Sliced Score Matching: A Scalable Approach to Density and Score Estimation

Yang Song[*] Stanford University

Sahaj Garg[*] Stanford University

Jiaxin Shi Tsinghua University

Stefano Ermon Stanford University

Abstract

Score matching is a popular method for estimating unnormalized statistical models. However, it has been so far limited to simple, shallow models or low-dimensional data, due to the difficulty of computing the Hessian of log-density functions. We show this difficulty can be mitigated by projecting the scores onto random vectors before comparing them. This objective, called sliced score matching, only involves Hessian-vector products, which can be easily implemented using reverse-mode automatic differentiation. Therefore, sliced score matching is amenable to more complex models and higher dimensional data compared to score matching. Theoretically, we prove the consistency and asymptotic normality of sliced score matching estimators. Moreover, we demonstrate that sliced score matching can be used to learn deep score estimators for implicit distributions. In our experiments, we show sliced score matching can learn deep energy-based models effectively, and can produce accurate score estimates for applications such as variational inference with implicit distributions and training Wasserstein Auto-Encoders.

1 INTRODUCTION

Score matching (Hyvärinen, 2005) is particularly suitable for learning unnormalized statistical models, such as energy based ones. It is based on minimizing the distance between the derivatives of the log-density functions (a.k.a., *scores*) of the data and model distributions. Unlike maximum likelihood estimation (MLE), the objective of score matching only depends on the scores, which are oblivious to the (usually) intractable partition functions. However, score matching requires the computation of the diagonal elements of the Hessian of the model's log-density function. This Hessian trace computation is generally expensive (Martens et al., 2012), requiring a number of forward and backward propagations proportional to the data dimension. This severely limits its applicability to complex models parameterized by deep neural networks, such as deep energy-based models (LeCun et al., 2006; Wenliang et al., 2019).

[*] Joint first authors. Correspondence to Yang Song <yangsong@cs.stanford.edu> and Stefano Ermon <ermon@cs.stanford.edu>.

Several approaches have been proposed to alleviate this difficulty: Kingma & LeCun (2010) propose approximate backpropagation for computing the trace of the Hessian; Martens et al. (2012) develop curvature propagation, a fast stochastic estimator for the trace in score matching; and Vincent (2011) transforms score matching to a denoising problem which avoids second-order derivatives. These methods have achieved some success, but may suffer from one or more of the following problems: inconsistent parameter estimation, large estimation variance, and cumbersome implementation.

To alleviate these problems, we propose sliced score matching, a variant of score matching that can scale to deep unnormalized models and high dimensional data. The key intuition is that instead of directly matching the high-dimensional scores, we match their projections along random directions. Theoretically, we show that under some regularity conditions, sliced score matching is a well-defined statistical estimation criterion that yields consistent and asymptotically normal parameter estimates. Moreover, compared to the methods of Kingma & LeCun (2010) and Martens et al. (2012), whose implementations require customized backpropagation for deep networks, sliced score matching only involves Hessian-vector products, thus can be easily and efficiently implemented in frameworks such as TensorFlow (Abadi et al., 2016) and PyTorch (Adam et al., 2017).

Beyond training unnormalized models, sliced score matching can also be naturally adapted as an objective for estimating the score function of a data generating distribution (Sasaki et al., 2014; Strathmann et al., 2015) by training a score function model parameterized by deep neural networks. This observation enables many new applications of sliced score matching. For example, we show that it can be used to provide accurate score estimates needed for variational inference with implicit distributions (Huszár, 2017) and learning Wasserstein Auto-Encoders (WAE, Tolstikhin et al. (2018)).

Finally, we evaluate the performance of sliced score matching on learning unnormalized statistical models (density estimation) and estimating score functions of a data generating process (score estimation). For density estimation, experiments on deep kernel exponential families (Wenliang et al., 2019) and NICE flow models (Dinh et al., 2015) show that our method is either more scalable or more accurate than existing score matching variants.

For score estimation, our method improves the performance of variational auto-encoders (VAE) with implicit encoders, and can train WAEs without a discriminator or MMD loss by directly optimizing the KL divergence between aggregated posteriors and the prior. In both situations we outperformed kernel-based score estimators (Li & Turner, 2018; Shi et al., 2018) by achieving better test likelihoods and better sample quality in image generation.

2 BACKGROUND

Given i.i.d. samples $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \subset \mathbb{R}^D$ from a data distribution $p_d(\mathbf{x})$, our task is to learn an unnormalized density, $\tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})$, where $\boldsymbol{\theta}$ is from some parameter space Θ. The model's partition function is denoted as $Z_{\boldsymbol{\theta}}$, which is assumed to be existent but intractable. Let $p_m(\mathbf{x}; \boldsymbol{\theta})$ be the normalized density determined by our model, we have

$$p_m(\mathbf{x}; \boldsymbol{\theta}) = \frac{\tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})}{Z_{\boldsymbol{\theta}}}, \quad Z_{\boldsymbol{\theta}} = \int \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}) \mathrm{d}\mathbf{x}.$$

For convenience, we denote the score functions of p_m and p_d as $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) \triangleq \nabla_{\mathbf{x}} \log p_m(\mathbf{x}; \boldsymbol{\theta})$ and $\mathbf{s}_d(\mathbf{x}) \triangleq \nabla_{\mathbf{x}} \log p_d(\mathbf{x})$ respectively. Note that since $\log p_m(\mathbf{x}; \boldsymbol{\theta}) = \log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}) - \log Z_{\boldsymbol{\theta}}$, we immediately conclude that $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ does not depend on the intractable partition function $Z_{\boldsymbol{\theta}}$.

2.1 SCORE MATCHING

Learning unnormalized models with maximum likelihood estimation (MLE) can be difficult due to the intractable partition function $Z_{\boldsymbol{\theta}}$. To avoid this, score matching (Hyvärinen, 2005) minimizes the Fisher divergence between p_d and $p_m(\cdot, \boldsymbol{\theta})$, which is defined as

$$L(\boldsymbol{\theta}) \triangleq \frac{1}{2} \mathbb{E}_{p_d}[\|\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) - \mathbf{s}_d(\mathbf{x})\|_2^2]. \tag{1}$$

Since $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ does not involve $Z_{\boldsymbol{\theta}}$, the Fisher divergence does not depend on the intractable partition function. However, Eq. (1) is still not readily usable for learning unnormalized models, as we only have samples and do not have access to the score function of the data $\mathbf{s}_d(\mathbf{x})$.

By applying integration by parts, Hyvärinen (2005) shows that $L(\boldsymbol{\theta})$ can be written as $L(\boldsymbol{\theta}) = J(\boldsymbol{\theta}) + \mathrm{C}$ (*cf.*, Theorem 1 in Hyvärinen (2005)), where

$$J(\boldsymbol{\theta}) \triangleq \mathbb{E}_{p_d}\left[\mathrm{tr}(\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})) + \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})\|_2^2\right], \tag{2}$$

C is a constant that does not depend on $\boldsymbol{\theta}$, $\mathrm{tr}(\cdot)$ denotes the trace of a matrix, and

$$\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}) = \nabla_{\mathbf{x}}^2 \log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta}) \tag{3}$$

is the Hessian of the log-density function. The constant can be ignored and the following unbiased estimator of the remaining terms is used to train $\tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})$:

$$\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N) \triangleq \frac{1}{N} \sum_{i=1}^{N} \left[\mathrm{tr}(\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})) + \frac{1}{2} \|\mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2\right],$$

where $\mathbf{x}_1^N$ is a shorthand used throughout the paper for a collection of N data points $\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\}$ sampled i.i.d. from p_d, and $\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})$ denotes the Hessian of $\log \tilde{p}_m(\mathbf{x}; \boldsymbol{\theta})$ evaluted at $\mathbf{x}_i$.

Computational Difficulty. While the score matching objective $\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N)$ avoids the computation of $Z_{\boldsymbol{\theta}}$ for unnormalized models, it introduces a new computational difficulty: computing the trace of the Hessian of a log-density function, $\nabla_{\mathbf{x}}^2 \log \tilde{p}_m$. A naïve approach of computing the trace of the Hessian requires D times more backward passes than computing the gradient $\mathbf{s}_m = \nabla_{\mathbf{x}} \log \tilde{p}_m$ (see Alg. 2 in the appendix). For example, the trace could be computed by applying backpropogation D times to $\mathbf{s}_m$ to get each diagonal term of $\nabla_{\mathbf{x}}^2 \log \tilde{p}_m$ sequentially. In practice, D can be many thousands, which can render score matching too slow for practical purposes. Moreover, Martens et al. (2012) argues from a theoretical perspective that it is unlikely that there exists an algorithm for computing the diagonal of the Hessian defined by an arbitrary computation graph within a constant number of forward and backward passes.

2.2 SCORE ESTIMATION FOR IMPLICIT DISTRIBUTIONS

Besides parameter estimation in unnormalized models, score matching can also be used to estimate scores of *implicit distributions*, which are distributions that have a tractable sampling process but without a tractable density. For example, the distribution of random samples from the generator of a GAN (Goodfellow et al., 2014) is an implicit distribution. Implicit distributions can arise in many more situations such as the marginal distribution of a non-conjugate model (Sun et al., 2019), and models defined by complex simulation processes (Tran et al., 2017). In many cases learning and inference become intractable due to the need of optimizing an objective that involves the intractable density.

In these cases, directly estimating the score function $\mathbf{s}_q(\mathbf{x}) = \nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})$ based on i.i.d. samples from an implicit density $q_{\boldsymbol{\theta}}(\mathbf{x})$ can be useful. For example, suppose our learning problem involves optimizing the entropy $H(q_{\boldsymbol{\theta}}(\mathbf{x}))$ of an implicit distribution. This situation is common when dealing with variational free energies (Kingma & Welling, 2014). Suppose $\mathbf{x} \sim q_{\boldsymbol{\theta}}$ can be reparameterized as $\mathbf{x} = g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$, where $\boldsymbol{\epsilon}$ is a simple random variable independent of $\boldsymbol{\theta}$, such as a standard normal, and $g_{\boldsymbol{\theta}}$ is a deterministic mapping. We can write the gradient of the entropy with respect to $\boldsymbol{\theta}$ as

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} H(q_{\boldsymbol{\theta}}) &\triangleq -\nabla_{\boldsymbol{\theta}} \mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{x})}[\log q_{\boldsymbol{\theta}}(\mathbf{x})] \\ &= -\nabla_{\boldsymbol{\theta}} \mathbb{E}_{p(\boldsymbol{\epsilon})}[\log q_{\boldsymbol{\theta}}(g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon}))] \\ &= -\mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})|_{\mathbf{x}=g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})} \nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})],\end{aligned}$$

where $\nabla_{\boldsymbol{\theta}} g_{\boldsymbol{\theta}}(\boldsymbol{\epsilon})$ is usually easy to compute. The score $\nabla_{\mathbf{x}} \log q_{\boldsymbol{\theta}}(\mathbf{x})$ is intractable but can be approximated by score estimation.

Score matching is an attractive solution for score estimation since (1) naturally serves as an objective to measure the difference between the a trainable score function and the score of a data generating process. We will discuss this in more detail in Section 3.2 and mention some other approaches of score estimation in Section 5.2.

3 DENSITY AND SCORE ESTIMATION WITH SLICED SCORE MATCHING

3.1 SLICED SCORE MATCHING

We observe that one dimensional problems are usually much easier to solve than high dimensional ones. Inspired by the idea of Sliced Wasserstein distance (Rabin et al., 2012), we consider projecting $\mathbf{s}_d(\mathbf{x})$ and $\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})$ onto some random direction $\mathbf{v}$ and propose to compare their average difference along that random direction. More specifically, we consider the following objective as a replacement of the Fisher divergence $L(\boldsymbol{\theta})$ in Eq. (1):

$$L(\boldsymbol{\theta}; p_{\mathbf{v}}) \triangleq \frac{1}{2}\mathbb{E}_{p_{\mathbf{v}}}\mathbb{E}_{p_d}\left[(\mathbf{v}^\intercal \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}) - \mathbf{v}^\intercal \mathbf{s}_d(\mathbf{x}))^2\right]. \quad (4)$$

Here $\mathbf{v} \sim p_{\mathbf{v}}$ and $\mathbf{x} \sim p_d$ are independent, and we require $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}\mathbf{v}^\intercal] \succ 0$ and $\mathbb{E}_{p_{\mathbf{v}}}[\|\mathbf{v}\|_2^2] < \infty$. Many examples of $p_{\mathbf{v}}$ satisfy these requirements. For instance, $p_{\mathbf{v}}$ can be a multivariate standard normal ($\mathcal{N}(0, I_D)$), a multivariate Rademacher distribution (the uniform distribution over $\{\pm 1\}^D$), or a uniform distribution on the hypersphere $\mathbb{S}^{D-1}$ (recall that D refers to the dimension of $\mathbf{x}$).

To eliminate the dependence of $L(\boldsymbol{\theta}; p_{\mathbf{v}})$ on $\mathbf{s}_d(\mathbf{x})$, we use integration by parts as in score matching (*cf*., the equivalence between Eq. (1) and (2)). Defining

$$\begin{aligned} J(\boldsymbol{\theta}; p_{\mathbf{v}}) \triangleq \mathbb{E}_{p_{\mathbf{v}}}\mathbb{E}_{p_d}\Big[&\mathbf{v}^\intercal \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\mathbf{v} \\ &+ \frac{1}{2}\left(\mathbf{v}^\intercal \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\right)^2\Big], \quad (5)\end{aligned}$$

the equivalence is summarized in the following theorem.

Theorem 1. *Under some regularity conditions (Assumption 1-3 in Appendix B.2), we have*

$$L(\boldsymbol{\theta}; p_{\mathbf{v}}) = J(\boldsymbol{\theta}; p_{\mathbf{v}}) + \mathrm{C}, \quad (6)$$

where C *is a constant w.r.t.* $\boldsymbol{\theta}$*.*

Other than our requirements on $p_{\mathbf{v}}$, the assumptions are exactly the same as in Theorem 1 of Hyvärinen (2005). We advise the interested readers to read Appendix B.2 for technical statements of the assumptions and a rigorous proof of the theorem.

In practice, given a dataset $\mathbf{x}_1^N$, we draw M projection vectors independently for each point $\mathbf{x}_i$ from $p_{\mathbf{v}}$. The collection of all such vectors $\{\mathbf{v}_{ij}\}_{1\le i\le N, 1\le j\le M}$ are abbreviated as $\mathbf{v}_{11}^{NM}$.We then use the following unbiased estimator of $J(\boldsymbol{\theta}; p_{\mathbf{v}})$

$$\begin{aligned}\hat{J}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) \triangleq \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M} &\mathbf{v}_{ij}^\intercal \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i;\boldsymbol{\theta})\mathbf{v}_{ij} \\ &+ \frac{1}{2}\left(\mathbf{v}_{ij}^\intercal \mathbf{s}_m(\mathbf{x}_i;\boldsymbol{\theta})\right)^2. \quad (7)\end{aligned}$$

Note that when $p_{\mathbf{v}}$ is a multivariate standard normal or multivariate Rademacher distribution, we have $\mathbb{E}_{p_{\mathbf{v}}}[(\mathbf{v}^\intercal \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}))^2] = \|\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\|_2^2$, in which case the second term of $J(\boldsymbol{\theta}; p_{\mathbf{v}})$ can be integrated analytically, and may lead to an estimator with reduced variance:

$$\begin{aligned}\hat{J}_{\mathrm{vr}}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM}) \triangleq \frac{1}{N}\frac{1}{M}\sum_{i=1}^{N}\sum_{j=1}^{M} &\mathbf{v}_{ij}^\intercal \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i;\boldsymbol{\theta})\mathbf{v}_{ij} \\ &+ \frac{1}{2}\|\mathbf{s}_m(\mathbf{x}_i;\boldsymbol{\theta})\|_2^2. \quad (8)\end{aligned}$$

Empirically, we do find that $\hat{J}_{\mathrm{vr}}$ has better performance than $\hat{J}$. We refer to this version as sliced score matching with variance reduction (SSM-VR). In fact, we can leverage $\mathbb{E}_{p_\mathbf{v}}[(\mathbf{v}^\intercal \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}))^2] = \|\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\|_2^2$ to create a control variate for guaranteed reduction of variance (Appendix D). $\hat{J}_{\text{vr}}$ is also closely related to Hutchinson's trace estimator (Hutchinson, 1990), which we will analyze later in Section 4.3.

For sliced score matching, the second derivative term $\mathbf{v}^\intercal \nabla_\mathbf{x} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\mathbf{v}$ is much less computationally expensive than $\operatorname{tr}(\nabla_\mathbf{x} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}))$. Using auto-differentiation systems that support higher order gradients, we can compute it using two gradient operations for a single $\mathbf{v}$, as shown in Alg. 1. Similarly, when there are M $\mathbf{v}$'s, the total number of gradient operations is $M+1$. In contrast, assuming the dimension of $\mathbf{x}$ is D and $D \gg M$, we typically need $D+1$ gradient operations to compute $\operatorname{tr}(\nabla_\mathbf{x} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}))$ because each diagonal entry of $\nabla_\mathbf{x} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})$ needs to be computed separately (see Alg. 2 in the appendix).

Algorithm 1 Sliced Score Matching

```
Input: $\tilde{p}_m(\cdot;\boldsymbol{\theta}), \mathbf{x}, \mathbf{v}$
 1: $\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}) \leftarrow \texttt{grad}(\log \tilde{p}_m(\mathbf{x};\boldsymbol{\theta}), \mathbf{x})$
 2: $\mathbf{v}^\intercal \nabla_\mathbf{x} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}) \leftarrow \texttt{grad}(\mathbf{v}^\intercal \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}), \mathbf{x})$
 3: $J \leftarrow \frac{1}{2}(\mathbf{v}^\intercal \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta}))^2$ (or $J \leftarrow \frac{1}{2}\|\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\|_2^2$)
 4: $J \leftarrow J + \mathbf{v}^\intercal \nabla_\mathbf{x} \mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})\mathbf{v}$
    return $J$
```

In practice, we can tune M to trade off variance and computational cost. In our experiments, we find that oftentimes $\boxed{M=1}$ is already a good choice.

3.2 SLICED SCORE ESTIMATION

As mentioned in section 2.2, the task of score estimation is to estimate $\nabla_\mathbf{x} \log q(\mathbf{x})$ at some test point $\mathbf{x}$, given a set of samples $\mathbf{x}_1^N \overset{\text{i.i.d.}}{\sim} q(\mathbf{x})$. In what follows, we show how our sliced score matching objective $\hat{J}(\boldsymbol{\theta};\mathbf{x}_1^N,\mathbf{v}_{11}^{NM})$ can be straightforwardly adapted for this task.

We propose to use a vector-valued deep neural network $\mathbf{h}(\mathbf{x};\boldsymbol{\theta}) : \mathbb{R}^D \to \mathbb{R}^D$ as our score model. Then, substituting $\mathbf{h}$ into $\hat{J}(\boldsymbol{\theta};\mathbf{x}_1^N,\mathbf{v}_{11}^{NM})$ for $\mathbf{s}_m$, we get

$$\frac{1}{N}\frac{1}{M}\sum_{i=1}^N\sum_{j=1}^M \mathbf{v}_{ij}^\intercal \nabla_\mathbf{x}\mathbf{h}(\mathbf{x}_i;\boldsymbol{\theta})\mathbf{v}_{ij} + \frac{1}{2}\left(\mathbf{v}_{ij}^\intercal \mathbf{h}(\mathbf{x}_i;\boldsymbol{\theta})\right)^2,$$

and we can optimize the objective to get $\hat{\boldsymbol{\theta}}$. Afterwards, $\mathbf{h}(\mathbf{x};\hat{\boldsymbol{\theta}})$ can be used as an approximation to $\nabla_\mathbf{x} \log q(\mathbf{x})$.[1]

[1]Note that $\mathbf{h}(\mathbf{x};\boldsymbol{\theta})$ may not correspond to the gradient of any scalar function. For $\mathbf{h}(\mathbf{x};\boldsymbol{\theta})$ to represent a gradient, one necessary condition is $\nabla_\mathbf{x} \times \mathbf{h}(\mathbf{x};\boldsymbol{\theta}) = \mathbf{0}$ for all $\mathbf{x}$, which may not be satisfied by general networks. However, this is oblivious to the fact that $\mathbf{h}(\mathbf{x};\hat{\boldsymbol{\theta}})$ will be close to $\nabla_\mathbf{x} \log q_{\boldsymbol{\theta}}(\mathbf{x})$ in ℓ_2 norm. As will be shown later, our argument based on integration by parts does not require $h(\mathbf{x};\boldsymbol{\theta})$ to be a gradient.

Using a similar argument of integration by parts (*cf.*, Eq. (4), (5) and (6)), we have

$$\mathbb{E}_{p_\mathbf{v}}\mathbb{E}_{p_d}\left[\mathbf{v}^\intercal \nabla_\mathbf{x}\mathbf{h}(\mathbf{x};\boldsymbol{\theta})\mathbf{v} + \frac{1}{2}(\mathbf{v}^\intercal \mathbf{h}(\mathbf{x};\boldsymbol{\theta}))^2\right] = \frac{1}{2}\mathbb{E}_{p_\mathbf{v}}\mathbb{E}_{p_d}[(\mathbf{v}^\intercal \mathbf{h}(\mathbf{x};\boldsymbol{\theta}) - \mathbf{v}^\intercal \nabla_\mathbf{x} \log q(\mathbf{x}))^2] + \mathrm{C},$$

which implies that minimizing $\hat{J}(\boldsymbol{\theta};\mathbf{x}_1^N,\mathbf{v}_{11}^{NM})$ with $\mathbf{s}_m(\mathbf{x};\boldsymbol{\theta})$ replaced by $\mathbf{h}(\mathbf{x};\boldsymbol{\theta})$ is approximately minimizing the average projected difference between $\mathbf{h}(\mathbf{x};\boldsymbol{\theta})$ and $\nabla_\mathbf{x} \log q(\mathbf{x})$. Hence, $\mathbf{h}(\mathbf{x};\hat{\boldsymbol{\theta}})$ should be close to $\nabla_\mathbf{x} \log q(\mathbf{x})$ and can serve as a score estimator.

4 THEORETICAL ANALYSIS

In this section, we present several theoretical results to justify sliced score matching as a principled objective. We also discuss the connection of sliced score matching to other methods. For readability, we will defer rigorous statements of assumptions and theorems to the Appendix.

4.1 CONSISTENCY

One important question to ask for any statistical estimation objective is whether the estimated parameter is consistent under reasonable assumptions. Our results confirm that for any M, the objective $\hat{J}(\boldsymbol{\theta};\mathbf{x}_1^N,\mathbf{v}_{11}^{NM})$ is consistent under suitable assumptions as $N \to \infty$.

We need several standard assumptions to prove the results rigorously. Let p_m be the normalized density induced by our unnormalized model $\tilde{p}_m$, which is assumed to be normalizable. First, we assume Θ is compact (Assumption 6), and our model family $\{p_m(\mathbf{x};\boldsymbol{\theta}) : \boldsymbol{\theta} \in \Theta\}$ is well-specified and identifiable (Assumption 4). These are standard assumptions used for proving the consistency of MLE (van der Vaart, 1998). We also adopt the assumption in Hyvärinen (2005) that all densities are strictly positive (Assumption 5). Finally, we assume that $p_m(\mathbf{x};\boldsymbol{\theta})$ has some Lipschitz properties (Assumption 7), and $p_\mathbf{v}$ has bounded higher-order moments (Assumption 2, true for all $p_\mathbf{v}$ considered in the experiments). Then, we can prove the consistency of $\hat{\boldsymbol{\theta}}_{N,M} \triangleq \arg\min_{\boldsymbol{\theta}\in\Theta} \hat{J}(\boldsymbol{\theta};\mathbf{x}_1^N,\mathbf{v}_{11}^{NM})$:

Theorem 2 (Consistency)**.** *Assume the conditions of Theorem 1 are satisfied. Assume further that the assumptions discussed above hold. Let $\boldsymbol{\theta}^*$ be the true parameter of the data distribution. Then for every $M \in \mathbb{N}^+$,*

$$\hat{\boldsymbol{\theta}}_{N,M} \xrightarrow{p} \boldsymbol{\theta}^*, \quad N \to \infty.$$

Sketch of proof. We first prove that $J(\boldsymbol{\theta};p_\mathbf{v}) = 0 \Leftrightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*$ (Lemma 1 and Theorem 1). Then we prove the uniform convergence of $\hat{J}(\boldsymbol{\theta};\mathbf{x}_1^N,\mathbf{v}_{11}^{NM})$ (Lemma 3), which holds regardless of M. These two facts lead to consistency. For a complete proof, see Appendix B.3. $\square$

Remark 1. *In Hyvärinen (2005), the authors only showed that $J(\boldsymbol{\theta}) = 0 \Leftrightarrow \boldsymbol{\theta} = \boldsymbol{\theta}^*$, which leads to "local consistency" of score matching. This is a weaker notion of consistency compared to our settings.*

4.2 ASYMPTOTIC NORMALITY

Asymptotic normality results can be very useful for approximate hypothesis testing and comparing different estimators. Below we show that $\hat{\boldsymbol{\theta}}_{N,M}$ is asymptotically normal when $N \to \infty$.

In addition to the assumptions in Section 4.1, we need an extra assumption to prove asymptotic normality. We require $p_m(\mathbf{x}; \boldsymbol{\theta})$ to have a stronger Lipschitz property (Assumption 9).

For simplicity, we denote $\nabla_{\mathbf{x}} h(\mathbf{x})|_{\mathbf{x}=\mathbf{x}'}$ as $\nabla_{\mathbf{x}} h(\mathbf{x}')$, where $h(\cdot)$ is an arbitrary function. In the following, we will only show the asymptotic normality result for a specific $p_{\mathbf{v}}$. More general results are in Appendix B.4.

Theorem 3 (Asymptotic normality, special case)**.** *Assume the assumptions discussed above hold. If $p_{\mathbf{v}}$ is the multivariate Rademacher distribution, we have*

$$\sqrt{N}(\hat{\boldsymbol{\theta}}_{N,M} - \boldsymbol{\theta}^*) \xrightarrow{d} \mathcal{N}(0, \Sigma),$$

where

$$\Sigma \triangleq [\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*)]^{-1} \Bigg(\sum_{1 \le i,j \le D} V_{ij} + \frac{2}{M} \sum_{1 \le i \ne j \le D} W_{ij} \Bigg) \cdot [\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*)]^{-1}. \quad (9)$$

Here D is the dimension of data; V_{ij} and W_{ij} are p.s.d matrices depending on $p_m(\mathbf{x}; \boldsymbol{\theta}^*)$, and their definitions can be found in Appendix B.4.

Sketch of proof. Once we get the consistency (Theorem 2), the rest closely follows the proof of asymptotic normality of MLE (van der Vaart, 1998). A rigorous proof can be found in Appendix B.4. $\square$

Remark 2. *As expected, larger M will lead to smaller asymptotic variance, as can be seen in Eq.* (9)*.*

Remark 3. *As far as we know, there is no published proof of asymptotic normality for the standard (not sliced) score matching. However, by using the same techniques in our proofs, and under similar assumptions, we can conclude that the asymptotic variance of the score matching estimator is $[\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*)]^{-1} \left(\sum_{ij} V_{ij}\right) [\nabla^2_{\boldsymbol{\theta}} J(\boldsymbol{\theta}^*)]^{-1}$ (Corollary 1), which will always be smaller than sliced score matching with multivariate Rademacher projections. However, the gap reduces when M increases.*

4.3 CONNECTION TO OTHER METHODS

Sliced score matching is widely connected to many other methods, and can be motivated from some different perspectives. Here we discuss a few of them.

Connection to NCE. Noise contrastive estimation (NCE), proposed by Gutmann & Hyvärinen (2010), is another principle for training unnormalized statistical models. The method works by comparing $p_m(\mathbf{x}; \boldsymbol{\theta})$ with a noise distribution $p_n(\mathbf{x})$. We consider a special form of NCE which minimizes the following objective

$$-\mathbb{E}_{p_d}[\log h(\mathbf{x}; \boldsymbol{\theta})] - \mathbb{E}_{p_n}[\log(1 - h(\mathbf{x}; \boldsymbol{\theta}))], \quad (10)$$

where $h(\mathbf{x}; \boldsymbol{\theta}) \triangleq \frac{p_m(\mathbf{x};\boldsymbol{\theta})}{p_m(\mathbf{x};\boldsymbol{\theta}) + p_m(\mathbf{x}-\mathbf{v};\boldsymbol{\theta})}$, and we choose $p_n(\mathbf{x}) = p_d(\mathbf{x}+\mathbf{v})$. Assuming $\|\mathbf{v}\|_2$ to be small, Eq. (10) can be written as the following by Taylor expansion

$$\frac{1}{4}\mathbb{E}_{p_d}\left[\mathbf{v}^\intercal \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})\mathbf{v} + \frac{1}{2}(\mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})^\intercal \mathbf{v})^2\right] + 2\log 2 + o(\|\mathbf{v}\|_2^2). \quad (11)$$

For derivation, see Proposition 1 in the appendix. A similar derivation can also be found in Gutmann & Hirayama (2011). As a result of (11), if we choose some $p_{\mathbf{v}}$ and take the expectation of (10) w.r.t. $p_{\mathbf{v}}$, the objective will be equivalent to sliced score matching whenever $\|\mathbf{v}\|_2 \approx 0$.

Connection to Hutchinson's Trick. Hutchinson's trick (Hutchinson, 1990) is a stochastic algorithm to approximate $\operatorname{tr}(A)$ for any square matrix A. The idea is to choose a distribution of a random vector $\mathbf{v}$ such that $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}\mathbf{v}^\intercal] = I$, and then we have $\operatorname{tr}(A) = \mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}^\intercal A \mathbf{v}]$. Hence, using Hutchinson's trick, we can replace $\operatorname{tr}(\nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta}))$ with $\mathbb{E}_{p_{\mathbf{v}}}[\mathbf{v}^\intercal \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}; \boldsymbol{\theta})\mathbf{v}]$ in the score matching objective $J(\boldsymbol{\theta})$. Then the finite sample objective of score matching becomes

$$\frac{1}{N}\sum_{i=1}^{N}\left(\frac{1}{M}\sum_{j=1}^{M} \mathbf{v}_{ij}^\intercal \nabla_{\mathbf{x}} \mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})\mathbf{v}_{ij}\right) + \frac{1}{2}\|\mathbf{s}_m(\mathbf{x}_i; \boldsymbol{\theta})\|_2^2,$$

which is exactly the sliced score matching objective with variance reduction $\hat{J}_{\text{vr}}(\boldsymbol{\theta}; \mathbf{x}_1^N, \mathbf{v}_{11}^{NM})$.

5 RELATED WORK

5.1 SCALABLE SCORE MATCHING

To the best of our knowledge, there are three existing methods that are able to scale up score matching to learning deep models on high dimensional data.

Denoising Score Matching. Vincent (2011) proposes denoising score matching, a variant of score matching that completely circumvents the Hessian. Specifically, consider a noise distribution $q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})$, and let $q_\sigma(\tilde{\mathbf{x}}) = \int q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x}) p_d(\mathbf{x}) \mathrm{d}\mathbf{x}$. Denoising score matching applies the original score matching to the noise-corrupted data distribution $q_\sigma(\tilde{\mathbf{x}})$, and the objective can be proven to be equivalent to the following up to a constant

$$\frac{1}{2}\mathbb{E}_{q_\sigma(\tilde{\mathbf{x}}|\mathbf{x})p_d(\mathbf{x})}[\|\mathbf{s}_m(\tilde{\mathbf{x}};\boldsymbol{\theta}) - \nabla_{\mathbf{x}} \log q_\sigma(\tilde{\mathbf{x}} \mid \mathbf{x})\|_2^2],$$

which can be estimated without computing any Hessian. Although denoising score matching is much faster than score matching, it has obvious drawbacks. First, it can only recover the noise corrupted data distribution. Furthermore, choosing the parameters of the noise distribution is highly non-trivial and in practice the performance can be very sensitive to σ, and heuristics have to be used in practice. For example, Saremi et al. (2018) propose a heuristic for choosing σ based on Parzen windows.

Approximate Backpropagation. Kingma & LeCun (2010) propose a backpropagation method to approximately compute the trace of the Hessian. Because the backpropagation of the full Hessian scales quadratically w.r.t. the layer size, the authors limit backpropagation only to the diagonal so that it has the same cost as the usual backpropagation for computing loss gradients. However, there are no theoretical guarantees for the approximation errors. In fact, the authors only did experiments on networks with a single hidden layer, in which case the approximation is exact. Moreover, there is no direct support for the proposed approximate backpropagation method in modern automatic differentiation frameworks.

Curvature Propagation. Martens et al. (2012) estimate the trace term in score matching by applying curvature propagation to compute an unbiased, complex-valued estimator of the diagonal of the Hessian. The work claims that curvature propagation will have the smallest variance among a class of estimators, which includes the Hutchinson's estimator. However, their proof evaluates the pseudo-variance of the complex-valued estimator instead of the variance. In practice, curvature propagation can have large variance when the number of nodes in the network is large, because it introduces noise for each node in the network. Finally, implementing curvature propagation also requires manually modifying the backpropagation code, handling complex numbers in neural networks, and will be inefficient for networks of more general structures, such as recurrent neural networks.

5.2 KERNEL SCORE ESTIMATORS

Two prior methods for score estimation are based on a generalized form of Stein's identity (Stein, 1981; Gorham & Mackey, 2017):

$$\mathbb{E}_{q(\mathbf{x})}[\mathbf{h}(\mathbf{x})\nabla_{\mathbf{x}} \log q(\mathbf{x})^\intercal + \nabla_{\mathbf{x}}\mathbf{h}(\mathbf{x})] = \mathbf{0}, \qquad (12)$$

where $q(\mathbf{x})$ is a continuously differentiable density and $\mathbf{h}(\mathbf{x})$ is a function satisfying some regularity conditions. Li & Turner (2018) propose to set $\mathbf{h}(\mathbf{x})$ as the feature map of some kernel in the *Stein class* (Liu et al., 2016) of q, and solve a finite-sample version of (12) to obtain estimates of $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ at the sample points. We refer to this method as *Stein* in the experiments. Shi et al. (2018) adopt a different approach but also exploit (12). They build their estimator by expanding $\nabla_{\mathbf{x}} \log q(\mathbf{x})$ as a spectral series of eigenfunctions and solve for the coefficients using (12). Compared to *Stein*, their method is argued to have theoretical guarantees and principled out-of-sample estimation at an unseen test point. We refer to their method as *Spectral* in the experiments.

6 EXPERIMENTS

Our experiments include two parts: (1) to test the effectiveness of sliced score matching (SSM) in learning deep models for density estimation, and (2) to test the ability of SSM in providing score estimates for applications such as VAEs with implicit encoders and WAEs. Unless specified explicitly, we choose $M = 1$ by default.

6.1 DENSITY ESTIMATION

We evaluate SSM and its variance-reduced version (SSM-VR) for density estimation and compare against score matching (SM) and its three scalable baselines: denoising score matching (DSM), approximate backpropagation (approx BP), and curvature propagation (CP). All SSM methods in this section use the multivariate Rademacher distribution as $p_{\mathbf{v}}$. Our results demonstrate that: (1) SSM is comparable in performance to SM, (2) SSM outperforms other computationally efficient approximations to SM, and (3) unlike SM, SSM scales effectively to high dimensional data.

6.1.1 Deep Kernel Exponential Families

Model. Deep kernel exponential families (DKEF) are unnormalized density models trained using SM (Wenliang et al., 2019). DKEFs parameterize the unnormalized log density as $\log \tilde{p}_f(\mathbf{x}) = f(\mathbf{x}) + \log q_0(\mathbf{x})$, where f is a mixture of Gaussian kernels evaluated at different inducing points: $f(\mathbf{x}) = \sum_{l=1}^{L} \alpha_l k(\mathbf{x}, \mathbf{z}_l)$. The kernel is defined on the feature space of a neural network, and the

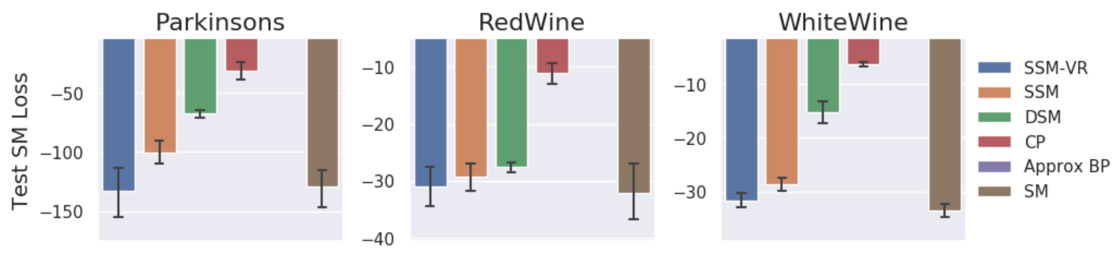

Figure 1: SM loss after training DKEF models on UCI datasets with different loss functions; lower is better. Results for approximate backprapogation are not shown because losses were larger than 10^9.

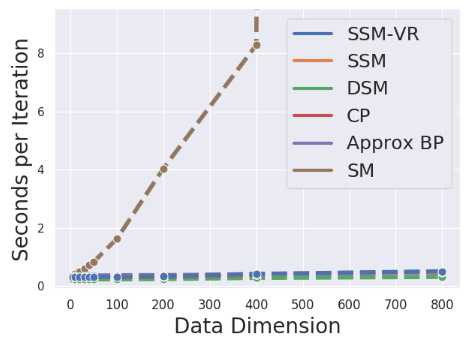

Figure 2: SM performance degrades linearly with the data dimension, while efficient approaches have relatively similar performance.

	Test SM Loss	Test LL
MLE	-579	**-791**
SSM-VR	**-8054**	-3355
SSM	-2428	-2039
DSM ($\sigma = 0.10$)	-3035	-4363
DSM ($\sigma = 1.74$)	-97	-8082
CP	-1694	-1517
Approx BP	-48	-2288

Table 1: Score matching losses and log-likelihoods for NICE models on MNIST. $\sigma = 0.1$ is by grid search and $\sigma = 1.74$ is from the heuristic of Saremi et al. (2018).

network parameters are trained along with the inducing points $\mathbf{z}_l$. Further details of the model, which is identical to that in Wenliang et al. (2019), are presented in Appendix C.1.

Setup. Following the settings of Wenliang et al. (2019), we trained models on three UCI datasets: Parkinsons, RedWine, and WhiteWine (Dua & Graff, 2017), and used their original code for SM. To compute the trace term exactly, Wenliang et al. (2019)'s manual implementation of backpropagation takes over one thousand lines for a model that is four layers deep, while the implementation of SSM only takes several lines. For DSM, the value of σ is chosen by grid search using the SM loss on a validation dataset. All models are trained with 15 different random seeds and training is stopped when validation loss does not improve for 200 steps.

Results. Results in Fig. 1 demonstrate that SSM-VR performs comparably to SM, when evaluated using the SM loss on a held-out test set. We observe that variance reduction substantially improves the performance of SSM. In addition, SSM outperforms other computationally efficient approaches. DSM can perform comparably to SSM on RedWine. However, it is challenging to select σ for DSM. Models trained using σ from the heuristic in Saremi et al. (2018) are far from optimal (on both SM losses and likelihoods), and extensive grid search is needed to find the best σ. CP performs substantially worse, which is likely because it injects noise for each node in the computation graph, and the amount of noise introduced is too large for a neural-network-based kernel evaluated at 200 inducing points, which supports our hypothesis that CP does not work effectively for deep models. Results for approx BP are omitted because the losses exceeded 10^9 on all datasets. This is because approx BP provides a biased estimate of the Hessian without any error guarantees. All the results are similar when evaluating according to log-likelihood metric (Appendix C.1).

Scalability to High Dimensional Data. We evaluate the computational efficiency of different losses on data of increasing dimensionality. We fit DKEF models to a multivariate standard normal in high dimensional spaces. The average running time per minibatch of 100 examples is reported in Fig. 2. SM performance degrades linearly with the dimension of the input data due to the computation of the Hessian, and creates out of memory errors for a 12GB GPU after the dimension increases beyond 400. SSM performs similarly to DSM, approx BP and CP, and they are all scalable to large dimensions.

6.1.2 Deep Flow Models

Setup. As a sanity check, we also evaluate SSM by training a NICE flow model (Dinh et al., 2015), whose likelihood is tractable and can be compared to results obtained by MLE. The model we use has 20 hidden layers, and 1000 units per layer. Models are trained to fit MNIST handwritten digits, which are 784 dimensional images. Data are dequantized by adding uniform noise in the range $[-\frac{1}{512}, \frac{1}{512}]$, and transformed using a logit transformation, $\log(x) - \log(1-x)$. We provide additional details in Appendix C.2.

Training with SM is extremely computationally expensive in this case. Our SM implementation based on auto-differentiation takes around 7 hours to finish one epoch of training, and 12 GB GPU memory cannot hold a batch size larger than 24, so we do not include these results. Since NICE has tractable likelihoods, we also evaluate MLE as a surrogate objective for minimizing the SM loss. Notably, likelihoods and SM losses might be uncorrelated when the model is mis-specified, which is likely to be the case for a complex dataset like MNIST.

Results. SM losses and log-likelihoods on the test dataset are reported in Tab. 1, where models are evaluated using the best checkpoint in terms of the SM loss on a validation dataset over 100 epochs of training. SSM-VR greatly outperforms all the other methods on the SM loss. DSM performs worse than SSM-VR, and σ is still hard to tune. Specifically, following the heuristic in Saremi et al. (2018) leads to $\sigma = 1.74$, which performed the worst (on both log-likelihood and SM loss) of the eight choices of σ in our grid search. Approx BP has more success on NICE than for training DKEF models. This might be because the Hessians of hidden layers of NICE are closer to a diagonal matrix, which results in a smaller approximation error for approx BP. As in the DKEF experiments, CP performs worse. This is likely due to injecting noise to all hidden units, which will lead to large variance for a network as big as NICE. Unlike the DKEF experiments, we find that good log-likelihoods are less correlated with good SM loss, probably due to model mis-specification.

6.2 SCORE ESTIMATION

We consider two typical tasks that require accurate score estimations: (1) training VAEs with an implicit encoder and (2) training Wasserstein Auto-Encoders. We show in both tasks SSM outperforms previous score estimators. Samples generated by various algorithms can be found in Appendix A.

	VAE		WAE	
Latent Dim	8	32	8	32
ELBO	96.87	**89.06**	N/A	N/A
SSM	**95.50**	89.25 (**88.29**†)	**98.24**	**90.37**
Stein	96.71	91.84	99.05	91.70
Spectral	96.60	94.67	98.81	92.55

Table 2: Negative log-likelihoods on MNIST, estimated by AIS. †The result of SSM with $M = 100$, in which case SSM matches the computational cost of kernel methods, which used 100 samples for each data point.

	Method \ Iteration	10k	40k	70k	100k
VAE	ELBO	**96.20**	73.70	69.42	66.32
	SSM	108.52	**70.28**	**66.52**	**62.50**
	Stein	126.60	118.87	120.51	126.76
	Spectral	131.90	125.04	128.36	133.93
WAE	SSM	84.11	**61.09**	**56.23**	**54.33**
	Stein	82.93	63.46	58.53	57.61
	Spectral	**82.30**	62.47	58.03	55.96

Table 3: FID scores of different methods versus number of training iterations on CelebA dataset.

6.2.1 VAE with Implicit Encoders

Background. Consider a latent variable model $p(\mathbf{x}, \mathbf{z})$, where $\mathbf{x}$ and $\mathbf{z}$ denote observed and latent variables respectively. A variational auto-encoder (VAE) is composed of two parts: 1) an encoder $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ modeling the conditional distribution of $\mathbf{x}$ given $\mathbf{z}$; and a decoder $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ that approximates the posterior distribution of the latent variable. A VAE is typically trained by maximizing the following evidence lower bound (ELBO)

$$\mathbb{E}_{p_d}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})p(\mathbf{z}) - \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})],$$

where $p(\mathbf{z})$ denotes a pre-specified prior distribution. The expressive power of $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is critical to the success of variational learning. Typically, $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is chosen to be a simple distribution with tractable densities so that $H(q_{\boldsymbol{\phi}}) \triangleq -\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ is tractable. We call this traditional approach "ELBO" in the experiments. With score estimation techniques, we can directly compute $\nabla_{\boldsymbol{\phi}} H(q_{\boldsymbol{\phi}})$ for implicit distributions, which enables more flexible options for $q_{\boldsymbol{\phi}}$. We consider 3 score estimation techniques: SSM, Stein (Li & Turner, 2018) and Spectral (Shi et al., 2018).

For a single data point $\mathbf{x}$, kernel score estimators need multiple samples from $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ to estimate its score. On MNIST, we use 100 samples, as done in Shi et al. (2018). On CelebA, however, we can only take 20 samples due to GPU memory limitations. In contrast, SSM learns a score network $h(\mathbf{z} \mid \mathbf{x})$ along with $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$, which amortizes the cost of score estimation. Unless noted explicitly, we use one projection per data point ($M = 1$) for SSM, which is scalable to deep networks.

Setup. We consider VAE training on MNIST and CelebA (Liu et al., 2015). All images in CelebA are first cropped to a patch of 140×140 and then resized to 64×64. We report negative likelihoods on MNIST, as estimated by AIS (Neal, 2001) with 1000 intermediate distributions. We evaluate sample quality on CelebA with FID scores (Heusel et al., 2017). For fast AIS evaluation on MNIST, we use shallow fully-connected networks with 3 hidden layers. For CelebA experiments we use deep convolutional networks. The architectures of implicit encoders and score networks are straightforward modifications to the encoders of ELBO. More details are provided in Appendix C.3.

Results. The negative likelihoods of different methods on MNIST are reported in the left part of Tab. 2. We note that SSM outperforms Stein and Spectral in all cases. When the latent dimension is 8, SSM outperforms ELBO, which indicates that the expressive power of implicit $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ pays off. When the latent dimension is 32, the gaps between SSM and kernel methods are even larger, and the performance of SSM is still comparable to ELBO. Moreover, when $M = 100$ (matching the computation of kernel methods), SSM outperforms ELBO.

For CelebA, we provide FID scores of samples in the top part of Tab. 3. We observe that after 40k training iterations, SSM outperforms all other baselines. Kernel methods perform poorly in this case because only 20 samples per data point can be used due to limited amount of GPU memory. Early during training, SSM does not perform as well. Since the score network is trained along with the encoder and decoder, a moderate number of training steps is needed to give an accurate score estimation (and better learning of the VAE).

6.2.2 WAEs

Background. WAE is another method to learn latent variable models, which generally produces better samples than VAEs. Similar to a VAE, it contains an encoder $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ and a decoder $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ and both can be implicit distributions. Let $p(\mathbf{z})$ be a pre-defined prior distribution, and $q_{\boldsymbol{\phi}}(\mathbf{z}) \triangleq \int q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) p_d(\mathbf{x}) \mathrm{d}\mathbf{x}$ denote the aggregated posterior distribution. Using a metric function $c(\cdot, \cdot)$ and KL divergence between $q_{\boldsymbol{\phi}}(\mathbf{z})$ and $p(\mathbf{z})$, WAE minimizes the following objective

$$\mathbb{E}_{p_d}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[c(\mathbf{x}, p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})) - \lambda \log p(\mathbf{z})]] - \lambda H(q_{\boldsymbol{\phi}}(\mathbf{z})).$$

Compared to $\nabla_{\boldsymbol{\phi}} H(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}))$, it is easier to estimate $\nabla_{\boldsymbol{\phi}} H(q_{\boldsymbol{\phi}}(\mathbf{z}))$ for kernel methods, because the samples of $q_{\boldsymbol{\phi}}(\mathbf{z})$ can be collected by first sampling $\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N \overset{\text{i.i.d.}}{\sim} p_d(\mathbf{x})$ and then sample one $\mathbf{z}$ for each $\mathbf{x}_i$ from $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}_i)$. In constrast, multiple $\mathbf{z}$'s need to be sampled for each $\mathbf{x}_i$ when estimating $\nabla_{\boldsymbol{\phi}} H(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}))$ with kernel approaches. For SSM, we use a score network $h(\mathbf{z})$ and train it alongside $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$.

Setup. The setup for WAE experiments is largely the same as VAE. The architectures are very similar to those used in the VAE experiments, and the only difference is that we made decoders implicit, as suggested in Tolstikhin et al. (2018). More details can be found in Appendix C.3.

Results. The negative likelihoods on MNIST are provided in the right part of Tab. 2. SSM outperforms both kernel methods, and achieves a larger performance gap when the latent dimension is higher. The likelihoods are lower than the VAE results as the WAE objective does not directly maximize likelihoods.

We show FID scores for CelebA experiments in the bottom part of Tab. 3. As expected, kernel methods perform much better than before, because it is faster to sample from $q_{\boldsymbol{\phi}}(\mathbf{z})$. The FID scores are generally lower than those in VAE experiments, which supports previous results that WAEs generally obtain better samples. SSM outperforms both kernel methods when the number of iterations is more than 40k. This shows the advantages of training a deep, expressive score network compared to using a fixed kernel in score estimation tasks.

7 CONCLUSION

We propose sliced score matching, a scalable method for learning unnormalized statistical models and estimating scores for implicit distributions. Compared to the original score matching and its variants, our estimator can scale to deep models on high dimensional data, while remaining easy to implement in modern automatic differentiation frameworks. Theoretically, our estimator is consistent and asymptotically normal under some regularity conditions. Experimental results demonstrate that our method outperforms competitors in learning deep energy-based models and provides more accurate estimates than kernel score estimators in training implicit VAEs and WAEs.

Acknowledgements

This research was supported by Intel Corporation, TRI, NSF (#1651565, #1522054, #1733686), ONR (N00014-19-1-2145), AFOSR (FA9550- 19-1-0024).

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pp. 265–283, 2016.

Adam, P., Soumith, C., Gregory, C., Edward, Y., Zachary, D., Zeming, L., Alban, D., Luca, A., and Adam, L. Automatic differentiation in pytorch. In *NIPS Autodiff Workshop*, 2017.

Canu, S. and Smola, A. Kernel methods and the exponential family. *Neurocomputing*, 69(7):714 – 720, 2006. ISSN 0925-2312. New Issues in Neurocomputing: 13th European Symposium on Artificial Neural Networks.

Dinh, L., Krueger, D., and Bengio, Y. NICE: Non-linear independent components estimation. *International Conference in Learning Representations Workshop Track*, 2015.

Dua, D. and Graff, C. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Dudley, R. M. The sizes of compact subsets of Hilbert space and continuity of Gaussian processes. *Journal of Functional Analysis*, 1(3):290–330, 1967.

Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pp. 2672–2680, 2014.

Gorham, J. and Mackey, L. Measuring sample quality with kernels. In *Proceedings of the 34th International Conference on Machine Learning*, pp. 1292–1301, 2017.

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

Gutmann, M. U. and Hirayama, J.-i. Bregman divergence as general framework to estimate unnormalized statistical models. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*, pp. 283–290. AUAI Press, 2011.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. GANs trained by a two time-scale update rule converge to a local Nash equilibrium. In *Advances in Neural Information Processing Systems*, pp. 6626–6637, 2017.

Huszár, F. Variational inference using implicit distributions. *arXiv preprint arXiv:1702.08235*, 2017.

Hutchinson, M. F. A stochastic estimator of the trace of the influence matrix for Laplacian smoothing splines. *Communications in Statistics-Simulation and Computation*, 19(2):433–450, 1990.

Hyvärinen, A. Estimation of non-normalized statistical models by score matching. *Journal of Machine Learning Research*, 6(Apr):695–709, 2005.

Kingma, D. P. and LeCun, Y. Regularized estimation of image statistics by score matching. In *Advances in Neural Information Processing Systems*, pp. 1126–1134, 2010.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. *International Conference on Learning Representations*, 2014.

LeCun, Y., Chopra, S., Hadsell, R., Ranzato, M. A., and Huang, F. J. A tutorial on energy-based learning. In *Predicting structured data*. MIT Press, 2006.

Li, Y. and Turner, R. E. Gradient estimators for implicit models. In *International Conference on Learning Representations*, 2018.

Liu, Q., Lee, J., and Jordan, M. A kernelized Stein discrepancy for goodness-of-fit tests. In *Proceedings of The 33rd International Conference on Machine Learning*, pp. 276–284, 2016.

Liu, Z., Luo, P., Wang, X., and Tang, X. Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*, 2015.

Martens, J., Sutskever, I., and Swersky, K. Estimating the Hessian by back-propagating curvature. *Proceedings of the 29th International Conference on Machine Learning*, 2012.

Neal, R. M. Annealed importance sampling. *Statistics and computing*, 11(2):125–139, 2001.

Owen, A. B. *Monte Carlo theory, methods and examples*. 2013.

Rabin, J., Peyré, G., Delon, J., and Bernot, M. Wasserstein barycenter and its application to texture mixing. In Bruckstein, A. M., ter Haar Romeny, B. M., Bronstein, A. M., and Bronstein, M. M. (eds.), *Scale Space and Variational Methods in Computer Vision*, pp. 435–446, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg. ISBN 978-3-642-24785-9.

Saremi, S., Mehrjou, A., Schölkopf, B., and Hyvärinen, A. Deep energy estimator networks. *arXiv preprint arXiv:1805.08306*, 2018.

Sasaki, H., Hyvärinen, A., and Sugiyama, M. Clustering via mode seeking by direct estimation of the gradient of a log-density. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 19–34. Springer, 2014.

Shi, J., Sun, S., and Zhu, J. A spectral approach to gradient estimation for implicit distributions. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 4651–4660, 2018.

Sriperumbudur, B., Fukumizu, K., Gretton, A., Hyvärinen, A., and Kumar, R. Density estimation in infinite dimensional exponential families. *Journal of Machine Learning Research*, 18(57):1–59, 2017.

Stein, C. M. Estimation of the mean of a multivariate normal distribution. *The annals of Statistics*, pp. 1135–1151, 1981.

Strathmann, H., Sejdinovic, D., Livingstone, S., Szabo, Z., and Gretton, A. Gradient-free Hamiltonian monte carlo with efficient kernel exponential families. In *Advances in Neural Information Processing Systems*, pp. 955–963, 2015.

Sun, S., Zhang, G., Shi, J., and Grosse, R. Functional variational Bayesian neural networks. In *International Conference on Learning Representations*, 2019.

Sutherland, D., Strathmann, H., Arbel, M., and Gretton, A. Efficient and principled score estimation with Nyström kernel exponential families. In *Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics*, pp. 652–660, 2018.

Tolstikhin, I., Bousquet, O., Gelly, S., and Schoelkopf, B. Wasserstein auto-encoders. In *International Conference on Learning Representations*, 2018.

Tran, D., Ranganath, R., and Blei, D. Hierarchical implicit models and likelihood-free variational inference. In *Advances in Neural Information Processing Systems*, pp. 5523–5533, 2017.

van der Vaart, A. W. *Asymptotic Statistics*. Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 1998. doi: 10.1017/CBO9780511802256.

Vincent, P. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7): 1661–1674, 2011.

Wenliang, L., Sutherland, D., Strathmann, H., and Gretton, A. Learning deep kernels for exponential family densities. In Chaudhuri, K. and Salakhutdinov, R. (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 6737–6746, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL `http://proceedings.mlr.press/v97/wenliang19a.html`.