When not to use an Adversarial Approach to Generative Modeling

David Alvarez-Melis
Microsoft Research
alvarez.melis@microsoft.com

Vikas Garg
MIT
vgarg@casil.mit.edu

Adam Tauman Kalai
Microsoft Research
adam.kalai@microsoft.com

Abstract

This work offers a novel theoretical perspective on why, despite numerous attempts, adversarial approaches to generative modeling (e.g., GANs) have not been as popular in certain domains, such as Natural Language Processing, as they have in others, such as Computer Vision. In particular, we argue that optimizing KL-divergence (or equivalently, maximizing likelihood) is a more efficient approach to effectively minimizing the same distinguishability criteria that adversarial models seek to optimize. Reductions show that minimizing distinguishability can be seen as simply boosting likelihood for certain families of models including n-gram models and neural networks with a softmax output layer. For the case of sequential data, we further show that these reductions yield efficient polynomial-time algorithms.

1 Introduction

Consider a situation where one has samples from a true distribution p over a set X and one wishes to learn to generate similar samples, such as learning to generate English sentences from a large English text corpus. One seeks an approximation q of p which is "close" in some sense and from which samples can efficiently be generated. A common approach to fit these models is Maximum Likelihood Estimation (MLE), which given a sample from p and a parametrized distribution q_θ seeks parameters θ that maximize the likelihood q_θ assigns to a training set. MLE has long been one of the most popular method for fitting generative models of sequential data, such as language, and recent advances in neural network architectures (e.g., transformer-based methods such as "GPT-3" [3]) have led to MLE-fit neural language models that generate remarkably realistic text. MLE, of course, requires the ability to compute likelihoods $q(x)$ in addition to generating examples from q.

An alternative approach, Generative Adversarial Networks (GANs), has become popular across several domains, such as Computer Vision, owing to breakthrough realism in the images they output [e.g., 11, 35]. GANs employ an adversarial approach to generation through a zero-sum game between a generator and a distinguisher in which the generator produces samples $x \in X$ which the distinguisher tries to distinguish from real samples from p. Both the generator and the distinguisher are differentiable neural networks, though this min-max approach of choosing a model whose outputs are nearly indistinguishable from real examples might be considered for any families of generative models and distinguishers. GANs do not require the ability to compute likelihood.

Curiously, the adversarial approach has yet to demonstrate significant improvements in some other domains such as Natural Language Processing (NLP). One well-known barrier to NLP GANs is that language models produce discrete outputs (words), so they are not naturally differentiable [10]. However, despite numerous works circumventing this limitation and adapting GANs to text generation

Preprint. Under review.

[34, 22, 13, 6], adversarial-based models have yet to achieve the same popularity or performance gains that were seen for images. In particular, language GANs have been shown to under-perform MLE in terms of quality [30] while facing the challenge of lack of diversity due to mode collapse [4], which is a well-known issue with GANs in other domains.

In this note, we suggest a different, fundamental barrier to adopting GANs in domains where MLE is prevalent: the adversarial approach of minimizing distinguishability is effectively a roundabout method of maximizing likelihood on observed data, and hence employing MLE directly can be more efficient. This is the case in NLP where, unlike computer vision, a measure of likelihood called *perplexity* has been the prevailing metric for training and evaluating language models for decades. We show how GANs boost likelihood in the spirit of, and inspired by, the related celebrated work of [8] that showed how boosting can be viewed as an iterative approach for logistic regression.

Consider a large finite set or countably infinite set X and a family Q of probability distributions over X. For language, these might be n-gram models or neural models. Also consider a family F of distinguishers $f : X \to [0, 1]$ that aim to distinguish random examples drawn from a distribution p from those sampled from q. For any such classifier f, we call the difference $\alpha(f) = \mathrm{E}_q[f(x)] - \mathrm{E}_p[f(x)]$ the distinguishability *advantage* of f because it quantifies the accuracy of f at the task of identifying "fake" examples. A perfect distinguisher would thus have $\alpha(f) = 1$, while $f(x) = 1/2$ which predicts at random has $\alpha(f) = 0$. More formally, imagine picking $y \in \{0, 1\}$ uniformly at random and picking a random example x from q if $y = 1$ and from p if $y = 0$. The (randomized) binary classifier that predicts, for any x, $\hat{y} = 1$ with probability $f(x)$, has (expected) accuracy:

$$\frac{1}{2}\sum_x q(x)f(x) + \frac{1}{2}\sum_x p(x)(1 - f(x)) = \frac{1}{2} + \frac{1}{2}\alpha(f).$$

Given a family F, we define the *distinguishability* of q from p to be $d(q) = \max_{f \in F} \alpha(f)$. Distinguishability is known to be a lower-bound on *total variation distance* (also called statistical distance), a measure of distance between distributions that is difficult to directly estimate for large domains X. The best distinguisher simply predicts 1 iff $q(x) > p(x)$, and has advantage equal to the total variation distance [see, e.g., 14]. Clearly $d(p) = 0$ since p is indistinguishable from itself. Motivated by this observation, numerous *adversarial* approaches to approximating q have been attempted to minimize distinguishability $d(q)$. If $p \in Q$ then d is minimized at $q = p$.

However, when $p \notin Q$, searching for a model q that is nearly indistinguishable from p may seem different than maximizing likelihood on a training set. For instance, consider modeling the age of U.S. high school students as a uniformly random integer in some range. The distribution over ages 14-18 would be less distinguishable from the true p than the distribution over 0-200, but the former distribution assigns zero probability and the latter assigns positive probability since the population has students who are over 18 but none who are over 200 years old. This type of example motivates generative adversarial approaches.

However, as we shall see this difference between maximizing likelihood and minimizing distinguishability is an illusion if Q is a sufficiently rich class of models. We point out that for models which are log-linear over F, which include n-gram models and neural networks whose top layer is a softmax, minimizing KL divergence $D_{\mathrm{KL}}(p \parallel q) = \sum_x p(x) \log \frac{p(x)}{q(x)}$ and distinguishability over F are in fact one and the same, even when $p \notin Q$. To fit the above interval example into a log-linear model with F consisting of just the 14-18 distinguisher and the 0-200 distinguisher, Q would have to be able to express distributions that put weight p_1 on ages 14-18 and p_2 on the remaining ages in 0-200. Among such distributions, the likelihood among distributions is in fact maximized exactly when distinguishability is 0, when p_1 and p_2 match the population statistics among these intervals. Put another way, if q can be distinguished from p by a function f, then folding f into q yields a model with greater likelihood.

Unfortunately, our general analysis only guarantees a reduction whose runtime is linear in the set size $|X|$, with the challenge being normalizing distributions. In the case of sequential distributions, however, Algorithm 1 presents an efficient reduction between distinguishing q with nontrivial advantage and increasing log-likelihood (or reducing log-loss). Unlike standard boosting analysis which requires a weak learner which always succeeds, Theorem 1 shows that it takes a distinguishing oracle and simply keeps increasing log-likelihood until it reaches a point where the distinguishing oracle can no longer ϵ-distinguish the current distribution from the true distribution. The number of iterations is bounded by a quantity that depends on the log-loss of the initial model and ϵ. This

implies that if $p \in Q$ and one has an optimal distinguisher, one will eventually converge to a model close to p. However, it is also meaningful in the more general case where one simply has a practical distinguisher.

Organization. We begin by summarizing related work on GANs, especially for text. We then illustrate how GANs can be overkill for the simple case of n-gram models in Section 3. Section 4 covers log-linear models. Section 5 gives general reductions, with specific bounds, between maximizing likelihood and minimizing distinguishability. Section 6 shows how the reduction can be efficiently computed in the case of sequential inputs, from which we propose a simple polynomial time algorithm that provably finds a distribution which is nearly-indistinguishable with respect to a given class of discriminator functions. Finally, we discuss the relevance of our work in Section 7.

2 Related Work

Ever since their introduction [11], there has been interest in adapting GANs to text data. The driving motivation was that —up until very recently— samples generated by traditional (likelihood-based) models had been easy to distinguish from human-generated text, and the success of image GANs at generating realistic-looking samples suggested a possible avenue to improve the quality of their natural language counterparts.

The first and most significant challenge in adapting GANS to text arises from the very nature of this data. [10] points out that GANs require the generator to be differentiable, which poses a challenge for discrete text representations such as one-hot word or character representations. Two of the most popular approaches to circumvent this obstacle are policy gradient techniques (e.g., REINFORCE [33]) —which when applied to language modeling nevertheless often require maximum likelihood pre-training [5, 34])— and the Gumbel-Softmax approximation [17]. The few adversarial methods that do not require pre-training (e.g., [22, 25]) have failed to show significant promise in all but a few artificial tasks.

This nascent but active line of work seemed to suggest for a period of time that GANs might provide a breakthrough in text generation. This promise did not fully materialize, and instead the most recent breakthrough came from models boasting very large transformer-based architectures like GPT [23, 24, 3] — which are trained with traditional cross-entropy (MLE) objectives.

Yet the question of how GAN-based methods for text compare with likelihood-based ones still garners significant interest, and while various works have provided an empirical comparison between them —with most of these suggesting the advantage of MLE-based ones [4]— theoretical explanations have been less explored.

The connection between maximum likelihood estimation, distinguishability and divergences between probability distributions has been explored before. For example, it is well known that the maximizing likelihood is equivalent to minimizing the KL divergence between fitted and reference distributions, though this is not the only divergence for which such a connection exists [26]. On the other hand, from the moment GANs were introduced, Goodfellow et al. [11] noted that —assuming a perfect discriminator— the adversarial objective corresponds to minimizing a Jensen-Shannon divergence. Furthermore, the minimal discrimination error is also directly related to the total variation distance (see, e.g., Hashimoto et al. [14]). On the other hand, for exponential families the gradient of the KL divergence is known to be related to the discrepancy between distributions [31]. While conceptually similar to this line of work, here instead we give an *explicit* reduction that shows how distinguishability and (log) likelihood are in direct correspondence.

Pinkser's inequality is a well-known result linking KL divergence and total variation distance (TVD): $\text{TVD} \leq \sqrt{\text{KL}/2}$. While related, this inequality is not directly relevant to the context of this work. First, while total variation provides an upper bound to distinguishability, it is not computable in general, so it is rarely used as a training objective for generative models. On the other hand, being one-sided,[1] it does not imply that reducing TVD reduces KL divergence. Furthermore, Pinkser's is in general a very loose inequality, particularly for the direction of KLD that is equivalent to MLE (i.e., $D_{\text{KL}}(p \parallel q_\theta)$), since if $p(x) > 0 \approx q_\theta(x)$ even for a single x leads to unbounded KL divergence.

[1] Reverse Pinkser's inequalities exist only for particular cases, but they too are very loose in general [28].

In contrast, in this work we provide a *direct reduction* directly linking the two criteria of interest: distinguishability and maximum likelihood.

In this work we focus our analysis on log-linear models [18, 20], which are widely used in natural language processing (often known in that community as Maximum Entropy –MaxEnt– models) for various tasks. In particular, these models have been a cornerstone of both neural [2, 21] and non-neural [27, 15] language modeling.

The reduction shown here bears resemblance to boosting. In pioneering work, Friedman et al. [8] analyzed boosting through the lens of maximum likelihood, while Lebanon and Lafferty [19] formalized the equivalence of AdaBoost and maximum likelihood training for exponential models. More recently, boosting has been applied to generative adversarial models [32, 12], albeit with a different approach and objective than the connection drawn in this work.

3 Overkill: GANs for n-gram language models

To illustrate our main point, consider first the simplest model of language: a unigram model where the probability of each word is independent, and the end of sentence token EOS has a given probability as well. If θ_w represents the log-probability of word w, then the log-probability of sentence $w_1 \dots w_t$ is:

$$\log q(w_1 w_2 \dots w_t) = \theta_{w_1} + \theta_{w_2} + \dots + \theta_{w_t} + \theta_{EOS}.$$

The MLE parameters θ^* can be computed in linear time by simply counting word frequencies.

A more roundabout approach to fitting a unigram language model would be to start with any initial unigram model q, generate random samples from q and compare them to those from p. One could then distinguish the two by finding a word that appears significantly more often in one than in the other. For example, if one generates text from the model q and finds that the word "the" occurs much more often in text generated from p, one would then update the parameters by increasing θ_{the} (and decreasing $\theta_{w'}$ for all other words w' to keep q a probability distribution). As we shall see, this more involved procedure would converge to the same maximum-likelihood estimator θ^*.

A similar argument applies to any n-gram model in which the probability of each subsequent word is determined only by the previous $n-1$ words. This is also optimized by frequency counts, though in practice a variety of "smoothing" techniques (e.g., adding 1 to counts, also known as Laplace Smoothing) or more involved methods [9, 16] are used as a form of regularization on top of these counts. Distinguishers could similarly be used to find a model q that is indistinguishable from p according to n-gram frequencies, but again this would simply converge to the MLE parameters.

4 Log-linear models

In this section, consider a log-linear model with features $f : X \to [0,1]^d$, i.e., d bounded features $f_i : X \to [0,1]$. The model predicts,

$$q_\theta(x) = \frac{\exp\left(\langle f(x), \theta \rangle\right)}{Z_\theta} \tag{1}$$

where $\langle \cdot, \cdot \rangle$ denotes inner product, $\theta \in \mathbb{R}^d$ is a parameter vector and $Z_\theta = \sum_x \exp\langle \theta, f(x) \rangle$ is a normalizing constant called the partition function.

In the unigram example, the features f_i would be word counts normalized by dividing by the maximum allowed sentence length (to ensure $f_i(x) \leq 1$). In a neural network the features f_i would correspond to the top layer and q computes a softmax. Multiple strategies have been studied for computing or estimating the partition function Z_θ [see, e.g., 7].

As discussed earlier, these feature functions can also be thought of as classifiers that distinguish examples drawn from p from those drawn from q_θ and the advantage of f_i is $\alpha(f_i) = \sum_x f_i(x)(q_\theta(x) - p(x))$. The advantage vector is $\alpha(f) = \langle \alpha(f_i) \rangle_{i=1}^d$. Note that a negative advantage can be used for distinguishing by using the reverse classifier $1 - f_i$ which has opposite advantage $\alpha(1 - f_i) = -\alpha(f_i)$.

Observation 1. *The gradient of $D_{KL}(p \parallel q_\theta)$ with respect to θ is the advantage vector $\alpha(f)$, i.e., for all $i = 1, 2, \ldots, d$:*

$$\frac{\partial D_{KL}(p \parallel q_\theta)}{\partial \theta_i} = \sum_x f_i(x)(q_\theta(x) - p(x)) = \alpha(f_i).$$

The above straightforward calculation is well-known as is the fact that $D_{\mathrm{KL}}(p \parallel q_\theta)$ is convex in θ. However, we interpret this fact in the context of GANs: searching for θ which gives a zero-gradient for KL divergence is equivalent to finding θ which is indistinguishable with respect to each f_i. While a number of GANs have be designed in various architectures that solve the seemingly more complex problem of $\min_\theta d(q_\theta)$, it can generally be more efficient to maximize likelihood, which (approximately) minimizes the KL divergence.

5 General reductions and sample bounds

In this section, we reduce the problem of minimizing log-loss to that of minimizing distinguishability with precise bounds on a given sample set, rather than abstract quantities such as the KL divergence. Our bounds show that the relationship between maximizing likelihood and minimizing distinguishability holds on any given training sample.

For the remainder of this section, we drop θ from the variable denoting the fitted distribution q_θ to avoid cluttering the notation. Fix a sample $\mathcal{S} = \langle x_1, \ldots, x_n \rangle \in X^n$ of n training examples drawn from p, and define the log-loss to be:

$$\hat{L}(q; \mathcal{S}) = -\frac{1}{n} \sum_{i=1}^{n} \log q(x_i) = -\hat{\mathrm{E}}_{\mathcal{S}}[\log q(x)],$$

where we use hat on L to denote that the loss is estimated on a (finite) training set $\mathcal{S}$. Likewise, $\hat{\mathrm{E}}_{\mathcal{S}}[g(x)]$ denotes the empirical expectation $\frac{1}{n} \sum_{i=1}^{n} g(x_i)$. Note that the expected log-loss over training sets is known as the cross-entropy

$$H(p, q) = \mathrm{E}_{\mathcal{S} \sim p^n}[\hat{L}(q; \mathcal{S})],$$

and hence the expected difference in log-loss between two candidate distributions q and q' is equal to the difference

$$\mathrm{E}_{\mathcal{S} \sim p^n}[\hat{L}(q; \mathcal{S}) - \hat{L}(q'; \mathcal{S})] = D_{\mathrm{KL}}(p \parallel q) - D_{\mathrm{KL}}(p \parallel q') \,,$$

so minimizing log-loss is approximately equivalent to minimizing the KL divergence. Also, we define the *training advantage* of distinguisher $f : X \to [0, 1]$ to be:

$$\hat{\alpha}(f) = \mathrm{E}_q[f(X)] - \hat{\mathrm{E}}_{\mathcal{S}}[f(x)] \tag{2}$$

which is independent of p, depending on the sample alone and can thus be estimated to arbitrary accuracy using samples generated from q. The lemmas below show how one can use a distinguisher to reduce log-loss on the same training sample, and how to use a distribution with a lower log-loss to distinguish the two distributions.

Lemma 1. *Let $a \geq 0$ and suppose $f : X \to [0, 1]$ has training advantage $\hat{\alpha}(f) \geq a$. Then, the probability distribution $q'(x) = q(x)e^{-af(x)}/Z_{q'}$ where $Z_{q'} = \sum_x q(x)e^{-af(x)}$, has lower log-loss:*

$$\hat{L}(q'; \mathcal{S}) \leq \hat{L}(q; \mathcal{S}) - a^2/2.$$

Before we give the proof, we note that if f is computed by a neural network and q is computed as a neural network with a softmax at the top layer, i.e., $q(x) = e^{\langle v, g(x) \rangle} / \sum_x e^{\langle v, g(x) \rangle}$ where $g : X \to \mathbb{R}^d$ is some neural network, then q' is naturally represented as the combined neural network with softmax $q'(x) \propto e^{\langle (v, -a), (g(x), f(x)) \rangle}$ in $d + 1$ dimensions.

Proof (Lemma 1).

$$\begin{aligned}
\hat{L}(q; \mathcal{S}) - \hat{L}(q'; \mathcal{S}) &= \hat{\mathrm{E}}_{\mathcal{S}}[\log q'(x) - \log q(x)] \\
&= \hat{\mathrm{E}}_{\mathcal{S}}\left[-af(x) - \log Z_{q'}\right] \\
&= a(\mathrm{E}_q\left[f(x)\right] - \hat{\mathrm{E}}_{\mathcal{S}}\left[f(x)\right]) \\
&\quad - a\mathrm{E}_q\left[f(x)\right] - \log Z_{q'} \\
&= a\hat{\alpha}(f) - a\mathrm{E}_q\left[f(x)\right] - \log Z_{q'}
\end{aligned} \tag{3}$$

Since $\hat{\alpha}(f) \geq a$ by assumption, it remains to show $a\mathrm{E}_q[f(x)] + \log Z_{q'} \leq a^2/2$. Using the bound $\log r \leq r - 1$ for any $r \in \mathbb{R}$, we get that,

$$\begin{aligned} a\mathrm{E}_q[f(x)] + \log Z_{q'} &\leq a\mathrm{E}_q[f(x)] + Z_{q'} - 1 \\ &= a\mathrm{E}_q[f(x)] + \mathrm{E}_q[e^{-af(x)}] - 1 \\ &= \mathrm{E}_q\left[af(x) + e^{-af(x)} - 1\right] \\ &\leq \mathrm{E}_q[(af(x))^2/2], \end{aligned}$$

where we have used the fact that $Z_{q'} = \mathrm{E}_q[e^{-af(x)}]$ and, to get to the last line we use $e^{-r} + r - 1 \leq r^2/2$ for $r \geq 0$ by Taylor expansion. Since $f(x) \in [0, 1]$, the last quantity is at most $a^2/2$, which together with (3), gives $\hat{L}(q; \mathcal{S}) - \hat{L}(q'; \mathcal{S}) \geq a^2/2$ □

This means that if we can distinguish $\mathcal{S}$ from q, then we can simply reduce log-loss by down-weighting suspicious samples that satisfy the distinguisher $f(x)$. The difference between this statement and Observation 1 is analogous to the difference between boosting and logistic regression [8]. In logistic regression, one typically fixes the set of features in advance, whereas in boosting this is not possible if there are infinitely many possible classifiers.

Conversely, we next show that if q' has a lower log-loss than q on the training samples, then we can distinguish q from these samples.

Lemma 2. *For any constant $C > 1$ and distributions q, q' such that $\frac{1}{C}q(x) \leq q'(x) \leq Cq(x)$ for all $x \in X$, the distinguisher $f : X \to [0, 1]$ defined by,*

$$f(x) = \frac{1}{2\log C} \log \frac{Cq(x)}{q'(x)},$$

has a training advantage of,

$$\hat{\alpha}(f) \geq \frac{\hat{L}(q) - \hat{L}(q')}{2\log C}.$$

Proof (Lemma 2). Let $g(x) = \log q(x) - \log q'(x)$. By Jensen's inequality,

$$\begin{aligned} \mathrm{E}_q[g(x)] - \hat{\mathrm{E}}_{\mathcal{S}}[g(x)] &= -\mathrm{E}_q\left[\log \frac{q'(x)}{q(x)}\right] - \hat{\mathrm{E}}_{\mathcal{S}}[g(x)] \\ &\geq -\log \mathrm{E}_q\left[\frac{q'(x)}{q(x)}\right] - \hat{\mathrm{E}}_{\mathcal{S}}[g(x)] \\ &= -\log(1) - \hat{\mathrm{E}}_{\mathcal{S}}[g(x)] \\ &= \mathrm{E}_{\mathcal{S}}[\log q'(x)] - \hat{\mathrm{E}}_{\mathcal{S}}[\log q(x)] \\ &= \hat{L}(q; S) - \hat{L}(q'; S) \end{aligned}$$

Since $f(x) = \frac{1}{2\log C}(g(x) + \log C)$, the training advantage of f is that of g scaled by a factor of $\frac{1}{2\log C}$. Finally, it is straightforward to verify that $f(x) \in [0, 1]$ by our assumptions on the ratio between q and q'. □

Importantly, the above theorem requires a bound C on the ratio q'/q and is meaningful even if q and q' are exponentially far apart so long as they have the same support. This assumption was violated for our example from the introduction where the two intervals had different supports.

6 Efficient Likelihood Boosting for Sequential Inputs

In this section we show how one can efficiently compute the normalization terms (partition function) on a token-by-token basis for black-box sequential (e.g., language or auto-regressive) models. The key insight for efficiency is that, rather than distinguishing entire sentences from p and q, one distinguishes the conditional next-word predictions. In particular, rather than generating entire sentences from q, one can generate next-word predictions on all sentence prefixes in the training data.

Lemma 1 in the previous section implies a reduction between the problem of distinguishing with nontrivial advantage to nontrivially reducing log-loss. Without the algorithm in this section, iteratively applying the reduction requires repeated computation of the normalization terms over X, and computing such partition functions is an area of active research—where it is known how to do it efficiently for some classes and not for others. For a general black-box distinguisher, one would have to sum over all possible inputs, which is computationally intractable even for a single step, let alone for a sequence of reductions.

An example would be a language model, where summing a neural network over all sentences is infeasible. However, in many applications such as NLP, the inputs have some special structure, where they are sequential $x = (x_1, \ldots, x_n)$, and where every symbol x_i is taken from a large discrete vocabulary. In such cases, the combinatorial nature of the data makes density estimation intractable unless the likelihood computation is broken into small sequential steps using the chain rule.

In this section we show how a natural extension of the framework described above allows us to achieve an efficient reduction for the common case of sequential data. To do so, we define a simple generalization of the training advantage criterion (2), which now relies on a *step-wise distinguisher* g operating on variable-length sequences. Formally, we consider a *language* of N-length sequences of symbols taken from a vocabulary $\mathcal{V}$, and distinguisher functions $g : \bigcup_{j=1}^{N} \mathcal{V}^j \to [0, 1]$, i.e., functions which can take subsequences of any size as input. Given a sample $\mathcal{S}$ of sequences, we say that g has *generalized training advantage* given by

$$\hat{\beta}(g) = \hat{\beta}(g, \mathcal{S}, q) = \frac{1}{N} \sum_{j=1}^{N} \hat{\mathrm{E}}_{x \sim \mathcal{S}} \big[\mathrm{E}_{w \sim q(\cdot | x_0, \ldots, x_{j-1})} g(x_0, \ldots, x_{j-1}, w) - g(x_0, \ldots, x_j) \big] \quad (4)$$

where, by convention, $x_0 = \emptyset$, so that $q(x_0, w) = q(w)$. This criterion can be interpreted as follows. For every length $j \in [0, .., N]$, g is tasked with distinguishing a subsequence consisting of the first j tokens in a *true* sequence sampled from $\mathcal{S}$ from another j-length sequence in which the last element is replaced by a randomly selected token from the alternate distribution q'.

Lemma 3. *Let $b \geq 0$ and suppose $g : \bigcup_{j=1}^{N} \mathcal{V}^j \to [0, 1]$ has generalized training advantage $\hat{\beta}(g) \geq b$. We define a distribution q' through its conditional probabilities as:*

$$q'(x_j \mid x_1, \ldots, x_{j-1}) = q(x_j \mid x_1, \ldots, x_{j-1}) e^{-g(x_1, \ldots, x_j)} / Z_{q'}(x_1, \ldots, x_{j-1})$$

where now $Z_{q'}(x_1, \ldots, x_{j-1}) = \sum_{\tilde{x}_j} q(\tilde{x}_j \mid x_1, \ldots, x_{j-1}) e^{-bg(x_1, \ldots, x_{j-1}, \tilde{x}_j)}$. Then q' incurs lower log-loss than q:

$$\hat{L}(q'; \mathcal{S}) \leq \hat{L}(q; \mathcal{S}) - Nb^2/2.$$

Proof. Provided in full in Appendix A. □

Next, we use Lemma 3 repeatedly to derive a simple algorithm that, given access to non-trivial weak distinguishers, returns a distribution that is nearly indistinguishable (by that class) from the the true distribution p. Formally, let $\mathcal{G} = \{g \mid g : \bigcup_{j=1}^{N} \mathcal{V}^j \to [0, 1]\}$ be a class of distinguishers. We assume access to an oracle $\mathrm{O}_d : \mathcal{Q} \mapsto \mathcal{G}$ which for any $q \in \mathcal{Q}$ returns a distinguisher g. We say that q is ϵ-indistinguishable by oracle O_d if its output g has advantage $\hat{\beta}(g, \mathcal{S}, q) \leq \epsilon$. We do not need to assume that O_d is optimal in any sense.

Theorem 1. *Let q_0 be a language model and let $\epsilon > 0$. Algorithm 1 returns a distribution q^* which is ϵ-indistinguishable from $\mathcal{S}$ by oracle O_d. It runs in $O\big(\frac{1}{\epsilon^2} L_0 (\frac{T_d}{N} + nT_g(m + n))\big)$ time, where $L_0 = \hat{L}(q_0; \mathcal{S})$ is the log-loss of q_0.*

Proof. The fact that Algorithm 1 terminates with a distribution q which is ϵ-indistinguishable by O_d is immediate from the stopping criterion.

Now, for the runtime analysis, note that —by construction— the iterates g_t, $t \in \{0, \ldots, T-1\}$ have training advantage $\hat{\beta}(g_t, \mathcal{S}, q_t) \geq \epsilon$. Thus, by Lemma 3, the algorithm makes at least $\frac{N\epsilon^2}{2}$ improvement in each iteration. Therefore, the total number of iterations T is at most $\frac{2L_0}{N\epsilon^2}$, where $L_0 := \hat{L}(q_0; \mathcal{S})$ is the log-loss of the initial model.

Algorithm 1: Boosted weak distinguishers.

Input: Initial model q_0, corpus $\mathcal{S}$, distinguisher oracle O_d, advantage threshold ϵ.
$t \leftarrow 0$
while *True* **do**
  $g_t \leftarrow \mathrm{O}_d(q_t)$
  $b_t \leftarrow \hat{\beta}(g_t, \mathcal{S}, q_t)$
  if $b_t < \epsilon$, **Output:** q_t
  Compute $q_{t+1}(x) \triangleq q_t(x)e^{-b_t g_t(x)} / \sum_{x \in \mathcal{S}} q_t(x)e^{-b_t g_t(x)}$ on entire corpus $\mathcal{S}$
  $t \leftarrow t + 1$
end

Let T_d be the runtime of oracle O_d, and T_g the complexity of evaluating any g on a single input. And recall that $|\mathcal{V}| = n$ is the vocabulary size, N is the sequence length and $|\mathcal{S}| = m$ is the training set size. Each iteration of Algorithm 1 requires calling O_d oracle once, evaluating $\hat{\beta}(\cdot)$ at an $\mathcal{O}(Nmn T_g)$ complexity, and updating each of the n next-symbol probabilities of q for each sequence length $1, \ldots, N$. Each of these updates involves evaluating g plus an $\mathcal{O}(n)$ partition normalization. Putting these together, we conclude that each iteration has $O(T_d + NnT_g(m+n))$ complexity.

Combining the the two arguments above, we conclude that Algorithm 1 has a total runtime of $O\big(\frac{1}{\epsilon^2} L_0 (\frac{T_d}{N} + nT_g(m+n))\big)$. □

7 Discussion and conclusions

In this work, we have argued that minimizing log-loss (i.e., KL-divergence) and minimizing statistical distinguishability are tightly related goals. This means in applications where it is natural to fit models by minimizing log-loss, it is indeed likely to be a more direct and efficient means of fitting a model. This is the case for n-gram language models (and other sequential tasks), for which perplexity (a measure of likelihood) is easy to compute, naturally meaningful, and allows for efficient sampling. Thus, for a long time, minimizing log-loss has been the objective with which most state-of-the-art models are trained. For such models, Lemma 1 implies that if one can distinguish the model from samples by a neural network then one can construct a larger neural network with lower log-loss. Hence, one may prefer to simply train a larger model in the first place.

One limitation of our work is that we do not address how samples can be efficiently generated or how the partition function can be computed, as this varies by architecture. In some architectures, such as unigram models, the new log-linear model can always be sampled from efficiently. For other models, we do not offer a general solution but a variety of sampling techniques have been studied.

Broader Impact

The contribution of this work is conceptual and theoretical, and as such, any nuanced discussion of the potential harms or benefits of its impact are irremediably tied to the applications where it might be put to use. We first make a few general observations regarding its immediate impact, and then discuss in a more informal manner downstream ramifications these might have in applications.

This work revolves around comparing to training paradigms: maximum likelihood and adversarial learning. We believe that the maxim of 'choosing the right tool for the job' applies in this context too, and can have important downstream consequences. For example, the amount of resources consumed by training large generative models has been growing substantially over the past few years [1]. This is particularly true for Natural Language Processing, where state-of-the-art models are increasingly large and trained on increasingly larger datasets, leading to striking computationally and environmental costs [29]. The key takeaway offered by this work, namely that training certain generative models like LMs through adversarial methods is less efficient that doing so via likelihood maximization approaches, could potentially lead to significant saving of these resources by steering practitioners away from adversarial approaches. On the other hand, it is not our intention for this work to lead to the opposite —but equally undesirable— effect of dissuading practitioners from choosing adversarial training whenever that is the sensible choice.

References

[1] D. Amodei and D. Hernandez. *AI and Compute*. https://openai.com/blog/ai-and-compute/. Accessed: 2020-6-1. 2018.

[2] Y. Bengio et al. "A Neural Probabilistic Language Model". In: *Journal of Machine Learning Research (JMLR)* 3 (2003), pp. 1137–1155.

[3] T. B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL].

[4] M. Caccia et al. *Language GANs Falling Short*. 2018. arXiv: 1811.02549 [cs.CL].

[5] T. Che et al. "Maximum-likelihood augmented discrete generative adversarial networks". In: *arXiv preprint arXiv:1702.07983* (2017).

[6] C. d. M. d'Autume et al. "Training language GANs from Scratch". In: *arXiv preprint arXiv:1905.09922* (2019).

[7] G. Desjardins et al. "On tracking the partition function". In: *Advances in neural information processing systems*. 2011, pp. 2501–2509.

[8] J. Friedman et al. "Additive Logistic Regression: a Statistical View of Boosting". In: *The Annals of Statistics* 38.2 (2000).

[9] I. J. Good. "The population frequencies of species and the estimation of population parameters". In: *Biometrika* 40.3-4 (1953), pp. 237–264.

[10] I. Goodfellow. "NIPS 2016 Tutorial: Generative Adversarial Networks". In: *arXiv preprint arXiv:1701.00160* (2017).

[11] I. Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems (NIPS)*. 2014, pp. 2672–2680.

[12] A. Grover and S. Ermon. "Boosted generative models". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[13] J. Guo et al. "Long text generation via adversarial training with leaked information". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. 2018.

[14] T. B. Hashimoto et al. "Unifying human and statistical evaluation for natural language generation". In: *Proceedings of NAACL-HLT*. 2019, pp. 1689–1701.

[15] S. Khudanpur and J. Wu. "Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling". In: *Computer Speech & Language* 14.4 (2000), pp. 355–372.

[16] R. Kneser and H. Ney. "Improved backing-off for m-gram language modeling". In: *1995 International Conference on Acoustics, Speech, and Signal Processing*. Vol. 1. IEEE. 1995, pp. 181–184.

[17] M. J. Kusner and J. M. Hernández-Lobato. "Gans for sequences of discrete elements with the gumbel-softmax distribution". In: *arXiv preprint arXiv:1611.04051* (2016).

[18] J. D. Lafferty et al. "Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data". In: *International Conference on Machine Learning (ICML)*. 2001, pp. 282–289.

[19] G. Lebanon and J. D. Lafferty. "Boosting and Maximum Likelihood for Exponential Models". In: *Advances in Neural Information Processing Systems (NIPS)*. Ed. by T. G. Dietterich et al. 2002, pp. 447–454.

[20] A. McCallum et al. "Maximum Entropy Markov Models for Information Extraction and Segmentation". In: *International Conference on Machine Learning (ICML)*. 2000, pp. 591–598.

[21] T. Mikolov et al. "Efficient estimation of word representations in vector space". In: *arXiv preprint arXiv:1301.3781* (2013).

[22] O. Press et al. "Language generation with recurrent generative adversarial networks without pre-training". In: *1st Workshop on Learning to Generate Natural Language at ICML 2017*. 2017.

[23] A. Radford et al. "Improving Language Understanding by Generative Pre-Training". In: (2018).

[24] A. Radford et al. "Language models are unsupervised multitask learners". In: *OpenAI Blog* 1.8 (2019).

[25] S. Rajeswar et al. "Adversarial generation of natural language". In: *arXiv preprint arXiv:1705.10929* (2017).

[26] P. Rigollet and J. Weed. "Entropic optimal transport is maximum-likelihood deconvolution". In: *Comptes Rendus Mathematique* 356.11-12 (2018), pp. 1228–1235.

[27] R. Rosenfeld. *Adaptive statistical language modeling; a maximum entropy approach*. Tech. rep. Carnegie-Mellon University, Department of Computer Science, 1994.

[28] I. Sason. "On reverse Pinsker inequalities". In: *arXiv preprint arXiv:1503.07118* (2015).

[29] E. Strubell et al. "Energy and Policy Considerations for Deep Learning in NLP". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. aclweb.org, 2019, pp. 3645–3650.

[30] G. Tevet et al. "Evaluating Text GANs as Language Models". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 2241–2247.

[31] L. Theis and M. D. Hoffman. "A Trust-region Method for Stochastic Variational Inference with Applications to Streaming Data". In: *International Conference on Machine Learning (ICML)*. 2015, pp. 2503–2511.

[32] I. O. Tolstikhin et al. "Adagan: Boosting generative models". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5424–5433.

[33] R. J. Williams. "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Machine learning* 8.3-4 (1992), pp. 229–256.

[34] L. Yu et al. "Seqgan: Sequence generative adversarial nets with policy gradient". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[35] H. Zhang et al. "Self-Attention Generative Adversarial Networks". In: *International Conference on Machine Learning (ICML)*. 2019, pp. 7354–7363.

A Proof of Lemma 3

We proceed analogously as in the proof of Lemma 1. We first note that

$$\hat{L}(q;\mathcal{S}) = -\hat{\mathrm{E}}_{\mathcal{S}}\left[\log\prod_{i=1}^{N} q(x_i \mid x_1,\ldots,x_{i-1})\right] = -\sum_{i=1}^{N}\hat{\mathrm{E}}_{\mathcal{S}}\log q(x_i \mid x_1,\ldots,x_{i-1}),$$

and

$$\begin{aligned}\hat{L}(q';\mathcal{S}) &= -\hat{\mathrm{E}}_{\mathcal{S}}\left[\log\prod_{i=1}^{N} q'(x_i \mid x_1,\ldots,x_{i-1})\right] \\ &= \sum_{i=1}^{N} -\hat{\mathrm{E}}_{\mathcal{S}}\log q(x_i \mid x_1,\ldots,x_{i-1}) + bg(x_1,\ldots,x_i) + \log Z_{q'}(x_1,\ldots,x_{i-1})\end{aligned}$$

Let us use the short-hand notation $x_{1:i} \triangleq (x_1,\ldots,x_i)$. Subtracting the two equalities above we obtain

$$\hat{L}(q;\mathcal{S}) - \hat{L}(q';\mathcal{S}) = \sum_{i=1}^{N}\hat{\mathrm{E}}_{\mathcal{S}}\left[-bg(x_{1:i}) - \log Z_{q'}(x_{1:i-1})\right],$$

which, after adding and subtracting $\mathrm{E}_{w\sim q(w|x_{1:i-1})}\hat{\mathrm{E}}_{\mathcal{S}}g(x_{1:i-1},w)$ and rearranging terms, yields

$$\hat{L}(q;\mathcal{S}) - \hat{L}(q';\mathcal{S}) = b\left[\sum_{i=1}^{N}\hat{\mathrm{E}}_{\mathcal{S}}\left(\mathrm{E}_{w\sim q(w|x_{1:i})}g(x_{1:i-1},w) - g(x_{1:i})\right)\right] \tag{5}$$

$$- \sum_{i=1}^{N}\hat{\mathrm{E}}_{\mathcal{S}}\left[\log Z_{q'}(x_{1:i-1}) - b\mathrm{E}_{w\sim q(w|x_{1:i-1})}g(x_{1:i-1},w)\right] \tag{6}$$

$$= bN\hat{\beta}(g) - \sum_{i=1}^{N}\hat{\mathrm{E}}_{\mathcal{S}}\left[b\mathrm{E}_w g(x_{1:i-1},w) + \log Z_{q'}(x_{1:i-1})\right] \tag{7}$$

By assumption we have $Nb\hat{\beta}(g) \geq Nb^2$, so it it remains to show that the second term is upper bounded by $Nb^2/2$. Using, as before, the bound $\log r \leq r-1$ for every $r = Z_{q'}(x_{1:i-1}) \geq 0$, we get that, for every $i = 1,\ldots,N$:

$$\begin{aligned}\hat{\mathrm{E}}_{\mathcal{S}}\left[b\mathrm{E}_w g(x_{1:i-1},w) + \log Z_{q'}(x_{1:i-1})\right] &\leq \hat{\mathrm{E}}_{\mathcal{S}}\left[b\mathrm{E}_w g(x_{1:i-1},w) + Z_{q'}(x_{1:i-1}) - 1\right] \\ &= \hat{\mathrm{E}}_{\mathcal{S}}\left[b\mathrm{E}_w g(x_{1:i-1},w) + \mathrm{E}_w e^{-bg(x_{1:i-1},w)} - 1\right] \\ &= \hat{\mathrm{E}}_{\mathcal{S}}\mathrm{E}_w\left[bg(x_{1:i-1},w) + e^{-bg(x_{1:i-1},w)} - 1\right] \\ &\leq \hat{\mathrm{E}}_{\mathcal{S}}\mathrm{E}_w\left(bg(x_{1:i-1},w)/2\right)^2 \leq \frac{b^2}{2}\end{aligned}$$

where the last inequality follows again from the fact that $g(x) \in [0,1]$ for any x. Therefore, the sum over these N terms is upper bounded by $N\frac{b^2}{2}$, which combined with (7), yields the desired result.