Lab on Intelligent Robotic Manipulation

TECHNISCHE UNIVERSITÄT DARMSTADT

Magnus Dierking
September 25, 2023

Contents

1. Motivation **2**

2. Foundations **2**
2.1. Information Theory . . . 2
2.2. Optimal Transport . . . 5
2.3. Entropic Regularization of Optimal Transport . . . 8
2.4. Sinkhorn's Algorithm . . . 10
2.5. Kernel Density Estimation . . . 11

3. Annotated InfoOT **12**
3.1. Measuring global Structure with Mutual Information . . . 12
3.2. Incorporating Domain Information . . . 15
3.3. Solving via Projected Gradient Descent . . . 16
3.4. The Conditional Projection . . . 17

4. Conclusions **20**
4.1. Discussion . . . 20
4.2. Outlook . . . 20

A. Linear Programming Example **22**

B. Mirror Descent Algorithm **22**

1. Motivation

The aim of this project is to extend the work of Younes, Schaub-Meyer and Chalvatzaki [25] in the scope of the "Lab on Intelligent Robotic Manipulation" course offered by PEARL lab at TU Darmstadt. Planned as a two-semester course, this summer term serves as a research phase for a more practical and problem-oriented second part of the project. Our goal is to study theoretical concepts from information theory and optimal transport in order to develop ideas and have a solid base of knowledge to be able to explore a variety of options later on. Hence, while the main focus is to gain an understanding of the work by Chuang et. al. [4], we also place a high value on keeping a good amount of flexibility with regard to the future of this work by also focusing on the underlying, more general concepts.
With this report, we try to document our learning process over the course of this semester and document the ideas we developed for the second part of this project. To do so, we provide the most important mathematical formulations alongside intuitive explanations and easy-to-grasp examples. The examples and visualizations are inspired by our discussions and whiteboard-sketches during the semester.

2. Foundations

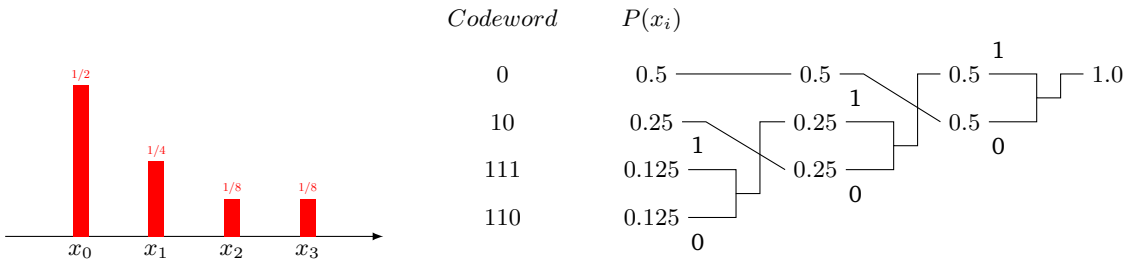
Before considering the papers for this research part of the project, we begin by discussing the core concepts from information theory and optimal transport in order to have an initial theoretical understanding of the most important topics. For the basics in information theory, we mainly rely on the standard book by Thomas M. Cover and Joy A. Thomas [5], while our discussions about optimal transport are heavily driven by the works of G. Peyré and M. Cuturi, e.g. [3], [6] or [18]. In order to provide visual intuition, we use the *Python Optimal Transport* [7] and *TikZ* [23] libraries.

2.1. Information Theory

Information Theory is a field of study concerned with the mathematical abstraction of information and the resulting implications towards its quantification, storage and compression in the form of data as well as reliable communication. In the following, all formulas were taken from the book by Thomas M. Cover and Joy A. Thomas [5].

Information and Entropy

Based on the well-known work by Claude Shannon [22], Information Theory is centered around the concept of information being the resolution of uncertainty about the realization x of a (discrete) random variable X. This resolution yields a gain in information that is assumed to be inversely proportional to the probability of the observation. A popularly used analogy is that the information is the level of surprise when observing the realization.
While this doesn't seem very intuitive at first, one has to keep in mind that information theory was developed as a theoretical framework for communication scenarios. For this reason, we will consider an example from coding theory to build understanding of the core concepts. Assume one wants to efficiently encode a letter in order to transmit it to a friend. For simplicity, we assume our



(a) Source distribution.

(b) Coding scheme.

Figure 1: Optimal and prefix-free encoding of a given random source via Huffman encoding scheme. The resulting codewords are depicted in the middle.

alphabet has only 4 letters x_1, x_2, x_3 and x_4 and the occurrence of each character is according to the distribution shown in Figure 1a. If we want to encode the message using binary –as it is the case in most applications– in the most efficient way, shorter codewords should be assigned to more frequent letters in order to keep the total message length as short as possible. One scheme of constructing a so-called optimal and prefix-free code for our distribution is given by the Huffmann coding scheme [9] shown in Figure 1b. As an intuition for the Huffman scheme, one can consider trying to guess the a realization of our random variable with binary questions, which is the result of our code having a binary alphabet. The most efficient questioning if all we have is the distribution can be done according to the decision tree depicted in Figure 2. The Huffman algorithm now essentially builds this tree from the bottom up and

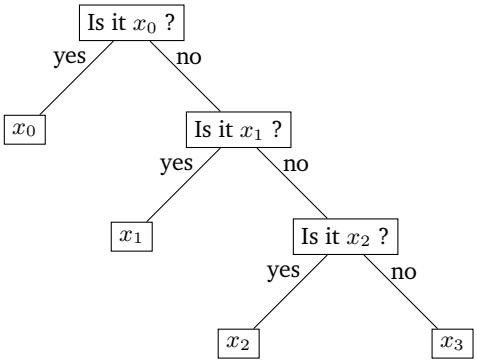


Figure 2: Decision tree used to optimally guess a realization of the random source given in Figure 1a.

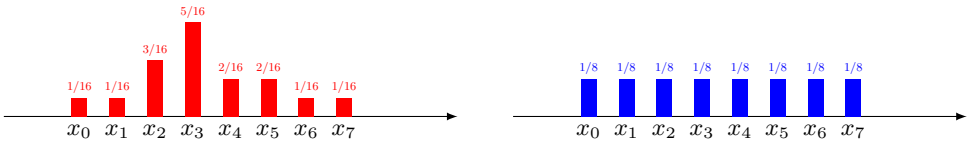
thereby assigns longer codewords to less frequent events and vice versa. From this, the information gained by observing x_0 should be lower than if we observe e.g. x_3, since the occurrence of x_0 is less surprising given the distribution and we would need to ask less questions to identify the outcome. Mathematically, the amount of information gained by observing the realization x of the random variable X that has non-zero probability $P(X = x)$ is defined as

$$I(x) = -\log(P(X = x)), \tag{1}$$

where the basis of the logarithm determines the unit. In our case the base-2 logarithm yields the unit bits, which we will implicitly use throughout this report. For example, observing x_0 yields 1 bit of information, since we would have to ask one binary question in order to guess it, while x_2 yields 3 bit of information. If we now consider that the random variable can take the values of an alphabet $\mathcal{X} = \{x_1, ..., x_N\}$, the Shannon entropy of the random variable –in this context often referred to as the random source– is defined as the average information gain

$$H(X) = \mathbb{E}[I(X)] = \sum_{i=1}^{N} P(x_i) \log_2 \left(\frac{1}{P(x_i)}\right) \quad \left[\frac{\text{bits}}{\text{symbol}}\right], \tag{2}$$

when observing a realization of this source. Intuitively, the entropy can be seen as a quantification of randomness for a given distribution, e.g. the two provided in Figure 3. Guessing a realization of the blue distribution in Figure 3b comes down to completely



(a) Random source with $H(X) \approx 2.727$ bit.

(b) Random source with $H(X) = 3$ bit.

Figure 3: 1-D histograms resulting in different entropies.

random guessing, as the structure of the distribution can't be employed to make the guesswork more efficient. Contrary, the red distribution in Figure 3a can be guessed more efficiently by using the same approach as before. This "higher" inherent randomness of the blue distribution results in a higher entropy.

It is important to note that the examples regarding the information and encoding were specifically chosen to yield an integer number of bits in every computation. In general, one faces fractions of bits for the information and therefore also in the decision tree. Additionally, because these fractions of bits can't be realized, the Huffman scheme won't encode every source quite as efficiently as this one. Apart from that, one can always construct a less efficient encoding by simply using additional, redundant bits for each codeword. According to Shannon's Theory, the entropy is a bound for how efficient an information source can be encoded or compressed.

Conditional and Joint Entropy

In applications, you often work with several random variables that are somehow correlated. For this, the conditional entropy of X given the observation of another random variable Y can be defined via

$$H(X|Y) = \sum_j P(y_j) \sum_i P(x_i|y_j) \log\left(\frac{1}{P(x_i|y_j)}\right) \quad \left[\frac{\text{bits}}{\text{symbol}}\right]. \tag{3}$$

Extending our example, we could drop the assumption that you want to encode your message in the most efficient way, but instead you want to encode it in order to prevent people who intercept the message from being able to decode it. You and your friend visited

the same cryptography lecture at university and you both have a book on cryptography describing several encoding schemes (keys) $\mathcal{K}$ at home. Before transmitting, you agree in a specific key $k \in \mathcal{K}$ to use. The random variable Y could describe the distribution of received symbols. Therefore, if you don't use encryption and instead the most efficient compression approach from the section before, the conditional entropy would approach zero the longer the message is, since the occurrence of each character is directly correlated to the used code symbol. Inversely, to reduce the probability that somebody could guess the encryption key by using statistics of the receivers message, one could either send very short messages or use a key that alters the distribution of symbols as much as possible. Summing up the uncertainty $H(Y)$ about which key was used and the conditional entropy $H(X|Y)$ yields the joint entropy

$$H(X,Y) = H(X|Y) + H(Y) = \sum_i \sum_j P(x_i, y_j) \log(\frac{1}{P(x_i, y_j)}), \tag{4}$$

which in the light of our previous examples denotes how much guesswork is needed on average to guess a pair from the joint distribution of two random variables. In the worst case, the joint distribution doesn't provide any additional structure to leverage for the guessing. This corresponds to statistical independence, where guessing both realizations of the pair independent from each other yields the same average number of questions. Consequently, the joint entropy is bounded by

$$H(X,Y) \leq H(X) + H(Y), \tag{5}$$

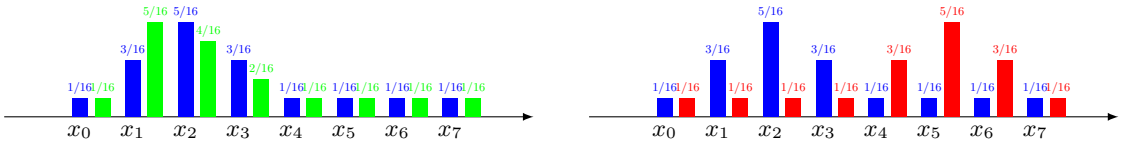
with equality only if both random variables are independent.

Kullback-Leibler Divergence

Assuming that we computed a statistic Q of the received message and used it to decode the un-encrypted message, is there a way to quantify how "close" our statistic is to the true distribution P ? A possible answer to this comes in the form of the well-known (at least in the machine learning community) Kullback-Leibler divergence

$$D_{P|Q}(X) = \sum_{i=1}^{N} P(x_i) \log\left(\frac{P(x_i)}{Q(x_i)}\right) \quad [\frac{\text{bits}}{\text{symbol}}], \tag{6}$$

which quantifies the average remaining uncertainty about a realization of X if we assume that Q instead of P is the true distribution.



(a) $D_{P|Q}(X) \approx 0.072$ bit. (b) $D_{P|Q}(X) \approx 0.977$ bit.

Figure 4: Two examples of the Kullblack-Leibler divergence using 1-D histogram.

We can also relate this back to our coding example and Figure 2: If we guess a realization using a decision tree based on the distribution Q, the Kullback-Leibler divergence tells us the average number of additional binary questions we have to ask in order to guess a realization x. These additional questions are the result of our decision tree being suboptimal for the true distribution P. Consider e.g. Figure 4a, where the decision tree for the green distribution may be suboptimal, but we will at least start asking about the highly-likely events first, although not in optimal order. In contrast, the situation in Figure 4b will lead to us always asking about relatively unlikely events, e.g. x_5 first, yielding more unnecessary additional questions. While this can gives us a notion of distance between two distributions intuitively, the Kullback-Leibler divergence is not a mathematical distance, as it is not symmetric in the sense of $D_{P|Q}(X) \neq D_{Q|P}(X)$. Additionally, a major limitation is that both distributions are required to have the same support. Considering Equation (6) again, a zero probability for either distributions yields infinite log terms.

Mutual Information

A more involved concept of information theory is mutual information, which is the average amount of information gained about X by observing realizations $y_j \in \mathcal{Y}$ of a distinct, correlated (discrete) random variable Y. Formally, it is defined via

$$I(X,Y) = \sum_{i=1}^{N} \sum_{j=1}^{M} P(x_i, y_j) \log(\frac{P(x_i, y_j)}{P(x_i)P(y_j)}), \tag{7}$$

which is equivalent to $H(X) - H(X|Y)$ and is therefore often phrased as the reduction of uncertainty about X due to the knowledge of Y. Continuing our example from the conditional entropy in Equation (3), the mutual information quantifies the reduction of uncertainty about the true decrypted symbol distribution by the distribution of the encrypted message. Consider again that we didn't encrypt the message at all and instead only focused on efficiency by using the Huffmann scheme., resulting in the conditional entropy approaching zero. The mutual information in this case is $I(X;Y) = H(X) - H(X|Y) = H(X)$, because the statistic we can obtain

about the received message will maximally inform us about the underlying distribution, reducing the uncertainty about X to zero. Using an encryption key k, the conditional entropy will increase, because the application of the key will alter the statistic and thereby create uncertainty about X. This also has an effect on the mutual information, as the amount of uncertainty created by the encryption will be subtracted in the formula stated before, decreasing the mutual information between the original distribution X and the statistic Y.
Another interesting intuition is given by the intimate relation of the mutual information to the Kullback-Leibler divergence. The formula stated in Equation (7) can be rephrased as

$$I(X;Y) = D_{P_{X,Y}|P_X P_Y}(X,Y) = \mathbb{E}_{(x,y)\sim P_{X,Y}}\left[\log(\frac{1}{P_X(x)P_Y(y)}) - \log(\frac{1}{P_{X,Y}(x,y)})\right], \tag{8}$$

stating that the mutual information corresponds to the Kullback-Leibler divergence between the actual joint probability distribution of X and Y and the joint distribution when assuming both random variables are independent. The second equality in Equation (8) reveals a connection to our initial example of guessing realizations of a random variable by leveraging known probabilities (compare Figure 2): Assume we want to guess a joint realization (x, y) of two random variables X and Y. The mutual information quantifies how much "guesswork" we can save by jointly guessing the pair (x, y) using the joint distribution instead of guessing x and y separately using the marginals. This encompasses how much additional structure or information is encoded in the joint distribution when comparing it to the assumption that both of them are independent.

2.2. Optimal Transport

In fields working with probabilistic models such as machine learning, comparing probability distributions is an essential and common task. While the Kullback-Leibler divergence of the preceding section provides a notion of dissimilarity, it is not symmetric and therefore doesn't provide the properties to be a distance in the mathematical sense.
Optimal Transport is a mathematical framework originally motivated by efficiently transporting and allocating resources, e.g. soldiers on the front-line or goods for factories. The original mathematical formulation of this problem dates back to Gaspard Monge in 1781, but his fomulation was limited in application. The main issue with the original Monge problem was that is restricted to deterministic maps, meaning that e.g. all the goods from one storage had to be assigned to a single factory. This limitation can lead to ill-posed problems, which was addressed by the Soviet mathematician Leonid Kantorovich with his Kantorovich relaxation [18, Sec. 2.3]. This formulation employs the idea of using a probabilistic transport rather than a deterministic one, allowing for mass-splitting by e.g. assigning fractions of goods from a storage to distinct factories. In the following, we will limit ourselves to the special case of discrete measures, although it is worth mentioning that the Kantorovich problem can be stated between arbitrary measures (see [18, Rem. 2.13]).
Consider two metric spaces $\mathcal{X}$ and $\mathcal{Y}$ with discrete probability measure α on $\mathcal{X}$ and β on $\mathcal{Y}$. These measures encode our source and target distributions, meaning that $X \sim \alpha$ and $Y \sim \beta$. Both of these can be represented by their respective probability mass function in form of histograms, $\mathbf{a}$ for the source and $\mathbf{b}$ for the target, such that

$$\alpha = \sum_{i=1}^{n} \mathbf{a}_i \delta_{x_i}, \quad x_i \in \mathcal{X} \quad \text{and} \quad \beta = \sum_{j=1}^{m} \mathbf{b}_j \delta_{y_j}, \quad y_i \in \mathcal{Y}. \tag{9}$$

Each entry of a histogram thereby tells us how much probability mass is available at a particular element of the respective support. The amount of mass that is transported from each element δ_{x_i} of the support of α to an element of the support of β can be encoded as entries of a matrix $\mathbf{\Gamma} \in \mathbb{R}_+^{n\times m}$, denoted transportation plan. This transportation plan has to satisfy the conditions of being a mapping from our exact source to our target. The resulting feasible set of plans/matrices is bounded and denoted via

$$\mathbf{U}(\mathbf{a},\mathbf{b}) = \left\{\mathbf{\Gamma} \in \mathbb{R}_+^{n\times m} : \mathbf{\Gamma}\mathbb{1}_m = \mathbf{a} \quad \text{and} \quad \mathbf{\Gamma}^{\mathrm{T}}\mathbb{1}_n = \mathbf{b}\right\}. \tag{10}$$

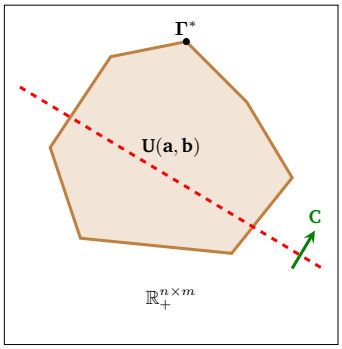


Figure 5: Qualitative sketch of transport polytope, where each edge depicts a linear constraint on the desired solution, while the arrow depicts the optimization direction enforced by the cost matrix $\mathbf{C}$.

In terms of convex optimization, the feasible set is also sometimes called the transport polytope, as each row of the two equalities in Equation (10) can be formalized as one of $n + m$ linear constraints of a linear program. For the full derivation we refer to [18, Sec. 3.1] and instead try to explain based on intuition inspired by [6]. An important thing to note from the derivation is that the resulting feasible set equates to a region inside a polytope, while a (possibly non-unique) optimal plan $\mathbf{\Gamma}^*$ will always be found on the boundary. For clarification consider the sketch in Figure 5, depicting such a polytope as a subset in the space of real matrices with fitting dimensions. In a similar fashion, the cost of moving mass between elements of the support of $\mathbf{a}$ and $\mathbf{b}$ can be written as a matrix $\mathbf{C} \in \mathbf{R}^{m\times n}$, such that the entry $\mathbf{C}_{ij}$ denotes the cost of moving a unit of mass from δ_{x_i} to δ_{y_j}. Given a transportation plan and the cost matrix, the full cost of the problem can thereby be computed by

$$\sum_{i=1}^{n}\sum_{j=1}^{m} \mathbf{\Gamma}_{ij}\mathbf{C}_{ij} = \langle \mathbf{\Gamma}, \mathbf{C}\rangle. \tag{11}$$

The dotted red line in Figure 5 indicates a hyperplane on which this cost is constant, while the green arrow shows the direction in

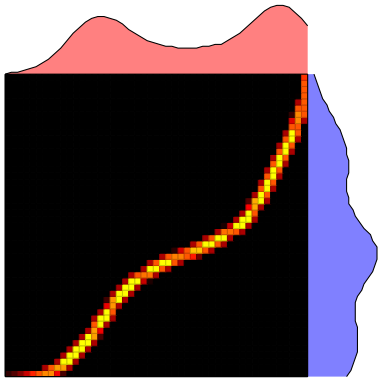


Figure 6: Exemplary optimal transport plan with source- (blue) and target-distribution (red) resulting from the Kantorovich formulation of Equation (12). Brighter colors correspond to higher values.

which to move this hyperplane in order to minimize the cost. If the described intuition for linear programming is unclear, please refer to Appendix A for a small example. Combining all of the above, the Kantorovich formulation of the optimal transport problem for discrete measures can be formulated as

$$L_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{\Gamma} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{\Gamma}, \mathbf{C} \rangle. \tag{12}$$

An example using one-dimensional distributions is provided in Figure 6, where the source- (blue) and target-distribution (red) are depicted along the resulting transportation plan. The support of both variables is the interval $\{0, ..., 49\} \subset \mathbb{N}_0$ and we use the squared euclidean distance to compute the cost matrix. Other metrics and higher dimensional distributions are possible, but we will stick to this setting to introduce the most important concepts.
Apart from better properties regarding solvability, the Kantorovich formulation also allows for a probabilistic interpretation of optimal transport. $\mathbf{U}(\mathbf{a}, \mathbf{b})$ can be interpreted as the set of all possible transport-weighted joint distributions of (X, Y) with marginals $\mathbf{a}$ and $\mathbf{b}$.

$$L_{\mathbf{C}}(\mathbf{a}, \mathbf{b}) = \min_{(X,Y)} \left\{ \mathbb{E}_{(X,Y)}[c(X, Y)] : \; X \sim \alpha, Y \sim \beta \right\}, \tag{13}$$

Under some assumptions about the underlying space and the cost matrix $\mathbf{C}$ in Equation (12), the optimal transport framework allows us to obtain a proper distance definition between histograms.

Wasserstein Distance
Assume both discrete probability measures for source and target are defined on the same metric space, namely $\mathcal{X} = \mathcal{Y}$ and $\mathcal{X}$ is equipped with the distance $d \colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$. The function d is considered a distance, if for $x_i, x_j, x_k \in \mathcal{X}$ it fulfills

(i) $d(x_i, x_j) = d(x_j, x_i) \geq 0$,

(ii) $d(x_i, x_j) = 0$ is and only if $x_i = x_j$,

(iii) $d(x_i, x_k) \leq d(x_i, x_j) + d(x_j, x_k)$.

Then, the optimal transport problem allows us to define the p-Wasserstein distance (given d) via

$$\mathcal{W}_p(\alpha, \beta) \coloneqq L_{\mathbf{C}'}(\mathbf{a}, \mathbf{b})^{\frac{1}{p}}, \tag{14}$$

where $\mathbf{C}'_{ij} = d(x_i, x_j)^p$.

Changing the Metric

The optimal transport framework introduced above exclusively relies on the cost matrix along with the source- and target distributions to compute the transportation plan. Consequently, every element of the source and target support is treated individually, considering only the information of how expensive each element is to move according to the used metric. One example is the squared euclidean metric as in Figure 6, where the cost of moving mass from one element of the support to another increases quadratically with their distance. Therefore, a plan that minimizes the cost tries to restrict the transportation of an element's probability mass to close-by elements. If source and target distribution are identical and no mass moving is required, the transportation plan corresponds to a diagonal line colored according to the distribution with a total cost of zero. Consequently, any transportation plan between distinct distributions tries to find a trade-off between keeping the transportation as local as possible, while also meeting the constraints enforced by the distributions, resulting in the bended diagonal structure observed in Figure 6.

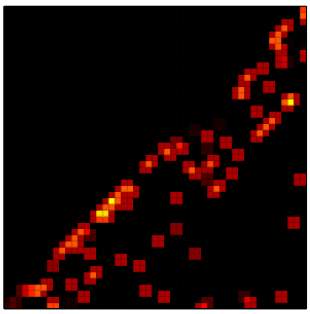
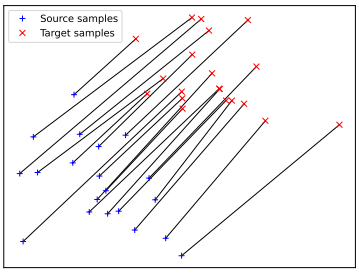


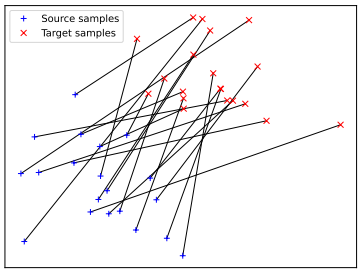
Figure 7: Transportation plan using the same source- and target distributions as in Figure 6, but the L^1/cityblock instead of the squared euclidean metric to compute the cost matrix.

This behavior can be weakened by instead choosing a different metric that punishes transportation between more distant elements less harshly, e.g. the *cityblock* or L^1-metric $d_{L^1}(x_1, x_2) = |x_1 - x_2|$. As Figure 7 indicates, choosing this linear-increasing cost results in fractions of mass being transported further as with the squared-euclidean metric, indicated by the colored blocks that are further away from the diagonal.

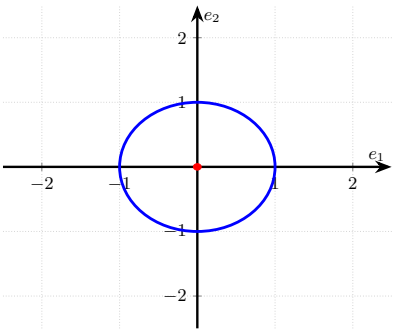
It is also possible to compute optimal transport maps solely on the basis of a cost matrix by assuming a uniform distribution for source and target. This allows us to align two datasets of samples purely based on a notion of distance between them. The uniform distributions and the fact that samples can occur multiple times in any order, results in 1-D plots such as in Figure 6 being less appealing. In order to still provide visual intuition, we therefore introduce a 2-D setting in Figures 8a and 8b, where black lines depict the alignments. In these examples, the amount of probability mass that is transported between two samples is encoded via a gray scale. The two plots contrast the transportation resulting from using the euclidean metric and the cityblock metric to compute the cost matrix. The distinct behavior can be explained by comparing the alignments with the plots below, where equidistant points from the origin are depicted for both metrics. While the euclidean metric on the left punishes deviations for each direction equally, resulting in a circle, the cityblock metric puts less emphasis on directions the more they align with the standard basis (e_1, e_2) of $\mathbb{R}^2$. As a result, the alignments resemble this behavior.



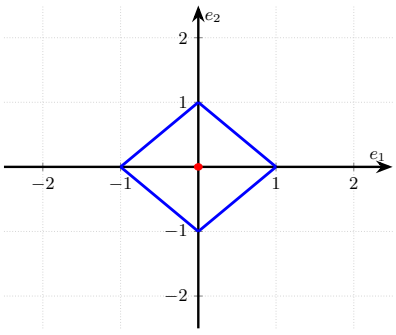
(a) Euclidean metric.



(b) Cityblock metric.



(c) $d(p, q) = \sqrt{\sum_i (p_i - q_i)^2}$.



(d) $d_{L^1}(p, q) = \sum_i |p_i - q_i|$.

Figure 8: 2-D results for sample-based optimal transport using uniform distributions and different metrics for the cost matrix. For each metric, equidistant points from the origin are depicted below.

2.3. Entropic Regularization of Optimal Transport

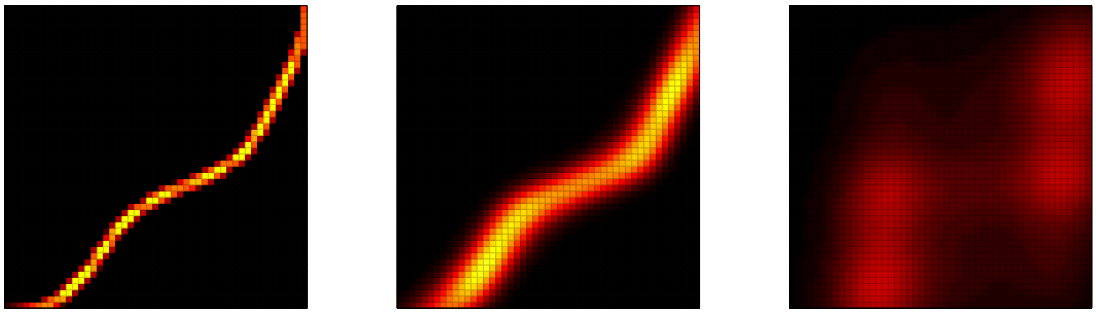
The mathematical framework of optimal transport provides a theoretic basis for computing distances between distributions. However, this rigor comes at the expense of high computational complexity. Computing optimal transport distances between n-dimensional histograms using linear programming requires at least $\mathcal{O}(n^3 \log(n))$ operations [16], which for a long time limited it's applicability. In order to speed up computations, M. Cuturi [6] proposed to regularize the objective in Equation (12) with an entropy term as in Section 2.1 of the transportation plan

$$H(\mathbf{\Gamma}) = \sum_{i,j} \mathbf{\Gamma}_{i,j} \log\left(\frac{1}{\mathbf{\Gamma}_{i,j}}\right). \tag{15}$$

Considering what we learned about the entropy of a distribution, the entropy of a plan is low if it has a distinct structure, meaning only limited "transportation routes" are used. Against that, the entropy is high if the probability mass of the source is split and transportation is "spread out" among many routes, leading to denser matrices for the plan. Because of this, the regularized objective

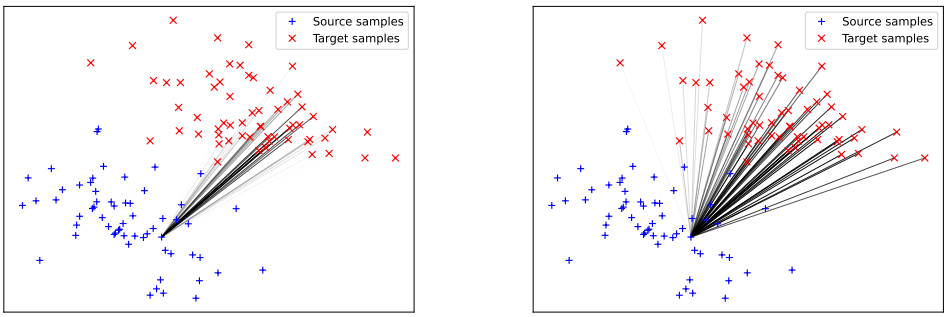
$$L_{\mathbf{C}}^{\epsilon}(\mathbf{a}, \mathbf{b}) = \min_{\mathbf{\Gamma} \in \mathbf{U}(\mathbf{a}, \mathbf{b})} \langle \mathbf{\Gamma}, \mathbf{C} \rangle - \epsilon H(\mathbf{\Gamma}), \tag{16}$$

favors sub-optimal transportation plans that aren't necessarily on the boundary of the polytope, if the transportation is sufficiently spread. The effect of this regularization is shown in Figures 9 and 10: increasing ϵ results in larger influence of the entropy on the minimization objective, which favors denser transportation plans and with that increasing fragmentation of the source mass into smaller fractions that are transported using more and more routes.



(a) $\epsilon = 0$. (b) $\epsilon = 10^{-2}$. (c) $\epsilon = 0.25$.

Figure 9: Regularized transportation plan for the same source- and target-distribution as in Figure 6 with increasing values of ϵ. With higher influence of the entropic regularizer, the plan becomes less and less sparse, which is often denoted "entropic blur".



(a) $\epsilon = 0.05$. (b) $\epsilon = 0.5$.

Figure 10: Example of entropic blur for 2-D samples and uniform distributions as in Figure 8. For clarity, the transport is only depicted for one source sample. The α-values for the lines are scaled with the normalized values of the respective transportation plan.

Using the provided background in information theory and [6], the regularization can be illustrated very elegantly by continuing the linear optimization example of Figure 5. We know from Equation (13) that the transportation plan Γ is interpretable as a joint distribution. Thereby, the entropy of such a plan is a joint entropy, which by using the results from Section 2.1 can be bounded by

$$H(\mathbf{\Gamma}) \leq H(\mathbf{a}) + H(\mathbf{b}) = H(\mathbf{a}\mathbf{b}^\intercal). \tag{17}$$

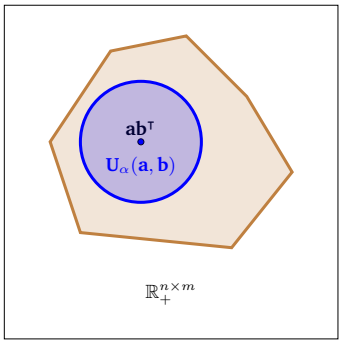


Figure 11: Qualitative sketch of transport polytope, with subset of transportation plans that have sufficiently small mutual information.

The transportation plan $\mathbf{a}\mathbf{b}^\intercal$ of the independent case is by construction an element of the feasible set (10). By using the relation between mutual information and the Kullback-Leibler Divergence of joint entropies in (8), we can rewrite Equation (3) using a threshold

$$H(\mathbf{\Gamma}) \leq H(\mathbf{a}\mathbf{b}^\intercal) - \alpha \tag{18}$$

$$I(\mathbf{\Gamma}; \mathbf{a}\mathbf{b}^\intercal) \leq \alpha. \tag{19}$$

Requiring $\mathbf{\Gamma} \in \mathbf{U}(\mathbf{a}, \mathbf{b})$, we can use the inequality above to define a subset $\mathbf{U}_\alpha(\mathbf{a}, \mathbf{b}) \subseteq \mathbf{U}(\mathbf{a}, \mathbf{b})$, that includes all transportation plans that yield sufficiently small mutual information between the input and output distribution. Before, when optimizing over $\mathbf{U}(\mathbf{a}, \mathbf{b})$, we were looking for a transportation plan that encoded as much information as possible about the underlying geometry (i.e. how to minimize the cost) by seeking a solution on the boundary of the transportation polytope in the direction dictated by $\mathbf{C}$. Now, by requiring $\mathbf{\Gamma} \in \mathbf{U}_\alpha(\mathbf{a}, \mathbf{b})$, we limit how much additional information can be encoded in the transportation plan compared to the independent case (for which all information is already present in the marginals), leading to a solution that is possibly found in the interior of the polytope. For α large enough, the boundary of $\mathbf{U}(\mathbf{a}, \mathbf{b})$ is inside of $\mathbf{U}_\alpha(\mathbf{a}, \mathbf{b})$, in which case the solution of the standard optimal transport problem over both sets coincides. However, if we use the entropy as a regularizer in Equation (16), we create a scenario in which the transportation plan can be modified to decrease the total cost either by moving inside the set $\mathbf{U}(\mathbf{a}, \mathbf{b})$ in the direction dictated by $\mathbf{C}$ or by increasing the plans entropy, moving it closer to $\mathbf{a}\mathbf{b}^\intercal$. To every $\epsilon \in [0, \infty]$, we can find a corresponding α given fixed $\mathbf{a}$, $\mathbf{b}$ and $\mathbf{C}$. The result is a path as in Figure 12 between $\mathbf{a}\mathbf{b}^\intercal$ and the optimal plan for the unregularized problem, where every point is a unique solution to the regularized problem depending on ϵ. The probabilistic interpretation of Equation (13) can thereby be extended for the regularized objective via

$$L_{\mathbf{C}}^{\epsilon}(\mathbf{a}, \mathbf{b}) = \min_{(X,Y)} \left\{ \mathbb{E}_{(X,Y)}[c(X,Y)] + \epsilon I(X;Y) : \; X \sim \alpha, Y \sim \beta \right\}. \tag{20}$$

The consequences of this regularization are extensive and have to be considered from the standpoint of theory and practice. First of

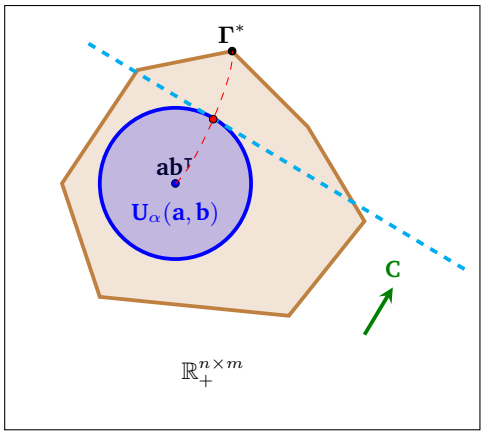


Figure 12: Regularization path resulting from adding an entropy term to the objective of optimal transport. The red point is the optimal solution of the optimal transport problem on $\mathbf{U}_\alpha(\mathbf{a}, \mathbf{b})$, resulting from the optimization direction given by $\mathbf{C}$. Every $\epsilon \in [0, \infty]$ in the objective (16) can be linked to an α, yielding a unique solution along the red line.

all, an essential result for the regularized objective is that it also satisfies the requirements to be considered a distance mathematically. For details, please refer to the proofs provided by [6]. The result is an approximation to the solution of the standard optimal transport problem.

In practice, the implications of a more spread out transportation are highly dependent on the application. While the example of assigning goods to factories may be limited by the economical benefit and the feasibility of distributing the goods among many smaller deliveries, other examples can benefit from this approach. Especially if the used cost is based on estimates it can be advantageous to introduce a notion of uncertainty into the plan by blurring it using the entropic regularization.

2.4. Sinkhorn's Algorithm

However, the main advantage of regularized optimal transport comes in the form of significantly reduced computational complexity. At first sight, the regularized objective is problematic, because solving via linear programming would require the function to be linear with respect to $\mathbf{\Gamma}$. Adding the entropy as in Equation (15) clearly contradicts this premise. Nonetheless, computing the Lagrangian of Equation (16) with dual variables $\boldsymbol{\upsilon} \in \mathbb{R}^n$ and $\boldsymbol{\varphi} \in \mathbb{R}^m$ yields

$$\mathcal{L}(\mathbf{\Gamma}, \boldsymbol{\upsilon}, \boldsymbol{\varphi}) = \sum_{i,j} \frac{1}{\epsilon} \mathbf{\Gamma}_{ij} \log\left(\mathbf{\Gamma}_{ij}\right) + \mathbf{\Gamma}_{ij}\mathbf{C}_{ij} + \boldsymbol{\upsilon}^{\mathrm{T}}\left(\mathbf{\Gamma}\mathbf{1}_n - \mathbf{a}\right) + \boldsymbol{\varphi}^{\mathrm{T}}\left(\mathbf{\Gamma}^{\mathrm{T}}\mathbf{1}_m - \mathbf{b}\right). \tag{21}$$

Element-wise differentiation reveals the general structure of a solution to the regularized optimal transport problem via

$$\frac{\partial \mathcal{L}}{\partial \mathbf{\Gamma}_{ij}} = \frac{1}{\epsilon} \log\left(\mathbf{\Gamma}_{ij}\right) + \frac{1}{\epsilon} + \mathbf{C}_{ij} + \boldsymbol{\upsilon}_i + \boldsymbol{\varphi}_j \overset{!}{=} 0 \tag{22}$$

$$\leftrightarrow \mathbf{\Gamma}_{ij} = e^{-\frac{1}{2} - \epsilon \boldsymbol{\upsilon}_i} e^{-\epsilon \mathbf{C}_{ij}} e^{-\frac{1}{2} - \epsilon \boldsymbol{\varphi}_j}. \tag{23}$$

Owing to the fact that any cost matrix is assumed to be computed based on a proper distance and thereby having only non-negative entries, the factor $e^{-\epsilon \mathbf{C}_{ij}}$ is always positive. With that in mind, the above Equation allows a connection to the following theorem.

Theorem 2.1 (Sinkhorn's Theorem, [10])**.** *If $\mathbf{K} \in \mathbb{C}^{n \times n}$ with strictly positive entries, there exist diagonal matrices $\mathbf{U}, \mathbf{V}$ with strictly positive diagonal entries, such that the product*

$$\mathbf{UKV}$$

is doubly-stochastic. The result is unique up to a scaling of $\mathbf{U}$ with χ and $\mathbf{V}$ with ψ, such that $\chi \cdot \psi = 1$.

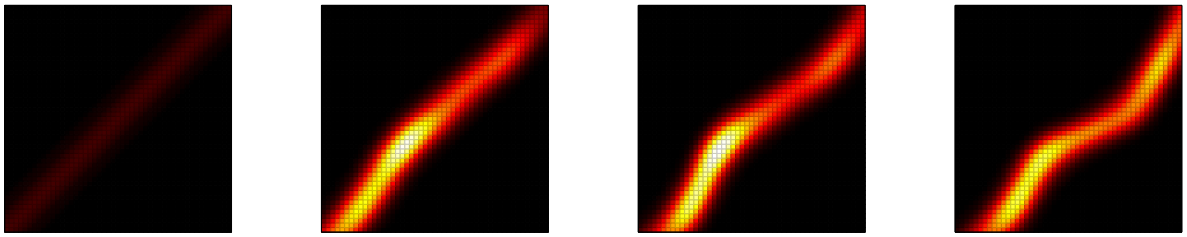
It is shown in [18, Rem. 4.8, 4.14] that if we can incorporate the constraints

$$\mathbf{UKv} = \mathbf{a}, \qquad \mathbf{VK}^{\mathsf{T}}\mathbf{u} = \mathbf{b}, \tag{24}$$

that are inherent to $\mathbf{U}(\mathbf{a}, \mathbf{b})$ into the computation, the solution of Equation (16) is equivalent to finding $\mathbf{u} = \mathrm{diag}(\mathbf{U})$ and $\mathbf{v} = \mathrm{diag}(\mathbf{V})$ to the matrix $\mathbf{K} = e^{-\epsilon \mathbf{C}}$ –the so-called *Gibbs Kernel*–. This can be achieved by alternatively iterating

$$\mathbf{u}^{(k+1)} = \frac{\mathbf{a}}{\mathbf{K}\mathbf{v}^{(k)}}, \qquad \mathbf{v}^{(k+1)} = \frac{\mathbf{b}}{\mathbf{K}\mathbf{u}^{(k+1)}}, \tag{25}$$

with element-wise division. The evolution of these updates is shown in Figure 13, where we stopped the solver with the example of Figure 9 at various iterations. In most cases, the algorithm iterates until a marginal constraint violation criterion is met. An alternative



(a) Iteration 0. (b) Iteration 1. (c) Iteration 5. (d) Iteration 25.

Figure 13: Sinkhorn's algorithm at different iterations. Starting from the Gibbs kernel $\mathbf{K} = e^{-\epsilon \mathbf{C}}$ (left), the algorithm alternates between scaling each row/column of the transportation plan to fit the distribution $\mathbf{a}/\mathbf{b}$.

perspective on this process can be obtained by defining the set $\mathbf{U}(\mathbf{a}, \mathbf{b})$ as the union $\mathcal{C}_1 \cap \mathcal{C}_2$ with

$$\mathcal{C}_1 = \{\mathbf{\Gamma} \colon \mathbf{\Gamma}\mathbf{1}_m = \mathbf{a}\}, \qquad \mathcal{C}_2 = \{\mathbf{\Gamma} \colon \mathbf{\Gamma}^{\mathsf{T}}\mathbf{1}_n = \mathbf{b}\}. \tag{26}$$

Each iterate can be seen as a certain kind of projection (see Appendix B), alternatively fitting the plan to the "closest" one in $\mathcal{C}_1/\mathcal{C}_2$ w.r.t. the Kullback-Leibler divergence, which converges to the union $\mathbf{U}(\mathbf{a}, \mathbf{b})$. An illustration is provided in Figure 14.

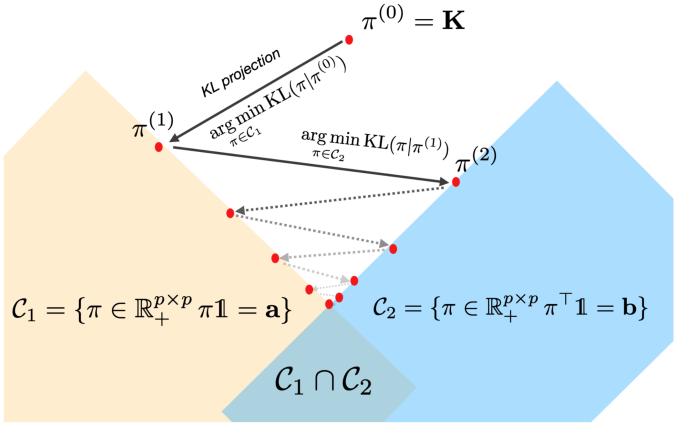


Figure 14: Sinkhorn's algorithm as a sequence of projections. This illustration was taken from [11], where the transportation plan is denoted π.

Convergence and Complexity

Computation-wise, the advantage of the iterations in Equation (25) is that they entirely consist of vector and matrix-vector operations, which are particularly well-suited for computation on e.g. GPU's. The complexity of the overall algorithm is reduced from $\mathcal{O}(n^3 \log(n))$ for the most efficient implementations of the standard problem down to $\mathcal{O}(n^2 \log(n) \tau^{-3})$, where the factor τ depends on ϵ [18, Rem. 4.6]. Increasing this parameter lowers the complexity at the expense of a worse approximation to the optimal plan of the unregularized case. Regarding convergence, the authors of [1] show that the Sinkhorn iterations exhibit linear convergence, which degrades if ϵ becomes too small. In practice, this can lead to significant speed ups in computation. The effect increases with ϵ and could already be observed when computing the examples for this report. However, the division by possibly smaller and smaller values in (25) can often lead to overflows. In these cases, increasing the parameter ϵ can help, because denser plans do not require the Gibbs kernel to be rescaled as drastically to concentrate the probability mass. The resulting plans are more blurred and a less accurate approximation of the unregularized problem, which can potentially limit the applicability of this strategy. In these cases, computations in the log-domain can be taken into account [18, Rem. 4.23].

2.5. Kernel Density Estimation

The preceding sections assumed that both source and target distribution are given via the histograms **a** and **b** as in Equation (9). However, in application scenarios one is often limited to samples collected in a dataset $\mathcal{D}$, which in Equation (9) corresponds to knowing e.g. δ_{x_i} without knowing the weights $\mathbf{a}_i$. To overcome this, one needs to estimate the density of the distribution underlying the empirical samples.

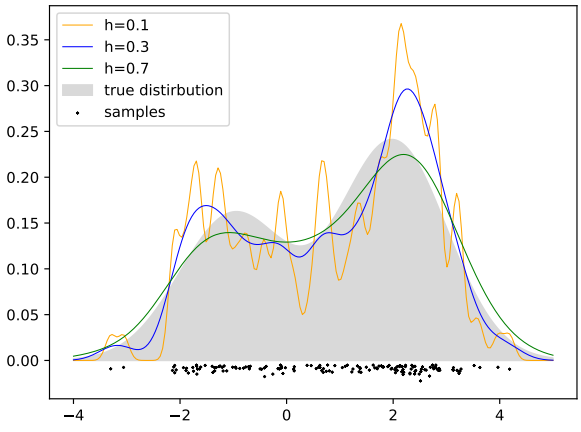


Figure 15: Exemplary kernel density estimation using Gaussian kernel and varying bandwidth parameter h.

A popular non-parametric method for estimating these densities is Kernel Density Estimation (KDE) [13, Sec. 14.7.2]. Given a dataset $\mathcal{D}$ with samples $\{x_i\}_{i=1}^N \in \mathcal{X}^N$, KDE estimates the density via

$$\hat{p}(x|\mathcal{D}) = \frac{1}{N} \sum_{i=1}^{N} K_h(d_{\mathcal{X}}(x, x_i)), \tag{27}$$

where $d_{\mathcal{X}}$ denotes a metric on the space $\mathcal{X}$. K_h is a non-negative function called kernel with a smoothing parameter $h > 0$. In the following, the authors of [4] employ the Gaussian kernel

$$K_h(d_{\mathcal{X}}(x, x_i)) = \frac{1}{Z_h} \exp(-\frac{d_{\mathcal{X}}(x, x_i)^2}{2h^2\sigma^2}) \tag{28}$$

with normalization parameter Z_h in all applications. The influence of the parameter h can be retraced in Figure 15: A higher h leads to more global averaging, meaning that every sample has influence on a larger neighborhood in the estimation. KDE can also be leveraged to estimate the joint distribution of two random variables (X, Y). Based on a dataset $\mathcal{D}' = \{x_i, y_i\}_{i=1}^N \in (\mathcal{X}^N, \mathcal{Y}^N)$ of paired samples, the joint distribution is estimated via

$$\hat{p}(x, y|\mathcal{D}') = \frac{1}{N} \sum_{i=1}^{N} K_{h_1}(d_{\mathcal{X}}(x, x_i)) K_{h_2}(d_{\mathcal{Y}}(y, y_i)). \tag{29}$$

3. Annotated InfoOT

In the following chapter, the main paper by Chuang, Jegelka and Alvarez-Melis [4] for this semester's part of the project is studied in detail. We draw inspiration from the blog post "The Annotated Transformer" [24] and try to relate the given mathematical expressions to the provided code and give intuitive explanations based on the preliminary work of the first chapter. Note that in the code, the transportation plan is denoted P instead of $\mathbf{\Gamma}$.

3.1. Measuring global Structure with Mutual Information

As mentioned before, the optimal transport framework treats every element ot the support individually, considering only the information of the used metric. Therefore, coherence structures of the data such as clusters ,that could otherwise be leveraged, are ignored. Additionally, obtaining a metric for the cost matrix requires a notion of distance between the source and target samples, which can be difficult to derive and / or compute.
To overcome these issues, the authors of [4] propose a new method utilizing ideas from information theory, seeking a transportation plan that maximizes the mutual information between the source and target distribution given unpaired samples. We know from Section 2.1 that computing the mutual information between two random variables requires the marginals and the joint distribution. Since the method assumes that only samples are given, we need to estimate all these distributions. Regarding the marginals, the Kernel Density estimation presented in Section 2.5 can be used to achieve this.

```
def compute_kernel(Cx, Cy, h):
    '''
    compute Gaussian kernel matrices
    Parameters
    ----------
    Cx: source pairwise distance matrix
    Cy: target pairwise distance matrix
    h : bandwidth
    Returns
    ----------
    Kx: source kernel
    Ky: targer kernel
    '''
    std1 = np.sqrt((Cx**2).mean() / 2)
    std2 = np.sqrt((Cy**2).mean() / 2)
    h1 = h * std1
    h2 = h * std2
    # Gaussian kernel (without normalization)
    Kx = np.exp(-(Cx / h1)**2 / 2)
    Ky = np.exp(-(Cy / h2)**2 / 2)
    return Kx, Ky
```

The provided implementation uses Gaussian kernel matrices, whose entries $(\mathbf{K}_X)_{ij} = K_h(d_{\mathcal{X}}(x_i - x_j))$ and $(\mathbf{K}_Y)_{ij} = K_h(d_{\mathcal{Y}}(y_i - y_j))$ correspond to elements of the sum in Equation (27). The joint distribution on the other hand cannot be estimated in this straightforward manner, because we want to consider unpaired samples. It is linked to the desired transportation plan via Equation (13), but directly using the plan as an estimate while also optimizing the whole expression with regard to the mutual information will yield the maximally achievable value for any arbitrary one-to-one mapping. Instead, the authors propose to use the entries of the transportation plan as weights for all possible sample pairs, yielding the kernelized joint density

$$\hat{f}_{\mathbf{\Gamma}}(x_i, y_j) = \sum_{i,j} \mathbf{\Gamma}_{ij} K_h(d_{\mathcal{X}}(x, x_i)) K_h(d_{\mathcal{Y}}(y, y_j)). \tag{30}$$

```
    f_x = Kx.sum(1) / Kx.shape[1]
    f_y = Ky.sum(1) / Ky.shape[1]

    f_x_f_y = np.outer(f_x, f_y)
    constC = np.zeros((len(Kx), len(Ky)))
    f_xy = -ot.gromov.tensor_product(constC, Kx, Ky, P)
```

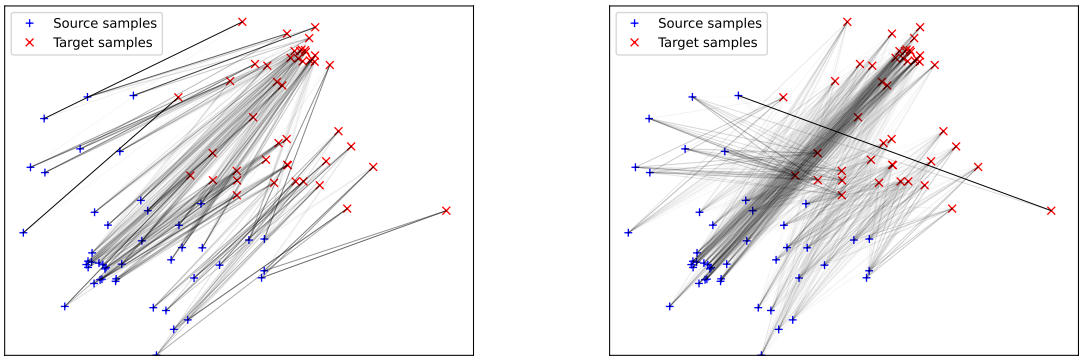
Computation-wise, the kernel matrices used in the KDE of the marginals can be combined with the transportation plan using a tensor product proposed in [19]. The resulting *Kernelized Mutual Information* is defined via

$$\hat{\mathbf{I}}_{\mathbf{\Gamma}}(X,Y) \coloneqq \sum_{i,j} \mathbf{\Gamma}_{ij} \log\Big(\frac{\hat{f}_{\mathbf{\Gamma}}(x_i,y_j)}{\hat{f}(y_j)\hat{f}(x_i)}\Big) = \sum_{i,j} \mathbf{\Gamma}_{ij} \log\left(\frac{nm \sum_{k,l} \mathbf{\Gamma}_{kl} K_h(d_{\mathcal{X}}(x_i,x_k)) K_h(d_{\mathcal{Y}}(y_j,y_l))}{\sum_k K_h(d_{\mathcal{X}}(x_i,x_k)) \cdot \sum_l K_h(d_{\mathcal{Y}}(y_j,y_l))}\right). \tag{31}$$

An intuition for this expression is that each local transportation is now weighted by how much information it provides about the underlying marginals. Similar to the formulation in Equation (8), this information content is quantified by how much additional structure is present in the joint distribution. Here, this influence on the global structure is incorporated by using the transportation plan entries as weights for each summand of the KDE. Maximizing this quantity via the new objective

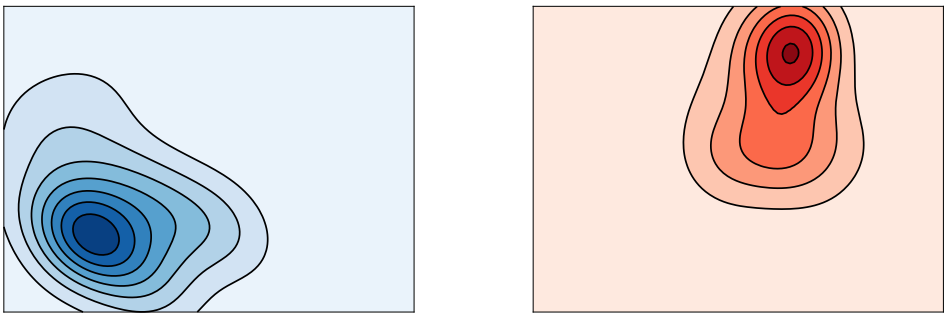
$$\max_{\Gamma \in \mathbf{U}(\mathbf{a},\mathbf{b})} \hat{\mathbf{I}}_{\mathbf{\Gamma}}(X,Y) = \min_{\mathbf{\Gamma} \in \mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j} \mathbf{\Gamma}_{ij} \log\Big(\frac{\hat{f}(y_i)\hat{f}(x_i)}{\hat{f}_{\mathbf{\Gamma}}(x_i,y_i)}\Big), \tag{32}$$

yields a method to generate transportation plans based on a different approach than that of Section 2.2, which is denoted ***Info**rmation-theoretic **O**ptimal **T**ransport* (InfoOT). The behavior of this new approach becomes clearer when observing Figure 16, where we use



(a) Standard OT. (b) InfoOT.

Figure 16: Comparison of 2-D results for regularized optimal transport and InfoOT, both using $\epsilon = 0.25$. InfoOT uses a bandwidth of $h = 0.7$.



(a) Source sample density. (b) Target sample density.

Figure 17: Kernel Density Estimation of the samples in Figure 16 using the same bandwidth of $h = 0.7$ as InfoOT.

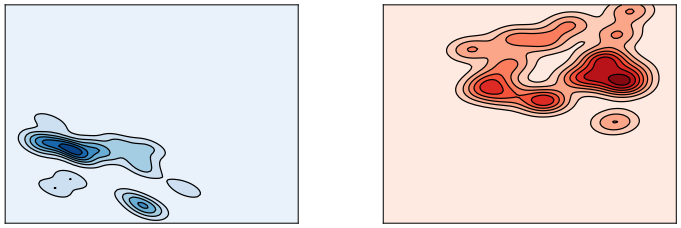
2-D distributions to compare regularized optimal transport with the squared euclidean metric to InfoOT. The objective in Equation 32 yields higher values, if for a pair of samples (i, j) with high transportation probability Γ_{ij}, pairs (k, l) that are "close-by" ($\to d_{\mathcal{X}}, \to d_{\mathcal{Y}}$

small) in the source and target distribution are also transported to one another with a high probability Γ_{kl}. The resulting transportation plan therefore enforces a mapping between samples in high-density regions. Figure 17 shows the densities of source and target samples resulting from a KDE using the same bandwidth as InfoOT. Comparing the plot to the transportation plans from before, we can confirm that InfoOT creates such a density-based mapping, while the standard optimal transport assigned most of the probability mass of source samples in the high-density region to more close-by target samples in order to reduce the quadratic transportation cost.

Hyperparameters

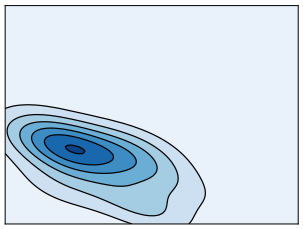
As with most machine learning algorithms, hyperparameter-tuning is an important task when solving problems with InfoOT. The provided algorithm uses a variant of the Sinkhorn algorithm introduced in Section 2.4 (more on that later), which is why the parameter ϵ controlling the entropic regularization has the same implications as with the standard regularized optimal transport problem: The higher ϵ, the more the algorithm distributes the transportation, leading to less sparse plans. As we used $\epsilon = 0.25$ to create the plots in Figure 16, the effect is also visible there. Apart from that, we saw that InfoOT internally performs Kernel Density Estimation based on Gaussian Kernels for the given samples to assess the underlying densities. Consequently, the bandwidth parameter h we introduced in Section 2.5 considerably influences the resulting transportation plan, because it heavily impacts of the estimated distribution this mapping is based on.

Figure 18 shows that a lower bandwidth results in a more complex distribution with a higher number of clusters / high-density

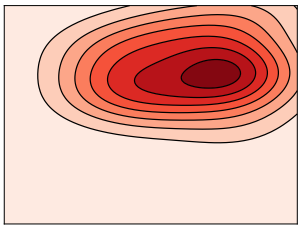


(a) Source $h = 0.3$.

(b) Target $h = 0.3$.



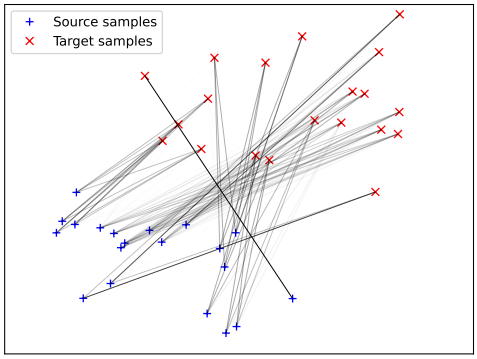
(c) Source $h = 0.8$.



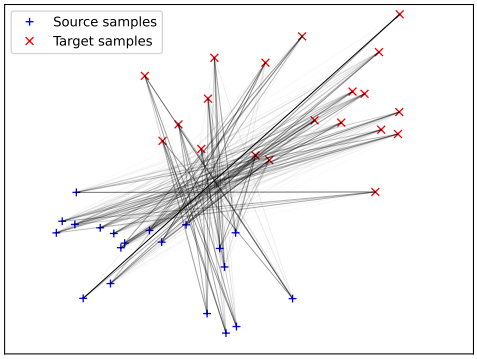
(d) Target $h = 0.8$.

Figure 18: Kernel Density Estimation for exemplary 2-D distribution with varying bandwidth parameter h.

regions. Just as with the 1-D example in Figure 15, this is the consequence of h scaling down the Gaussian kernel functions' variance, which results in a more local influence of each sample on the estimated distribution. Implications of this on the transportation plan can be observed in Figure 19. While the regions with the highest density are mapped to each other in both cases, the transport for other samples shows significant deviations, because the lower bandwidth $h = 0.3$ implies a different cluster structure.



(a) $h = 0.3$ and $\epsilon = 0.6$.



(b) $h = 0.8$ and $\epsilon = 0.1$

Figure 19: Transportation plans computed using InfoOT with different hyperparameters. Although the same samples are used, the mappings deviate significantly due to the bandwidth of the internal KDE controlling what is considered as a cluster.

Implications

To summarize, InfoOT consists of a new optimal transport objective which yields transportation plans that map unpaired source and target samples based on their respective densities. Consequently, probability mass from clusters in the source distribution is predominantly transported to clusters of the target. The most significant advantage of this approach consists of the fact that the metric spaces of source and target don't have to be comparable, since the densities are estimated via KDE purely based on intra-domain distances. This allows for settings with different geometries and / or dimensions for these spaces. While this opens up a lot of possibilities, it is important to note that if both distributions lie in the same metric space, InfoOT on its own does not allow to consider any notion of transportation cost.

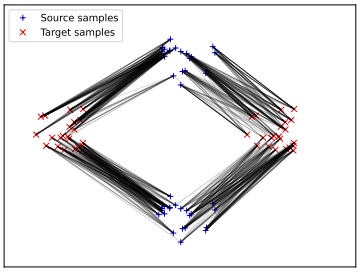
3.2. Incorporating Domain Information

However, in many cases the cost matrix encodes important information about the underlying domain and / or its geometry. For a simple intuition, consider e.g. assigning wood from various logging camps to lumber mills. With InfoOT, we are able to compute a mapping that assigns wood from areas with a high concentration of logging camps to areas with many lumber mills, e.g. cities, while more isolated camps are assigned to areas with sparsely scattered mills. However, this mapping would completely ignore information about the transportation distances, the available infrastructure and / or natural barriers that influence the transport, such as rivers or mountains.

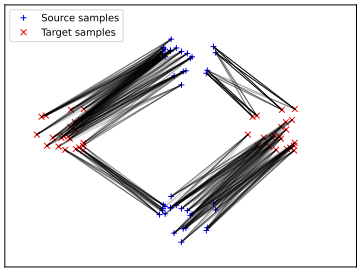
In order to combine the advantages mentioned in the previous section while also being able to take the geometry information into account, the authors of [4] propose to use the InfoOT objective in Equation (31) as a regularizer for the original discrete optimal transport objective in Equation (12). Mathematically, this leads to the Fused InfoOT objective

$$\min_{\mathbf{\Gamma}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \left[\langle \mathbf{\Gamma}, \mathbf{C}\rangle - \lambda \hat{\mathbf{I}}_{\mathbf{\Gamma}}(X,Y)\right] = \min_{\mathbf{\Gamma}\in\mathbf{U}(\mathbf{a},\mathbf{b})} \sum_{i,j} \mathbf{\Gamma}_{ij}\Big(C_{ij} + \lambda \log\Big(\frac{\hat{f}(x_i)\hat{f}(y_j)}{\hat{f}_{\mathbf{\Gamma}}(x_i,y_j)}\Big)\Big), \tag{33}$$

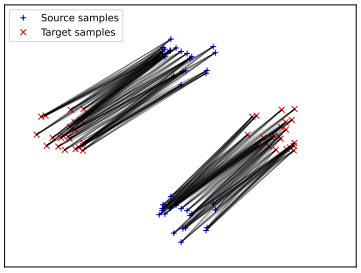
where the parameter $\lambda \in \mathbb{R}_+$ controls the influence of the mutual information term. The higher λ, the more $\mathbf{\Gamma}$ tries to preserve cluster-structures, with less weight given to the fact that preserving these structures through the mapping may cause high total transportation cost. The effect is depicted in Figure 20: In 20a a λ of 0 leads to the fact that every sample is again treated individually, creating the alignment solely with the purpose of minimizing the total cost. With 20b, it becomes more attractive for the algorithm to use the InfoOT regularizer to reduce the cost by preserving clusters, an effect that then dominates the solution when increasing λ to 7 in 20c.



(a) $\lambda = 0.0$.



(b) $\lambda = 0.95$.



(c) $\lambda = 7.0$.

Figure 20: Transportation plans computed using Fused InfoOT with different values for λ. All examples use $h = 0.6$ and $\epsilon = 0.1$.

3.3. Solving via Projected Gradient Descent

Both of the newly introduced objectives in Equation (32) and Equation (33) are nonlinear in $\mathbf{\Gamma}$, because the Kernelized Mutual Information of Equation 31 inside the logarithm depends on it. In the same way as the regularized optimal transport objective, solving via linear programming like in Section 2.2 is not an option any more, because this method requires all constraints and the objective itself to be linear.

```
def migrad(P, Kx, Ky):
    '''
    compute the gradient w.r.t. KDE mutual information
    Parameters
    ----------
    P : transportation plan
    Ks: source kernel matrix
    Kt: target kernel matrix

    Returns
    ----------
    negative gradient w.r.t. MI
    '''
    f_x = Kx.sum(1) / Kx.shape[1]
    f_y = Ky.sum(1) / Ky.shape[1]
    f_x_f_y = np.outer(f_x, f_y)
    constC = np.zeros((len(Kx), len(Ky)))
    # there's a negative sign in ot.gromov.tensor_product
    f_xy = -ot.gromov.tensor_product(constC, Kx, Ky, P)
    P_f_xy = P / f_xy
    P_grad = -ot.gromov.tensor_product(constC, Kx, Ky, P_f_xy)
    P_grad = np.log(f_xy / f_x_f_y) + P_grad
    return -P_grad
```

```
    def solve(self, numIter=50, verbose='True'):
        '''
        solve projected gradient descent via sinkhorn iteration
        '''
        p = np.zeros(len(self.Xs)) + 1. / len(self.Xs)
        q = np.zeros(len(self.Xt)) + 1. / len(self.Xt)
        P = np.outer(p, q)
        if verbose:
            print('solve projected gradient descent...')
            for i in tqdm(range(numIter)):
                grad_P = migrad(P, self.Ks, self.Kt)
```

```
            P = ot.bregman.sinkhorn(p, q, self.C + self.lam * grad_P, reg=self.reg)
    else:
        for i in range(numIter):
            grad_P = migrad(P, self.Ks, self.Kt)
            P = ot.bregman.sinkhorn(p, q, self.C + self.lam * grad_P, reg=self.reg)
    self.P = P
    return P
```

Instead, an algorithm based on the mirror descent framework is used. A short conceptual introduction is provided in Appendix B. When applied to the Fused InfoOT, the update for the transportation plan reads

$$\Gamma_{i+1} \leftarrow \arg\min_{\Gamma \in \mathbf{U}(\mathbf{p},\mathbf{q})} \langle \Gamma, \mathbf{C} - \lambda \nabla_\Gamma \hat{\mathbf{I}}_{\Gamma_i}(X,Y) \rangle - \epsilon H(\Gamma), \tag{34}$$

which is solved by using Sinkhorn's algorithm at every iteration. For InfoOT, set $\mathbf{C} = 0$ and $\lambda = 1$. The gradient of the InfoOT objective with respect to the transportation plan can be computed analytically, yielding

$$\frac{\partial \hat{I}_\Gamma(X,Y)}{\partial \mathbf{\Gamma}_{ij}} = \log\Big(\frac{\hat{f}_\Gamma(x_i,y_j)}{\hat{f}(y_j)\hat{f}(x_i)}\Big) + \sum_{k,l} \mathbf{\Gamma}_{kl} \frac{K_h(d_\mathcal{X}(x_i,x_k))K_h(d_\mathcal{Y}(y_j,y_l))}{\hat{f}_\Gamma(x_k,y_l)}. \tag{35}$$

Using element-wise division $\oslash$, this term can be computed using the kernel matrices $\mathbf{K}_X, \mathbf{K}_Y$ and the (estimated) marginal density vectors $\mathbf{M}_X, \mathbf{M}_Y$ of the samples via

$$\nabla_\Gamma \hat{I}(X;Y) = \log(\mathbf{K}_X \mathbf{\Gamma} \mathbf{K}_Y^\intercal \oslash \mathbf{M}_X \mathbf{M}_Y^\intercal) + \mathbf{K}_X(\mathbf{\Gamma} \oslash \mathbf{K}_X \mathbf{\Gamma} \mathbf{K}_Y^\intercal)\mathbf{K}_Y^\intercal. \tag{36}$$

Hyperparameters

Apart from the effect on the resulting alignments, hyperparameter tuning also greatly influences the convergence speed. In accordance to Section 2.2, increasing ϵ leads to faster convergence for both algorithms. Apart from that, the possibility of encountering overflows is reduced significantly, but the resulting plans can be overly blurred, which can negatively affect interpretability in downstream tasks. Regarding the bandwidth of the KDE used in (Fused) InfoOT, values lower than approx. 0.5 are problematic in that the resulting complex densities negatively affect convergence speeds in general and can even prevent the Sinkhorn iterations from converging. If lower bandwidths are required, the only way to work against these issues is to increase ϵ, which can again yield overly blurred plans. Apart from h and ϵ, the weighting parameter λ of Fused InfoOT is the most difficult to tune. If neither the regularization of the standard optimal transport objective nor the kernelized mutual information regularization clearly dominates the objective (very high or low λ), convergence can be difficult to achieve. For example, even small changes in the parameters used to create Figure 20b lead to these problems, even though the setting is minimalistic.

3.4. The Conditional Projection

With a transportation plan, we compute an alignment for the given source- and target distribution. As the Kantorovich formulation of optimal transport introduced in Section 2.2 allows for mass splitting, which is additionally emphasized when using entropic regularization, this alignment is in itself not suitable to be used as a deterministic mapping of specific samples. A very simple idea is the *barycentric projection*

$$x_i \to \arg\min_{y \in \mathcal{Y}} \sum_{j=1}^m \mathbf{\Gamma}_{ij} ||y - y_j||^2 = \frac{1}{\sum_{j=1}^m \mathbf{\Gamma}_{ij}} \sum_{j=1}^m \mathbf{\Gamma}_{ij} y_j, \tag{37}$$

which uses the transportation weights of $\mathbf{\Gamma}$ that result from the mass splitting as weights for a weighted average [17]. It thereby maps source samples $x_i \in \mathcal{X}$ to the target domain $\mathcal{Y}$ by minimizing the weighted cost to the m target samples that receive fractions of the probability mass at x_i when transporting according to $\mathbf{\Gamma}$. While easy to compute, this approach has key limitations:

(1) **Imbalanced Data and Outliers**
If the source and target distribution exhibit clusters consisting of a strongly varying number of samples, the mass constraint enforced via the KDE will lead to wrong mappings. Also, since the barycentric projection uses a weighted average, even very small values in the plan that would result from the sample being an outlier and therefore having a very low density / probability mass can greatly influence the projection of that sample.

(2) **New Samples**
Because the conditional projection relies on the data in the transportation plan as weights, it can only be used to map the exact samples that were used to compute this plan. In Equation (37), this relates to the index i in both sums. In applications, it is easy to think about settings where it would be advantageous to be able to extend this ability to unseen samples.

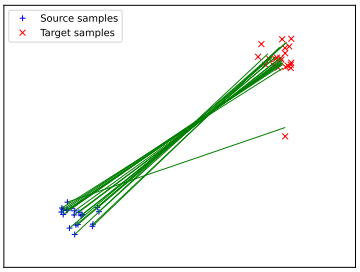


Figure 21: Barycentric projections based on a transportation plan computed via InfoOT. Because the weighted average is based on the transportation of a single sample, the outlier leads to a false alignment.

To overcome this, the authors of [4] propose a probabilistic approach that again leverages the joint density estimated using the kernelized mutual information. As we already discussed with Equation (13), a transportation plan based on the Kantorovich formulation allows a probabilistic interpretation as a joint probability mass function. In this light, the weighted average in Equation (37) is interpretable as taking an expectation, in this case the conditional expectation $\mathbb{E}[Y|X = x_i]$ given a source sample x_i

$$x_i \to \frac{1}{\sum_{j=1}^{m} \mathbf{\Gamma}_{ij}} \sum_{j=1}^{m} \mathbf{\Gamma}_{ij} y_j = \mathbb{E}_{P_{Y|X=x}}[y]. \tag{38}$$

The idea is to generalize this formulation using importance sampling via

$$x \to \mathbb{E}_{P_{Y|X=x}}[y] = \int_{\mathcal{Y}} f_{Y|X=x}(y) y \mathrm{d}y = \int_{\mathcal{Y}} f_{Y|X=x}(y) \frac{f_Y(y)}{f_Y(y)} y \mathrm{d}y \tag{39}$$

$$= \mathbb{E}_{y \sim P_Y}\left[\frac{f_{Y|X=x}(y)}{f_Y(y)} y\right] = \mathbb{E}_{y \sim P_Y}\left[\frac{f_{YX}(x,y)}{f_Y(y) f_X(x)} y\right], \tag{40}$$

where, because we are now using a joint probability instead of the conditional one, we are no longer limited to the samples used to compute the transportation plan. If –as in our case– the setting is limited to unpaired samples from source and target distribution, we do not have access to the joint distribution e.g. by KDE estimation, but it can be observed that the importance weight in the formulation above exactly corresponds to the ratio used when computing the mutual information.

```
            if method == 'conditional':
                _Cs = pairwise_distances(X, self.Xs)
                _Ct = pairwise_distances(self.Xt, self.Xt)
                _Ks, _Kt = compute_kernel(_Cs, _Ct, h)

                P = ratio(self.P, _Ks, _Kt)
                return projection(P, self.Xt)
```

```
def projection(P, X):
    '''
    compute the projection based on similarity matrix
    Parameters
    ----------
    P : transportation plan or similarity matrix
    X : target data

    Returns
    ----------
    projected source data
    '''
    weights = np.sum(P, axis = 1)
    X_proj = np.matmul(P, X) / weights[:, None]
    return X_proj
```

This means we can use the same estimators introduced in Section 3.1 for the InfoOT objective to compute

$$\mathbb{E}_{y\sim P_Y}\left[\frac{f_{YX}(x,y)}{f_Y(y)f_X(x)}y\right] \approx \frac{1}{\sum_{j=1}^{m}\hat{f}_{\mathbf{\Gamma}}(x,y_j)/(\hat{f}_X(x)\hat{f}_Y(y_j))}\frac{\hat{f}_{\mathbf{\Gamma}}(x,y_j)}{\hat{f}_X(x)\hat{f}_Y(y_j)}y_j, \tag{41}$$

which is referred to as the *conditional projection*. Apart from being able to extend to new samples, this projection is more robust against outliers in the data, because it is based on the kernelized mutual information that is computed using the InfoOT or Fused InfoOT objective. This way, the projection considers global information about how close-by points are mapped, which can work against wrong alignments if outliers exist in the target dataset.

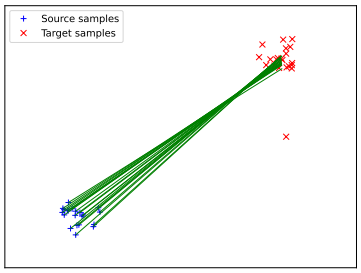


Figure 22: Conditional projections based on a transportation plan computed via InfoOT. In contrast to the barycentric projection in Figure 21 the outlier does not lead to false projections.

Color Matching

Visually appealing examples can be generated by using the conditional projection to map colors of one picture to another based on samples from both images. We randomly sample the RGB values of 500 pixels from both images shown in Figure 23 to create two 500×3 matrices. These are then used as source and target samples to compute a transportation plan using the InfoOT algorithm. This plan is then used to map the RGB values of the source in Figure 23a to every pixel of the target in Figure 23b. As we learned in

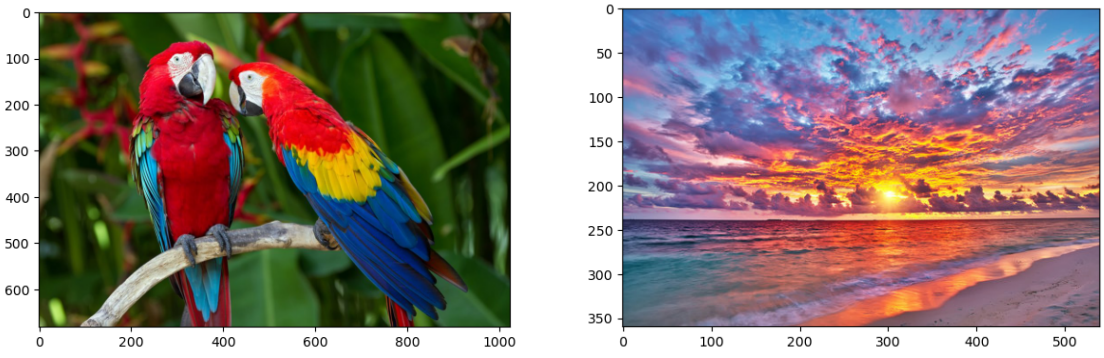


Figure 23: Images for the color matching example, where we want ot find a matching between the color distributions of the given images using optimal transport.

Section 3.1, this happens based on the densities that are estimated using the samples. This effect can be retraced with the results in Figure 24, where the shades of purple that are most dominant in Figure 23b are mapped to shades of green, which make up most of the pixels in Figure 23a. However, comparing Figures 24a and 24b, we observe that the mapping is not consistent over multiple runs. The reason for this behavior is that we used only 500 randomly selected samples from images with 194400 and 698368 pixels, respectively. The resulting estimate of the densities in RGB space can therefore vary widely between runs and can be very different to the true distribution that would result from incorporating every single pixel. To work against this effect , we would have to increase the number of samples dramatically, up to dimensions that wouldn't be computationally feasible in most applications. Apart from that, the parameter h has to be chosen a lot smaller than in the examples before, because too much averaging of the color samples leads to a reduced variety of colors, which can cause the whole image showing only shades of brown.

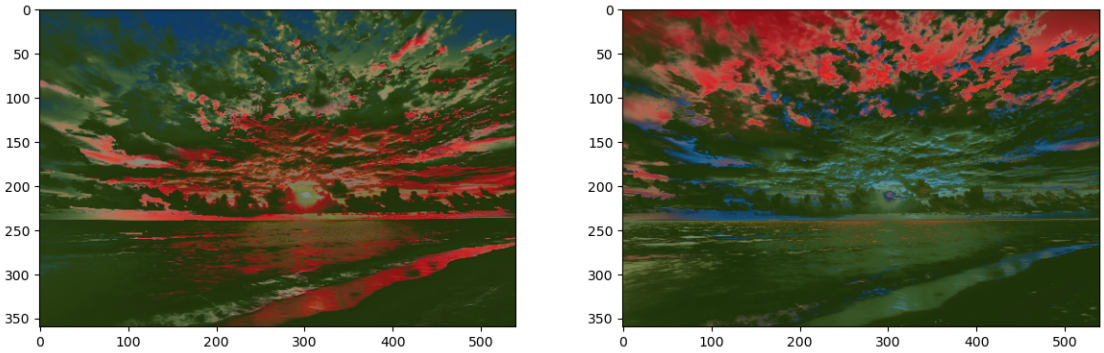


Figure 24: Results for two separate runs of the color matching example using 500 samples from source and target distributions and parameters $h = 0.3$, $\epsilon = 0.3$ and $\lambda = 0.3$. The resulting transportation plan was then used to project each pixel of the image in Figure 23b to a color of Figure 23a using the conditional projection.

4. Conclusions

4.1. Discussion

We saw that InfoOT yields an alternative method to compute transportation plans that map high density regions between arbitrary metric spaces. While this can be very powerful if source and target dataset are not defined on the same space, the usefulness of the resulting transportation plan is highly problem-dependent. While InfoOT solely relies on the described density-driven approach and therefore is not able to incorporate other domain information, Fused InfoOT provides a method to do so via a cost matrix.
Both algorithms are solved as described in Section 3.3 via Sinkhorn's algorithm applied at each iteration, which yielded reasonable computation most of the time. However, the convergence speed proved to be highly sensitive to hyperparameter tuning. Even for the toy examples we provided in this report small changes in one of the parameters h, ϵ and λ could yield significantly slower convergence or lead to the overflow problems mentioned in Section 2.4. For us, this became most noticeable when using the Fused InfoOT objective in combination with a value for λ that did not strongly favor the regularization nor the cost in objective (33), e.g. the example in Figure 20b. In addition to convergence, the hyperparameters could also heavily influence the resulting alignments, the most notably being the KDE bandwidth h. The reason for this is that the bandwidth has a profound effect on the smoothness of the estimated distribution, which determines what is deemed as a cluster.
Another implication of this is encountered with the conditional projection we used in the color matching example. While for most cases a higher KDE bandwidth between 0.6 and 0.9 proved to be the best option regarding performance and results, it was problematic in this application because it negatively affected color variety. To counteract the poor convergence resulting from the low bandwidth required to work against this, we had to use a relatively high ϵ of 0.3. Therefore, the question of how to reliably set these hyperparameters in a possible future application remains a concern.
If the provided code is to be used in applications, several improvements are necessary. Firstly, the implementation uses a pre-defined number of iterations without a stopping criterion. For the figures in this report, we tested multiple values and compared the resulting transportation plans to determine when the changes between iterations become negligible. Additionally, the authors limit themselves to the euclidean metric to compute distances, while other metrics could prove beneficial.
Apart from what we observed, one OpenReview critic [15] of the paper mentions that for high dimensional settings the KDE can break down and suffer from the curse of dimensionality. As applications will more likely than not feature higher dimensions than the examples we tested, this is something to keep in mind.
Interestingly, the authors themselves mention in a response to another review that the InfoOT framework can be interpreted as a special case of the weak optimal transport formulation [8]. Should the problems of (Fused) InfoOT prevail even if the behavior proves advantageous, further research into the direction of the generalization is an interesting possibility.
It remains to be seen whether these methods have an advantage over the standard regularized optimal transport objective for the applications we encounter during the second part of the project. With that in mind, we also shortly discussed numerous other optimal transport-based approaches that we came across during our research, e.g. [12], [14] or [21], which could prove to be useful.

4.2. Outlook

Apart from the discussed possible drawbacks, (Fused) InfoOT and optimal transport in general provide a very versatile mathematical framework for the second part of this project. In particular, the following three aspects can potentially be leverage in various

applications.

(1) **Sample Space**
In this report, we investigated examples with 1-D, 2-D and 3-D (RGB) sample spaces. However, as long as we can provide a metric to compute distances between the samples, an arbitrary number of dimension is possible.

(2) **Geometry**
In Section 2.2, we learned that the metric for computing the cost matrix for the classic optimal transport problem has a profound effect on the resulting plan. While InfoOT only works with intra-domain distances and therefore does not require source and target to live in the same metric space, it is entirely based on the notion of individual densities in these separate spaces. Therefore, every framework we discussed in this report depends on a metric, either between source and target samples (optimal transport), among the samples of one set themselves (InfoOT) or both (Fused InfoOT). For example, we could consider a metric based on a spherical instead of euclidean geometry to obtain densities that are more suitable when considering transportation on a planet's surface.

(3) **Cost Matrix**
If prior domain knowledge is available or no assumptions on the geometry can be made, handcrafted cost matrices can be used to enforce a desired behavior. Continuing the example, we could use the spherical metric to compute the densities for the kernelized mutual information and then combine this with a cost matrix encoding possible transportation routes or barriers on the planet.

A. Linear Programming Example

A better intuitive understanding of the linear programming insights used in Section 2.2 can be gathered by considering a minimalist example in 2D. In this case, our transportation plan is denoted $\boldsymbol{\gamma}$ and has only 2 entries x and y. The cost matrix is a 2D-vector $\mathbf{c}$, which yields the linear program

$$\begin{aligned} \min_{\mathbf{c}} (\mathbf{c}^\intercal \cdot \boldsymbol{\gamma}) \\ \mathbf{A}\boldsymbol{\gamma} \leq \mathbf{b}, \end{aligned}$$

where each row of the matrix-vector inequality encodes one constraint. As an example, the edge marked blue in Figure 25 would correspond to the entry

$$\begin{pmatrix} \vdots & \vdots \\ -1 & -4 \\ \vdots & \vdots \end{pmatrix} \cdot \begin{pmatrix} x \\ y \end{pmatrix} \leq \begin{pmatrix} \vdots \\ 0.25 \\ \vdots \end{pmatrix}.$$

If the cost matrix is e.g. $\mathbf{c} = (1.5\ 2.5)$, then the total cost can be computed via the dot product in the equation above. Extracting all plans that have a common total cost η can be achieved by rearranging the equation, resulting in

$$y = \frac{\eta - 1.5x}{2.5},$$

which is depicted as a red line for $\eta = 8.5$. The direction in which to move this line closer to the required solution is given by the negative gradient $-\nabla(\mathbf{c}^\intercal \cdot \boldsymbol{\gamma}) = (-1.5\ -2.5)^\intercal$ displayed in green.

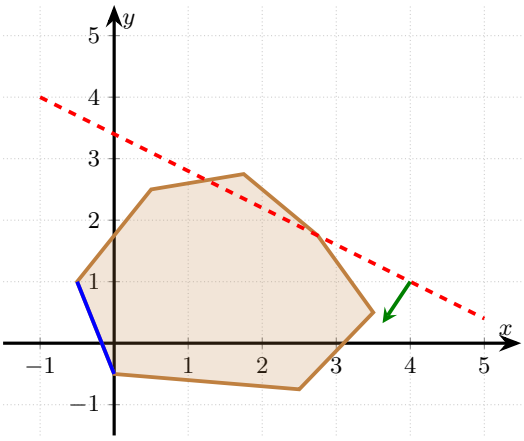


Figure 25: Illustration for 2D example of optimization via linear programming.

B. Mirror Descent Algorithm

Perhaps the most well-known algorithm for performing unconstrained optimization is the gradient descent or steepest descent algorithm [13, Sec. 8.3.2]. Assume one wants to find a minimum of a differentiable function $f(\mathbf{x})$. Starting from an initial guess x_0, we want achieve that every new guess x_{i+1} moves closer to the desired solution than the previous guess x_i. Because the original function f might be very complex or impossible to handle for the minimization, we use a first order Taylor-expansion at x_0 to approximate f, resulting in

$$\mathbf{x}_{i+1} = \arg\min_{\mathbf{x}} \big(f(\mathbf{x}_i) + \big\langle \nabla f(\mathbf{x}_i), (\mathbf{x} - \mathbf{x}_i) \big\rangle\big). \tag{42}$$

This expression is a linear function and therefore yields an infinite solution if the gradient is non-zero. Apart form that, it's common knowledge that the approximation via the Taylor-expansion gets worse the further we move away from the evaluated point $\mathbf{x}_i$. To overcome these issues, we introduce a penalty term $\frac{1}{2}||\mathbf{x}_i - \mathbf{x}||_2^2$ that effectively constraints our search for a minimum to the surroundings of our previous point $\mathbf{x}_i$, resulting in

$$\mathbf{x}_{i+1} = \arg\min_{\mathbf{x}} \big(f(\mathbf{x}_i) + \big\langle \nabla f(\mathbf{x}_i), (\mathbf{x} - \mathbf{x}_i) \big\rangle + \gamma||\mathbf{x}_i - \mathbf{x}||_2^2\big), \tag{43}$$

where the parameter γ controls the trade-off between taking larger or more accurate steps. Taking the gradient of above expression and setting it to zero results in the expression used in most introductory lectures.

Bregman Distance and Projection

In the short conceptual derivation of the gradient descent algorithm above, we carelessly employed the euclidean distance measure $\frac{1}{2}||\mathbf{x}_{i+1} - \mathbf{x}_i||_2^2$ in order to incorporate a notion of distance between our guesses. However, the results of this are more far-fetching than one might think initially, because we implicitly enforce a (possibly sub-optimal) euclidean geometry on our problem.
We saw in Section 2.2 that the euclidean metric punishes deviations equally in all directions, but we know that e.g. the Kullback-Leibler divergence from Section 2.1 can also give us a notion of dissimilarity if we are dealing with distributions. Depending on the application, alternative penalty terms can prove beneficial. Therefore, the idea of mirror descent [2] is to use a generalized notion of distance in form of a Bregman divergence to potentially enforce a different geometry. This divergence is defined via

$$B_\phi(\mathbf{x}, \mathbf{y}) = \phi(\mathbf{x}) - \phi(\mathbf{y}) - \langle \nabla\phi(\mathbf{x}), \mathbf{x} - \mathbf{y} \rangle, \tag{44}$$

where $\phi\colon \Omega \to \mathbb{R}$ is a continuously-differentiable, strictly convex function. This convexity leads to favorable properties e.g. regarding projections that are well understood from convex analysis. For points $\mathbf{x}$ and $\mathbf{y}$, the Bregman divergence represents the deviation of the

difference between function values at those the points with a linear approximation computed in $\mathbf{x}$ and evaluated at $\mathbf{y}$. Substituting the euclidean penalty term in Equation (43) with the general term from above, the new update reads

$$\mathbf{x}_{i+1} = \arg\min_{\mathbf{x}} \left(\langle \nabla f(\mathbf{x}_i), \mathbf{x} - \mathbf{x_i} \rangle + \frac{1}{\eta} B_\phi(\mathbf{x}, \mathbf{x_i}) \right). \tag{45}$$

For InfoOT and Fused InfoOT, using the negative entropy function of a transportation plan $\phi = \sum_{i,j} \mathbf{\Gamma}_{ij} \ln(\mathbf{\Gamma}_{ij})$ yields the Bregman divergence

$$B_\theta(\mathbf{\Gamma}', \mathbf{\Gamma}) = \sum_{i,j} \mathbf{\Gamma}'_{ij} \ln(\frac{\mathbf{\Gamma}'_{ij}}{\mathbf{\Gamma}_{ij}}) - \sum_{i,j} \mathbf{\Gamma}'_{ij} + \sum_{ii,j} \mathbf{\Gamma}_{ij} = D_{KL}(\mathbf{\Gamma}' || \mathbf{\Gamma}). \tag{46}$$

The second equality follows from the fact that $\sum_{i,j} \mathbf{\Gamma}'_{ij} = \sum_{ii,j} \mathbf{\Gamma}_{ij} = 1$ because of the marginal constraints for every transportation plan. Intuitively, this notion of distance is more suited than the euclidean one, since both plans can be interpreted as joint distributions. Remembering the mutual information relation from Equation (8), we can thereby limit the amount of additional information that is encoded in the transportation plan with each step. This is important, because each update is based on an approximation that is only valid in a trust region around the last iterate.
The framework we discussed so far only considers unconstrained optimization objectives. However, the convexity of the function used in the Bregman divergence leads to the fact that a projection

$$P_{\mathcal{K},\phi}(\mathbf{y}) = \arg\min_{\mathbf{x} \in \mathcal{K}} B_\phi(\mathbf{x}, \mathbf{y}) \tag{47}$$

on a convex set $\mathcal{K}$ has a unique solution [20]. Consequently, if we are dealing with constrained optimization, we can use this projection with the result of Equation (45) to obtain a solution on the feasible (convex) set.

Mirror Descent Algorithm
The name of the algorithm is due to the fact that one can rewrite the generalized update of the algorithm as

$$\mathbf{x}'_{i+1} = (\nabla\phi)^{-1}(\nabla\phi(\mathbf{x}_i) - \eta \nabla f_i(\mathbf{x}_i), \tag{48}$$

which can be interpreted as carrying out the following steps at each iteration:

(i) Map the current iterate $\mathbf{x}_i$ to a point in the dual space using the gradient of the convex function ϕ

$$\theta_i \leftarrow \nabla\phi(\mathbf{x}_i).$$

(ii) Take the gradient step in the dual space via

$$\theta_{i+1} \leftarrow \theta_i - \eta_i \nabla f(\mathbf{x}_i).$$

(iii) Map the result back to point $\mathbf{x}'_{i+1}$ of the initial space

$$\mathbf{x}'_{i+1} \leftarrow (\nabla\phi)^{-1}(\theta_{i+1}).$$

(iv) In the constrained case project the point $\mathbf{x}'_{i+1}$ to the "closest" point of the feasible set $\mathcal{K}$ w.r.t. the Bregman divergence used in the first steps

$$\mathbf{x}_{i+1} \leftarrow \min_{\mathbf{x} \in \mathcal{K}} B_\phi(\mathbf{x} || \mathbf{x}'_{i+1})$$

When following steps (i) to (iii) with $\mathbf{\Gamma}$ for $\mathbf{x}$, the Bregman divergence from above and the Fused InfoOT objective in place of $f(\mathbf{x})$, the update is

$$\mathbf{\Gamma}'_{i+1} = \left(\mathbf{\Gamma}_i \odot e^{-\frac{1}{\epsilon}(\mathbf{C} - \lambda \nabla_{\mathbf{\Gamma}_i} \hat{I}_{\mathbf{\Gamma}_i}(X,Y)} e^{-\nabla H(\mathbf{\Gamma}_i))} \right). \tag{49}$$

Projecting this back onto the feasible set $\mathbf{U}(\mathbf{a}, \mathbf{b})$ from Section 2.2 as in step (iv) results in

$$\mathbf{\Gamma}_{i+1} = \arg\min_{\mathbf{\Gamma}} D_{KL}(\mathbf{\Gamma} || \mathbf{\Gamma}'_{i+1}). \tag{50}$$

In [3] it is shown that this is equivalent to solving the objective

$$\Gamma_{i+1} \leftarrow \arg\min_{\Gamma \in \Pi(\mathbf{p},\mathbf{q})} \langle \Gamma, \mathbf{C} - \lambda \nabla_\Gamma \hat{\mathbf{I}}_{\Gamma_i}(X,Y) \rangle - \epsilon H(\Gamma), \tag{51}$$

at every iteration using the Sinkhorn algorithm.

Further Reading

[1] *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009.

[2] A. Beck and M. Teboulle. “Mirror descent and nonlinear projected subgradient methods for convex optimization”. In: *Operations Research Letters* 31.3 (2003), pp. 167–175.

[3] J.-D. Benamou, G. Carlier, M. Cuturi, L. Nenna, and G. Peyré. *Iterative Bregman Projections for Regularized Transportation Problems*. 2014.

[4] C.-Y. Chuang, S. Jegelka, and D. Alvarez-Melis. “InfoOT: Information Maximizing Optimal Transport”. In: ().

[5] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 2005.

[6] M. Cuturi. “Sinkhorn Distances: Lightspeed Computation of Optimal Transportation Distances”. In: *Advances in Neural Information Processing Systems 26* ().

[7] R. Flamary, N. Courty, A. Gramfort, M. Z. Alaya, A. Boisbunon, S. Chambon, L. Chapel, A. Corenflos, K. Fatras, N. Fournier, L. Gautheron, N. T. Gayraud, H. Janati, A. Rakotomamonjy, I. Redko, A. Rolet, A. Schutz, V. Seguy, D. J. Sutherland, R. Tavenard, A. Tong, and T. Vayer. “POT: Python Optimal Transport”. In: *Journal of Machine Learning Research* 22.78 (2021), pp. 1–8.

[8] N. Gozlan, C. Roberto, P.-M. Samson, and P. Tetali. *Kantorovich duality for general transport costs and applications*. 2015.

[9] D. Huffman. “A Method for the Construction of Minimum-Redundancy Codes”. In: *Proceedings of the IRE* 40.9 (1952), pp. 1098–1101.

[10] M. Idel. *A review of matrix scaling and Sinkhorn’s normal form for matrices and positive maps*. 2016.

[11] H. Janati. “Advances in optimal transport and applications to neuroscience”. PhD thesis. Institut Polytechnique de Paris, 2021.

[12] Y. Liu, L. Zhu, M. Yamada, and Y. Yang. “Semantic Correspondence as an Optimal Transport Problem”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020, pp. 4463–4472.

[13] K. Murphy. *Machine Learning - A Probabilistic Perspective*. Adaptive Computation and Machine Learning. Cambridge: MIT Press, 2014.

[14] J. Ni, Y. Li, Z. Huang, H. Li, H. Bao, Z. Cui, and G. Zhang. *PATS: Patch Area Transportation with Subdivision for Local Feature Matching*. 2023.

[15] *OpenReview InfoOT: Information Maximizing Optimal Transport*. 2022. URL: https://openreview.net/forum?id=nG08xiRT2As (visited on 09/21/2023).

[16] O. Pele and M. Werman. “Fast and robust Earth Mover’s Distances”. In: *2009 IEEE 12th International Conference on Computer Vision*. IEEE, 2009, pp. 460–467.

[17] M. Perrot, N. Courty, R. Flamary, and A. Habrard. “Mapping Estimation for Discrete Optimal Transport”. In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett. Vol. 29. Curran Associates, Inc., 2016.

[18] G. Peyré and M. Cuturi. “Computational Optimal Transport”. In: *Foundations and Trends in Machine Learning* ().

[19] G. Peyré, M. Cuturi, and J. Solomon. “Gromov-wasserstein averaging of kernel and distance matrices”. In: *International conference on machine learning*. PMLR. 2016, pp. 2664–2672.

[20] G. Raskutti and S. Mukherjee. “The Information Geometry of Mirror Descent”. In: *IEEE Transactions on Information Theory* 61.3 (2015), pp. 1451–1457.

[21] I. Redko, T. Vayer, R. Flamary, and N. Courty. *CO-Optimal Transport*. 2020.

[22] C. E. Shannon. *The mathematical theory of communication*. Urbana: Univ. of Illinois Press, 1998.

[23] T. Tantau. *The TikZ and PGF Packages. Manual for version 3.1.10 - 2023*. 2023.

[24] *The Annotated Transformer*. harvardnlp. 2018. URL: https://nlp.seas.harvard.edu/2018/04/03/attention.html (visited on 08/09/2023).

[25] A. Younes, S. Schaub-Meyer, and G. Chalvatzaki. *Entropy-driven Unsupervised Keypoint Representation Learning in Videos*. 2022.