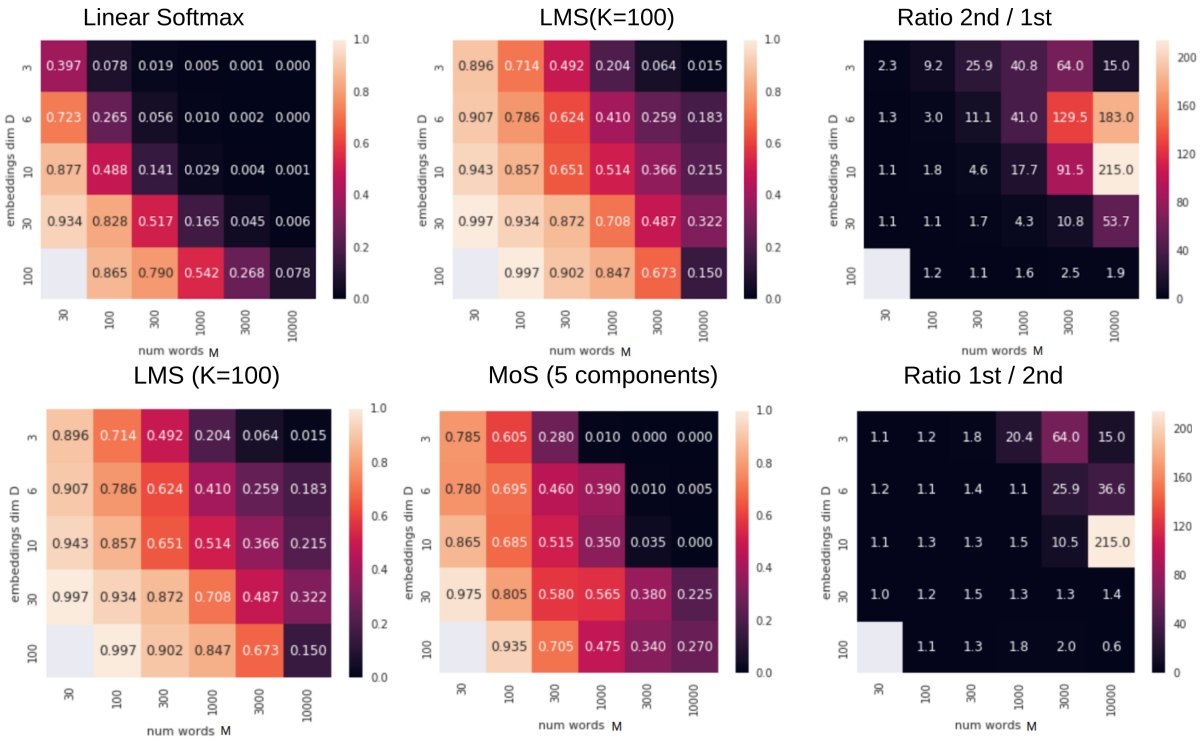

Figure 2. Percentage of contexts j for which the modes of true and parametric distributions match, i.e $\arg\max_i P^*(x_i|c_j) = \arg\max_i Q_\Theta(x_i|c_j)$. Higher the better. Dirichlet concentration $\alpha = 0.1$.

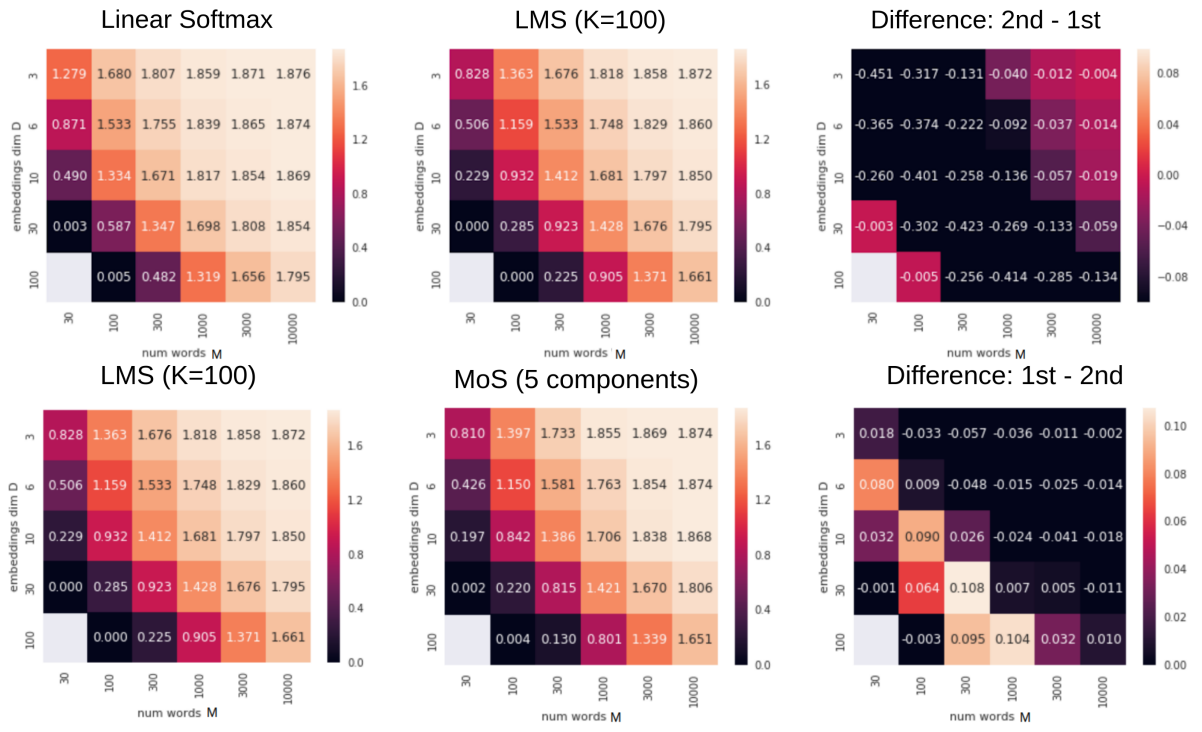

Figure 3. Average $KL(P^*||Q_\Theta)$ (across all contexts). Lower the better. Dirichlet concentration $\alpha = 0.1$.

Table 1. Single model perplexities on validation and test sets on Penn Treebank and WikiText-2 datasets. For a fair comparison, baseline results are obtained by running the respective open-source implementations locally, however, being comparable to the published results. We also show the training time per epoch when using a single Tesla P100 GPU.

	PENN TREEBANK				WIKITEXT-2			
	#PARAM	VALID PPL	TEST PPL	#SEC/EP	#PARAM	VALID PPL	TEST PPL	#SEC/EP
LINEAR-SOFTMAX W/ AWD-LSTM, W/O FINETUNE (MERITY ET AL., 2017)	24.2M	60.83	58.37	∼60	33M	68.11	65.22	∼120
OURS LMS-PLIF, 10^5 KNOTS W/ AWD-LSTM, W/O FINETUNE	24.4M	59.45	57.25	∼70	33.2M	67.87	64.86	∼150
MOS, K = 15 W/ AWD-LSTM, W/O FINETUNE (YANG ET AL., 2017)	26.6M	58.58	56.43	∼150	33M	66.01	63.33	∼550
MOS(15 COMP) + OUR PLIF (10^6 KNOTS) W/ AWD-LSTM, W/O FINETUNE	28.6M	58.20	56.02	∼220	-	-	-	-

2010) and WikiText-2 (Merity et al., 2016). These datasets have word vocabulary sizes of 10,000 and 33,000.

Baselines. We integrate our PLIF layer on top of the state of the art language models of AWD-LSTM (Merity et al., 2017) and AWD-LSTM+MoS (Yang et al., 2017). Additionally, our PLIF architecture can also be combined with MoS instead of standard Softmax. We call this model "MoS + PLIF". We use the AWD-LSTM open source implementation [4]. All the models in table 1 [5] were ran locally and we report these results; we did this to understand how different Softmax models compare with each other when using the exact same context embedding architecture. We note that (Yang et al., 2017) redo architecture search after integrating their MoS model, their goal being to reduce the number of parameters to the same size as AWD-LSTM.

All the models are trained without finetuning (Merity et al., 2017). We use embedding dimension 400 for all the models in table 1. For optimization, we use the strategy described in (Merity et al., 2017) consisting of running stochastic gradient descent (SGD) with constant learning rate (20.0) until the cross entropy loss starts stabilizing, and then switching to averaged SGD. This strategy was shown to improve state of the art language models (Takase et al., 2018) and to consistently and by a large margin outperform popular adaptive methods such as ADAM (Kingma & Ba, 2015).

We did not include the Sigsoftmax baseline model as no significant improvement over Linear-Softmax was seen, neither locally nor in the original paper (Kanai et al., 2018) (the w/o finetune setting). We note that this method is a particular case of our LMS model.

Results and Discussion. Table 1 shows the results. Our LMS-PLIF layer consistently improves over Linear-Softmax when combined with the same state-of-the-art AWD-LSTM context embedding architecture. The computational prices (memory and training time) we pay for using LMS-PLIF are negligible compared to Linear-Softmax. However, while MoS outperforms our simple LMS-PLIF model, it is computationally several orders of magnitude more expensive, which is a practical advantage of our method. Finally, combining MoS and our PLIF model gives the best Penn TreeBank result, outperforming all baselines (but at the highest computational cost).

[4] http://github.com/salesforce/awd-lstm-lm

[5] Except MoS on WT2 which took too long to run on a single GPU, thus reporting the published results.

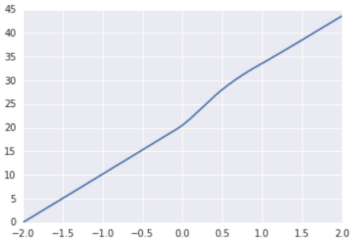

Figure 4. Learned function for the model in table 2.

Table 2. Statistics on the slope values of the PLIF pointwise function trained on WikiText-2.

MEAN	STD	MIN	MAX
1.10	0.62	0.02	5.16

We show in table 2 statistics of the slope values of a learned PLIF function, revealing its highly non-linear nature.

6. Conclusion

We re-analyzed the Softmax bottleneck here from multiple perspectives and confirmed, both theoretically and empirically, that the widely used Softmax layer is not flexible enough to model arbitrarily distributions over large vocabularies. We proposed LMS-PLIF, a model that learns parametric monotonic functions to make Softmax more flexible, and show some of its capabilities.

Acknowledgements

We thank Josip Djolonga for useful discussions and anonymous reviewers for suggestions.

Octavian Ganea is funded by the Swiss National Science Foundation (SNSF) under grant agreement number 167176. Gary Bécigneul is funded by the Max Planck ETH Center for Learning Systems.

References

Amini, A., Karbasi, A., and Marvasti, F. Low-rank matrix approximation using point-wise operators. *IEEE Transactions on Information Theory*, 58(1):302–310, 2012.

Bahdanau, D., Cho, K., and Bengio, Y. Neural machine translation by jointly learning to align and translate. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Chopra, S., Auli, M., and Rush, A. M. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 93–98, 2016.

Daniels, H. and Velikova, M. Monotone and partially monotone neural networks. *IEEE Transactions on Neural Networks*, 21(6):906–917, 2010.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Grave, E., Joulin, A., and Usunier, N. Improving neural language models with a continuous cache. *arXiv preprint arXiv:1612.04426*, 2016.

Graves, A. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Inan, H., Khosravi, K., and Socher, R. Tying word vectors and word classifiers: A loss framework for language modeling. *arXiv preprint arXiv:1611.01462*, 2016.

Kanai, S., Fujiwara, Y., Yamanaka, Y., and Adachi, S. Sigsoftmax: Reanalysis of the softmax bottleneck. *Advances in Neural Information Processing Systems*, 2018.

Kim, Y., Jernite, Y., Sontag, D., and Rush, A. M. Character-aware neural language models. 2016.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2015.

Krause, B., Kahembwe, E., Murray, I., and Renals, S. Dynamic evaluation of neural sequence models. *arXiv preprint arXiv:1709.07432*, 2017.

Kruskal, J. B. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, 1964a.

Kruskal, J. B. Nonmetric multidimensional scaling: a numerical method. *Psychometrika*, 29(2):115–129, 1964b.

Manning, C. D. and Schütze, H. Foundations of statistical natural language processing. 1999.

Melis, G., Dyer, C., and Blunsom, P. On the state of the art of evaluation in neural language models. *arXiv preprint arXiv:1707.05589*, 2017.

Merity, S., Xiong, C., Bradbury, J., and Socher, R. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.

Merity, S., Keskar, N. S., and Socher, R. Regularizing and optimizing lstm language models. *arXiv preprint arXiv:1708.02182*, 2017.

Mikolov, T. Statistical language models based on neural networks.

Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., and Khudanpur, S. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pp. 1310–1318, 2013.

Peters, M., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pp. 2227–2237, 2018.

Rush, A. M., Chopra, S., and Weston, J. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Shalit, U., Weinshall, D., and Chechik, G. Online learning in the embedded manifold of low-rank matrices. *Journal of Machine Learning Research*, 13(Feb):429–458, 2012.

Sill, J. Monotonic networks. In *Advances in neural information processing systems*, pp. 661–667, 1998.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems (NIPS)*, pp. 3104–3112, 2014.

Takase, S., Suzuki, J., and Nagata, M. Direct output connection for a high-rank language model. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4599–4609, 2018.

Vinyals, O. and Le, Q. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

Yang, Z., Dai, Z., Salakhutdinov, R., and Cohen, W. W. Breaking the softmax bottleneck: A high-rank rnn language model. *International Conference on Learning Representations 2018*, 2017.

Zaremba, W., Sutskever, I., and Vinyals, O. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

Zolna, K., Arpit, D., Suhubdy, D., and Bengio, Y. Fraternal dropout. *arXiv preprint arXiv:1711.00066*, 2017.

Zoph, B. and Le, Q. V. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.