Stochastic Optimization for Large-scale Optimal Transport
Aude Genevay, Marco Cuturi, Gabriel Peyré, Francis Bach

To cite this version:
Aude Genevay, Marco Cuturi, Gabriel Peyré, Francis Bach. Stochastic Optimization for Large-scale Optimal Transport. NIPS 2016 - Thirtieth Annual Conference on Neural Information Processing System, Dec 2016, Barcelona, Spain. hal-01321664v2

HAL Id: hal-01321664
https://hal.science/hal-01321664v2
Submitted on 2 Nov 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Stochastic Optimization for Large-scale Optimal Transport

Aude Genevay
CEREMADE, Université Paris-Dauphine
INRIA – Mokaplan project-team
genevay@ceremade.dauphine.fr

Marco Cuturi
CREST, ENSAE
Université Paris-Saclay
marco.cuturi@ensae.fr

Gabriel Peyré
CNRS and DMA, École Normale Supérieure
INRIA – Mokaplan project-team
gabriel.peyre@ens.fr

Francis Bach
INRIA – Sierra project-team
DI, ENS
francis.bach@inria.fr

Abstract
Optimal transport (OT) defines a powerful framework to compare probability distributions in a geometrically faithful way. However, the practical impact of OT is still limited because of its computational burden. We propose a new class of stochastic optimization algorithms to cope with large-scale OT problems. These methods can handle arbitrary distributions (either discrete or continuous) as long as one is able to draw samples from them, which is the typical setup in high-dimensional learning problems. This alleviates the need to discretize these densities, while giving access to provably convergent methods that output the correct distance without discretization error. These algorithms rely on two main ideas: (a) the dual OT problem can be re-cast as the maximization of an expectation; (b) the entropic regularization of the primal OT problem yields a smooth dual optimization which can be addressed with algorithms that have a provably faster convergence. We instantiate these ideas in three different setups: (i) when comparing a discrete distribution to another, we show that incremental stochastic optimization schemes can beat Sinkhorn’s algorithm, the current state-of-the-art finite dimensional OT solver; (ii) when comparing a discrete distribution to a continuous density, a semi-discrete reformulation of the dual program is amenable to averaged stochastic gradient descent, leading to better performance than approximately solving the problem by discretization; (iii) when dealing with two continuous densities, we propose a stochastic gradient descent over a reproducing kernel Hilbert space (RKHS). This is currently the only known method to solve this problem, apart from computing OT on finite samples. We backup these claims on a set of discrete, semi-discrete and continuous benchmark problems.

1 Introduction
Many problems in computational sciences require to compare probability measures or histograms. As a set of representative examples, let us quote: bag-of-visual-words comparison in computer vision [18], color and shape processing in computer graphics [22], bag-of-words for natural language processing [12] and multi-label classification [9]. In all of these problems, a geometry between the features (words, visual words, labels) is usually known, and can be leveraged to compare probability distributions in a geometrically faithful way. This underlying geometry might be for instance the planar Euclidean domain for 2-D shapes, a perceptual 3D color metric space for image processing or a high-dimensional semantic embedding for words. Optimal transport (OT) [24] is the canonical
way to automatically lift this geometry to define a metric for probability distributions. That metric is
known as the Wasserstein or earth mover’s distance. As an illustrative example, OT can use a metric
between words to build a metric between documents that are represented as frequency histograms of
words (see [12] for details). All the above-cited lines of work advocate, among others, that OT is the
natural choice to solve these problems, and that it leads to performance improvement when compared
to geometrically-oblivious distances such as the Euclidean or $\chi^2$ distances or the Kullback-Leibler
divergence. However, these advantages come at the price of an enormous computational overhead.
This is especially true because current OT solvers require to sample beforehand these distributions
on a pre-defined set of points, or on a grid. This is both inefficient (in term of storage and speed)
and counter-intuitive. Indeed, most high-dimensional computational scenarios naturally represent
distributions as objects from which one can sample, not as density functions to be discretized.
Our goal is to alleviate these shortcomings. We propose a class of provably convergent stochastic
optimization schemes that can handle both discrete and continuous distributions through sampling.

Previous works. The prevalent way to compute OT distances is by solving the so-called Kantorovitch
problem [11] (see Section 2 for a short primer on the basics of OT formulations), which boils down
to a large-scale linear program when dealing with discrete distributions (i.e., finite weighted sums of
Dirac masses). This linear program can be solved using network flow solvers, which can be further
refined to assignment problems when comparing measures of the same size with uniform weights [3].
Recently, regularized approaches that solve the OT with an entropic penalization [6] have been shown
to be extremely efficient to approximate OT solutions at a very low computational cost. These regular-
ized approaches have supported recent applications of OT to computer graphics [22] and machine
learning [9]. These methods apply the celebrated Sinkhorn algorithm [21], and can be extended
to solve more exotic transportation-related problems such as the computation of barycenters [22].
Their chief computational advantage over competing solvers is that each iteration boils down to
matrix-vector multiplications, which can be easily parallelized, streams extremely well on GPU, and
enjoys linear-time implementation on regular grids or triangulated domains [22].

These methods are however purely discrete and cannot cope with continuous densities. The only
known class of methods that can overcome this limitation are so-called semi-discrete solvers [1],
that can be implemented efficiently using computational geometry primitives [13]. They can compute
distance between a discrete distribution and a continuous density. Nonetheless, they are restricted to
the Euclidean squared cost, and can only be implemented in low dimensions (2-D and 3-D). Solving
these semi-discrete problems efficiently could have a significant impact for applications to density
fitting with an OT loss [4] for machine learning applications, see [14]. Lastly, let us point out that
there is currently no method that can compute OT distances between two continuous densities, which
is thus an open problem we tackle in this article.

Contributions. This paper introduces stochastic optimization methods to compute large-scale optimal
transport in all three possible settings: discrete OT, to compare a discrete vs. another discrete measure;
semi-discrete OT, to compare a discrete vs. a continuous measure; and continuous OT, to compare a
continuous vs. another continuous measure. These methods can be used to solve classical OT
problems, but they enjoy faster convergence properties when considering their entropic-regularized
versions. We show that the discrete regularized OT problem can be tackled using incremental
algorithms, and we consider in particular the stochastic averaged gradient (SAG) method [20]. Each
iteration of that algorithm requires $N$ operations ($N$ being the size of the supports of the input
distributions), which makes it scale better in large-scale problems than the state-of-the-art Sinkhorn
algorithm, while still enjoying a convergence rate of $O(1/k)$, $k$ being the number of iterations. We
show that the semi-discrete OT problem can be solved using averaged stochastic gradient descent
(SGD), whose convergence rate is $O(1/\sqrt{k})$. This approach is numerically advantageous over the
brute force approach consisting in sampling first the continuous density to solve next a discrete OT
problem. Lastly, for continuous optimal transport, we propose a novel method which makes use of an
expansion of the dual variables in a reproducing kernel Hilbert space (RKHS). This allows us for
the first time to compute with a converging algorithm OT distances between two arbitrary densities,
under the assumption that the two potentials belong to such an RKHS.

Notations. In the following we consider two metric spaces $\mathcal{X}$ and $\mathcal{Y}$. We denote by $\mathcal{M}_+^1(\mathcal{X})$ the set
of positive Radon probability measures on $\mathcal{X}$, and $\mathcal{C}(\mathcal{X})$ the space of continuous functions on $\mathcal{X}$. Let
$\mu \in \mathcal{M}_+^1(\mathcal{X})$, $\nu \in \mathcal{M}_+^1(\mathcal{Y})$, we define

$$\Pi(\mu, \nu) \overset{\text{def}}{=} \{ \pi \in \mathcal{M}_+^1(\mathcal{X} \times \mathcal{Y}) \, ; \, \forall (A,B) \subset \mathcal{X} \times \mathcal{Y}, \pi(A \times \mathcal{Y}) = \mu(A), \pi(\mathcal{X} \times B) = \nu(B) \},$$
the set of joint probability measures on $\mathcal{X} \times \mathcal{Y}$ with marginals $\mu$ and $\nu$. The Kullback-Leibler divergence between joint probabilities is defined as

$$\forall (\pi, \xi) \in \mathcal{M}_+^1(\mathcal{X} \times \mathcal{Y})^2, \quad \text{KL}(\pi|\xi) \overset{\text{def}}{=} \int_{\mathcal{X} \times \mathcal{Y}} \left( \log \left( \frac{d\pi}{d\xi}(x,y) \right) - 1 \right) d\xi(x,y),$$

where we denote $\frac{d\pi}{d\xi}$ the relative density of $\pi$ with respect to $\xi$, and by convention $\text{KL}(\pi|\xi) \overset{\text{def}}{=} +\infty$ if $\pi$ does not have a density with respect to $\xi$. The Dirac measure at point $x$ is $\delta_x$. For a set $C$, $\mu_C(x) = 0$ if $x \in C$ and $\mu_C(x) = +\infty$ otherwise. The probability simplex of $N$ bins is $\Sigma_N = \{ \mu \in \mathbb{R}^N_+ : \sum_i \mu_i = 1 \}$. Element-wise multiplication of vectors is denoted by $\odot$ and $K^\top$ denotes the transpose of a matrix $K$. We denote $1_N = (1, \ldots, 1)^T \in \mathbb{R}^N$ and $0_N = (0, \ldots, 0)^T \in \mathbb{R}^N$.

## 2 Optimal Transport: Primal, Dual and Semi-dual Formulations

We consider the optimal transport problem between two measures $\mu \in \mathcal{M}_+^1(\mathcal{X})$ and $\nu \in \mathcal{M}_+^1(\mathcal{Y})$, defined on metric spaces $\mathcal{X}$ and $\mathcal{Y}$. No particular assumption is made on the form of $\mu$ and $\nu$, we only assume that they both can be sampled from to be able to apply our algorithms.

### Primal, Dual and Semi-dual Formulations


$$\forall (\mu, \nu) \in \mathcal{M}_+^1(\mathcal{X}) \times \mathcal{M}_+^1(\mathcal{Y}), \quad W_\varepsilon(\mu, \nu) \overset{\text{def}}{=} \min_{\pi \in \Pi(\mu, \nu)} \int_{\mathcal{X} \times \mathcal{Y}} c(x,y)d\pi(x,y) + \varepsilon \text{KL}(\pi|\mu \otimes \nu). \quad (P_\varepsilon)$$

Here $c \in C(\mathcal{X} \times \mathcal{Y})$ and $c(x, y)$ should be interpreted as the “ground cost” to move a unit of mass from $x$ to $y$. This $c$ is typically application-dependent, and reflects some prior knowledge on the data to process. We refer to the introduction for a list of previous work where various examples (in imaging, vision, graphics or machine learning) of such costs are given.

When $\mathcal{X} = \mathcal{Y}$, $\varepsilon = 0$ and $c = d^p$ for $p \geq 1$, where $d$ is a distance on $\mathcal{X}$, then $W_0(\mu, \nu)^{\frac{1}{p}}$ is known as the $p$-Wasserstein distance on $\mathcal{M}_+^1(\mathcal{X})$. Note that this definition can be used for any type of measure, both discrete and continuous. When $\varepsilon > 0$, problem $(P_\varepsilon)$ is strongly convex, so that the optimal $\pi$ is unique, and algebraic properties of the KL regularization result in computations that can be tackled using the Sinkhorn algorithm [6].

For any $c \in C(\mathcal{X} \times \mathcal{Y})$, we define the following constraint set

$$U_c \overset{\text{def}}{=} \{(u, v) \in C(\mathcal{X}) \times C(\mathcal{Y}) : \forall (x, y) \in \mathcal{X} \times \mathcal{Y}, u(x) + v(y) \leq c(x, y)\},$$

and define its indicator function as well as its “smoothed” approximation

$$\iota_{U_c}(u, v) \overset{\text{def}}{=} \begin{cases} u_c(u, v) & \text{if } \varepsilon = 0, \\ \varepsilon \int_{\mathcal{X} \times \mathcal{Y}} \exp\left( \frac{u(x) + v(y) - c(x, y)}{\varepsilon} \right) d\mu(x)d\nu(y) & \text{if } \varepsilon > 0. \end{cases} \quad (1)$$

For any $v \in C(\mathcal{Y})$, we define its $c$-transform and its “smoothed” approximation

$$\forall x \in \mathcal{X}, \quad v^{c, \varepsilon}(x) \overset{\text{def}}{=} \begin{cases} \min_{y \in \mathcal{Y}} c(x, y) - v(y) & \text{if } \varepsilon = 0, \\ -\varepsilon \log \left( \int_{\mathcal{Y}} \exp\left( \frac{v(y) - c(x, y)}{\varepsilon} \right) d\nu(y) \right) & \text{if } \varepsilon > 0. \end{cases} \quad (2)$$

The proposition below describes two dual problems. It is central to our analysis and paves the way for the application of stochastic optimization methods.

### Proposition 2.1 (Dual and semi-dual formulations)

For $\varepsilon \geq 0$, one has

$$W_\varepsilon(\mu, \nu) = \max_{u \in C(\mathcal{X}), v \in C(\mathcal{Y})} F_\varepsilon(u, v) \overset{\text{def}}{=} \int_\mathcal{X} u(x)d\mu(x) + \int_\mathcal{Y} v(y)d\nu(y) - \iota_{U_c}(u, v), \quad (D_\varepsilon)$$

$$= \max_{v \in C(\mathcal{Y})} H_\varepsilon(v) \overset{\text{def}}{=} \int_\mathcal{X} v^{c, \varepsilon}(x)d\mu(x) + \int_\mathcal{Y} v(y)d\nu(y) - \varepsilon, \quad (S_\varepsilon)$$

where $\iota_{U_c}$ is defined in [1] and $v^{c, \varepsilon}$ in [2]. Furthermore, $u$ solving $(D_\varepsilon)$ is recovered from an optimal $v$ solving $(S_\varepsilon) as u = v^{c, \varepsilon}$. For $\varepsilon > 0$, the solution $\pi$ of $(P_\varepsilon)$ is recovered from any $(u, v)$ solving $(D_\varepsilon)$ as $d\pi(x,y) = \exp\left( \frac{u(x) + v(y) - c(x, y)}{\varepsilon} \right) d\mu(x)d\nu(y)$.
Proof. Problem $\mathcal{P}_0$ is the convex dual of $\mathcal{P}_2$, and is derived using Fenchel-Rockafellar’s theorem. The relation between $u$ and $v$ is obtained by writing the first order optimality condition for $v$ in $\mathcal{D}_2$. Plugging this expression back in $\mathcal{D}_2$ yields $\mathcal{S}_2$. □

Problem $\mathcal{P}_2$ is called the primal while $\mathcal{D}_2$ is its associated dual problem. We refer to $\mathcal{S}_2$ as the “semi-dual” problem, because in the special case $\varepsilon = 0$, $\mathcal{S}_2$ boils down to the so-called semi-discrete OT problem [1]. Both dual problems are concave maximization problems. The optimal dual variables $(u,v)$—known as Kantorovitch potentials—are not unique, since for any solution $(u,v)$ of $\mathcal{D}_2$, $(u + \lambda, v - \lambda)$ is also a solution for any $\lambda \in \mathbb{R}$. When $\varepsilon > 0$, they can be shown to be unique up to this scalar translation [6]. We refer to the supplementary material for a discussion (and proofs) of the convergence of the solutions of $\mathcal{P}_2$, $\mathcal{D}_2$ and $\mathcal{S}_2$ towards those of $(\mathcal{P}_0)$, $(\mathcal{D}_0)$ and $(\mathcal{S}_0)$ as $\varepsilon \to 0$.

A key advantage of $\mathcal{S}_2$ over $\mathcal{D}_2$ is that, when $\nu$ is a discrete density (but not necessarily $\mu$), then $\mathcal{S}_2$ is a finite-dimensional concave maximization problem, which can thus be solved using stochastic programming techniques, as highlighted in Section 4. By contrast, when both $\mu$ and $\nu$ are continuous densities, these dual problems are intrinsically infinite dimensional, and we propose in Section 5 more advanced techniques based on RKHSs.

Stochastic Optimization Formulations. The fundamental property needed to apply stochastic programming is that both dual problems $\mathcal{D}_2$ and $\mathcal{S}_2$ must be rephrased as maximizing expectations:

$$
\forall \varepsilon > 0, \quad F_\varepsilon(u,v) = \mathbb{E}_{X,Y} [f_\varepsilon(X,Y,u,v)] \quad \text{and} \quad \forall \varepsilon \geq 0, \quad H_\varepsilon(v) = \mathbb{E}_X [h_\varepsilon(X,v)],
$$

(3)

where the random variables $X$ and $Y$ are independent and distributed according to $\mu$ and $\nu$ respectively, and where, for $(x,y) \in \mathcal{X} \times \mathcal{Y}$ and $(u,v) \in \mathcal{C}(\mathcal{X}) \times \mathcal{C}(\mathcal{Y})$,

$$
\forall \varepsilon > 0, \quad f_\varepsilon(x,y,u,v) \overset{\text{def.}}{=} u(x) + v(y) - \varepsilon \exp \left( \frac{u(x) + v(y) - c(x,y)}{\varepsilon} \right),
$$

$$
\forall \varepsilon \geq 0, \quad h_\varepsilon(x,v) \overset{\text{def.}}{=} \int_{\mathcal{Y}} v(y) \text{d} \nu(y) + v^\varepsilon(x) - \varepsilon.
$$

This reformulation is at the heart of the methods detailed in the remainder of this article. Note that the dual problem $\mathcal{D}_2$ cannot be cast as an unconstrained expectation maximization problem when $\varepsilon = 0$, because of the constraint on the potentials which arises in that case.

When $\nu$ is discrete, i.e $\nu = \sum_{j=1}^J \nu_j \delta_{y_j}$, the potential $v$ is a $J$-dimensional vector $(v_j)_{j=1}^J$ and we can compute the gradient of $h_\varepsilon$. When $\varepsilon > 0$ the gradient reads $\nabla_v h_\varepsilon(v,x) = \nu - \pi(x)$ and the hessian is given by $\nabla_v^2 h_\varepsilon(v,x) = \frac{1}{\varepsilon} (\pi(x)\pi(x)^T - \text{diag}(\pi(x)))$ where $\pi(x)_i = \exp \left( \frac{\pi(x,y_j)}{\varepsilon} \right) \left( \sum_{j=1}^J \exp \left( \frac{-\pi(x,y_j)}{\varepsilon} \right) \right)^{-1}$. The eigenvalues of the hessian can be upper bounded by $\frac{1}{\varepsilon}$, which guarantees a lipschitz gradient, and lower bounded by $0$ which does not ensure strong convexity. In the unregularized case, $h_0$ is not smooth and a subgradient is given by $\nabla_v h_0(v,x) = \nu - \tilde{\pi}(x)$, where $\tilde{\pi}(x)_i = \chi_{i=j^*}$, and $j^* = \arg \min_{j \in \{1,...,J\}} c(x,y_j) - v_j$ (when several elements are in the argmin, we arbitrarily choose one of them to be $j^*$). We insist on the lack of strong convexity of the semi-dual problem, as it impacts the convergence properties of the stochastic algorithms (stochastic averaged gradient and stochastic gradient descent) described below.

3 Discrete Optimal Transport

We assume in this section that both $\mu$ and $\nu$ are discrete measures, i.e. finite sums of Diracs, of the form $\mu = \sum_{i=1}^I \mu_i \delta_{x_i}$, and $\nu = \sum_{j=1}^J \nu_j \delta_{y_j}$, where $(x_i)_i \subset \mathcal{X}$ and $(y_j)_j \subset \mathcal{Y}$, and the histogram vector weights are $\mu \in \Sigma_I$ and $\nu \in \Sigma_J$. These discrete measures may come from the evaluation of continuous densities on a grid, counting features in a structured object, or be empirical measures based on samples. This setting is relevant for several applications, including all known applications of the earth mover’s distance. We show in this section that our stochastic formulation can prove extremely efficient to compare measures with a large number of points.

Discrete Optimization and Sinkhorn. In this setup, the primal $\mathcal{P}_2$, dual $\mathcal{D}_2$ and semi-dual $\mathcal{S}_2$ problems can be rewritten as finite-dimensional optimization problems involving the cost matrix.
\( \mathbf{c} \in \mathbb{R}^{L \times J} \) defined by \( c_{i,j} = c(x_i, y_j) \):

\[
W_\varepsilon(\mu, \nu) = \min_{\pi \in \Pi_{\varepsilon}(\mu, \nu)} \left\{ \sum_{i,j} c_{i,j} \pi_{i,j} + \varepsilon \sum_{i,j} \left( \log \frac{\pi_{i,j}}{\mu_{i,j}} - 1 \right) \pi_{i,j} \right\}, \quad (P_\varepsilon)
\]

\[
= \max_{\mu \in \Pi_{\varepsilon}, \nu \in \Pi_{\varepsilon}} \sum_{i,j} u_i \mu_i + \sum_j v_j \nu_j - \varepsilon \sum_{i,j} \exp \left( \frac{u_i + v_j - c_{i,j}}{\varepsilon} \right) \mu_i \nu_j, \quad (D_\varepsilon)
\]

\[
\hat{H}_\varepsilon(x, \nu) = \sum_{j} v_j \nu_j + \left\{ \begin{array}{l}
- \varepsilon \log \left( \sum_{j} \exp \left( \frac{v_j - c(x,y_j)}{\varepsilon} \right) \nu_j \right) - \varepsilon & \text{if } \varepsilon > 0, \\
\min_j \left( c(x,y_j) - v_j \right) & \text{if } \varepsilon = 0,
\end{array} \right. \quad (S_\varepsilon)
\]

The state-of-the-art method to solve the discrete regularized OT (i.e. when \( \varepsilon > 0 \)) is Sinkhorn’s algorithm [6, Alg.1], which has linear convergence rate [8]. It corresponds to a block coordinate maximization, successively optimizing \((D_\varepsilon)\) with respect to either \( \mu \) or \( \nu \). Each iteration of this algorithm is however costly, because it requires a matrix-vector multiplication. Indeed, this corresponds to a “batch” method where all the samples \((x_i), (y_j)\) are used at each iteration, which has thus complexity \( O(N^2) \) where \( N = \max(I, J) \). We now detail how to alleviate this issue using online stochastic optimization methods.

**Incremental Discrete Optimization when \( \varepsilon > 0 \).** Stochastic gradient descent (SGD), in which an index \( k \) is drawn from distribution \( \mu \) at each iteration can be used to minimize the finite sum that appears in \((S_\varepsilon)\). The gradient of that term \( \hat{H}_\varepsilon(x, \nu) \) can be used as a proxy for the full gradient in a standard gradient ascent step to maximize \( \hat{H}_\varepsilon \).

When \( \varepsilon > 0 \), the finite sum appearing in \((S_\varepsilon)\) suggests to use incremental gradient methods—rather than purely stochastic ones—which are known to converge faster than SGD. We propose to use the stochastic averaged gradient (SAG) [20]. As SGD, SAG operates at each iteration by sampling a point \( x_k \) from \( \mu \), to compute the gradient corresponding to that sample for the current estimate \( \nu \). Unlike SGD, SAG keeps in memory a copy of that gradient. Another difference is that SAG applies a fixed length update, in the direction of the average of all gradients stored so far, which provides a better proxy of the gradient corresponding to the entire sum. This improves the convergence rate to \( \| \hat{H}_\varepsilon(\nu^*_\varepsilon) - \hat{H}_\varepsilon(\nu_k) \| = O(1/k) \), where \( \nu^*_\varepsilon \) is a minimizer of \( \hat{H}_\varepsilon \), at the expense of storing the gradient for each of the \( I \) points. This expense can be mitigated by considering mini-batches instead of individual points. Note that the SAG algorithm is adaptive to strong-convexity and will be linearly convergent around the optimum. The pseudo-code for SAG is provided in Algorithm 1 and we defer more details on SGD for Section 4 in which it will be shown to play a crucial role. Note that the Lipschitz constant of all these terms is upperbounded by \( L = \max_{i,j} \mu_i / \varepsilon \).

**Numerical Illustrations on Bags of Word-Embeddings.** Comparing texts using a Wasserstein distance on their representations as clouds of word embeddings has been recently shown to yield state-of-the-art accuracy for text classification [12]. The authors of [12] have however highlighted that this accuracy comes at a large computational cost. We test our stochastic approach to discrete OT in this scenario, using the complete works of 35 authors (names in supplementary material). We use Glove word embeddings [15] to represent words, namely \( X = Y = \mathbb{R}^{300} \). We discard all most frequent 1,000 words that appear at the top of the file glove.840B.300d provided on the authors’ website. We sample \( N = 20,000 \) words (found within the remaining huge dictionary of relatively rare words) from each authors’ complete work. Each author is thus represented as a cloud of 20,000 points in \( \mathbb{R}^{300} \). The cost function \( c \) between the word embeddings is the squared-Euclidean distance, re-scaled so that it has a unit empirical median on 2,000 points sampled randomly among all vector embeddings. We set \( \varepsilon \) to 0.01 (other values are considered in the supplementary material). We compute all \((35 \times 34)/2 = 595\) pairwise regularized Wasserstein distances using both the Sinkhorn algorithm and SAG. Following the recommendations in [20], SAG’s stepsize is tested for 3 different settings, \( 1/L, 3/L \) and \( 5/L \). The convergence of each algorithm is measured by computing the \( \ell_1 \) norm of the gradient of the full sum (which also corresponds to the marginal violation of the primal transport solution that can be recovered with these dual variables[9]), as well as the \( \ell_2 \) norm of the
deviation to the optimal scaling found after 4,000 passes for any of the three methods. Results are presented in Fig. 1 and suggest that SAG can be more than twice faster than Sinkhorn on average for all tolerance thresholds. Note that SAG retains exactly the same parallel properties as Sinkhorn: all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamlined on GPUs. We used 4 Tesla K80 cards to compute both SAG and Sinkhorn results. For each computation, all of these computations can be streamed...
Figure 2: (a) Plot of $\|\mathbf{v}_k - \mathbf{v}^*_0\|_2 / \|\mathbf{v}^*_0\|_2$ as a function of $k$, for SGD and different values of $\varepsilon$ ($\varepsilon = 0$ being un-regularized). (b) Plot of $\|\mathbf{v}_k - \mathbf{v}^*_0\|_2 / \|\mathbf{v}^*_0\|_2$ as a function of $k$, for SGD and SAG with different number $N$ of samples, for regularized OT using $\varepsilon = 10^{-2}$.

Figure 2(a) shows the evolution of $\|\mathbf{v}_k - \mathbf{v}^*_0\|_2 / \|\mathbf{v}^*_0\|_2$ as a function of $k$. It highlights the influence of the regularization parameters $\varepsilon$ on the iterates of SGD. While the regularized iterates converge faster, they do not converge to the correct unregularized solution. This figure also illustrates the convergence theorem of solution of $(S_\varepsilon)$ toward those $(S_0)$ when $\varepsilon \to 0$, which can be found in the supplementary material.

Figure 2(b) shows the evolution of $\|\mathbf{v}_k - \mathbf{v}^*_0\|_2 / \|\mathbf{v}^*_0\|_2$ as a function of $k$, for a fixed regularization parameter value $\varepsilon = 10^{-2}$. It compares SGD to SAG using different numbers $N$ of samples for the empirical measures $\hat{\mu}_N$. While SGD converges to the true solution of the semi-discrete problem, the solution computed by SAG is biased because of the approximation error which comes from the discretization of $\mu$. This error decreases when the sample size $N$ is increased, as the approximation of $\mu$ by $\hat{\mu}_N$ becomes more accurate.

## 5 Continuous optimal transport using RKHS

In the case where neither $\mu$ nor $\nu$ are discrete, problem $\mathcal{P}$ is infinite-dimensional, so it cannot be solved directly using SGD. We propose in this section to solve the initial dual problem $\mathcal{P}_\varepsilon$, using expansions of the dual variables in two reproducing kernel Hilbert spaces (RKHS). Recall that contrarily to the semi-discrete setting, we can only solve the regularized problem here (i.e. $\varepsilon > 0$), since $\mathcal{P}_\varepsilon$ cannot be cast as an expectation maximization problem when $\varepsilon = 0$. In the particular case where $c(x, y) = |x - y|$, note that the unregularized problem is equivalent to the maximum mean discrepancy problem tackled in [10].

**Stochastic Continuous Optimization.** We consider two RKHS $\mathcal{H}$ and $\mathcal{G}$ defined on $\mathcal{X}$ and on $\mathcal{Y}$, with kernels $\kappa$ and $\ell$, associated with norms $|| \cdot ||_{\mathcal{H}}$ and $|| \cdot ||_{\mathcal{G}}$. Recall the two main properties of RKHS: (a) if $u \in \mathcal{H}$, then $u(x) = \langle u, \kappa(\cdot, x) \rangle_{\mathcal{H}}$ and (b) $\kappa(x, x') = \langle \kappa(\cdot, x), \kappa(\cdot, x') \rangle_{\mathcal{H}}$.

The dual problem $\mathcal{P}_\varepsilon$ is conveniently re-written in (3) as the maximization of the expectation of $f^*(X, Y, u, v)$ with respect to the random variables $(X, Y) \sim \mu \otimes \nu$. The SGD algorithm applied to this problem reads, starting with $u_0 = 0$ and $v_0 = 0$,

$$ (u_k, v_k) \overset{\text{def.}}{=} (u_{k-1}, v_{k-1}) + \frac{C}{\sqrt{k}} \nabla f^*_\varepsilon(x_k, y_k, u_{k-1}, v_{k-1}) \in \mathcal{H} \times \mathcal{G}, \quad (5) $$

where $(x_k, y_k)$ are i.i.d. samples from $\mu \otimes \nu$. The following proposition shows that these $(u_k, v_k)$ iterates can be expressed as finite sums of kernel functions, with a simple recursion formula.

**Proposition 5.1.** The iterates $(u_k, v_k)$ defined in (5) satisfy

$$ (u_k, v_k) \overset{\text{def.}}{=} \sum_{i=1}^{k} \alpha_i(\kappa(\cdot, x_i), \ell(\cdot, y_i)), \quad \text{where} \quad \alpha_i \overset{\text{def.}}{=} \Pi_{B_r} \left( \frac{C}{\sqrt{k}} \left( 1 - e^{-\frac{\varepsilon - \varepsilon_{i-1}}{\varepsilon}(x_i, y_i)} \right) \right), \quad (6) $$

where $(x_i, y_i)_{i=1,...,k}$ are i.i.d samples from $\mu \otimes \nu$ and $\Pi_{B_r}$ is the projection on the centered ball of radius $r$. If the solutions of $\mathcal{P}_\varepsilon$ are in the $\mathcal{H} \times \mathcal{G}$ and if $r$ is large enough, the iterates $(u_k, v_k)$ converge to a solution of $\mathcal{P}_\varepsilon$.

The proof of proposition 5.1 can be found in the supplementary material.
We have shown in this work that the computations behind (regularized) optimal transport can be numerically illustrated. Perspectives for the application of OT to high-dimensional problems address the challenging semi-discrete and continuous cases. All of these three settings can open new possibilities in practical applications. We have also proposed the first known methods that can be implemented with SGD as a proxy for the solution and we denote it by \( \hat{u}^* \), evaluated on a grid where \( \mu \) has non negligible mass.

Algorithm 3 describes our kernel SGD approach, in which both potentials \( u \) and \( v \) are approximated by a linear combination of kernel functions. The main cost is the computation of \( u_{k-1}(x_k) = \sum_{i=1}^{k-1} \alpha_i \kappa(x_k, x_i) \) and \( v_{k-1}(y_k) = \sum_{i=1}^{k-1} \alpha_i \ell(y_k, y_i) \) which yields complexity \( O(k^2) \). Several methods exist to alleviate the running time complexity of kernel algorithms, e.g. random Fourier features [17] or incremental incomplete Cholesky decomposition [25].

Kernels that are associated with dense RHKS are called universal [23] and can thus approach any arbitrary potential. When working over Euclidean spaces \( \mathcal{X} = \mathcal{Y} = \mathbb{R}^d \) for some \( d > 0 \), a natural choice of universal kernel is the kernel defined by \( \kappa(x, x') = \exp(-\|x - x'\|^2/\sigma^2) \). Tuning its bandwidth \( \sigma \) is crucial to obtain a good convergence of the algorithm.

Finally, let us note that, while entropy regularization of the primal problem \( [P_\epsilon] \) was instrumental to be able to apply semi-discrete methods in Sections 3 and 4, this is not the case here. Indeed, since the kernel SGD algorithm is applied to the dual \( [P^*_\epsilon] \), it is possible to replace KL(\( \pi \| \mu \otimes \nu \)) appearing in \( [P^*_\epsilon] \) by other regularizing divergences. A typical example would be a \( \chi^2 \) divergence \( \int_{\mathcal{X} \times \mathcal{Y}} (\pi \text{tr}((x, y))^2 d\mu(x) d\nu(y) \) (with positivity constraints on \( \pi \)).

Numerical Illustrations. We consider optimal transport in 1D between a Gaussian \( \mu \) and a Gaussian mixture \( \nu \) whose densities are represented in Figure 3(a). Since there is no existing benchmark for continuous transport, we use the solution of the semi-discrete problem \( W_\epsilon(\mu, \nu_N) \) with \( N = 10^3 \) computed with SGD as a proxy for the solution and we denote it by \( \hat{u}^* \). We focus on the convergence of the potential \( u \), as it is continuous in both problems contrarily to \( v \). Figure 3(b) represents the plot of \( \|u_k - \hat{u}^*\|_2 / \|\hat{u}^*\|_2 \) where \( u \) is the evaluation of \( u \) on a sample \( (x_i)_{i=1...N'} \) drawn from \( \mu \). This gives more emphasis to the norm on points where \( \mu \) has more mass. The convergence is rather slow but still noticeable. The iterates \( u_k \) are plotted on a grid for different values of \( k \) in Figure 3(c), to emphasize the convergence to the proxy \( \hat{u}^* \). We can see that the iterates computed with the RKHS converge faster where \( \mu \) has more mass, which is actually where the value of \( u \) has the greatest impact in \( F_\epsilon(u) \) (\( u \) being integrated against \( \mu \)).

Conclusion

We have shown in this work that the computations behind (regularized) optimal transport can be considerably alleviated, or simply enabled, using a stochastic optimization approach. In the discrete case, we have shown that incremental gradient methods can surpass the Sinkhorn algorithm in terms of efficiency, taken for granted that the (constant) stepsize has been correctly selected, which should be possible in practical applications. We have also proposed the first known methods that can address the challenging semi-discrete and continuous cases. All of these three settings can open new perspectives for the application of OT to high-dimensional problems.
Acknowledgements  GP was supported by the European Research Council (ERC SIGMA-Vision); AG by Région Ile-de-France; MC by JSPS grant 2670002.

References


A Convergence of \((S_ε)\) as \(ε \to 0\)

The convergence of the solution of \((P_ε)\) toward a solution of \((P_0)\) as \(ε \to 0\) is proved in \([4]\). The convergence of solutions of \((P_ε)\) toward solutions of \((D_0)\) as \(ε \to 0\) is proved for the special case of discrete measures in \([5]\). To the best of our knowledge, the behavior of \((S_ε)\) has not been studied in the literature, and we propose a convergence result in the case where \(ν\) is discrete, which is the setting in which this formulation is most advantageous.

**Proposition A.1.** We assume that \(∀y ∈ \mathcal{Y}, c(⋅,y) ∈ L^1(μ)\), that \(ν = \sum_{j=1}^J ν_j δ_{y_j}\), and we fix \(x_0 ∈ X\). For all \(ε > 0\), let \(v^ε\) be the unique solution of \((S_ε)\) such that \(v^ε(x_0) = 0\). Then \((v^ε)_ε\) is bounded and all its converging sub-sequences for \(ε \to 0\) are solutions of \((S_0)\).

We first prove a useful lemma.

**Lemma A.2.** If \(∀y, x ↦ c(x,y) ∈ L^1(μ)\) then \(H_ε\) converges pointwise to \(H_0\).

**Proof.** Let \(α_j(x) \overset{def}{=} v_j - c(x,y_j)\) and \(j^* \overset{def}{=} \arg \max_j α_j(x)\).

On the one hand, since \(∀y_j, \ ν_j(x) ≤ \ ν_j(x)\) we get

\[
ε \log(\sum_{j=1}^J e^{α_j(x)/ε}) = ε \log(e^{α_{j^*}(x)/ε} \sum_{j=1}^J e^{α_j(x)-α_{j^*}(x)/ε}) \leq α_{j^*}(x) + ε \log(\sum_{j=1}^J ν_j) = α_{j^*}(x)
\]

(7)

On the other hand, since log is increasing and all terms in the sum are non negative we have

\[
ε \log(\sum_{j=1}^J e^{α_j(x)/ε}) ≥ ε \log(e^{α_{j^*}(x)/ε} ν_{j^*}) = α_{j^*}(x) + ε \log(ν_{j^*}) \xrightarrow{ε \to 0} α_{j^*}(x)
\]

(8)

Hence \(ε \log(\sum_{j=1}^J e^{α_j(x)/ε}) \xrightarrow{ε \to 0} α_{j^*}(x)\) and \(ε \log(\sum_{j=1}^J e^{α_j(x)/ε}) \leq α_j(x)\).

Since we assumed \(x ↦ c(x,y) ∈ L^1(μ)\), then \(α_j \in L^1(μ)\) and by dominated convergence we get that \(H_ε(ν) \xrightarrow{ε \to 0} H_0(ν)\). \(\square\)

**Proof of Proposition A.1.** First, let’s prove that \((v^ε)_ε\) has a converging subsequence. With similar computations as in Proposition 2.1 we get that \(v^ε(y_i) = − ε \log(\int_X e^{c(x,y)-c(x,y_i)} dμ(x))\). We denote by \(\tilde{v}_ε\) the c-transform of \(u_ε\) such that \(\tilde{v}_ε(y_i) = \min_{y ∈ X} c(x,y) - u_ε(x)\). From standard results on optimal transport (see \([19]\), p.11) we know that \(\|\tilde{v}_ε(y_i) - \tilde{v}_ε(y_j)\| ≤ ω(\|y_i - y_j\|)\) where \(ω\) is the modulus of continuity of the cost \(c\). Besides, using once again the soft-max argument we can bound \(\|v_ε(y) - \tilde{v}_ε(y)\|\) by some constant \(C\).

Thus we get that:

\[
|v_ε(y_i) - v_ε(y_j)| ≤ |v_ε(y_i) - \tilde{v}_ε(y_i)| + |\tilde{v}_ε(y_i) - \tilde{v}_ε(y_j)| + |\tilde{v}_ε(y_j) - v_ε(y_j)| \leq C + ω(\|y_i - y_j\|) + C
\]

(9)

(10)

Besides, the regularized potentials are unique up to an additive constant. Hence we can set without loss of generality \(v_ε(y_0) = 0\). So from the previous inequality yields:

\[
v_ε(y_i) ≤ 2C + ω(\|y_i - y_0\|)
\]

(11)

So \(v^ε\) is bounded on \(\mathbb{R}^J\) which is a compact set and thus we can extract a subsequence which converges to a certain limit that we denote by \(\tilde{v}\).

Let \(v^* \in \arg \max_ν H_0\). To prove that \(\tilde{v}\) is optimal, it suffices to prove that \(H_0(v^*) ≤ H_0(\tilde{v})\).

By optimality of \(v_ε\),

\[
H_ε(v^*) ≤ H_ε(v_ε)
\]

(12)

The term on the left-hand side of the inequality converges to \(H_0(v^*)\) since \(H_ε\) converges pointwise to \(H_0\). We still need to prove that the right-hand term converges to \(H_0(\tilde{v})\).

By the Mean Value Theorem, there exists \(\tilde{v}_ε \overset{def}{=} (1 - t^ε)v_ε + t^ε\tilde{v}\) for some \(t^ε \in [0,1]\) such that

\[
|H_ε(v_ε) - H_ε(\tilde{v})| ≤ ∥∇H_ε(\tilde{v}_ε)∥ ∥v_ε - \tilde{v}∥
\]

(13)
The gradient of $H_\varepsilon$ reads
\[ \nabla_v H_\varepsilon(v) = \nu - \pi(v) \] (14)
where \( \pi_i(v) = \frac{\int_x e^{v_i - c(x,y_j)} \nu_i d\mu(x)}{\int_x \sum_{j=1}^J e^{v_j - c(x,y_j)} \nu_j d\mu(x)} \).

It is the difference of two elements in the simplex thus it is bounded by a constant $C$ independently of $\varepsilon$.

Using this bound in (13) get
\[ H_\varepsilon(\bar{v}) - C \| \nu_\varepsilon - \bar{v} \| \leq H_\varepsilon(\nu_\varepsilon) \leq H_\varepsilon(\bar{v}) + C \| \nu_\varepsilon - \bar{v} \| \] (15)

By pointwise convergence of $H_\varepsilon$ we know that $H_\varepsilon(\bar{v}) \to H_0(\bar{v})$, and since $\bar{v}$ is a limit point of $\nu_\varepsilon$ we can conclude that the left and right hand term of the inequality converge to $H_0(\bar{v})$. Thus we get $H_\varepsilon(\nu_\varepsilon) \to H_0(\bar{v})$.

**B Discrete-Discrete Setting**

The list of authors we consider is: KEATS, CERVANTES, SHELLEY, WOOLF, NIETZSCHE, PLUTARCH, FRANKLIN, COLERIDGE, MAUPASSANT, NAPOLEON, AUSTEN, BIBLE, LINCOLN, PAINE, DELAFONTAINE, DANTE, VOLTAIRE, MOORE, HUME, BURROUGHS, JEFFERSON, DICKENS, KANT, ARISTOTLE, DOYLE, HAWTHORNE, PLATO, STEVENSON, TWAIN, IRVING, EMERSON, POE, WILDE, MILTON, SHAKESPEARE.
Proof of convergence of stochastic gradient descent in the RKHS

The SGD algorithm applied to the regularized continuous OT problem reads, starting with $u_0 = 0$ and $v_0 = 0$,

$$(u_k, v_k) \overset{\text{def.}}{=} (u_{k-1}, v_{k-1}) + \frac{C}{\sqrt{k}} \nabla f_{\varepsilon}(x_k, y_k, u_{k-1}, v_{k-1}) \in \mathcal{H} \times \mathcal{G},$$

where $(x_k, y_k)$ are i.i.d. samples from $\mu \otimes \nu$. The following proposition shows that these $(u_k, v_k)$ iterates can be expressed as finite sums of kernel functions, with a simple recursion formula.

**Proposition C.1.** The iterates $(u_k, v_k)$ defined in (16) satisfy

$$(u_k, v_k) \overset{\text{def.}}{=} \sum_{i=1}^{k} \alpha_i \langle \kappa(\cdot, x_i), \ell(\cdot, y_i) \rangle,$$

where $(x_i, y_i)_{i=1}^{\ldots k}$ are i.i.d samples from $\mu \otimes \nu$ and $\Pi_{B_r}$ is the projection on the centered ball of radius $r$. If the solutions of $(D_\varepsilon)$ are in the $\mathcal{H} \times \mathcal{G}$ and if $r$ is large enough, the iterates $(u_k, v_k)$ converge to a solution of $(D_\varepsilon)$.

**Proof.** Rewriting $u(x)$ and $v(y)$ as scalar products in $f_{\varepsilon}(X, Y, u, v)$ yields

$$f_{\varepsilon}(x, y, u, v) = \langle u, \kappa(x, \cdot) \rangle_{\mathcal{H}} + \langle v, \ell(y, \cdot) \rangle_{\mathcal{G}} - \varepsilon \exp \left( \frac{\langle u, \kappa(x, \cdot) \rangle_{\mathcal{H}} + \langle v, \ell(y, \cdot) \rangle_{\mathcal{G}} - c(x, y)}{\varepsilon} \right).$$

Plugging this formula in iteration (16) yields: $(u_k, v_k) = u_{k-1} + \omega(k(x_k), \ell(\cdot, y_k))$, where $\omega_k$ is defined in (17). Since the parameters $(\alpha_i)_{i<k}$ are not updated at iteration $k$, we get the announced formula. Putting a bound on the iterates of $\alpha$ by projecting on $B_r$ ensures convergence [7].