When and why are log-linear models self-normalizing?

Jacob Andreas and **Dan Klein**
Computer Science Division
University of California, Berkeley
{jda,klein}@cs.berkeley.edu

Abstract

Several techniques have recently been proposed for training "self-normalized" discriminative models. These attempt to find parameter settings for which unnormalized model scores approximate the true label probability. However, the theoretical properties of such techniques (and of self-normalization generally) have not been investigated. This paper examines the conditions under which we can expect self-normalization to work. We characterize a general class of distributions that admit self-normalization, and prove generalization bounds for procedures that minimize empirical normalizer variance. Motivated by these results, we describe a novel variant of an established procedure for training self-normalized models. The new procedure avoids computing normalizers for most training examples, and decreases training time by as much as factor of ten while preserving model quality.

1 Introduction

This paper investigates the theoretical properties of log-linear models trained to make their unnormalized scores approximately sum to one.

Recent years have seen a resurgence of interest in log-linear approaches to language modeling. This includes both conventional log-linear models (Rosenfeld, 1994; Biadsy et al., 2014) and neural networks with a log-linear output layer (Bengio et al., 2006). On a variety of tasks, these LMs have produced substantial gains over conventional generative models based on counting n-grams. Successes include machine translation (Devlin et al., 2014) and speech recognition (Graves et al., 2013). However, log-linear LMs come at a significant cost for computational efficiency. In order to output a well-formed probability distribution over words, such models must typically calculate a normalizing constant whose computational cost grows linearly in the size of the vocabulary.

Fortunately, many applications of LMs remain well-behaved even if LM scores do not actually correspond to probability distributions. For example, if a machine translation decoder uses output from a pre-trained LM as a feature inside a larger model, it suffices to have all output scores on approximately the same scale, even if these do not sum to one for every LM context. There has thus been considerable research interest around training procedures capable of ensuring that unnormalized outputs for every context are "close" to a probability distribution. We are aware of at least two such techniques: noise-contrastive estimation (NCE) (Vaswani et al., 2013; Gutmann and Hyvärinen, 2010) and explicit penalization of the log-normalizer (Devlin et al., 2014). Both approaches have advantages and disadvantages. NCE allows fast training by dispensing with the need to ever compute a normalizer. Explicit penalization requires full normalizers to be computed during training but parameterizes the relative importance of the likelihood and the "sum-to-one" constraint, allowing system designers to tune the objective for optimal performance.

While both NCE and explicit penalization are observed to work in practice, their theoretical properties have not been investigated. It is a classical result that empirical minimization of *classification error* yields models whose predictions generalize well. This paper instead investigates a notion of *normalization error*, and attempts to understand the conditions under which unnormalized model scores are a reliable surrogate for probabilities. While language modeling serves as a motivation and running example, our results apply to any log-linear model, and may be of general use for efficient classification and decoding.

Our goals are twofold: primarily, to provide intuition about how self-normalization works, and why it behaves as observed; secondarily, to back these intuitions with formal guarantees, both about classes of normalizable distributions and parameter estimation procedures. The paper is built around two questions:

Human Language Technologies: The 2015 Annual Conference of the North American Chapter of the ACL, pages 244–249,
Denver, Colorado, May 31 – June 5, 2015. ©2015 Association for Computational Linguistics

When can self-normalization work—for which distributions do good parameter settings exist? And *why* should self-normalization work—how does variance of the normalizer on held-out data relate to variance of the normalizer during training? Analysis of these questions suggests an improvement to the training procedure described by Devlin et al., and we conclude with empirical results demonstrating that our new procedure can reduce training time for self-normalized models by an order of magnitude.

2 Preliminaries

Consider a log-linear model of the form

$$p(y|x;\boldsymbol{\theta}) = \frac{\exp\{\theta_y^\top x\}}{\sum_{y'} \exp\{\theta_{y'}^\top x\}} \tag{1}$$

We can think of this as a function from a *context* x to a probability distribution over *decisions* y_i, where each decision is parameterized by a weight vector θ_y.[1] For concreteness, consider a language modeling problem in which we are trying to predict the next word after the context *the ostrich*. Here x is a vector of features on the context (e.g. $x = \{$`1-2=the_ostrich, 1=the, 2=ostrich,` $\ldots\}$), and y ranges over the full vocabulary (e.g. $y_1 =$ `the`, $y_2 =$ `runs`, $\ldots$).

Our analysis will focus on the standard log-linear case, though later in the paper we will also relate these results to neural networks. We are specifically concerned with the behavior of the normalizer or *partition function*

$$Z(x;\boldsymbol{\theta}) \overset{\text{def}}{=} \sum_y \exp\{\theta_y^\top x\} \tag{2}$$

and in particular with choices of $\boldsymbol{\theta}$ for which $Z(x;\boldsymbol{\theta}) \approx 1$ *for most* x.

To formalize the questions in the title of this paper, we introduce the following definitions:

Definition 1. A log-linear model $p(y|x,\boldsymbol{\theta})$ is *normalized* with respect to a set $\mathcal{X}$ if for every $x \in \mathcal{X}$, $Z(x;\boldsymbol{\theta}) = 1$. In this case we call $\mathcal{X}$ *normalizable* and $\boldsymbol{\theta}$ *normalizing*.

Now we can state our questions precisely: What distributions are normalizable? Given data points from a normalizable $\mathcal{X}$, how do we find a normalizing $\boldsymbol{\theta}$?

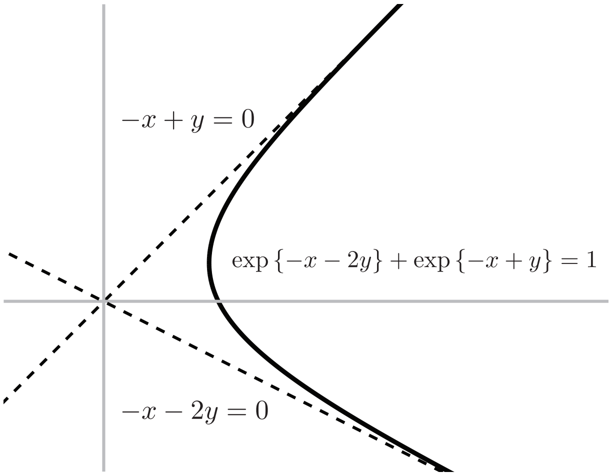

Figure 1: A normalizable set, the solutions $[x, y]$ to $Z([x,y]; \{[-1,1],[-1,-2]\}) = 1$. The set forms a smooth one-dimensional manifold bounded on either side by the hyperplanes normal to $[-1,1]$ and $[-1,-2]$.

In sections 3 and 4, we do not analyze whether the setting of $\boldsymbol{\theta}$ corresponds to a good classifier—only a good normalizer. In practice we require both good normalization *and* good classification; in section 5 we provide empirical evidence that both are achievable.

Some notation: Weight vectors θ (and feature vectors x) are d-dimensional. There are k output classes, so the total number of parameters in $\boldsymbol{\theta}$ is kd. $||\cdot||_p$ is the ℓ_p vector norm, and $||\cdot||_\infty$ specifically is the max norm.

3 When should self-normalization work?

In this section, we characterize a large class of datasets (i.e. distributions $p(y|x)$) that are normalizable either exactly, or approximately in terms of their marginal distribution over *contexts* $p(x)$. We begin by noting simple features of Equation 2: it is convex in x, so in particular its level sets enclose convex regions, and are manifolds of lower dimension than the embedding space.

As our definition of normalizability requires the existence of a normalizing $\boldsymbol{\theta}$, it makes sense to begin by fixing $\boldsymbol{\theta}$ and considering contexts x for which it is normalizing.

Observation. *Solutions x to $Z(x;\boldsymbol{\theta}) = 1$, if any exist, lie on the* boundary *of a convex region in* $\mathbb{R}^d$.

[1] An alternative, equivalent formulation has a single weight vector and a feature function from contexts and decisions onto feature vectors.

This follows immediately from the definition of a convex function, but provides a concrete example of a set for which $\boldsymbol{\theta}$ is normalizing: the solution set of $Z(x; \boldsymbol{\theta}) = 1$ has a simple geometric interpretation as a particular kind of smooth surface. An example is depicted in Figure 1.

We cannot expect real datasets to be this well behaved, so seems reasonable to ask whether "good-enough" self-normalization is possible for datasets (i.e. distributions $p(x)$) which are only close to some exactly normalizable distribution.

Definition 2. A context distribution $p(x)$ is *D-close* to a set $\mathcal{X}$ if

$$\mathbb{E}_p \left[\inf_{x^* \in \mathcal{X}} ||X - x^*||_\infty \right] = D \tag{3}$$

Definition 3. A context distribution $p(x)$ is *ε-approximately normalizable* if $\mathbb{E}_p |\log Z(X; \boldsymbol{\theta})| \leq \varepsilon$.

Theorem 1. *Suppose $p(x)$ is D-close to $\{x : Z(x; \boldsymbol{\theta}) = 1\}$, and each $||\theta_i||_\infty \leq B$. Then $p(x)$ is dBD-approximately normalizable.*

Proof sketch.[2] Represent each X as $X^* + X^-$, where X^* solves the optimization problem in Equation 3. Then it is possible to bound the normalizer by $\log \exp \{\tilde{\theta}^\top X^-\}$, where $\tilde{\theta}$ maximizes the magnitude of the inner product with X^- over $\boldsymbol{\theta}$.

In keeping with intuition, data distributions that are close to normalizable sets are themselves approximately normalizable on the same scale.[3]

4 Why should self-normalization work?

So far we have given a picture of what approximately normalizable distributions look like, but nothing about how to find normalizing $\boldsymbol{\theta}$ from training data in practice. In this section we prove that any procedure that causes training contexts to approximately normalize will also have log-normalizers close to zero in unseen contexts. As noted in the introduction, this does not follow immediately from corresponding results for *classification* with log-linear models. While the two problems are related (it would be quite surprising to have uniform convergence for classification but not normalization), we nonetheless have a different function class and a different loss, and need new analysis.

Theorem 2. *Consider a sample $(X_1, X_2, \dots)$, with all $||X||_\infty \leq R$, and $\boldsymbol{\theta}$ with each $||\theta_i||_\infty \leq B$. Additionally define $\hat{\mathcal{L}} = \frac{1}{n} \sum_i |\log Z(X_i)|$ and $\mathcal{L} = \mathbb{E}|\log Z(X)|$. Then with probability $1 - \delta$,*

$$|\hat{\mathcal{L}} - \mathcal{L}| \leq 2\sqrt{\frac{dk(\log dBR + \log n) + \log \frac{1}{\delta}}{2n}} + \frac{2}{n} \tag{4}$$

Proof sketch. Empirical process theory provides standard bounds of the form of Equation 4 (Kakade, 2011) in terms of the size of a *cover* of the function class under consideration (here $Z(\cdot; \boldsymbol{\theta})$). In particular, given some α, we must construct a finite set of $\hat{Z}(\cdot; \theta)$ such that some $\hat{Z}$ is everywhere a distance of at most α from every Z. To provide this cover, it suffices to provide a cover $\hat{\boldsymbol{\theta}}$ for $\boldsymbol{\theta}$. If the $\hat{\boldsymbol{\theta}}$ are spaced at intervals of length D, the size of the cover is $(B/D)^{kd}$, from which the given bound follows.

This result applies uniformly across choices of $\boldsymbol{\theta}$ regardless of the training procedure used—in particular, $\boldsymbol{\theta}$ can be found with NCE, explicit penalization, or the variant described in the next section.

As hoped, sample complexity grows as the number of features, and not the number of contexts. In particular, skip-gram models that treat context words independently will have sample efficiency multiplicative, rather than exponential, in the size of the conditioning context. Moreover, if some features are correlated (so that data points lie in a subspace smaller than d dimensions), similar techniques can be used to prove that sample requirements depend only on this effective dimension, and not the true feature vector size.

We emphasize again that this result says nothing about the *quality* of the self-normalized model (e.g. the likelihood it assigns to held-out data). We defer a theoretical treatment of that question to future work. In the following section, however, we provide experimental evidence that self-normalization does not significantly degrade model quality.

5 Applications

As noted in the introduction, previous approaches to learning approximately self-normalizing distributions have either relied on explicitly computing the

[2] Full proofs of all results may be found in the Appendix.

[3] Here (and throughout) it is straightforward to replace quantities of the form dB with B by working in ℓ_2 instead of ℓ_∞.

normalizer for each training example, or at least keeping track of an estimate of the normalizer for each training example.

Our results here suggest that it should be possible to obtain approximate self-normalizing behavior without *any* representation of the normalizer on some training examples—as long as a sufficiently large fraction of training examples are normalized, then we have some guarantee that with high probability the normalizer will be close to one on the remaining training examples as well. Thus an unnormalized likelihood objective, coupled with a penalty term that looks at only a small number of normalizers, might nonetheless produce a good model. This suggests the following:

$$l(\boldsymbol{\theta}) = \sum_i \theta_{y_i}^\top x_i + \frac{\alpha}{\gamma} \sum_{h \in H} (\log Z(x_h; \boldsymbol{\theta}))^2 \qquad (5)$$

where the parameter α controls the relative importance of the self-normalizing constraint, H is the set of indices to which the constraint should be applied, and γ controls the size of H, with $|H| = \lceil n\gamma \rceil$. Unlike the objective used by Devlin et al. (2014) most examples are *never* normalized during training. Our approach combines the best properties of the two techniques for self-normalization previously discussed: like NCE, it does not require computation of the normalizer on all training examples, but like explicit penalization it allows fine-grained control over the tradeoff between the likelihood and the quality of the approximation to the normalizer.

We evaluate the usefulness of this objective with a set of small language modeling experiments. We train a log-linear LM with features similar to Biadsy et al. (2014) on a small prefix of the Europarl corpus of approximately 10M words.[4] We optimize the objective in Equation 5 using Adagrad (Duchi et al., 2011). The normalized set H is chosen randomly for each new minibatch. We evaluate using two metrics: BLEU on a downstream machine translation task, and *normalization risk R*, the average magnitude of the log-normalizer on held-out data. We measure the response of our training to changes in γ and α. Results are shown in Table 1 and Table 2.

	Normalized fraction (γ)				
	0	0.001	0.01	0.1	1
R_{train}	22.0	1.7	1.5	1.5	1.5
R_{test}	21.6	1.7	1.5	1.5	1.5
BLEU	1.5	19.1	19.2	20.0	20.0

Table 1: Result of varying normalized fraction γ, with $\alpha = 1$. When no normalization is applied, the model's behavior is pathological, but when normalizing only a small fraction of the training set, performance on the downstream translation task remains good.

	Normalization strength (α)			
α	0.01	0.1	1	10
R_{train}	20.4	9.7	1.5	0.5
R_{test}	20.1	9.7	1.5	0.5
BLEU	1.5	2.6	20.0	16.9

Table 2: Result of varying normalization parameter α, with $\gamma = 0.1$. Normalization either too weak or too strong results in poor performance on the translation task, emphasizing the importance of training procedures with a tunable normalization parameter.

Table 1 shows that with small enough α, normalization risk grows quite large. Table 2 shows that forcing the risk closer to zero is not necessarily desirable for a downstream machine translation task. As can be seen, no noticeable performance penalty is incurred when normalizing only a tenth of the training set. Performance gains are considerable: setting $\gamma = 0.1$, we observe a roughly tenfold speedup over $\gamma = 1$.

On this corpus, the original training procedure of Devlin et al. with $\alpha = 0.1$ gives a BLEU score of 20.1 and R_{test} of 2.7. Training time is equivalent to choosing $\gamma = 1$, and larger values of α result in decreased BLEU, while smaller values result in significantly increased normalizer risk. Thus we see that we can achieve smaller normalizer variance and an order-of-magnitude decrease in training time with a loss of only 0.1 BLEU.

6 Relation to neural networks

Our discussion has focused on log-linear models. While these can be thought of as a class of single-layer neural networks, in practice much of the demand for fast training and querying of log-linear LMs

[4] This prefix was chosen to give the fully-normalized model time to finish training, allowing a complete comparison. Due to the limited LM training data, these translation results are far from state-of-the-art.

comes from deeper networks. All of the proof techniques used in this paper can be combined straightforwardly with existing tools for covering the output spaces of neural networks (Anthony and Bartlett, 2009). If optimization of the self-normalizing portion of the objective is deferred to a post-processing step after standard (likelihood) training, and restricted to parameters in the output layers, then Theorem 2 applies exactly.

7 Conclusion

We have provided both qualitative and formal characterizations of "self-normalizing" log-linear models, including what we believe to be the first theoretical guarantees for self-normalizing training procedures. Motivated by these results, we have described a novel objective for training self-normalized log-linear models, and demonstrated that this objective achieves significant performance improvements without a decrease in the quality of the models learned.

A Quality of the approximation

Proof of Theorem 1. Using the definitions of X^*, X^- and $\tilde{\theta}$ given in the proof sketch for Theorem 1,

$$\begin{aligned}
&\mathbb{E}|\log(\textstyle\sum \exp\{\theta_i^\top X\})| \\
&\quad = \mathbb{E}|\log(\textstyle\sum \exp\{\theta_i^\top (X^* + X^-)\})| \\
&\quad \leq \mathbb{E}|\log(\exp\{\tilde{\theta}^\top X^-\} \textstyle\sum \exp\{\theta_i^\top X^*\})| \\
&\quad \leq \mathbb{E}|\log(\exp\{\tilde{\theta}^\top X^-\})| \\
&\quad \leq dDB \qquad \square
\end{aligned}$$

B Generalization error

Lemma 3. *For any $\boldsymbol{\theta}_1$, $\boldsymbol{\theta}_2$ with $||\theta_{1,i} - \theta_{2,i}||_\infty \leq D \overset{\text{def}}{=} \alpha/dR$ for all i,*

$$||\log Z(x;\boldsymbol{\theta}_1)| - |\log Z(x;\boldsymbol{\theta}_2)|| \leq \alpha \qquad (6)$$

Proof.

$$\begin{aligned}
&||\log Z(x;\boldsymbol{\theta}_1)| - |\log Z(x;\boldsymbol{\theta}_2)|| \\
&\quad \leq |\log Z(x;\boldsymbol{\theta}_1) - \log Z(x;\boldsymbol{\theta}_2)| \\
&\quad \leq \log \frac{Z(x;\boldsymbol{\theta}_1)}{Z(x;\boldsymbol{\theta}_2)} \qquad \text{(w.l.o.g.)} \\
&\quad = \log \frac{\sum_i \exp\left\{(\theta_{1i} - \theta_{2i})^\top x\right\} \exp\left\{\theta_{2i}^\top x\right\}}{\sum_i \exp\left\{\theta_{2i}^\top x\right\}} \\
&\quad \leq dDR + \log \frac{Z(x;\boldsymbol{\theta}_2)}{Z(x;\boldsymbol{\theta}_2)} = \alpha \qquad \square
\end{aligned}$$

Corollary 4. *The set of partition functions $\mathcal{Z} = \{Z(\cdot;\boldsymbol{\theta}) : ||\theta||_\infty \leq B \; \forall \theta \in \boldsymbol{\theta}\}$ can be covered on on the ℓ_∞ ball of radius R by a grid of $\hat{\boldsymbol{\theta}}$ with distance D. The size of this cover is*

$$|\hat{\mathcal{Z}}| = \left(\frac{B}{D}\right)^{dk} = \left(\frac{dBR}{\alpha}\right)^{dk} \qquad (7)$$

Proof of Theorem 2. From a standard discretization lemma (Kakade, 2011) and Corollary 4, we immediately have that with probabilty $1 - \delta$,

$$\begin{aligned}
&\sup_{Z \in \mathcal{Z}} |\hat{\mathcal{L}} - \mathcal{L}| \leq \\
&\quad \leq \inf_\alpha 2\sqrt{\frac{dk(\log dBR - \log \alpha) + \log \frac{1}{\delta}}{2n}} + 2\alpha
\end{aligned}$$

Taking $\alpha = 1/n$,

$$\leq 2\sqrt{\frac{dk(\log dBR + \log n) + \log \frac{1}{\delta}}{2n}} + \frac{2}{n} \qquad \square$$

Acknowledgements

The authors would like to thank Peter Bartlett, Robert Nishihara and Maxim Rabinovich for useful discussions. This work was partially supported by BBN under DARPA contract HR0011-12-C-0014. The first author is supported by a National Science Foundation Graduate Fellowship.

References

Martin Anthony and Peter Bartlett. 2009. *Neural network learning: theoretical foundations*. Cambridge University Press.

Yoshua Bengio, Holger Schwenk, Jean-Sébastien Senécal, Fréderic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In *Innovations in Machine Learning*, pages 137–186. Springer.

Fadi Biadsy, Keith Hall, Pedro Moreno, and Brian Roark. 2014. Backoff inspired features for maximum entropy language models. In *Proceedings of the Conference of the International Speech Communication Association*.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. 2014. Fast and robust neural network joint models for statistical machine translation. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics*.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Alex Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid speech recognition with deep bidirectional LSTM. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 273–278.

Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pages 297–304.

Sham Kakade. 2011. Uniform and empirical covering numbers. `http://stat.wharton.upenn.edu/~skakade/courses/stat928/lectures/lecture16.pdf`.

Ronald Rosenfeld. 1994. *Adaptive statistical language modeling: a maximum entropy approach*. Ph.D. thesis.

Ashish Vaswani, Yinggong Zhao, Victoria Fossum, and David Chiang. 2013. Decoding with large-scale neural language models improves translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.