Supplementary Material for Estimating the Rate-Distortion Function by Wasserstein Gradient Descent

Anonymous Author(s)
Affiliation
Address
email

We review probability theory background and explain our notation from the main text in Section 1, elaborate on the connections between the R-D estimation problem and variational inference/learning in Section 2, give proofs of formal results for Wasserstein gradient descent in Section 3, provide an example implementation in Section 4 (the full implementation can be found in the zip file), and finally provide additional experimental results and details in Section 5.

1 Notions from probability theory

In this section we collect notions of probability theory used in the main text. See, e.g., [Çinlar, 2011] or [Folland, 1999] for more background.

Marginal and conditional distributions. The source and reproduction spaces $\mathcal{X}, \mathcal{Y}$ are equipped with sigma-algebras $\mathcal{A}_\mathcal{X}$ and $\mathcal{A}_\mathcal{Y}$, respectively. Let $\mathcal{X} \times \mathcal{Y}$ denote the product space equipped with the product sigma algebra $\mathcal{A}_\mathcal{X} \otimes \mathcal{A}_\mathcal{Y}$. For any probability measure π on $\mathcal{X} \times \mathcal{Y}$, its first **marginal** is

$$\pi_1(A) := \pi(A \times \mathcal{Y}), \quad A \in \mathcal{A}_\mathcal{X},$$

which is a probability measure on $\mathcal{X}$. When π is the distribution of a random vector (X, Y), then π_1 is the distribution of X. The second marginal of π is defined analogously as

$$\pi_2(B) := \pi(\mathcal{X} \times B), \quad B \in \mathcal{A}_\mathcal{Y}.$$

A Markov **kernel** or **conditional distribution** $K(x, dy)$ is a map $\mathcal{X} \times \mathcal{A}_\mathcal{Y} \to [0, 1]$ such that

1. $K(x, \cdot)$ is a probability measure on $\mathcal{Y}$ for each $x \in \mathcal{X}$;
2. the function $x \mapsto K(x, B)$ is measurable for each set $B \in \mathcal{A}_\mathcal{Y}$.

When speaking of the conditional distribution of a random variable Y given another random variable X, we occasionally also use the notation $Q_{Y|X}$ from information theory [Polyanskiy and Wu, 2014]. Then, $Q_{Y|X=x}(B) = K(x, B)$ is the conditional probability of the event $\{Y \in B\}$ given $X = x$.

Suppose that a probability measure μ on $\mathcal{X}$ is given, in addition to a kernel $K(x, dy)$. Together they define a unique measure $\mu \otimes K$ on the product space $\mathcal{X} \times \mathcal{Y}$. For a rectangle set $A \times B \in \mathcal{A}_\mathcal{X} \otimes \mathcal{A}_\mathcal{Y}$,

$$\mu \otimes K(A \times B) = \int_A \mu(dx) K(x, B), \quad A \in \mathcal{A}_\mathcal{X}, B \in \mathcal{A}_\mathcal{Y}.$$

The measure $\pi := \mu \otimes K$ has first marginal $\pi_1 = \mu$.

The classic product measure is a special case of this construction. Namely, when a measure ν on $\mathcal{Y}$ is given, using the constant kernel $K(x, dy) := \nu(dy)$ (which does not depend on x) gives rise to the product measure $\mu \otimes \nu$,

$$\mu \otimes \nu(A \times B) = \mu(A)\nu(B), \quad A \in \mathcal{A}_\mathcal{X}, B \in \mathcal{A}_\mathcal{Y}.$$

Submitted to 37th Conference on Neural Information Processing Systems (NeurIPS 2023). Do not distribute.

Table 1: Guide to notation and their interpretations in various problem domains. "LVM" stands for latent variable modeling, "NPMLE" stands for non-parametric MLE. The R-D problem (3) is equivalent to a "projection" problem in entropic optimal transport (discussed in Sec. 2.2) and statistical problems involving maximum-likelihood estimation (see discussion in Sec. 2.3 and below).

Context	$\mu = P_X$	$\rho(x, y)$	$K = Q_{Y\|X}$	$\nu = Q_Y$
OT	source distribution	transport cost	"transport plan"	target distribution
R-D	data distribution	distortion criterion	compression algorithm	codebook distribution
LVM/NPMLE	data distribution	"$-\log p(x\|y)$"	variational posterior	prior distribution
deconvolution	noisy measurements	"noise kernel"	—	noiseless model

Under mild conditions (for instance when $\mathcal{X}, \mathcal{Y}$ are Polish spaces equipped with their Borel sigma algebras, as in the main text), any probability measure π on $\mathcal{X} \times \mathcal{Y}$ is of the above form. Namely, the **disintegration** theorem asserts that π can be written as $\pi = \pi_1 \otimes K$ for some kernel K. When π is the joint distribution of a random vector (X, Y), this says that there is a measurable version of the conditional distribution $Q_{Y|X}$.

Optimal transport. Given a measure μ on $\mathcal{X}$ and a measurable function $T : \mathcal{X} \to \mathcal{Y}$, the **pushforward** (or image measure) of μ under T is a measure on $\mathcal{Y}$, given by

$$T_{\#}\mu(B) = \mu(T^{-1}(B)), \quad B \in \mathcal{A}_{\mathcal{Y}}.$$

If T is seen as a random variable and μ as the baseline probability measure, then $T_{\#}\mu$ is simply the distribution of T.

Suppose that μ and ν are probability measures on $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ with finite second moment. As introduced in the main text, $\Pi(\mu, \nu)$ denotes the set of couplings, i.e., measures π on $\mathcal{X} \times \mathcal{Y}$ with $\pi_1 = \mu$ and $\pi_2 = \nu$. The 2-Wasserstein distance $W_2(\mu, \nu)$ between μ and ν is defined as

$$W_2(\mu, \nu) = \left(\inf_{\pi \in \Pi(\mu,\nu)} \int \|y - x\|^2 \pi(dx, dy) \right)^{1/2}.$$

This indeed defines a metric on the space of probability measures with finite second moment.

2 R-D estimation and variational inference/learning

In this section, we give a more detailed explanation of how the the R-D problem (3) relates to variational inference and learning in latent variable models.

To facilitate the discussion and make clearer the connections, we adopt notation more common in statistics and information theory. Table 1 summarizes the notation and the correspondence to the measure-theoretic notation used in the main text.

In statistical modeling, the goal is to fit a density $\hat{p}(x)$ to the true (unknown) data distribution P_X. Consider specifying $\hat{p}(x)$ as a latent variable model, where $\mathcal{Y}$ takes on the role of a latent space, and $Q_Y = \nu$ is the distribution of a latent variable Y (which may encapsulate the model parameters). As we shall see, the optimization objective defining the rate functional (5) corresponds to an aggregate Evidence LOwer Bound (ELBO) [Blei et al., 2017]. Thus, computing the rate functional corresponds to variational inference [Blei et al., 2017] in a given model (see Sec. 2.2), and the parametric R-D estimation problem, i.e.,

$$\inf_{\nu \in \mathcal{H}} \mathcal{L}_{BA}(\nu),$$

is equivalent to estimating a model using the variational EM algorithm [Beal and Ghahramani, 2003] (see Sec. 2.3). The variational EM algorithm can be seen as a restricted version of the BA algorithm (see Sec. 2.3), whereas the EM algorithm [Dempster et al., 1977] shares its E-step with the BA algorithm but can differ in its M-step (see Sec. 2.4).

2.1 Setup

For concreteness, fix a reference measure ζ on $\mathcal{Y}$, and suppose Q_Y has density $q(y)$ w.r.t. ζ. Often the latent space $\mathcal{Y}$ is a Euclidean space, and $q(y)$ is the usual probability density function w.r.t. the

Lebesgue measure ζ; or when the latent space is discrete/countable, ζ is the counting measure and $q(y)$ is the usual probability mass function. We will consider the typical parametric estimation problem and choose a particular parametric form for Q_Y indexed by a parameter vector θ. This defines a parametric family $\mathcal{H} = \{Q_Y^\theta : \theta \in \Theta\}$ for some parameter space Θ. Finally, suppose the distortion function ρ induces a conditional likelihood density, $p(x|y) \propto e^{-\lambda\rho(x,y)}$, with a normalization constant that has no y-dependence.

A latent variable model is then specified by the joint density $q(y)p(x|y)$. We use it to posit a density for the data by

$$\hat{p}(x) = \int_{\mathcal{Y}} p(x|y) dQ_Y(y) = \int_{\mathcal{Y}} p(x|y)q(y)\zeta(dy). \tag{16}$$

As a simple example, a Gaussian mixture model with isotropic component variances can be specified as follows. Let Q_Y be a mixing distribution on $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ parameterized by component weights $w_{1,\ldots,k}$ and locations $\mu_{1,\ldots,k}$, such that $Q_Y = \sum_{k=1}^K w_k \delta_{\mu_k}$. Let $p(x|y) = \mathcal{N}(y, \sigma^2)$ be a conditional Gaussian density with mean y and variance σ^2. Now formula (16) gives the usual Gaussian mixture density on $\mathbb{R}^d$.

Maximum-likelihood estimation then ideally maximizes the population log (marginal) likelihood,

$$\mathbb{E}_{x\sim P_X}[\log \hat{p}(x)] = \int \log \hat{p}(x) P_X(dx) = \int \log \left(\int_{\mathcal{Y}} p(x|y) dQ_Y(y) \right) P_X(dx). \tag{17}$$

To deal with the often intractable marginal likelihood in the inner integral, we turn to variational inference and learning [Jordan et al., 1999, Wainwright et al., 2008].

2.2 Connection to variational inference

Given a latent variable model and any data observation x, a central task in Bayesian statistics is to infer the Bayesian posterior [Jordan, 1999], which we formally view as a conditional distribution $Q^*_{Y|X=x}$. It is given by

$$\frac{dQ^*_{Y|X=x}(y)}{dQ_Y(y)} = \frac{p(x|y)}{\hat{p}(x)},$$

or, using the density $q(y)$ of Q_Y, given by the following conditional density via the familiar Bayes' rule,

$$q^*(y|x) = \frac{p(x|y)q(y)}{\hat{p}(x)} = \frac{p(x|y)q(y)}{\int_{\mathcal{Y}} p(x|y)q(y)\zeta(dy)}.$$

Unfortunately, the true Bayesian posterior is typically intractable, as the (marginal) data likelihood in the denominator involves an often high-dimensional integral. Variational inference [Jordan et al., 1999, Wainwright et al., 2008] therefore aims to approximate the true posterior by a variational distribution $Q_{Y|X=x} \in \mathcal{P}(\mathcal{Y})$ by minimizing their relative divergence $H(Q_{Y|X=x}|Q^*_{Y|X=x})$. The problem is equivalent to maximizing the following lower bound on the marginal log-likelihood, known as the Evidence Lower BOund (ELBO) [Blei et al., 2017]:

$$\begin{aligned}
\arg\min_{Q_{Y|X=x}} H(Q_{Y|X=x}|Q^*_{Y|X=x}) &= \arg\max_{Q_{Y|X=x}} \mathrm{ELBO}(Q_Y, x, Q_{Y|X=x}), \\
\mathrm{ELBO}(Q_Y, x, Q_{Y|X=x}) &= \mathbb{E}_{y\sim Q_{Y|X=x}}[\log p(x|y)] - H(Q_{Y|X=x}|Q_Y) \\
&= \log \hat{p}(x) - H(Q_{Y|X=x}|Q^*_{Y|X=x}).
\end{aligned} \tag{18}$$

Translating the definition of the rate functional (5) into the present scenario,

$$\begin{aligned}
\mathcal{L}_{BA}(Q_Y) &= \inf_{Q_{Y|X}} \mathbb{E}_{x\sim P_X, y\sim Q_{Y|X=x}}[-\log p(x|y)] + \mathbb{E}_{x\sim P_X}[H(Q_{Y|X=x}|Q_Y)] + \text{const} \\
&= \inf_{Q_{Y|X}} \mathbb{E}_{x\sim P_X}[-\mathrm{ELBO}(Q_Y, x, Q_{Y|X=x})] + \text{const},
\end{aligned} \tag{19}$$

we recognize that the rate functional optimizes the population ELBO, and this optimization problem decouples over x and can be solved by the variational inference problem (18) involving $Q_{Y|X=x}$. At optimality, $Q_{Y|X} = Q^*_{Y|X}$, the ELBO (18) is tight and recovers $\log \hat{p}(x)$, and the rate functional takes on the form of a (negated) population marginal log likelihood (17), as given earlier by (6) in Sec. 2.1.

2.3 Connection to variational EM

The discussion so far concerns *probabilistic inference*, where a latent variable model $(Q_Y, p(x|y))$ has been given and we saw that computing the rate functional amounts to variational inference. Suppose now we wish to *learn* a model from data. The R-D problem (4) then corresponds to model estimation using the variational EM algorithm [Beal and Ghahramani, 2003].

To estimate a latent variable model by (approximate) maximum-likelihood, the variational EM algorithm maximizes the population ELBO

$$\mathbb{E}_{x\sim P_X}[\mathrm{ELBO}(Q_Y, x, Q_{Y|X=x})] = \mathbb{E}_{x\sim P_X, y\sim Q_{Y|X=x}}[\log p(x|y)] - \mathbb{E}_{x\sim P_X}[H(Q_{Y|X=x}|Q_Y)], \tag{20}$$

w.r.t. Q_Y and $Q_{Y|X}$. This precisely corresponds to the R-D problem $\inf_{Q_Y\in\mathcal{H}} \mathcal{L}_{BA}(Q_Y)$, using the form of $\mathcal{L}_{BA}(Q_Y)$ from (19).

In popular implementations of variational EM such as the VAE [Kingma and Welling, 2013], Q_Y and $Q_{Y|X}$ are restricted to parametric families. When they are allowed to range over all of $\mathcal{P}(\mathcal{Y})$ and all conditional distributions, variational EM then becomes equivalent to the BA algorithm.

2.4 The Blahut–Arimoto and EM algorithms

The BA and EM algorithms share the same objective function, namely the negative of the population ELBO from (20). Both also perform coordinate descent / alternating projection, but they define the coordinates slightly differently — the BA algorithm uses $(Q_{Y|X}, Q_Y)$ with $Q_Y \in \mathcal{P}(\mathcal{Y})$, whereas the EM algorithm uses $(Q_{Y|X}, \theta)$ with θ indexing a parametric family $\mathcal{H} = \{Q_Y^\theta : \theta \in \Theta\}$. Thus the coordinate update w.r.t. $Q_{Y|X}$ (the "E-step") is the same in both algorithms, but the subseuquent "M-step" potentially differs depending on the role of θ.

Given the optimization objective,

$$\mathbb{E}_{x\sim P_X, y\sim Q_{Y|X=x}}[-\log p(x|y)] + H(P_X Q_{Y|X}|P_X \otimes Q_Y),$$

both the BA and EM algorithms optimize the transition kernel $Q_{Y|X}$ the same way in the E-step, as

$$\frac{dQ^*_{Y|X=x}}{dQ_Y}(y) = \frac{p(x|y)}{p(x)}. \tag{21}$$

For the M-step, the BA algorithm performs

$$\min_{Q_Y\in\mathcal{P}(\mathcal{Y})} H(P_X Q^*_{Y|X}; P_X \otimes Q_Y),$$

whereas the EM algorithm minimizes the full objective w.r.t. the parameters θ of Q_Y,

$$\min_{\theta\in\Theta} \mathbb{E}_{(x,y)\sim P_X Q^*_{Y|X}}[-\log p(x|y)] + H(P_X Q^*_{Y|X}; P_X \otimes Q_Y). \tag{22}$$

The difference comes from the fact that when we parameterize Q_Y by θ in the parameter estimation problem, $Q^*_{Y|X}$ — and consequently both terms in the objective of (22) — will have functional dependence on θ through the E-step optimality condition (21).

In the Gaussian mixture example, $Q_Y = \sum_{k=1}^K w_k \delta_{\mu_k}$, and its parameters θ consist of the components weights $(w_1, ..., w_K) \in \Delta^{d-1}$ and location vectors $\{\mu_1, ..., \mu_K\} \subset \mathbb{R}^d$. The E-step computes $Q^*_{Y|X=x} = \sum_k w_k \frac{p(x|\mu_k)}{p(x)} \delta_{\mu_k}$. For the M-step, if we regard the locations as known so that $\theta = (w_1, ..., w_K)$ only consists of the weights, then the two algorithms perform the same update; however if θ also includes the locations, then the M-step of the EM algorithm will not only update the weights as in the BA algorithm, but also the locations, due to the distortion term $\mathbb{E}_{(x,y)\sim P_X Q^*_{Y|X}}[-\log p(x|y)] = -\int \sum_k w_k \frac{p(x|\mu_k)}{p(x)} \log p(x|\mu_k) P_X(dx)$.

3 Wasserstein gradient descent

3.1 Wasserstein gradient for the rate functional

Below we compute the Wasserstein gradient of $\mathcal{L}_{BA}(\nu) = \int -\log \int \exp(-\lambda\rho(x,y))\nu(dy)\mu(dx)$. Under sufficient integrability on μ and ν to exchange the order of limit and integral, we can calculate

the first variation as

$$\begin{aligned}\lim_{\varepsilon\to 0}\frac{\mathcal{L}((1-\varepsilon)\nu+\varepsilon\tilde{\nu})-\mathcal{L}(\nu)}{\varepsilon} &= -\int \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\log\left[\frac{\int \exp(-\lambda\rho(x,y))(\nu+\varepsilon(\tilde{\nu}-\nu))(dy)}{\int \exp(-\lambda\rho(x,y))\nu(dy)}\right]\mu(dx)\\ &= -\int \lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\log\left[1+\frac{\int \exp(-\lambda\rho(x,y))\varepsilon(\tilde{\nu}-\nu)(dy)}{\int \exp(-\lambda\rho(x,y))\nu(dy)}\right]\mu(dx)\\ &= \iint -\frac{\exp(-\lambda\rho(x,y))}{\int \exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx)\,(\tilde{\nu}-\nu)(dy),\end{aligned}$$

where the last equality uses $\lim_{\varepsilon\to 0}\frac{1}{\varepsilon}\log(1+\varepsilon x)=x$ and Fubini's theorem. Thus the first variation ψ^ν of $\mathcal{L}_{BA}$ at ν is

$$\psi^\nu(y)=\int -\frac{\exp(-\lambda\rho(x,y))}{\int \exp(-\lambda\rho(x,\tilde{y}))\nu(d\tilde{y})}\mu(dx). \tag{23}$$

To find the desired Wasserstein gradient of $\mathcal{L}_{BA}$, it remains to take the Euclidean gradient of ψ^ν, i.e., $\nabla\mathcal{L}_{BA}(\nu)=\nabla\psi^\nu$.

3.2 Proof of Lemma 4.2 (convergence of Wasserstein gradient descent)

We first provide an auxiliary result.

Lemma 3.1. *Let $\gamma_1\geq\gamma_2\geq\cdots\geq 0$ and $a_t\geq 0$, $t\in\mathbb{N}$, $C>0$ satisfy $\sum_{t=1}^\infty\gamma_t=\infty$, $\sum_{t=1}^\infty\gamma_t^2<\infty$, $\sum_{t=1}^\infty a_t\gamma_t<\infty$ and $|a_t-a_{t+1}|\leq C\gamma_t$ for all $t\in\mathbb{N}$. Then $\lim_{t\to\infty}a_t=0$.*

Proof. The conclusion remains unchanged when rescaling a_t by the constant C, and thus without loss of generality $C=1$.

Clearly $\gamma_t\to 0$ as $\sum_{t=1}^\infty\gamma_t^2<\infty$. Moreover, there exists a subsequence of $(a_t)_{t\in\mathbb{N}}$ which converges to zero (otherwise there exists $\delta>0$ such that $a_t\geq\delta>0$ for all but finitely many t, contradicting $\sum_{t=1}^\infty\gamma_t a_t<\infty$).

Arguing by contradiction, suppose that the conclusion fails, i.e., that there exists a subsequence of $(a_t)_{t\in\mathbb{N}}$ which is uniformly bounded away from zero, say $a_t\geq\delta>0$ along that subsequence. Using this subsequence and the convergent subsequence mentioned above, we can construct a subsequence $a_{i_1},a_{i_2},a_{i_3},\ldots$ where $a_{i_n}\approx 0$ for n odd and $a_{i_n}\geq\delta$ for n even. We will show that

$$\sum_{t=i_{2n-1}}^{i_{2n}} a_t\gamma_t\gtrsim\delta^2/2\qquad\text{for all } n\in\mathbb{N},$$

contradicting the finiteness of $\sum_t\gamma_t a_t$. (The notation $\approx$ ($\gtrsim$) indicates (in)equality up to additive terms converging to zero for $n\to\infty$.)

To ease notation, fix n and set $m=i_{2n-1}$ and $M=i_{2n}$. We show that $\sum_{t=m}^M a_t\gamma_t\gtrsim\delta^2/2$. To this end, using $|a_t-a_{t+1}|\leq\gamma_t$ we find

$$a_t\geq a_M-\sum_{j=k}^{M-1}\gamma_j\geq\delta-\sum_{j=k}^{M-1}\gamma_j.$$

Since $a_m\approx 0$, there exists a largest $n_0\in\mathbb{N}$, $n_0\geq m$, such that $\sum_{j=n_0}^{M-1}\gamma_j\gtrsim\delta$ (and thus $\sum_{j=n_0}^{M-1}\gamma_j\lesssim\delta-\gamma_{n_0}\approx\delta$ as well). We conclude

$$\begin{aligned}\sum_{t=m}^M\gamma_t a_t\geq\sum_{t=n_0}^M\gamma_t a_t\geq\sum_{t=n_0}^M\gamma_t\left(\delta-\sum_{j=k}^{M-1}\gamma_j\right)&\gtrsim\delta^2-\sum_{t=n_0}^M\sum_{j=n_0}^M\gamma_t\gamma_j\mathbf{1}_{\{j\geq k\}}\\ &=\delta^2-\frac{1}{2}\left(\sum_{t=n_0}^M\gamma_t\right)^2-\frac{1}{2}\sum_{t=n_0}^M\gamma_t^2\approx\delta^2/2,\end{aligned}$$

where we used that $\sum_{t=n_0}^M\gamma_t^2\approx 0$. This completes the proof. □

Proof of Lemma 4.2. Using the linear approximation property in (15), we calculate

$$\begin{aligned}\mathcal{L}(\nu^{(n)}) - \mathcal{L}(\nu^{(0)}) &= \sum_{t=0}^{n-1} \mathcal{L}(\nu^{(t+1)}) - \mathcal{L}(\nu^{(t)}) \\ &= \sum_{t=0}^{n-1} -\gamma_t \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} + \gamma_t^2 \, o\left(\int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)}\right).\end{aligned}$$

As $\mathcal{L}(\nu^0)$ is finite and $\mathcal{L}(\nu^{(n)})$ is bounded from below, it follows that

$$\sum_{t=0}^{\infty} \gamma_t \int \|\nabla V_{\mathcal{L}}(\nu^{(t)})\|^2 \, d\nu^{(t)} < \infty.$$

The claim now follow by applying Lemma 3.1 with $a_t = \int \|\nabla \psi^{\nu^{(t)}}\|^2 \, d\nu^{(t)}$; note that the assumption in the lemma is satisfied due to the second inequality in (15). □

3.3 Proof of Proposition 4.3 (sample complexity)

Recall that $\mathcal{X} = \mathcal{Y} = \mathbb{R}^d$ and $\rho(x, y) = \|x - y\|^2$ in this proposition. For the proof, we will need the following lemma which is of independent interest. We write $\nu \leq_c \mu$ if ν is dominated by μ in convex order, i.e., $\int f \, d\nu \leq \int f \, d\mu$ for all convex functions $f : \mathbb{R}^d \to \mathbb{R}$.

Lemma 3.2. *Let μ have finite second moment. Given $\nu \in \mathcal{P}(\mathbb{R}^d)$, there exists $\tilde{\nu} \in \mathcal{P}(\mathbb{R}^d)$ with $\tilde{\nu} \leq_c \mu$ and*

$$\mathcal{L}_{EOT}(\mu, \tilde{\nu}) \leq \mathcal{L}_{EOT}(\mu, \nu).$$

This inequality is strict if $\nu \not\leq_c \mu$. In particular, any optimizer ν^* of (8) satisfies $\nu^* \leq_c \mu$.

Proof. Because this proof uses disintegration over $\mathcal{Y}$, it is convenient to reverse the order of the spaces in the notation and write a generic point as $(x, y) \in \mathcal{Y} \times \mathcal{X}$. Consider $\pi \in \Pi(\nu, \mu)$ and its disintegration $\pi = \nu(dx) \otimes K(x, dy)$ over $x \in \mathcal{Y}$. Define $T : \mathbb{R}^d \to \mathbb{R}^d$ by

$$T(x) := \int y \, K(x, dy).$$

Define also $\tilde{\pi} := (T, \mathrm{id})_{\#}\pi$ and $\tilde{\nu} := \tilde{\pi}_1$. From the definition of T, we see that $\tilde{\pi}$ is a martingale, thus $\tilde{\nu} \leq_c \mu$. Moreover, $\tilde{\nu} \otimes \mu = (T, \mathrm{id})_{\#}\nu \otimes \mu$. The data-processing inequality now shows that

$$H(\tilde{\pi}|\tilde{\nu} \otimes \mu) \leq H(\pi|\nu \otimes \mu).$$

On the other hand, $\int \| \int \tilde{y} \, K(x, d\tilde{y}) - y\|^2 \, K(x, dy) \leq \int \|x - y\|^2 K(x, dy)$ since the barycenter minimizes the squared distance, and this inequality is strict whenever $x \neq \int \tilde{y} K(x, d\tilde{y})$. Thus

$$\int \|x - y\|^2 \, \tilde{\pi}(dx, dy) \leq \int \|x - y\|^2 \, \pi(dx, dy),$$

and the inequality is strict unless $T(x) = x$ for ν-a.e. x, which in turn is equivalent to π being a martingale. The claims follow. □

Proof of Proposition 4.3. Subgaussianity of the optimizer follows directly from Lemma 3.2.

Recalling that $\inf_\nu \mathcal{L}_{EOT}(\nu)$ and $\inf_\nu \lambda^{-1}\mathcal{L}_{BA}(\nu)$ have the same values and minimizers, it suffices to show the claim for $\mathcal{L} = \mathcal{L}_{EOT}$. Let ν^* be an optimizer of (8) and ν^n its empirical measure from n samples, then clearly

$$\begin{aligned}\left| \min_{\nu_n \in \mathcal{P}_n(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu_n) - \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu) \right| &= \min_{\nu_n \in \mathcal{P}_n(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu_n) - \min_{\nu \in \mathcal{P}(\mathbb{R}^d)} \mathcal{L}_{EOT}(\mu, \nu) \\ &\leq \mathbb{E}\left[|\mathcal{L}_{EOT}(\mu, \nu^n) - \mathcal{L}_{EOT}(\mu, \nu^*)|\right]\end{aligned}$$

where the expectation is taken over samples for ν^n. The first inequality of Proposition 4.3 now follows from the sample complexity result for entropic optimal transport in [Mena and Niles-Weed, 2019, Theorem 2].

Denote by ν_m^* the optimizer for the problem (8) with μ replaced by μ^m. Similarly to the above, we obtain

$$\mathbb{E}\left[\left|\min_{\nu\in\mathcal{P}(\mathbb{R}^d)}\mathcal{L}_{EOT}(\mu,\nu)-\min_{\nu\in\mathcal{P}(\mathbb{R}^d)}\mathcal{L}_{EOT}(\mu^m,\nu)\right|\right] \leq \mathbb{E}\left[\max_{\nu\in\{\nu^*,\nu_m^*\}}\left|\mathcal{L}_{EOT}(\mu,\nu)-\mathcal{L}_{EOT}(\mu^m,\nu)\right|\right],$$

where the expectation is taken over samples from μ^m. In this situation, we cannot directly apply [Mena and Niles-Weed, 2019, Theorem 2]. However, the bound given by [Mena and Niles-Weed, 2019, Proposition 2] still applies, and the only dependence on $\nu \in \{\nu^*, \nu_m^*\}$ is through their subgaussianity constants. By Lemma 3.2, these constants are bounded by the corresponding constants of μ and μ^m. Thus, the arguments in the proof of [Mena and Niles-Weed, 2019, Theorem 2] can be applied, yielding the second inequality of Proposition 4.3.

The final inequality of Proposition 4.3 follows from the first two inequalities (the first one being applied with μ^m) and the triangle inequality, where we again use the arguments in the proof of [Mena and Niles-Weed, 2019, Theorem 2] to bound the expectation over the subgaussianity constants of μ^m. □

4 Example implementation of WGD

We provide a self-contained minimal implementation of Wasserstein gradient descent on $\mathcal{L}_{BA}$, using the Jax library [Bradbury et al., 2018]. To compute the Wasserstein gradient, we evaluate the first variation of the rate functional in `compute_psi_sum` according to (23), yielding $\sum_{i=1}^n \psi^\nu(x_i)$, then simply take its gradient w.r.t. the particle locations $x_{1,\dots n}$ using Jax's autodiff tool on line 51.

The implementation of WGD on $\mathcal{L}_{EOT}$ is similar, except the first variation is computed using Sinkhorn's algorithm. Both versions can be easily just-in-time compiled and enjoy GPU acceleration.

```
# Wasserstein GD on the rate functional L_{BA}.
import jax.numpy as jnp
import jax
from jax.scipy.special import logsumexp

# Define the distortion function \rho.
squared_diff = lambda x, y: jnp.sum((x - y) ** 2)
pairwise_distortion_fn = jax.vmap(jax.vmap(squared_diff, (None, 0)), (0, None))


def wgrad(mu_x, mu_w, nu_x, nu_w, rd_lambda):
  """
  Compute the Wasserstein gradient of the rate functional, which we will use
  to move the \nu particles.
  :param mu_x: locations of \mu atoms.
  :param mu_w: weights  of \mu atoms.
  :param nu_x: locations of \nu atoms.
  :param nu_w: weights  of \nu atoms.
  :param rd_lambda: R-D tradeoff hyperparameter.
  :return:
  """

  def compute_psi_sum(nu_x):
    """
    Here we compute a surrogate loss based on the first variation \psi, which
    allows jax autodiff to compute the desired Wasserstein gradient.
    :param nu_x:
    :return: psi_sum = \sum_i \psi(nu_x[i])
    """
    C = pairwise_distortion_fn(mu_x, nu_x)
```

```
    scaled_C = rd_lambda * C  # [m, n]
    log_nu_w = jnp.log(nu_w)  # [1, n]

    # Solve BA inner problem with a fixed nu.
    phi = - logsumexp(-scaled_C + log_nu_w, axis=1, keepdims=True)  # [m, 1]
    loss = jnp.sum(mu_w * phi)  # Evaluate the rate functional. Eq (6) in paper.

    # Let's also report rate and distortion estimates (discussed in Sec. 4.4 of the paper).
    # Find \pi^* via \phi
    pi = jnp.exp(phi - scaled_C) * jnp.outer(mu_w, nu_w)  # [m, n]
    distortion = jnp.sum(pi * C)
    rate = loss - rd_lambda * distortion

    # Now evaluate \psi on the atoms of \nu.
    phi = jax.lax.stop_gradient(phi)
    psi = - jnp.sum(jnp.exp(jax.lax.stop_gradient(phi) - scaled_C) * mu_w, axis=0)
    psi_sum = jnp.sum(psi)  # For computing gradient w.r.t. each nu_x atom.
    return psi_sum, (loss, rate, distortion)

  # Evaluate the Wasserstein gradient, i.e., \nabla \psi, on nu_x.
  psi_prime, loss = jax.grad(compute_psi_sum, has_aux=True)(nu_x)
  return psi_prime, loss


def wgd(X, n, rd_lambda, num_steps, lr, rng):
  """
  A basic demo of Wasserstein gradient descent on a discrete distribution.
  :param X: a 2D array [N, d] of data points defining the source \mu.
  :param n: the number of particles to use for \nu.
  :param rd_lambda: R-D tradeoff hyperparameter.
  :param num_steps: total number of gradient updates.
  :param lr:  step size.
  :param rng: jax random key.
  :return: (nu_x, nu_w), the locations and weights of the final \nu.
  """
  # Set up the source measure \mu.
  m = jnp.size(X, 0)
  mu_x = X
  mu_w = 1 / m * jnp.ones((m, 1))
  # Initialize \nu atoms using random training samples.
  rand_idx = jax.random.permutation(rng, m)[:n]
  nu_x = X[rand_idx]  # Locations of \nu atoms.
  nu_w = 1 / n * jnp.ones((1, n))  # Uniform weights.
  for step in range(num_steps):
    psi_prime, (loss, rate, distortion) = wgrad(mu_x, mu_w, nu_x, nu_w, rd_lambda)
    nu_x -= lr * psi_prime
    print(f'step={step}, loss={loss:.4g}, rate={rate:.4g}, distortion={distortion:.4g}')

  return nu_x, nu_w


if __name__ == '__main__':
  # Run a toy example on 2D Gaussian samples.
  rng = jax.random.PRNGKey(0)
  X = jax.random.normal(rng, [10, 2])
  nu_x, nu_w = wgd(X, n=4, rd_lambda=2., num_steps=100, lr=0.1, rng=rng)
```

5 Further experimental results

Our deconvolution experiments were run on Intel(R) Xeon(R) CPUs, while the rest of the experiments were run on Titan RTX GPUs.

In most experiments, we use the Adam [Kingma and Ba, 2015] optimizer for updating the ν particle locations in WGD and for updating the variational parameters in other methods. For our hybrid WGD algorithm, which adjusts the particle weights in addition to their locations, we found that applying momentum to the particle locations can in fact slow down convergence, and therefore use plain gradient descent with a decaying step size.

5.1 Deconvolution

Loss curves and solutions from various methods. In Figure 1 we plot both the training and test losses for the various methods. The test losses are evaluated on freshly drawn samples from the source distribution, and provide estimates of the true population losses. As expected, the train losses appear similar to the test losses since we use a large sample size for training.

In Figure 2, we visualize the fitted ν measure after performing the optimization illustrated in Figure 1. We plot the location of the $n = 20$ particles from the BA, WGD, and hybrid algorithms, additionally coloring the particles from BA and the hybrid algorithm by their weights. To visualize the (continuous) ν learned by RD-VAE and NERD, we plot a scatter of 300 random samples drawn from each.

Characterizing the optimal solution. In the deconvolution problem, $\mu = S * \mathcal{N}(0, \sigma^2 I)$, and whenever $\lambda \geq \frac{1}{\sigma^2}$ the optimal solution to the R-D problem (3) is given by $\nu^* = S * \mathcal{N}(0, \sigma^2 - \frac{1}{\lambda})$ and $K^*(x, dy) = \mathcal{N}(x, \frac{1}{\lambda})$. This follows from a basic property of the Gaussian distribution and an argument based on characteristic functions.

Knowing the optimal ν^*, we can therefore numerically compute the optimal loss,

$$OPT := \mathcal{L}_{BA}(\nu^*) = \int_{\mathcal{X}} -\log\left(\int_{\mathcal{Y}} e^{-\lambda\rho(x,y)}\nu^*(dy)\right)\mu(dx), \tag{24}$$

using the plug-in Monte Carlo estimator

$$\frac{1}{m}\sum_{i=1}^{m} -\log\left(\frac{1}{n}\sum_{j=1}^{n} e^{-\lambda\rho(x_i,y_j)}\right), \tag{25}$$

where $x_{1,\ldots,m}$ are drawn from μ and $y_{1,\ldots,n}$ from ν^*. To reduce the bias of this estimator (also discussed in the context of NERD in Sec. 3.2), we use $m = 10000$ and the very large $n = 10^6$ in our Monte-Carlo estimation above.

Similarly, we can sample $\{(x_i, y_i)\}_{i=1}^m$ from $\nu^* \otimes K^*$ to compute the ground truth distortion and rate with high accuracy as follows,

$$\mathcal{D} = \frac{1}{m}\sum_{i=1}^{m}\rho(x_i, y_i),$$
$$\mathcal{R} = OPT - \lambda\mathcal{D}.$$

We can thus obtain the segment of the ground truth $R(D)$ where $\lambda \geq \frac{1}{\sigma^2}$.

R-D upper bounds. We rerun the various algorithms with $\lambda \in \{1, 3, 10, 30, 100, 300\}$ to produce upper bounds on $R(D)$, and plot the results in Figure 3-Right. We observe that WGD gives the tightest upper bound out of all the methods (the hybrid WGD algorithm produces overlapping curves and is omitted for clarity). As we increase n to 50 and 1000 (Figure 3-Middle, Left), the various methods increase linearly in their computational complexity (except for RD-VAE, which used a fixed architecture and didn't benefit noticeably from further increase in its neural network sizes), and eventually give qualitatively similar R-D upper bounds that generally agree with the true $R(D)$. Note that in large scale problems (e.g., those considered in Sec. 5.3), we are much more likely to operate in the "small n" regime due to computational constraints.

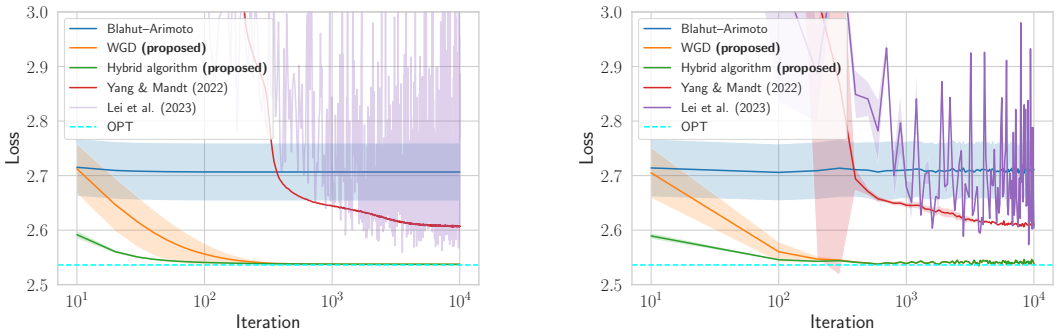

Figure 1: **Left:** The objective functions of the various methods across training iterations. **Right:** The same objective functions evaluated on random empirical measures of the source. The curve for each method is averaged over 5 reruns with different random seeds, with the shading corresponding to one standard deviation. The proposed WGD algorithms (orange, green) converge quickly to the theoretically optimal value OPT (cyan). BA [Blahut, 1972, Arimoto, 1972] (blue) converges quickly to a highly suboptimal solution, while the RD-VAE [Yang and Mandt, 2022] (red) converges more slowly, also to an inferior solution. NERD [Lei et al., 2023] (purple) fails to converge due to inaccuracy of its Monte-Carlo estimator when n is relatively small (see discussion in Sec. 3.2), leading to oscillating objective values.

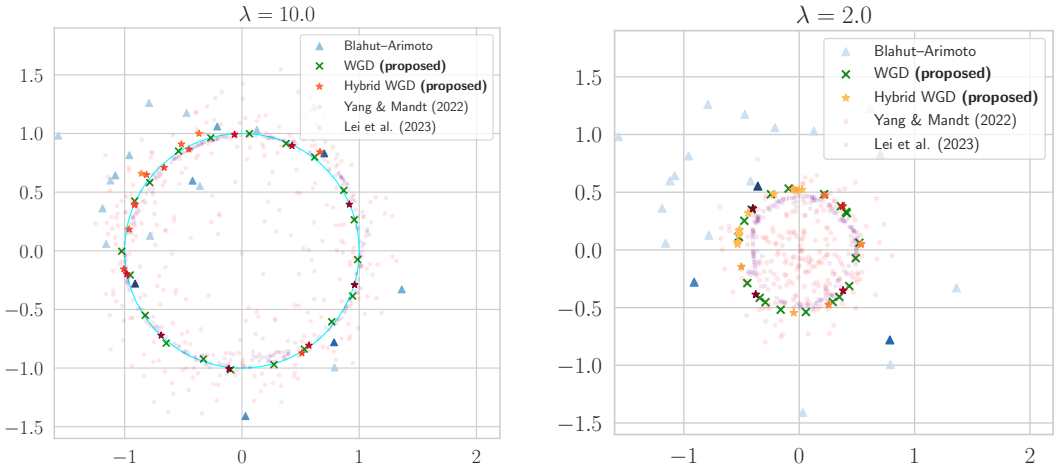

Figure 2: Visualizing the optimized ν measures from various algorithms. **Left:** $\lambda = 10 = \frac{1}{\sigma^2}$, the same as in Fig. 1. Here ν^* is precisely the uniform distribution on the unit circle, colored in cyan. The proposed WGD algorithm places almost all its ν atoms (green crosses) exactly on the circle. The proposed hybrid algorithm occasionally places atoms off the circle, and assigns them lower weights (orange stars) than the ones on the circle (red stars). This extra flexibility explains its faster convergence compared to the plain WGD algorithm seen in Fig. 1, while achieving the same optimized loss close to OPT. The BA algorithm is stuck with the randomly initialized set of ν atoms (blue) and can only manage to assign higher weights to atoms closer to the unit circle. RD-VAE and NERD have difficulty learning the true ν^*, as seen from the misplaced samples of ν from the two methods (faint red dots for RD-VAE and purple squares for NERD, respectively). **Right:** We repeat the experiment but with $\lambda = 2$. ν^* is now uniform on a circle with a smaller radius. The algorithms maintain their respective behavior from the $\lambda = 10$ case, with the BA, RD-VAE, and NERD algorithms failing to recover the support of ν^*. As $\lambda \to 0$, ν^* shrinks towards the mean of μ (the origin in this case), making it exceedingly difficult for the BA algorithm with a randomly discretized $\mathcal{Y}$-space to locate the true support of ν^*.

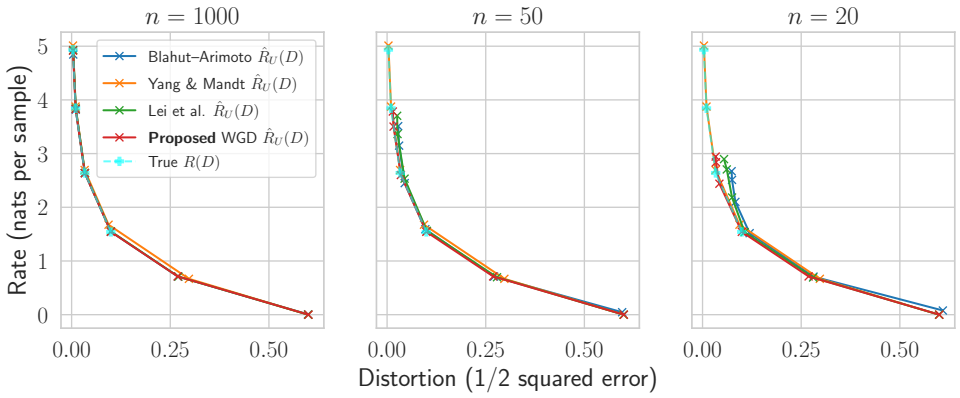

Figure 3: Final R-D upper bounds for the source $\mu = S * \mathcal{N}(0, 0.1I)$ in the maximum-likelihood deconvolution problem (Sec. 5.2), with different settings of n for BA [Blahut, 1972, Arimoto, 1972], NERD [Lei et al., 2023], and the WGD algorithm. The result using the hybrid WGD algorithm (Sec. 4.2) overlaps with that of WGD, hence is omitted for better readability. The ground truth $R(D)$ is known analytically for $\lambda \geq \frac{1}{\sigma^2}$ and computed numerically (see discussion in the text), and is drawn in cyan. The RD-VAE upper bound (orange; the same in each subplot) agrees fairly well with the true $R(D)$ except for some looseness when the distortion is between 0.1 and 0.25. **Left:** when the number of particles is large ($n = 1000$), BA, WGD, and NERD give similarly R-D upper bound estimates close to the true $R(D)$. **Middle:** as we allow ourselves to use fewer particles, e.g., $n = 50$, the bounds from BA, WGD, and NERD start to deviate from the true $R(D)$, with WGD appearing the least affected out of the three. **Right:** as we decrease n further to 20, WGD still mostly preserves the true $R(D)$, while BA and NERD shows much larger deviation.

References

Erhan Çinlar. *Probability and stochastics*, volume 261. Springer, 2011.

Gerald B Folland. *Real analysis: modern techniques and their applications*, volume 40. John Wiley & Sons, 1999.

Yury Polyanskiy and Yihong Wu. Lecture notes on information theory. *Lecture Notes for ECE563 (UIUC) and*, 6(2012-2016):7, 2014.

David M Blei, Alp Kucukelbir, and Jon D McAuliffe. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

MJ Beal and Z Ghahramani. The variational bayesian em algorithm for incomplete data: with application to scoring graphical model structures. *Bayesian statistics*, 7(453-464):210, 2003.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1): 1–22, 1977.

Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An introduction to variational methods for graphical models. *Machine learning*, 37:183–233, 1999.

Martin J Wainwright, Michael I Jordan, et al. Graphical models, exponential families, and variational inference. *Foundations and Trends® in Machine Learning*, 1(1–2):1–305, 2008.

Michael Irwin Jordan. *Learning in graphical models*. MIT press, 1999.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Gonzalo Mena and Jonathan Niles-Weed. Statistical bounds for entropic optimal transport: sample complexity and the central limit theorem. *Advances in Neural Information Processing Systems*, 32, 2019.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Diederik P Kingma and Jimmy Lei Ba. Adam: A method for stochastic gradient descent. In *International Conference on Learning Representations*, 2015.

R. Blahut. Computation of channel capacity and rate-distortion functions. *IEEE Transactions on Information Theory*, 18(4):460–473, 1972. doi: 10.1109/TIT.1972.1054855.

Suguru Arimoto. An algorithm for computing the capacity of arbitrary discrete memoryless channels. *IEEE Transactions on Information Theory*, 18(1):14–20, 1972.

Yibo Yang and Stephan Mandt. Towards empirical sandwich bounds on the rate-distortion function. In *International Conference on Learnning Representations*, 2022.

Eric Lei, Hamed Hassani, and Shirin Saeedi Bidokhti. Neural estimation of the rate-distortion function with applications to operational source coding. *IEEE Journal on Selected Areas in Information Theory*, 2023.