The Entropy-Regularized Wasserstein Distance as a Metric for Machine Learning Based Post-Processing of Structural MR Images of the Brain

Master thesis
for attainment of the academic degree of
Master of Science

Westfälische Wilhelms-Universität Münster
Department of Mathematics und Informatics
Institute for Applied Mathematics

Submitted by:
Juliane Braunsmann

Thesis supervised by:
Prof. Dr. Benedikt Wirth
Prof. Dr. Xiaoyi Jiang
PD Dr. Tim Hahn

Münster, 30th September 2018

Abstract

This thesis treats the Wasserstein distance and its applicability as a metric between magnetic resonance images of the brain, represented as densities on a voxel grid. We aim to use the Wasserstein distance for dimensionality reduction of MR images by using it as a loss function in autoencoders. This requires a computationally feasible approach to calculate a large amount of distances, which can further be incorporated into machine learning frameworks. For this reason, we consider entropy regularization. We introduce this regularization for probability measures on compact metric spaces and explain why it suits these requirements. We further introduce an extension of the Wasserstein distance to measures with arbitrary mass, to which the same regularization approach can be applied. To evaluate the suitability of the Wasserstein distance to MR images, we incorporate it in kernel support vector machines and kernel principal component analysis to predict gender and depression, using a dataset provided by the Department of Psychiatry. Finally, we show that the Wasserstein distance can indeed be used to train autoencoders. However, we also address the question whether the application of the Wasserstein distance as a loss function for comparing MR images is advisable in practice.

Declaration of Academic Integrity

I hereby confirm that this thesis on *"The Entropy-Regularized Wasserstein Distance as a Metric for Machine Learning Based Post-Processing of Structural MR Images of the Brain"* is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

(date and signature of student)

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

(date and signature of student)

Acknowledgements

I would like to express my gratitude to everyone who made this thesis possible through their support:

- Prof. Dr. Benedikt Wirth for taking time to help me work out the mathematical problems that arose during the process of writing this thesis, for answering all my questions and for supporting my pursuit of a dual master's degree in mathematics and computer science.
- Prof. Dr. Xiaoyi Jiang for offering me an interesting and challenging topic that combines mathematics and computer science and for establishing the cooperation with the department of psychiatry and for making time for group meetings.
- PD Dr. Tim Hahn for introducing me to the topic of machine learning in psychiatry, for many interesting discussions, for offering me a space to work and for making the computational resources of the department and the MRI data available to me.
- The AI and Machine Learning in Psychiatry Group for helping me get started and for being pleasant office co-workers, always willing to answer my questions. I would like to especially thank Claas Kähler for always being approachable and Kelvin Sarink for being good at his job of looking after and fixing technical things.
- Dr. Bernhard Schmitzer for giving me advice regarding the implementation and derivation of the algorithm and for providing results obtained from his personal implementation of the algorithm for comparison.
- My parents for their support throughout my studies and especially during the last months of my thesis.
- Last but not least, I would like to thank my friends in Münster for always having a sympathetic ear, cheering me up and providing me with distraction when I struggled and of course for proofreading my thesis.

Contents

Introduction

The human brain is one of the most important organs of the human body and there has been much research into neuroimaging techniques such as Magnetic Resonance Imaging (MRI), which simplifies the (early) diagnosis of conditions such as brain tumors and aneurysms as well as chronic conditions such as multiple sclerosis. The obtained MR images are subsequently analyzed by a specifically trained professional and a report is sent to the treating physician.

Professionals are however not yet able to use neuroimaging techniques to diagnose a very common condition of the brain, which is one of the leading causes of disability worldwide: depression. It can affect people independently of their age, gender, wealth and location and has a large impact on the life of the affected individual as well as on their family and friends. The worst consequence of depression is suicide, which makes depression a disease with significant mortality. Despite depression being such a major issue, a reliable diagnosis of depression as well as the identification of a suitable treatment for each individual case remain difficult. To date, depression is diagnosed on the basis of behavioral symptoms, which means that the diagnosis depends on the patient's cooperation and perception. This makes the distinction between *major depressive disorder* and *bipolar disorder* an especially hard challenge: while bipolar patients pass through manic as well as depressive episodes, they only perceive depressive episodes as unusual. Time limitations and patient denial are additional factors that complicate diagnosis. For these reasons, the interest in a diagnosis based on neurobiological markers has grown in recent years.

Using high-resolution structural magnetic resonance imaging, researchers have found that structural changes are present in the brains of patients with depression. These findings motivate the quest for an automated system that could assist medical specialists in their decision-making. Such a system could be acquired by making use of machine learning models. Some research has been carried out in the direction of supervised learning, the dominant classification method being *support vector machines* (SVMs). Since the introduction of SVMs in the 1960s, significant progress has been made in the domain of supervised learning with *neural networks* (NNs) and *convolutional neural networks* (CNNs). While they had been introduced not much later than SVMs, their usage only became feasible recently thanks to advanced hardware, notably high performance graphics processing units (GPUs) and software developed for the use with GPUs, like `Tensorflow`. While CNNs have been very successful for example in object classification, there have been no comparable successes in the field of psychiatry. This has various reasons, one of them being the lack of large labeled datasets. As a reference, the largest database of labeled images, ImageNet, consists of over 14 million images, while datasets acquired from psychiatric studies usually only consist of less than one hundred subjects. The data set made available for this thesis by the Department of Psychiatry consists of almost two thousand labeled MR images, which is still not much compared to ImageNet, the data set is bound to grow further in the following years.

Much of the success of deep learning in the field of computer vision can be attributed to the fact that it renders the manual selection of suitable features for the task at hand unnecessary. Instead, "raw" images are fed into the network and suitable features are extracted automatically. In order to exploit this advantage for classification of depression, raw MR data has to be fed into the network. In this thesis, the raw images are unsmoothed, realigned, spatially normalized T1-weighted gray matter MR scans. Since MR images have many more voxels than images have pixels, this can be problematic memory-wise. For this reason it can be useful to perform post-processing in the form of dimensionality reduction, where optimally, redundant information is eliminated in the process. A common dimensionality reduction technique is *principal component analysis* (PCA), which gives a linear transformation of the data into a lower dimensional space. A more modern technique for performing dimensionality reduction can be found in the field of machine learning. The concept is to use a network consisting of an *encoder* and a *decoder* and train it in a way such that the output of the decoder is similar to the input of the encoder. Such a network is called an *autoencoder*. In contrast to PCA, autoencoders are in theory able to model arbitrary functions. Another advantage compared to PCA is that the representation resulting from dimensionality reduction with an autoencoder is not fixed after the end of training, but can be further adapted to any specific classification problem. This can be achieved by replacing the decoder with a neural network for classification, which could be further advantageous over directly using a neural net, since autoencoders do not require labels for training. It is thus possible to perform unsupervised learning with an autoencoder on a large unlabeled dataset, which makes it possible to feed more information on the structure of MR images into the network without requiring further labeling.

In order for an autoencoder to work well, a useful notion of similarity between two MR images is needed. Since taking the whole brain image would still be computationally infeasible, in this thesis we restrict ourselves to smaller brain regions which have been shown to be linked to depression. One possibility is to take the sum of squared voxel-wise differences, also referred to as the Euclidean distance or the 2-distance. However, this distance does not take into account the additional information of the spatial structure of the MR image. For this reason, the *Wasserstein distance* might define a better metric between MR images of brain regions. This thesis deals with the investigation of the usability of this idea.

The Wasserstein distance is an optimal transport based distance between probability measures. MR images can be understood as measures over a three dimensional grid, where the density of the gray matter determines the density of the measure. However, several problems arise. For one, the measure defined in this way does not usually define a probability measure. This cannot be alleviated by simply scaling the density values, because the total gray matter volume of regions may be subject to individual variation. We consider two ways of dealing with this problem: normalizing the MR images to have constant total gray matter volume, and extending the Wasserstein distance to unnormalized measures. We present an extension that was originally proposed by Leonid Kantorovich in the 1950s. An alternative formulation for computation was introduced in the 1990s, which we will make use of. A different problem is the high computational power required for calculating the Wasserstein distance. For this reason, we use the entropy-regularized variant of the Wasserstein distance instead, which was proposed in 2013 by Marco Cuturi. Compared to the original Wasserstein distance, a fast, parallelizable algorithm exists, which can be implemented in `Tensorflow`. The resulting regularized distance has the further advantage of being differentiable, which is important for the use with an autoencoder.

We first aim to evaluate whether the Wasserstein distance defines a useful notion of similarity between MR images. Our attempt is to use the Wasserstein distance to define *kernel functions*, which are usually defined using the squared Euclidean distance. Such kernel functions can be used for classification by integrating them into kernel SVMs. Kernel functions can also be integrated into PCA. Using kernel PCA in a pipeline in combination with standard SVM for classification supplies another way to evaluate the performance of the Wasserstein distance as a metric between MR images. The classification tasks used for evaluation are the classification of patients with major depressive disorder and healthy controls, as well as the classification regarding gender, which is less difficult. The same evaluation metrics can be used to assess the value of autoencoders trained with the Wasserstein distance as a loss function.

This thesis is organized as follows. In Chapter 1, a short introduction to machine learning is given, including neural networks and specifically autoencoders, (kernel) support vector machines and (kernel) principal component analysis. Chapter 2 gives a short overview of the mathematical background. Chapter 3 treats the mathematical theory behind Wasserstein distances and their extension to general measures and entropic regularization. In this chapter, we give rigorous proofs of some statements for which no such proofs could be found in the literature. Chapter 4 is a short chapter about the application to MR images. Chapter 5 explains how the Wasserstein distance can be used in machine learning. In Chapter 6 a description of the implementation follows, which contains also a short description of the packages used. In Chapter 7, the experiments conducted using the implementation from Chapter 6 are described and the results are stated. Finally, Chapter 8 gives a conclusion and an outlook on future work.

1 Machine Learning and Neural Networks

This chapter gives a short introduction to machine learning and neural networks. The following citation gives a good impression of what machine learning is about:

> [Machine Learning is about] searching for useful representations of some input data, within a predefined space of possibilities, using guidance from a feedback signal.[1]

This predefined space, which is called "hypothesis space", is defined by the architecture of a neural network. We describe two special kinds of neural networks: *feedforward neural networks* and *autoencoders*.

1.1 Feedforward Neural Networks

This section describes feedforward neural networks. We introduce some useful notations and give an overview of different layers. References for this section are [2, §5] and [1, §1].

We start with the definition of a *tensor*.

1.1.1. Definition (Tensor). Let

$$[n] := \left\{ 0, \dots , n-1 \right\} \text{ for } n \in \mathbb{N}$$

and $d \in \mathbb{N}$. Let $n_1, \dots , n_d \in \mathbb{N}$. Define

$$I := [n_1] \times \dots \times [n_d] = \left\{ i = (i_1, \dots , i_d) \ : \ i_j \in [n_j] \text{ for } 1 \le j \le d \right\}.$$

We reserve the letter I for such objects. Then a *tensor a* of dimension d is a map $a : I \to \mathbb{R}$ and can be identified with an element in $\mathbb{R}^{n_1 \times \dots \times n_d} \cong \mathbb{R}^{n_1 \cdots n_d}$. We say that a has shape $n_1 \times \dots \times n_d$.

This means basically that a tensor is a multidimensional array. A tensor of dimension 1 is a vector, and a tensor of dimension 2 is a matrix. A tensor of dimension 0 is simply a scalar.

Let x be a tensor, identified as $x \in \mathbb{R}^n$ for some $n \in \mathbb{N}$. A *feedforward neural network* is defined by a mapping of an input tensor x to some output tensor $y \in \mathbb{R}^m$ for some $m \in \mathbb{N}$,

$$y = f(x; \theta),$$

where f further depends on some parameters $\theta \in \mathbb{R}^l$ for some $l \in \mathbb{N}$. Usually, f consists of the concatenation of many different functions, which is where the name "network" comes from. For example, f could be the concatenation of three functions $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}, i = 1, 2, 3$, with $n_i \in \mathbb{N}, n_4 := m$ and $n_1 := n$, where each function also depends on some parameters $\theta_i \in \mathbb{R}^{l_i}$ for some $l_i \in \mathbb{N}, i = 1, 2, 3$. In this case, $\theta = (\theta_1, \theta_2, \theta_3) \in \mathbb{R}^{l_1 + l_2 + l_3}$, and

$$f(x; \theta) = f_3\Big(f_2(f_1(x; \theta_1); \theta_2) \ ; \theta_3 \Big) \ .$$

[1] [1, §1.1.3]

The individual functions f_i are usually of a specific form and can be interpreted as *layers* of the network, and the amount of layers is called the *depth* of the network. In the above case, we have a network with three layers, where f_1 defines the *first layer*, f_2 defines the *second layer* and f_3 defines the last layer, which is called the *output layer*. The other layers, whose output is not seen, are also called *hidden layers*. The goal is to use the network to approximate some function $f^* : \mathbb{R}^n \to \mathbb{R}^m$. In order to achieve this, one needs *training data* $(x^{(1)}, \dots, x^{(k)})$ with $x^{(i)} \in \mathbb{R}^n, i = 1, \dots, k,\ k \in \mathbb{N}$, associated to *training labels* $(y_1, \dots, y_k)$ with $y_i \approx f^*(x^{(i)}), i = 1, \dots, k$. We want the neural network to learn to output y_i whenever the input is $x^{(i)}$. For this we need a *cost function* or *loss function*, specifying a penalty if $f(x^{(i)})$ deviates from y_i. Let $L : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}_{\geq 0}$ be such a loss function. Then, the goal is to find

$$\theta^* = \arg\inf_{\theta} \sum_{i=1}^{k} L\big(f(x^{(i)}; \theta), y_i\big).$$

This can be done using gradient descent based methods. It is also possible for the loss function to depend on the function f or on θ.

Next, we will introduce some typical functions that define layers. The most simple layer is the *dense* or *fully-connected* layer. In this case, the parameter θ is a matrix W of *weights* combined with a *bias vector* b. Let $x \in \mathbb{R}^n, W \in \mathbb{R}^{m \times n}$ and $b \in \mathbb{R}^m$. Then the function describing the fully connected layer is

$$f_{\text{fc}} : \mathbb{R}^n \to \mathbb{R}^m, (x; (W, b)) \mapsto Wx + b.$$

This means the output y of the fully connected layer is calculated by

$$y_i = \sum_{j=1}^{n} w_{i,j} x_j + b_i, i = 1, \dots m.$$

The name "fully connected" emphasises that every output variable is connected with every input variable, so that every input variable contributes to the result of every output variable.

In some cases, the input has some geometrical structure, more specifically a grid-like structure. In such cases it might be sensible that not every input variable contributes to an output variable, but only a neighborhood of variables of a specified size. This results in *locally connected* layers.

Another variant are *convolutional layers*. In this case, the same weights are used for every neighborhood. The name "convolutional" comes from the mathematical concept of convolution. However, in machine learning, usually a similar operation is implemented, namely the *cross-correlation*. The definition is based on the implementation in Tensorflow, see [3].

1.1.2. Definition (Cross-correlation with zero-padding). Let $a : I \to \mathbb{R}$ be a d-dimensional tensor and let $m_1, \dots, m_d \in \mathbb{N}$. Define

$$J := \left\{-\left\lceil \tfrac{m_1}{2} \right\rceil + 1, \dots, \left\lfloor \tfrac{m_1}{2} \right\rfloor\right\} \times \dots \times \left\{-\left\lceil \tfrac{m_d}{2} \right\rceil + 1, \dots, \left\lfloor \tfrac{m_d}{2} \right\rfloor\right\}.$$

Let $k : J \to \mathbb{R}$ be a tensor, called *filter*. Let

$$\hat{a} : I + J \to \mathbb{R},\ k \mapsto \begin{cases} a(k), & \text{if } k \in I \\ 0, & \text{else.} \end{cases}$$

Then the *cross-correlation* $a * k$ of a, an $n_1 \times \dots \times_d$ input, and k, an $m_1 \times \dots \times m_d$ filter, is defined by

$$(a * k)(i_1, \dots, i_d) = \sum_{j \in J} \hat{a}(i_1 + j_1, \dots, i_d + j_d) k(j_1, \dots, j_d)$$

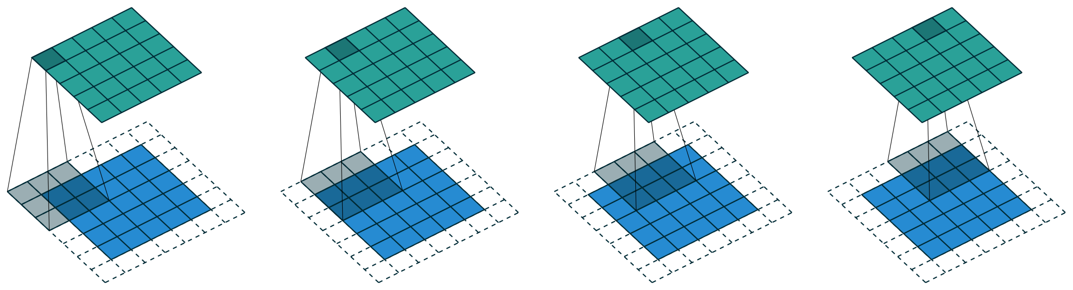


Figure 1.1: Computing the output values of cross-correlation for $d = 2$. The blue field corresponds to the tensor a on I, the dashed grid corresponds to the tensor $\hat{a}$ on $I + J$, and the grey shadow corresponds to the filter k on J. The green field is the resulting output $a \star k$.

for $(i_1, \dots, i_d) \in I$.

The definition of J in this way assures that the tensor $k : J \to \mathbb{R}$ always has shape $m_1 \times \dots \times m_d$, regardless of the parity of the m_i. For a convolutional layer, the parameter θ thus takes the form of the filter k. The function of a convolutional layer is

$$f_{\text{conv}} : \mathbb{R}^{n_1 \times \dots \times n_d} \to \mathbb{R}^{n_1 \times \dots \times n_d},\ (a; k) \mapsto a \star k.$$

The domain J, on which the filter is defined, is like a window that slides across the domain I, which has been enlarged to $I + J$. This enlargement is known as *padding* or *zero-padding*. The window is often called *receptive field*. For example, if $d = 2, n_1 = n_2 = 5, m_1 = m_2 = 5$, we have $I = \{0, 1, 2, 3, 4, 5\}^2, J = \{-1, 0, 1\}^2$ and $I + J = \{-1, 0, 1, 2, 3, 4, 5, 6\}^2$, so we have a padding of one in each direction. The concept of receptive fields and padding with these values is visualized in Figure 1.1. It is possible to further define *strides*, so the sliding window skips some pixels, which results in a smaller output. The operation of cross-correlation using striding is illustrated in Figure 1.2. Cross-correlation can be used with different filters in order to examine local properties of images.

For example, the well-known *Sobel operators*

$$S_x = \begin{pmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{pmatrix} \text{ and } S_y = \begin{pmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{pmatrix}$$

are used to detect vertical and horizontal edges, because they can be seen as a discrete version of the gradient in x- and y-direction, respectively.

Using convolution in neural networks means that the filters are not predefined, but are being learned. This means that the network learns by itself which features to look for in the input in order to better approximate the wanted function.

Other layers using the sliding window technique are *pooling layers*. Here, a reduction is performed on the receptive field, usually by either taking the maximum or by averaging, resulting in *max pooling* and *average pooling*, respectively. Usually, pooling is used to reduce the amount of neurons, which is achieved by using strides and no padding, i. e. the output is only defined for those $i \in I$ for which $i + j \in I$ for all $j \in J$. An illustration of max pooling on a 5×5 input with a 3×3 filter with no strides and no padding is shown Figure 1.3. It can be seen that many values are redundant, which can be avoided when using strides. Another important part

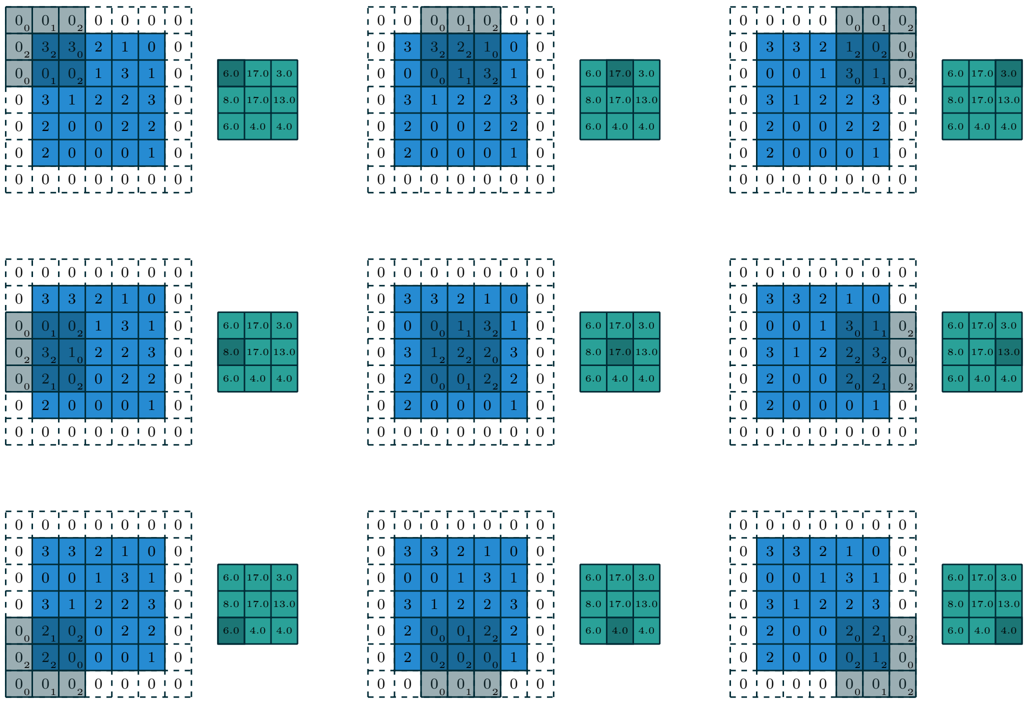


Figure 1.2: Computing the output values of cross-correlation with a 3 × 3 filter on a 5 × 5 input and striding 2 in both directions. The gray highlighting in the blue image shows which values are considered to compute the value of the pixel highlighted in gray in the green output image.

Source: [4]

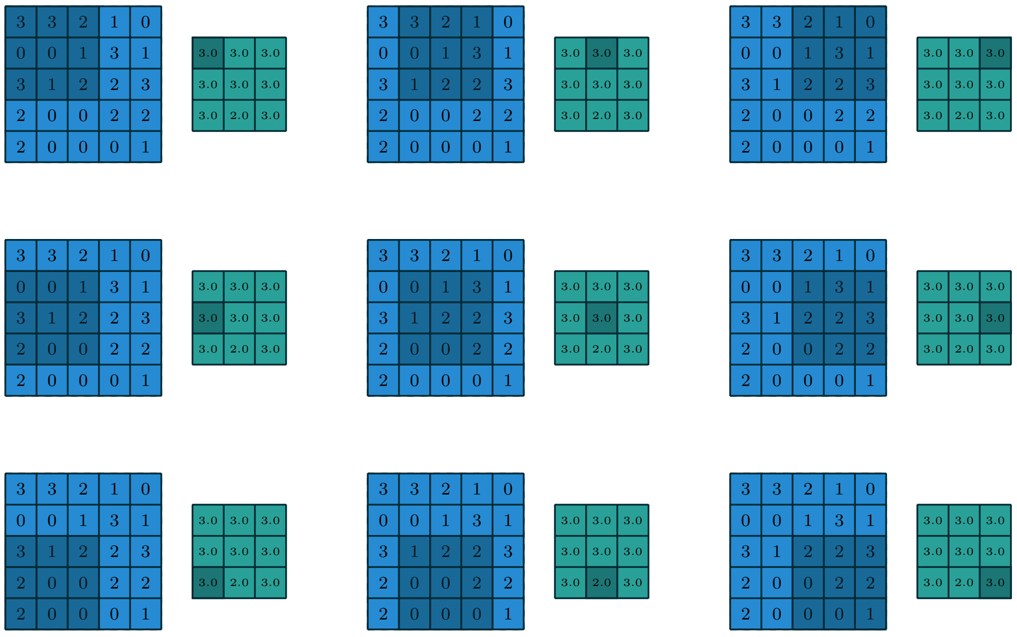


Figure 1.3: Computing the output values of a 3 × 3 max pooling operation on a 5 × 5 input using no strides and zero padding. The gray highlighting in the blue image shows which values are considered to compute the value of the pixel highlighted in gray in the green output image.

Source: [4]

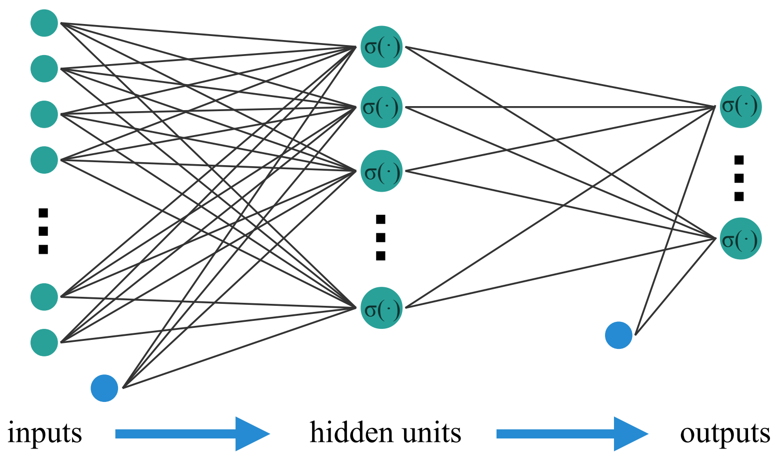


Figure 1.4: Illustration of a feedforward neural network consisting of two dense layers including activation functions. The input, hidden and output variables are represented as nodes, and the weights w_{ij} correspond to the lines connecting the nodes. The bias parameter b corresponds to the lines starting at the blue (hidden) node. The arrows emphasize the forward direction of the data flow through the network.

of neural networks are so called *activation functions*, which introduce non-linearities into the network. Especially fully connected layers are followed by an activation function σ of some kind. These activation functions are not considered to be separate layers. Let $y = (y_1, \dots, y_m) \in \mathbb{R}^m$. Some popular activation functions are

- **Sigmoid function:** $\sigma_{\text{sig}}(y)_i = \frac{1}{1+\exp(-y_i)}$,
- **Tanh function:** $\sigma_{\tanh}(y)_i = \frac{\exp(y_i)-\exp(-y_i)}{\exp(y_i)+\exp(-y_i)}$,
- **ReLu function:** $\sigma_{\text{ReLu}}(y)_i = \begin{cases} y_i, & \text{if } y_i \geq 0 \\ \alpha \cdot y_i, & \text{if } y_i < 0, \end{cases}$ where $\alpha \geq 0$ is some small constant,
- **Softmax function:** $\sigma_{\text{softmax}}(y)_i = \frac{\exp(y_i)}{\sum_{j=1}^{m} \exp(y_j)}$

The softmax function can be used to obtain normalized outputs. The sigmoid function outputs values between 0 and 1. An illustration of a general feedforward neural network is depicted in Figure 1.4.

1.2 Autoencoders

There are many different types of neural networks, suited to the task at hand, or the type of function to be approximated, respectively. In this section, we want to address *autoencoders*. The purpose of an autoencoder is to find an encoding function which reduces the dimension of its input by eliminating irrelevant information. This is attempted by simultaneously training an encoder and a decoder to produce a faithful copy of its input when concatenated. The main source for this section is [5, §14].

An autoencoder consists of two parts, the *encoder* and the *decoder*. A schematic diagram of an autoencoder network can be seen in Figure 1.5. More formally an autoencoder consists of the concatenation of two functions f_{enc} and f_{dec}, the encoder and the decoder function. The encoder produces a hidden code h representing an input x. Taking this hidden code h as input, the

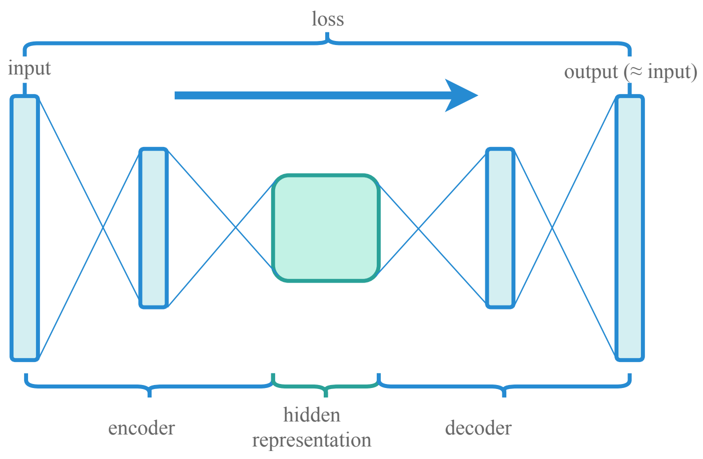


Figure 1.5: A schematic diagram of an autoencoder. It consists of an encoder function, a hidden representation and a decoder function. The autoencoder is trained to produce an output similar to the respective input. The loss function is a dissimilarity measure between input and output.

decoder then produces a decoding y. Summarized in formulas, this means

$$h = f_{\text{enc}}(x), \quad y = f_{\text{dec}}(h), \quad y = (f_{\text{dec}} \circ f_{\text{enc}})(x).$$

If the code has a lower dimension than the input, the encoder can be used for dimensionality reduction. Such types of autoencoders are called *undercomplete*. The objective of an undercomplete autoencoder is to minimize a loss of the type

$$L(x, f_{\text{dec}}(f_{\text{enc}}(x)))$$

for some dissimilarity function $L : \mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R}_{\geq 0}$. Ideally, forcing the encoder to use a smaller dimension leads to the encoder capturing the most distinguishing features of the input. However, if the encoder and decoder could learn any arbitrary function, it might happen that no useful information on the distribution of the data is extracted. An extreme example would be an autoencoder with a one-dimensional encoding layer, see [5, §14.1]. If this autoencoder is very powerful and no further restrictions are made, it could be possible that the encoder learns to encode each training example $x^{(i)}$ with the code i,

$$f_{\text{enc}}(x^{(i)}) = i \ \forall 1 \leq i \leq k.$$

The decoder could in turn learn to map these codes back to the original inputs,

$$f_{\text{dec}}((i)) = x^{(i)} \ \forall 1 \leq i \leq k.$$

Thus the autoencoder has simply memorized the training data. For any loss function penalizing only the dissimilarity of input and reconstruction, this would be optimal. However, such an autoencoder would perform very badly on unseen test data. Thus, only learning to copy the input to the output is not sufficient to learn meaningful representations. This means that some constraints on the encoding or decoding function are necessary to guarantee that the autoencoder does not overfit on the training data. One option to put a constraint on the functions is by choosing an architecture that limits the capacity of the network. For example, this can be achieved by limiting the parameters of the neural network. One possibility to limit parameters is given by using convolutions, if the input to reproduce has some spatial geometric structure. By using convolutional filters, the autoencoders are constrained to capture information on the data by using spatial filters. An autoencoder using convolutions is called a *convolutional autoencoder*.

Many other ideas to encourage the autoencoder to learn useful representations have been proposed. One of them is the *denoising autoencoder*, which gets as input a vector $\bar{x}$, which is a version of the original input x that has been corrupted by noise. Forcing the autoencoder to learn to reproduce x by minimizing a loss

$$L(x, f_{\text{dec}}(f_{\text{enc}}(\bar{x})))$$

makes the autoencoder learn implicitly the distribution of the data. A nice byproduct of denoising autoencoders is that it can be applied to data that is naturally noisy instead of artificially corrupted data and can thus effectively denoise data.

Another variant of an autoencoder is the *contractive autoencoder*.The idea of contractive autoencoders is to encourage the derivatives of the encoder to be small by adding a regularization term to the loss function. This is intended to prevent that small variations in the input data lead to distinct changes in the hidden presentation. More precisely, let

$$R(f_{\text{enc}}(x)) = \|Df_{\text{enc}}(x)\|_F^2 ,$$

where $Df_{\text{enc}}(x)$ is the Jacobian matrix of f_{enc} at the point x and $\|\cdot\|_F$ is the Frobenius norm. The goal is then to minimize

$$L(x, f_{\text{dec}}(f_{\text{enc}}(x))) + \lambda R(f_{\text{enc}}(x))$$

for some $\lambda > 0$.

So far, we have only considered autoencoders as a dimensionality reduction method. As already noted for denoising autoencoders, there are further usecases for autoencoders. One such use case is pretraining neural networks: an autoencoder does not only provide a lower dimensional representation of the input, but with the encoder it also provides a trainable function to produce these representations. Using this encoder, a classification network can be constructed by appending a classification function. A schematic diagram of such a network can be seen in Figure 1.6, compare also the diagram of the corresponding autoencoder in Figure 1.5. This means that autoencoders can effectively be used to pretrain networks, which can be useful to avoid getting stuck in a poor local minimum while training. Since autoencoders do not need labels, the set of available training data for autoencoders might be also larger, and thus more information on the structure of the data can be introduced into a network. Pretraining is further discussed in [6]. An example where such a scheme was successful is [7]. They use a stack of two-layer convolutional autoencoders, which are successively trained. The trained convolutional layers are then stacked and classification layers are added. This network is then fine-tuned to a specific task. They use this architecture successfully to diagnose Alzheimer's Disease using structural MR brain scans.

To conclude, at first glance autoencoders are neural networks to perform dimensionality reduction. Such a reduction is normally done with a specific goal in mind, such as classification. In this case, the encoder function can further be used, while the decoder function becomes irrelevant. This suggests that the dissimilarity measure used as a loss function does not have to guarantee a one-to-one reconstruction, but should rather assure that all features relevant for classification are kept. The choice of a suitable loss function is thus not trivial. In the course of this thesis, we

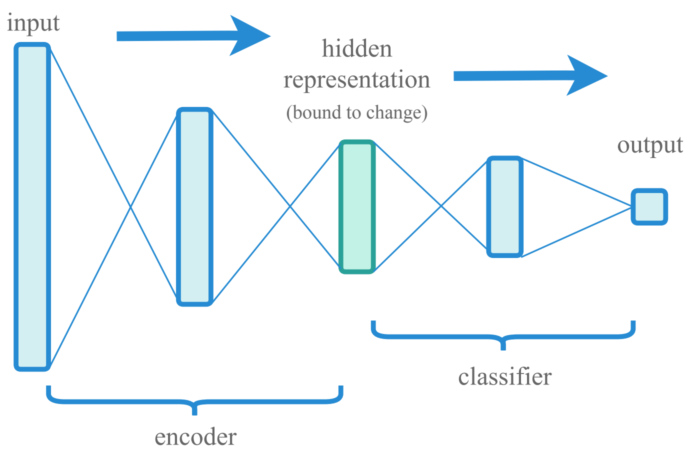


Figure 1.6: A schematic diagram of a network whose first layers consists of the pretrained encoding layers of an autoencoder. Instead of a decoding function, a classification function is added.

want to examine if the Wasserstein distance could be a replacement for the Euclidean distance in the case of MR images: the Euclidean distance as a loss function aims to reconstruct the image voxelwise, which might not be necessary to keep relevant features.

1.3 Support Vector Machines

Support vector machines (SVMs) solve classification problems between two sets of points belonging to two classes A and B by finding a good *decision boundary*. This decision boundary is a hyperplane separating the two sets. A good decision boundary is characterized by a large distance between the hyperplane and the closest data point of each class. The goal of an SVM is to find the hyperplane that maximizes this distance.

The following introduction to classical SVMs is based on [8]. Let

$$\left(x^{(1)}, \dots, x^{(k)}\right),\ x^{(i)} \in \mathbb{R}^n,\ i = 1, \dots, k \text{ for } k, n \in \mathbb{N}$$

with labels

$$(y_1, \dots, y_k),\ y_i \in \{\ \pm 1\}\ ,$$

where

$$y_i = \begin{cases} 1, & \text{if } x^{(i)} \text{ is in class } A, \\ -1, & \text{if } x^{(i)} \text{ is in class } B. \end{cases} \tag{1.1}$$

The goal is to learn a linear decision function $D(x) = \langle w, x \rangle + b$ with parameters $w \in \mathbb{R}^n$ and $b \in \mathbb{R}$, satisfying

$$D(x^{(i)}) > 0, \text{ if } x^{(i)} \text{ is in class A}$$
$$D(x^{(i)}) < 0, \text{ if } x^{(i)} \text{ is in class B}$$

for all $1 \leq i \leq k$.

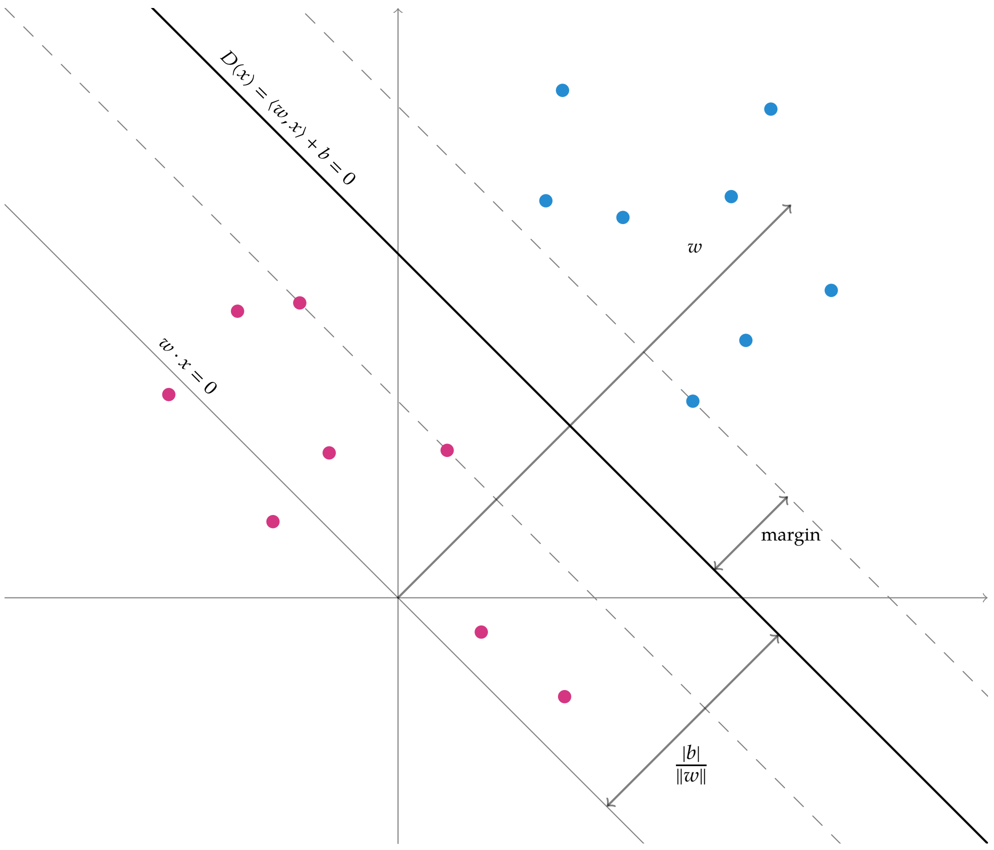


Figure 1.7: Illustration of a separating hyperplane. The hyperplane is visualized as a thick line. The dashed lines are defined by the support vectors. The margin M can be calculated as the distance of two points v and u that are multiples of w.

Then, an unknown point x is classified according to the rule

$$x \text{ is in class } \begin{cases} A, & \text{if } D(x) > 0, \\ B, & \text{otheriwse.} \end{cases}$$

The decision boundary defined as $\{x \in \mathbb{R}^n : D(x) = \langle w, x \rangle + b = 0\}$ is a hyperplane. Then vector w is orthogonal to the hyperplane. This representation of the hyperplane is not unique, because w and b can be multiplied by some nonzero constant. We want to avoid this by taking a canonical hyperplane. First we see that using the class labels, the conditions can be reformulated as

$$y_i D(x^{(i)}) > 0 \text{ for all } 1 \leq i \leq k. \tag{1.2}$$

Then a canonical hyperplane is a hyperplane with

$$\min_{i=1,\ldots,k} y_i D(x^{(i)}) = \min_{i=1,\ldots,k} y_i(\langle w, x \rangle^{(i)} + b) = 1. \tag{1.3}$$

Of course such a function can only be found if the data is linearly separable. Let us assume that this is the case. Then we wish to find a decision function, i. e. a separating hyperplane, that generalizes well. To achieve this, we search for the hyperplane which maximizes the smallest perpendicular distance between the training samples and the decision boundary. We call this distance the *margin*. An illustration can be seen in Figure 1.7. The distance between any point x and the hyperplane defined above is given by $\frac{|D(x)|}{\|w\|}$.

Now, if a hyperplane separating the classes A and B exists, it follows from (1.1) and (1.2) that the distance between a point and the hyperplane is

$$\frac{y_i D(x^{(i)})}{\|w\|} \text{ for all } 1 \le i \le k.$$

We can now consider this distance using canonical hyperplane as defined in (1.3), i. e. a hyperplane that satisfies

$$y_i D(x^{(i)}) = 1$$

for all points $x^{(i)}$ with the closest distance to the hyperplane. Then smallest distance between a point and the hyperplane, i. e. the margin, will be $\frac{1}{\|w\|}$.

Now we want to maximize the margin, which amounts to maximizing $\frac{1}{\|w\|}$, which is in turn equivalent to minimizing $\|w\|^2$. This leads to the consideration of the following optimization problem:

$$\begin{aligned} &\min_{w\in\mathbb{R}^n,\, b\in\mathbb{R}} \frac{1}{2}\|w\|^2 \\ &\text{subject to } y_i\big(\langle w, x^{(i)}\rangle + b\big) \ge 1 \text{ for all } 1 \le i \le k. \end{aligned} \tag{1.4}$$

It can be shown that in the case of linearly separable classes, the above problem is equivalent to

$$\begin{aligned} \max_{a\in\mathbb{R}^n} &\sum_{i=1}^{k} a_i - \frac{1}{2}\sum_{i,j=1}^{k} a_i a_j y_i y_j \langle x^{(i)}, x^{(j)}\rangle \\ \text{subject to } &a_i \ge 0, 1 \le i \le k \\ \text{and } &\sum_{i=1}^{k} a_i y_i = 0, \end{aligned} \tag{1.5}$$

and a solution to this problem exists and can be calculated using quadratic programming. The elements $x^{(i)}$ for which $a_i > 0$ are called *support vectors*, and these are exactly the elements that fulfill

$$y_i D(x^{(i)} = 1,$$

thus their margin is exactly $\frac{1}{\|w\|}$. The parameters of the optimal hyperplane can be recovered from the optimal solution a of (1.5) as

$$w = \sum_{i=1}^{k} a_i y_i x^{(i)},$$

where only support vectors contribute to the sum. Using some support vector $x^{(i_0)}$, the optimal offset b can be calculated by

$$y_{i_0}\langle w, x^{(i_0)}\rangle + b = 1.$$

In practice, we often encounter cases where two classes are not linearly separable. To solve this problem, Vapnik and Cortes introduced the following modification of support vector classification in [9], called *soft margin hyperplanes*. The main idea is to replace the constraints (1.2) with relaxed constraints

$$y_i\langle x^{(i)}, w\rangle + b \ge 1 - \zeta_i,\ 1 \le i \le k, \tag{1.6}$$

where

$$\zeta_i \ge 0,\ 1 \le i \le k, \tag{1.7}$$

are the so-called *slack variables*. Adding the slack variables assures that the constraints can always be fulfilled, since ζ_i only has to be chosen large enough. In order to still get sensible hyperplanes, we thus need to penalize large slack variables in the objective function.
Thus we want to solve

$$\min_{w\in\mathbb{R}^n,\ \zeta\in\mathbb{R}^k,\ b\in\mathbb{R}^n} \frac{1}{2}\|w\|^2 + \frac{C}{k}\sum_{i=1}^{k}\zeta_i$$

subject to the constraints (1.6) and (1.7) for some $C > 0$. If $\zeta_i = 0$, this means that the constraints are met, and $x^{(i)}$ lies behind margin line. If $\zeta_i > 0$, the constraint is violated. The element $x^{(i)}$ might lie between the margin line and the hyperplane, or it might even lie on the wrong side of the hyperplane if $\zeta_i > 1$. The value C controls how much margin errors are penalized. For small values of C, margin errors are more likely than for bigger values. As before, the optimization problem can be shown to be equivalent to a maximization problem, which can be solved using quadratic programming:

$$\begin{aligned} &\max_{a\in\mathbb{R}^k}\sum_{i=1}^{k} a_i - \frac{1}{2}\sum_{i,j=1}^{k} a_i a_j y_i y_j \langle x^{(i)}, x^{(j)}\rangle, \\ &\text{subject to } 0 \le a_i \le \frac{C}{m}, 1 \le i \le k, \\ &\text{and} \sum_{i=1}^{k} a_i y_i = 0. \end{aligned} \tag{1.8}$$

Again, the optimal w can be obtained through the optimal a by

$$w = \sum_{i=1}^{k} a_i y_i x^{(i)},$$

and b can be calculated using the support vectors. A further generalization of support vector classification consists of the *kernel trick*. This is based upon the fact that in the above problems, only the inner product of the training samples $x^{(i)}$ is required. Now one can consider replacing the inner product $\langle x^{(i)}, x^{(j)}\rangle$ by a kernel function $k(x^{(i)}, x^{(j)})$. This means that the samples $x^{(i)}$ need not lie in $\mathbb{R}^n$, but can lie in any set $\mathcal{X}$. We now give a short interpretation of this idea. First, we consider which functions can be used as kernels.

1.3.1. Definition (Positive Definite Kernel). Let $\mathcal{X}$ be a nonempty set. A function $k : \mathcal{X}\times\mathcal{X} \to \mathbb{R}$ is called *positive definite* if k is symmetric and for any $m \ge 0$, any family $x^{(1)}, \dots, x^{(m)} \in \mathcal{X}$ and any sequence $c_1, \dots, c_m \in \mathbb{R}$ the inequality

$$\sum_{i,j=1}^{m} c_i c_j k(x^{(i)}, x^{(j)}) \ge 0$$

holds. Such kernels are also called *Mercer kernels* and the above condition is called *Mercer's condition*.

Exchanging the standard inner product in $\mathbb{R}^n$ with the kernel k can be interpreted as mapping the elements $x^{(i)}$ to a different feature space and thus effectively performing support vector classification in a higher dimensional feature space. More specifically, if k is a positive definite kernel, it can be shown that there exists a map $\varphi : \mathcal{X} \to \mathcal{X}^{\mathbb{R}} = \{f : \mathcal{X} \to \mathbb{R}\}$ and an inner product $\langle\cdot,\cdot\rangle$ on $\mathcal{X}^{\mathbb{R}}$ such that

$$k(x, x') = \langle\varphi(x), \varphi(x')\rangle. \tag{1.9}$$

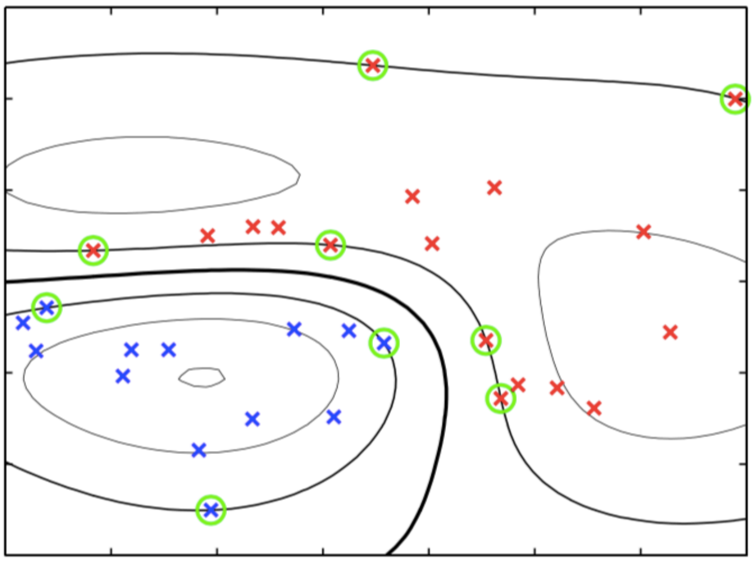


Figure 1.8: Example of synthetic data from two classes. The decision boundary using a Gaussian kernel is shown as the thick black line. The support vectors are circled in green. The contours show the points of constant $y(x)$.

Source: [2, §7.1]

This map is given by

$$\varphi(x) = k(\cdot, x) \text{ for all } x \in \mathcal{X}. \tag{1.10}$$

The closure of the image of φ under this map is a Hilbert Space, and it is called *Reproducing Kernel Hilbert Space (RKHS)*. We will refer to this space as the *feature space* and to the map φ as the *feature map*. Since all calculations above are valid in arbitrary Hilbert spaces, using a Mercer kernel in SVM is the same as performing SVM in the feature space. However, it is not necessary to calculate the mapping, nor to even know the mapping or the resulting feature space explicitly.

An example of a kernel is the popular *Gaussian radial basis function kernel*. For $\mathcal{X} = \mathbb{R}^n$ it is defined as

$$k^{\text{gauss}}(x, y) = \exp\left(-\frac{\|x-y\|^2}{2\sigma^2}\right) \quad \text{for } x, y \in \mathbb{R}^n,\ \sigma \in \mathbb{R}. \tag{1.11}$$

It is indeed a positive semidefinite Mercer kernel, a shown for example in [2, §6.2]. The form of the map φ is not explicitly given. However, it can be shown that for l distinct elements $x^{(1)}, \dots, x^{(l)}$, the vectors $\varphi(x^{(1)}, \dots, \varphi(x^{(l)})$ are linearly independent, see [10, Thm. 2.18]. Thus, the corresponding feature space has infinite dimension if the number of training samples is not restricted. This kernel describes a measure of similarity between two points x and x': the value $k_{\text{RBF}}(x, x')$ increases if the Euclidean distance between the points decreases. So for two points very close to each other, the kernel will give a high value. The parameter σ controls how fast the function decreases. Since the feature map is given by $\varphi(x) = k(\cdot, x)$, the image of x in feature space can be seen as a vector of similarities of x to all other points in $\mathbb{R}^n$. An illustration of a decision boundary obtained using a Gaussian kernel can be seen in Figure 1.8.

It makes sense to use this kernel if the Euclidean distance describes a meaningful distance. But other distances are imaginable: one can define a general radial basis function kernel for a metric space (Ω, d) through

$$k_d^{\text{rbf}}(x, y) = \exp\left(-\varepsilon d(x, y)^2\right) \quad \text{for } x, y \in \Omega,\ \varepsilon \in \mathbb{R}_{>0}, \tag{1.12}$$

see [11]. For example, if Ω is a set of strings, d could be the string edit distance, see [12]. However, this kernel is not guaranteed to be positive semidefinite, which means that the existence of an RKHS feature space is not guaranteed. In [13] it is remarked that it is still possible to formulate the optimization problems (1.5) and (1.8) and they obtain good results using non-Mercer kernels. They also remark, that it is not guaranteed that the resulting hyperplane maximizes some margin in a hidden space. But it can be shown that the symmetry of the kernel still guarantees the existence of a feature map φ and a symmetric bilinear form $Q(\cdot,\cdot)$ such that $k(x,x') = Q(\varphi(x),\varphi(x'))$, see [10, Prop. 2.25]. The lack of the positive definiteness means that Q does not define an inner product, and the resulting feature space is not a Hilbert space. In [14], *Reproducing Kernel Krein Spaces* have been introduced to study the feature spaces associated to non-Mercer kernels and a feature space interpretation of classical SVM with non-Mercer kernels has been given in [15].

1.4 Principal Component Analysis

In this section we want to introduce *Principal Component Analysis*. This technique, used for dimensionality reduction, has already been introduced in 1901 in [16] and has become a widely used tool. An extension of PCA making it possible to perform PCA in feature space has been introduced as *Kernel PCA* in 1992 by [17]. We will first explain the idea behind PCA and describe standard PCA, using [2, §12] as a source. Then we will present the extension of Kernel PCA, using [17] as a source.

The idea behind PCA is to search new axes in the input space, orthogonal to each other, such that most of the variance of the given data set is along those axes. For dimensionality reduction, one can then consider the subspace spanned only by a subset of the axes which consists of those axes along which the points are most dispersed, which is measured by variance. A point is then represented by the coordinates in that new subspace, which is called the *principal subspace*.

Another approach to reduce the dimensionality of some data is to find a linear projection onto a lower dimensional space such that the mean squared error between the original data points and their projections is minimized. It can be shown that this approach is equivalent to the first approach, which means that the subspace constructed through the directions with most variance is also the subspace that minimizes the mean squared distance between data points and projections. An illustration can be seen in Figure 1.9. We will now formalize the concept of maximum variance and describe how to find the axes with most variance.

Let $\left(x^{(1)},\dots,x^{(k)}\right),\ x^{(i)} \in \mathbb{R}^n,\ i = 1,\dots,k$ with $k,n \in \mathbb{N}$ be a set of *centered* sample points, i. e.

$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x^{(i)} = 0.$$

The maximum variance formulation aims to successively find normed, orthogonal directions (axes) $u^{(1)},\dots,u^{(l)}$ for some $l < n$ that maximize the *empirical variance* of the projection $h_u(x) = \langle u,x\rangle$ defined as

$$\operatorname{var}(u) = \frac{1}{k}\sum_{i=1}^{k} h_u(x^{(i)})^2 = \frac{1}{k}\sum_{i=1}^{k}\langle u,x^{(i)}\rangle^2.$$

Now, let X be the $(n \times k)$-matrix whose i-th column is the sample $x^{(i)}$, $X = (x^{(1)} \dots x^{(k)})$ and let $S = \frac{1}{k}XX^T$. Then we have, since $X^T u = (x^{(1)} \dots x^{(k)})^T u = (\langle x^{(1)}, u\rangle \dots \langle x^{(k)}, u\rangle)^T$,

$$\operatorname{var}(u) = \frac{1}{k}\left\langle X^T u, X^T u\right\rangle = \frac{1}{k}\left\langle u, XX^T u\right\rangle = \langle u, Su\rangle. \tag{1.13}$$

Now, the objective is to maximize $\operatorname{var}(u)$ under the constraint $\|u\| = 1$. Using Lagrange multipliers, it can be shown that a stationary point fulfills

$$Su = \lambda u$$

and using $u^T u = 1$ this means the variance is

$$\operatorname{var}(u) = u^T S u = u^T \lambda u = \lambda.$$

This means that the first direction is the eigenvector of S with maximal eigenvalue. It is called the first *principal component*. The next principal component will be the direction amongst all directions orthogonal to the first principal component that again maximizes the empirical variance. It can be shown by induction that the principal components $u^{(1)}, \dots, u^{(p)}$ are the eigenvectors corresponding to the p largest eigenvalues $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ of S. Note that for every real, symmetric matrix there always exists an orthonormal basis of real eigenvectors of the respective vector space by the spectral theorem, thus the $u^{(i)}$ can be chosen to be pairwise orthogonal. For any point $x \in \mathbb{R}^n$, its new coordinates in the principal subspace with respect to the orthonormal basis $u^{(1)}, \dots, u^{(p)}$ can be calculated as

$$x_{\text{code}} = \left(\left\langle u^{(1)}, x\right\rangle, \dots, \left\langle u^{(p)}, x\right\rangle\right)^T.$$

Denoting by U the matrix whose i-th column is $u^{(i)}$, this can be rewritten as

$$x_{\text{code}} = U^T x,$$

which shows that the reduction defines a linear map. Forming the linear combination of the coordinates and the basis vectors $u^{(1)}, \dots, u^{(p)}$, we get a vector $\tilde{x}$ in the original space $\mathbb{R}^n$ as

$$\tilde{x} = \sum_{i=1}^{p} x_{\text{code},i} u^{(i)} = \sum_{i=1}^{p} \left\langle u^{(i)}, x\right\rangle u^{(i)} \tag{1.14}$$

which can be interpreted as a reconstruction or decoding of x.

This leads to the second formulation: let $f_{\text{enc}} : \mathbb{R}^n \to \mathbb{R}^p,\ x \mapsto D^T x$ for any matrix D with normed orthogonal columns be an encoding function, and let the decoding function be given by $f_{\text{dec}} : \mathbb{R}^p \to \mathbb{R}^n,\ x \mapsto Dx$. It can be shown that the matrix U consisting of the eigenvectors corresponding to the maximal eigenvalues of XX^T as defined above minimizes the mean squared error

$$\sum_{i=1}^{k} \left\| x^{(i)} - f_{\text{dec}}(f_{\text{enc}}(x^{(i)})) \right\|^2 = \sum_{i=1}^{k} \left\| x^{(i)} - DD^T x^{(i)} \right\|^2$$

between the sample points $x^{(i)}$ and the reconstruction $\tilde{x}^{(i)} = DD^T x^{(i)}$, which corresponds to the projection of $x^{(i)}$ onto the principal subspace, see Figure 1.9.

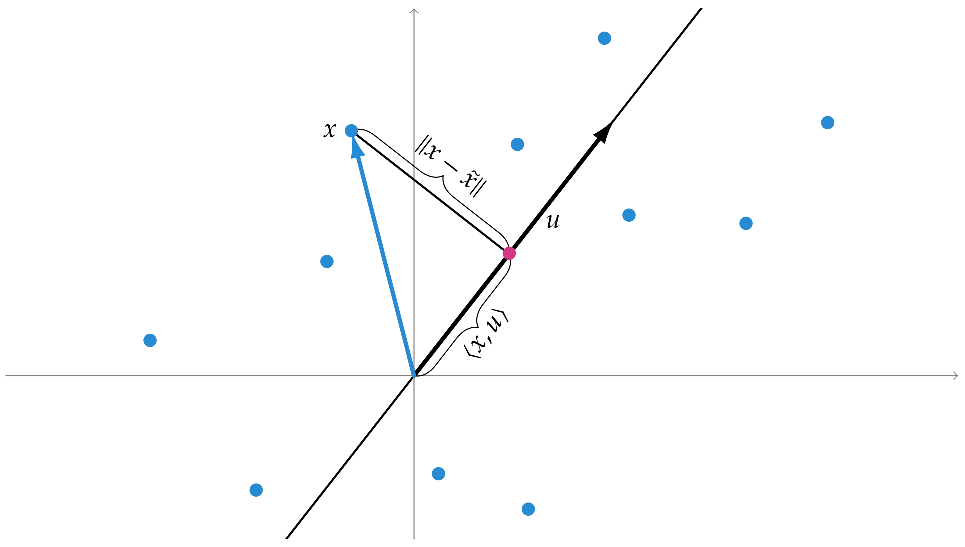


Figure 1.9: One formulation of PCA is to minimize the error $\|x-\tilde{x}\|$ between the point x (blue circle) and its projection $\tilde{x}$ (pink circle) onto the principal subspace, illustrated as the diagonal axis u. An equivalent formulation is to maximize the variance of x along u, i. e. the length of x along u, calculated as $\langle x, u\rangle$ for a normed vector u.

Let us now look at the optimal reconstruction error. Since the principal components are a part of a complete basis $u^{(1)}, \dots, u^{(n)}$ of $\mathbb{R}^n$, x can be written as $\sum_{i=1}^n \langle u^{(i)}, x\rangle u^{(i)}$. Then the optimal reconstruction error is, using the orthonormality of the basis and (1.13),

$$\begin{aligned}
\sum_{j=1}^k \left\|x^{(j)} - \tilde{x}^{(j)}\right\|^2 &= \sum_{i=1}^k \left\| \sum_{i=1}^n \left\langle u^{(i)}, x^{(j)} \right\rangle u^{(i)} - \sum_{i=1}^p \left\langle u^{(i)}, x^{(j)} \right\rangle u^{(i)} \right\|^2 \\
&= \sum_{j=1}^k \left\| \sum_{i=p+1}^n \left\langle u^{(i)}, x^{(j)} \right\rangle u^{(i)} \right\|^2 = \sum_{j=1}^k \sum_{i=p+1}^n \left\langle u^{(i)}, x^{(j)} \right\rangle^2 \\
&= \sum_{i=p+1}^n \operatorname{var}(u^{(i)}) = \sum_{i=p+1}^n \left\langle u^{(i)}, S u^{(i)} \right\rangle = \sum_{i=p+1}^n \left\langle u^{(i)}, \lambda_i u^{(i)} \right\rangle \\
&= \sum_{i=p+1}^n \lambda_i.
\end{aligned}$$

On the other hand, the maximum variance is

$$\begin{aligned}
\sum_{i=1}^p \operatorname{var}(u^{(i)}) &= \sum_{i=1}^p \left\langle u^{(i)}, S u^{(i)} \right\rangle = \sum_{i=1}^p \left\langle u^{(i)}, \lambda_i u^{(i)} \right\rangle \\
&= \sum_{i=1}^p \lambda_i.
\end{aligned}$$

Thus, the amount of eigenvalues controls both the reconstruction error and variance. These quantities can be used to choose a fitting value for p.

We now want to use the kernel trick in order to be able to perform non-linear PCA. An illustration of why this can be useful is shown in Figure 1.10.

So far, the method is not formulated in terms of inner products, but it can be reformulated. Denote by φ the feature map, and by $\mathcal{H}$ the corresponding RKHS. The matrix S in this setting

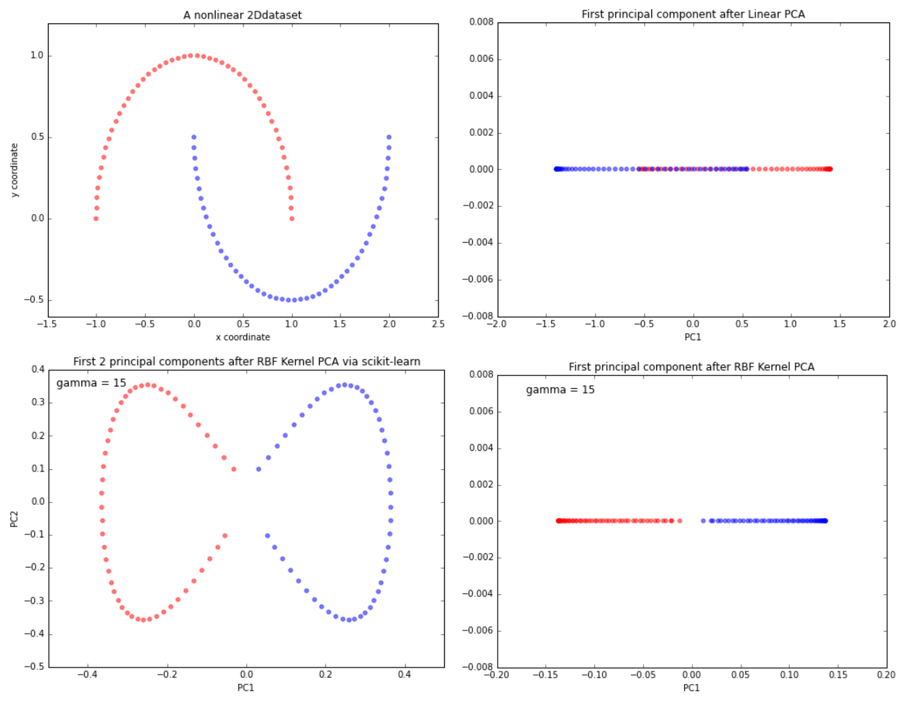


Figure 1.10: Top row: a synthetic nonlinearly separable dataset and the projection onto one dimension using standard PCA. Bottom row: the projection of the dataset onto two dimensions and one dimension using RBF kernel PCA. The projection onto two dimensions shows the distribution of the data in feature space.

Source: [18]

has to be seen as a linear operator $S : \mathcal{H} \to \mathcal{H}$ and is defined by

$$Sx = \frac{1}{k} \sum_{j=1}^{k} \varphi(x^{(j)}) \left\langle \varphi(x^{(j)}), x \right\rangle \text{ for all } x \in \mathcal{H}.$$

It is easily seen that in the case where $\varphi(x) = x$, this definition yields exactly the definition of S as above. Denote by K the kernel matrix defined through $K_{ij} = \langle \varphi(x^{(i)}), \varphi(x^{(j)}) \rangle$.

So far, we have assumed the data to be centered. Now we have to assume that the data is centered in the feature space, which cannot be done explicitly, since the feature map is usually not explicitly known. However, it is easy to see that the kernel of the centered data can be expressed through the kernel of the uncentered data as

$$\tilde{K} = K - 1_k K - K 1_k + 1_k K 1_k,$$

where $(1_k)_{ij} = \frac{1}{k}$ for all $1 \leq i, j \leq k$, which is sufficient, since we aim for a formulation in terms of the kernel anyway.

As before, we aim to find eigenvectors $u \neq 0 \in \mathcal{H}$ and eigenvalues $\lambda \geq 0$ satisfying

$$Su = \lambda u, \tag{1.15}$$

since all calculations done up to this point can be done in arbitrary Hilbert spaces. Our goal now is to rewrite this equation in a way such that it only depends on inner products. First, we see that every solution of (1.15) is also a solution of

$$\left\langle \varphi(x^{(j)}), Su \right\rangle = \lambda \left\langle \varphi(x^{(j)}), u \right\rangle \text{ for every } 1 \leq j \leq k. \tag{1.16}$$

Denote by V the span of $\varphi(x^{(1)}), \dots, \varphi(x^{(k)})$. We claim that the solutions of (1.15) are exactly the solutions of (1.16) which lie in V. Assume that u is a solution of (1.16). Then, by taking linear combinations, (1.16) implies that $\langle v, Su\rangle = \lambda\langle v, u\rangle$ for all $v \in V$, or $\langle v, Su - \lambda u\rangle = 0$ for all $v \in V$. If we now assume $u \in V$, we get that $Su - \lambda u \in V$ and thus

$$\langle Su - \lambda u, Su - \lambda u\rangle = \|Su - \lambda u\|^2 = 0,$$

which shows $Su - \lambda u = 0$. This means that any $u \in V$ which is a solution to (1.16) is also a solution to (1.15). On the other hand, if u is a solution of (1.15), it follows immediately from the definition of S that for $\lambda \neq 0$ any solution u of (1.16) lies in the span of $\varphi(x^{(1)}), \dots, \varphi(x^{(k)})$. We conclude that the solutions to (1.15) are indeed exactly the solutions to (1.16) which lie in V. So instead of looking for solutions to (1.15), we can constrain (1.16) to V by expressing u as

$$u = \sum_{i=1}^{k} \alpha_i \varphi(x^{(i)}) \tag{1.17}$$

for some $\alpha \in \mathbb{R}^k$. The problem of finding eigenvectors u is then equivalent to finding coefficients α. By putting u into (1.16), we get

$$\sum_{i=1}^{k} \alpha_i \left\langle \varphi(x^{(j)}), S\varphi(x^{(i)}) \right\rangle = \lambda \sum_{i=1}^{k} \alpha_i \left\langle \varphi(x^{(j)}), \varphi(x^{(i)}) \right\rangle \tag{1.18}$$

and by further replacing the definition of S we get

$$\frac{1}{k} \sum_{i=1}^{k} \alpha_i \Big\langle\, \varphi(x^{(j)}), \sum_{l=1}^{k} \varphi(x^{(l)}) \left\langle \varphi(x^{(l)}), \varphi(x^{(i)}) \right\rangle \Big\rangle \; = \lambda \sum_{i=1}^{k} \alpha_i \left\langle \varphi(x^{(j)}), \varphi(x^{(i)}) \right\rangle.$$

This formula can be expressed in terms of the kernel as

$$K^2\alpha = k\lambda K\alpha. \tag{1.19}$$

It can be shown that solving the equation

$$K\alpha = k\lambda\alpha \tag{1.20}$$

instead yields all solution that are of interest for us, see [17, Appendix A].

Let $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_p$ denote the first p eigenvalues of K (which are the solutions $k\lambda$ of (1.20)), where p is chosen such that λ_p is positive, and let $\alpha^{(1)}, \dots, \alpha^{(p)}$ be the corresponding eigenvectors. Note that the eigenvectors $\alpha^{(l)}$ now describe the coefficients of the principal components $u^{(l)}$ as a linear combination of the feature vectors. In order to get normalized principal components, we have to normalize the coefficients $\alpha^{(l)}$. By using (1.17) and (1.20), we see that

$$\begin{aligned}
\left\langle u^{(l)}, u^{(l)} \right\rangle &= \left\langle \sum_{i=1}^{k} \alpha_i^{(l)} \varphi(x^{(i)}), \sum_{i=1}^{k} \alpha_i^{(l)} \varphi(x^{(i)}) \right\rangle \\
&= \sum_{i,j=1}^{k} \alpha_i^{(l)} \alpha_j^{(l)} \left\langle \varphi(x^{(i)}), \varphi(x^{(j)}) \right\rangle = \sum_{i,j=1}^{k} \alpha_i^{(l)} \alpha_j^{(l)} K_{ij} \\
&= \left\langle \alpha^{(l)}, K\alpha^{(l)} \right\rangle = k\lambda \left\langle \alpha^{(l)}, \alpha^{(l)} \right\rangle.
\end{aligned}$$

Thus, requiring $\|u\| = 1$ amounts to normalizing $\alpha^{(l)}$ to have $\left\|\alpha^{(l)}\right\|^2 = \frac{1}{k\lambda_l}$. Now, in order to compute the new coordinates of a test point $\varphi(x)$, we have to calculate the projection of $\varphi(x)$ onto the eigenvectors $u^{(l)}$, which can be done as

$$\varphi(x)_{\text{code},l} = \left\langle u^{(l)}, \varphi(x) \right\rangle = \sum_{i=1}^{k} \alpha_i^{(l)} \left\langle \varphi(x^{(i)}), \varphi(x) \right\rangle.$$

Again, it can be shown that choosing the vectors $u^{(l)}$ as the eigenvectors to the maximal eigenvalues leads to a minimal reconstruction error, with the same reconstruction error value as before. However, this reconstruction error has to be interpreted as a reconstruction error in the feature space. The reconstruction formula (1.14) in the case of Kernel PCA reads

$$\widetilde{\varphi(x)} = \sum_{i=1}^{p} \beta^{(i)} u^{(i)} = \sum_{i=1}^{p} \beta^{(i)} \sum_{l=1}^{k} \alpha^{(l)} \varphi(x^{(l)})$$

for $\beta^{(i)} = \sum_{j=1}^{k} \alpha_j^{(i)} \left\langle \varphi(x^{(j)}, \varphi(x) \right\rangle$. In order to get a reconstruction in the original space, we thus need to invert the (unknown) function φ. Further, there might not even exist a preimage for $\widetilde{\varphi(x)}$. Methods to handle this problem are discussed in [19] and [20].

As a last point, let us remark on indefinite kernels. Since in order to calculate the principal components one really only needs to calculate eigenvectors, PCA can be implemented for indefinite kernels without any problems. As remarked before, the feature space corresponding to indefinite kernels is not a Hilbert space, which means that the interpretation of minimum reconstruction error and maximum variance do not remain correct, but in [21] it is shown that there exists an alternative formulation in the feature space. They also demonstrate the usability of indefinite kernels in the case of a *truncated* ℓ_1 *distance* kernel compared to RBF kernel PCA and linear PCA. Using six different datasets, they show that the indefinite kernel outperforms the RBF kernel and linear kernel on some datasets and gives similar accuracy as the RBF kernel on the others.

2 Mathematical Background

In this chapter we give a brief overview of the mathematical background we will need in the next chapter. The first section treats the space of measures on a compact metric space and its dual, the space of continuous functions. We further state some well known theorems such as the Radon-Nikodym and Hahn Decomposition theorems. The second section collects some results from the field of convex analysis that we will need for the next chapter, most notably the Fenchel Duality theorem. We will start off with a short motivation for choosing the setting of a compact metric space.

Our motivation is to consider Wasserstein distances between MR images of brain regions. Since the Wasserstein metric is a metric between probability measures, it is thus necessary to model MR images of brain regions as probability measures on some space. After preprocessing, an MR image is a three-dimensional image, which can be understood to be a function on a three-dimensional grid as a subset of $\mathbb{R}^3$. Two-dimensional images, as functions on subsets of $\mathbb{R}^2$, are usually represented as matrices. They are defined on grids consisting of *pixels*, which have two coordinates $(i, j) \in \mathbb{N}^2$, where usually the i is the row and j is the column. The three-dimensional analogon to a pixel is a *voxel*, which has three coordinates $(i, j, k) \in \mathbb{N}^3$. The analogon to a matrix in higher dimension is often called *tensor*, as defined in Definition 1.1.1. An MR image is a rank three tensor of shape $(n_1, n_2, n_3) \in \mathbb{N}^3$. Since a voxel grid is a subset of $\mathbb{R}^3$, a natural definition of a metric on a voxel grid is through the Euclidean metric. It consists of finitely many elements, which means that it can be seen as a compact metric space. Now unlike images in general, MR images can be interpreted as measures, since they describe gray matter volume. This means we can see an MR image as a measure on a compact metric space. We could concentrate on finite metric spaces, but the mathematical theory becomes much more conceptual if we consider general compact metric spaces. However, we always assume the metric space to be compact, since it is sufficient for our purposes.

2.1 The Space of Measures and its Dual

In this section we will introduce the space of Radon measures and its dual. The definitions are based on [22, §6]. We start with the definition of a Radon measure.

2.1.1. Definition. Let (X, d) be a compact metric space, equipped with the Borel sigma algebra. A *Radon measure* is a finite, signed Borel measure on X. We define $\mathcal{M}(X)$ as the set of all Radon measures. Further, we define $\mathcal{M}_+(X)$ as the set of all non-negative Radon measures. The set of all probability measures, i.e. non-negative Radon measures μ with $\mu(X) = 1$, is denoted by $\mathcal{P}(X)$.

If $X = \{1, \dots, n\}$ is finite, then a Radon measure can be identified with a vector in $\mathbb{R}^{|X|}$, since every measure μ is uniquely determined by the mass of singletons, since $\mu(A) = \sum_{i \in A} \mu(\{i\})$ for all $A \subseteq X$.

2.1.2. Example (Dirac Measure and Counting Measure). Let X be any set.

- For some $x \in X$, the *Dirac measure* is the measure δ_x defined by
$$\delta_x(E) = \begin{cases} 1, & \text{if } x \in E \\ 0, & \text{if } x \notin E, \end{cases}$$
for any subset $E \subseteq X$. Since $\delta_x \geq 0$ and $\delta_x(X) = 1$, it is a probability measure. If $X = \{1, \dots, n\}$ is finite, then δ_k for $k \in X$ is the unit vector $e_k \in \mathbb{R}^n$.
- The *counting measure* is the measure λ_{count} defined by
$$\lambda_{\text{count}}(E) = \begin{cases} \infty, & \text{if } E \text{ is infinite} \\ \text{number of elements in } E, & \text{if } E \text{ is finite,} \end{cases}$$
for any subset of $E \subseteq X$. The counting measure is not a probability measure if X has more than one element. If X is finite, it is possible to consider the *normalized counting measure*, defined as
$$\lambda^1_{\text{count}}(E) = \frac{\text{number of elements in } E}{\text{number of elements in } X},$$
which fulfills $\lambda^1_{\text{count}} \geq 0$ and $\lambda^1_{\text{count}}(X) = 1$ and is thus a probability measure.

So far, we have only considered $\mathcal{M}(X)$ as a set. Since we allow measures to be negative in $\mathcal{M}(X)$, we can scalar multiplication and addition of measures by
$$\begin{aligned} (\mu + \nu)(E) &= \mu(E) + \nu(E) \\ (c\mu)(E) &= c\mu(E) \end{aligned} \tag{2.1}$$
for all measurable sets E, measures $\mu, \nu \in \mathcal{M}(X)$ and $c \in \mathbb{R}$. The axioms of a vector space are satisfied. For example, using this vector space structure, we can write the counting measure as a sum of Dirac measures,
$$\lambda_{\text{count}} = \sum_{x \in X} \delta_x.$$

For finite $X = \{1, \dots, n\}$, this structure corresponds to the vector space structure of $\mathbb{R}^n$. In order to define a norm on the vector space $\mathcal{M}(X)$, we need the following definition.

2.1.3. Definition (Total, Positive and Negative Variation Measures). The *total variation measure* $|\mu| \in \mathcal{M}_+(X)$ of a measure $\mu \in \mathcal{M}(X)$ is defined as
$$|\mu|(E) = \sup_{\mathcal{E}} \sum_{E_i \in \mathcal{E}} |\mu(E_i)|,$$
where the supremum is taken over all partitions $\mathcal{E}$ of E into measurable sets E_i. The *positive variation measure* $\mu_+ \in \mathcal{M}_+(X)$ is defined as
$$\mu^+ = \frac{1}{2}(|\mu| + \mu),$$
and the *negative variation measure* $\mu_- \in \mathcal{M}_+(X)$ is defined as
$$\mu^- = \frac{1}{2}(|\mu| - \mu).$$

If μ is a positive measure, we have $|\mu| = \mu, \mu^+ = \mu$ and $\mu^- = 0$. If X is finite and μ is given as a vector $(\mu_1, \dots, \mu_n) \in \mathbb{R}^n$, then $|\mu| = (|\mu_1|, \dots, |\mu_n|)$. The following relations exist between the measure, which is easy to see from the definition.

$$\mu = \mu^+ - \mu^- \quad \text{and} \quad |\mu| = \mu^+ + \mu^-. \tag{2.2}$$

An intuitive description of positive and negative variation is given by the following theorem.

2.1.4. Theorem (Hahn Decomposition Theorem). Let $\mu \in \mathcal{M}(X)$. Then there exist two measurable sets P and N with $P \cup N = X$, $P \cap N = \emptyset$, such that the positive and negative variations μ^+ and μ^- of μ satisfy

$$\mu^+(E) = \mu(P \cap E) \quad \text{and} \quad \mu^-(E) = -\mu(P \cap E)$$

for all measurable sets $E \subseteq X$. The pair (P, N) is called *Hahn decomposition* of X.

This means that X is the union of two disjoint sets P and N such that "P carries all the positive mass of μ" and "N carries all the negative mass of μ". This "disjointness" of μ^+ and μ^- can be formalized by the notion of *singularity*. As a reference see [23, §4].

2.1.5. Definition. Let $\mu, \nu \in \mathcal{M}(X)$ be two measures. Then μ is said to be *singular* w. r. t. ν, $\mu \perp \nu$, if there exists a set $B \subseteq X$ such that

$$\mu(B) = 0 \quad \text{and} \quad \nu(B^c) = 0,$$

where B^c denotes the complement of B in X. Since this relation is symmetric, we also say that μ and ν are *mutually singular*. It further holds that for all measurable sets E,

$$\mu(E) = \mu(E \cap B^c) \quad \text{and} \quad \nu(E) = \nu(E \cap B).$$

Since μ^+ and μ^- satisfy

$$\mu^+(N) = \mu^+(P \cap N) = \mu^+(\emptyset) = 0$$

and

$$\mu^-(N^c) = \mu^-(P) = \mu^-(P \cap N) = \mu^-(\emptyset) = 0,$$

we see that μ^+ and μ^- are mutually singular, $\mu^+ \perp \mu^-$.

We can now define a norm on $\mathcal{M}(X)$ by $\|\mu\| = |\mu|(X)$, where $|\mu|$ is the total variation measure. For non-negative measures, this simplifies to $\|\mu\| = \mu(X)$. All axioms for a norm are satisfied and $\mathcal{M}_+(X)$ is thus a normed vector space. When X is finite and μ is a vector, $\|\mu\| = \|\mu\|_1$ is the 1-norm.

Next, we want to describe the relation of the space $\mathcal{M}(X)$ to the space of continuous real-valued functions. The following discussion is based on [24, §13].

2.1.6. Definition. The space $C(X)$ is the space of all continuous real-valued functions on X. It is a Banach space with the norm

$$\|f\| = \sup_{x \in X} |f(x)|.$$

As a first relation between the two spaces, we have the following lemma.

2.1.7. Lemma. Every $\rho \in \mathcal{M}(X)$ defines a bounded functional on $C(X)$ through

$$\alpha(f) = \int_X f \mathrm{d}\rho.$$

This means that every measure in $\mathcal{M}(X)$ defines an element in the dual space of $C(X)$. The inverse is also true, which is stated in the following lemma.

2.1.8. Lemma. Let (X, d) be a compact metric space. For every bounded functional α on $C(X)$, there exists a unique finite signed measure $\rho \in \mathcal{M}(X)$ such that

$$\alpha(f) = \int_X f \mathrm{d}\rho,$$

for all $f \in C(X)$.

We thus get a bijection between the space of Radon measures and the dual of the space of continuous real-valued functions. We can further identify the subset of continuous functions that stands in bijection with the set of non-negative Radon measures.

2.1.9. Definition (Positive Functional). A functional α on $C(X)$ is called positive if and only if

$$f \geq 0 \Rightarrow \alpha(f) \geq 0 \text{ for all } f \in C(X).$$

Now, it can be shown that the functional α is positive if and only if ρ is non-negative, i.e. $\rho \in \mathcal{M}_+(X)$, which means that non-negative measures correspond exactly to positive functionals. We finally summarize everything in the following theorem. It is known in a more general setting as the *Riesz–Markov–Kakutani Representation Theorem* or *Riesz Representation Theorem*.

2.1.10. Theorem (Riesz Representation Theorem). Let (X, d) be a compact metric space. Then the map

$$\omega : \mathcal{M}(X) \to C(X)^*, \quad \rho \mapsto \alpha, \text{ where } \alpha(f) = \int_X f \mathrm{d}\rho$$

defines an isometric isomorphism between $C(X)^*$ and $\mathcal{M}(X)$. In particular,

$$\|\rho\| = |\rho|(X) = \|\alpha\|.$$

Finally, α is positive if and only if ρ is non-negative.

This shows that we can interpret $\mathcal{M}(X)$ as the dual space of $C(X)$. This means we can use the notion of weak-* convergence on $\mathcal{M}(X)$.

2.1.11. Definition. A sequence of measures $(\mu_n)_n \subseteq \mathcal{M}(X)$ converges weakly-* to $\mu \in \mathcal{M}(X)$ if and only if

$$\int_X f \mathrm{d}\mu_n \longrightarrow \int_X f \mathrm{d}\mu$$

in $\mathbb{R}$ for all $f \in C(X)$. We write $\mu_n \xrightarrow{*} \mu$.

The subset of probability measures $\mathcal{P}(X)$, i. e. the set of all non-negative measures μ with $\mu(X) = 1$ is of special interest to us. It can be written as

$$\mathcal{P}(X) = \{\mu \in \mathcal{M}(X) : \|\mu\| \leq 1 \text{ and } \mu(X) = 1\},$$

since the conditions $\|\mu\| = \mu^+(X) + \mu^-(X) \leq 1$ and $\mu(X) = \mu^+(X) - \mu^-(X) = 1$ imply $\mu^- = 0$ and hence $\mu \geq 0$. It can thus be seen as a subset of the unit ball in $\mathcal{M}(X)$.

This gives rise to the following theorem, which is a corollary of the Banach-Alaoglu theorem. For details see [24, Cor. 13.9].

2.1.12. Theorem (Sequential Compactness). Let X be a compact metric space. Then the set $\mathcal{P}(X)$ is sequentially compact in the weak-* topology.

In the next chapter, we will need the concept of *densities* additionally to the concept of measures. Some measures can be expressed as a density with respect to another measure. This is the case if and only if that measure is *absolutely continuous* with respect to the other measure. This is the content of the following definition and theorem.

2.1.13. Definition (Absolute Continuity and Equivalence of Measures). A measure $\mu \in \mathcal{M}(X)$ is called *absolutely continuous* with respect to a measure $\nu \in \mathcal{M}(X)$, written $\mu \ll \nu$, if $\mu(A) = 0$ for any measurable subset A with $\nu(A) = 0$. Two measures $\mu_1, \mu_2 \in \mathcal{M}(X)$ are called *equivalent*, $\mu_1 \sim \mu_2$, if $\mu_1 \ll \mu_2$ and $\mu_2 \ll \mu_1$.

2.1.14. Theorem (Radon-Nikodym). Let $\mu, \lambda \in \mathcal{M}(X)$ be measures. If $\mu \ll \lambda$, then there exists a function $f \in L^1(X, \lambda)$ such that

$$\mu(A) = \int_A f(x) \mathrm{d}\lambda(x) \tag{2.3}$$

for every measurable set $A \subseteq X$. The function f is called the *Radon-Nikodym derivative* of μ with respect to λ and is denoted by $\frac{\mathrm{d}\mu}{\mathrm{d}\lambda}$. Further, there always exists a unique pair of measures μ_a and μ_s such that

$$\mu = \mu_a + \mu_s, \qquad \mu_a \ll \lambda \text{ and } \mu_s \perp \lambda,$$

and the measure μ can be written as

$$\mu = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda} \cdot \lambda + \mu_s, \qquad \mu_s \perp \lambda.$$

This is called the *Radon-Nikodym decomposition.*

As a reference see [22, Thm. 6.10].

Thus, if we fix some non-negative measure λ, the space $L^1(X, \lambda)$ corresponds to a subset of measures $\mu \in \mathcal{M}(X)$ by forming $\mu = f\lambda$ for some $f \in L^1(X, \lambda)$.

2.2 Convex Analysis and Nonlinear Optimization

In this section we give a short summary of definitions and results from convex analysis and nonlinear optimization that will be of use to us. An important tool in convex analysis is given by *Fenchel conjugation.*

2.2.1. Definition (Fenchel Conjugate). Let $\mathcal{X}$ be a Banach space and let $g : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ be a functional. Then the *Fenchel conjugate* $g^* : \mathcal{X}^* \to \mathbb{R} \cup \{\infty\}$ of g is defined as

$$g^*(x^*) = \sup_{x \in \mathcal{X}} \langle x^*, x \rangle - g(x).$$

For non-reflexive spaces, we further define the *Fenchel preconjugate.* Let $f : \mathcal{X}^* \to \mathbb{R} \cup \{\infty\}$, then the *Fenchel preconjugate* ${}^*f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is defined as

$${}^*f(x) = \sup_{x^* \in \mathcal{X}^*} \langle x^*, x \rangle - f(x^*).$$

We say that a function $f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ is *proper* if there exists $x_0 \in \mathcal{X}$ such that $f(x_0) < \infty$.

2.2.2. Proposition. Let $f : \mathcal{X}^* \to \mathbb{R} \cup \{\infty\}$ be proper, convex and weak-* lower semicontinuous. Then

$$({}^*f)^* = f.$$

This is a variant of the well-known Fenchel Moreau Theorem, and a proof can be found in [25, Thm. 9.3.4].

We now want to look at a special family of functionals, examined in [26], which we use as a reference.

2.2.3. Definition (Integral functional). Let (X, λ) be a compact measure space.

Let $f : X \times \mathbb{R} \to \mathbb{R} \cup \{\infty\}$ such that $f(x,t)$ is convex in t for every $x \in X$. Then the *integral functional* $I_f : L^1(X, \lambda) \to \mathbb{R} \cup \{\infty\}$ is defined by

$$I_f(u) = \int f(x, u(x)) \mathrm{d}\lambda(x).$$

We can then look at the Fenchel conjugate of integral functionals. This is examined in [26, Thm.2].

2.2.4. Proposition. Let $f : X \times \mathbb{R}$ such that $f(x,t)$ is measurable in x for each t and convex in t for each x. Let $f^* : X \times \mathbb{R}$ be the conjugate of f in t, i. e. $f^*(x,t) = f_x^*(t)$ for fixed $x \in X$. Let $f^*(x, u^*(x))$ be integrable in x for at least one $u^* \in L^\infty(X, \lambda)$ and let $f(x, u(x))$ be integrable in x for at least one $u \in L^1(X, \lambda)$.

Then the Fenchel conjugate I_f^* of the integral functional I_f is I_{f^*}.

Another definition we need is the *adjoint* of a linear map.

2.2.5. Definition (Adjoint). Let $A : \mathcal{X} \to \mathcal{Y}$ be a bounded, linear map. Then the *adjoint* $A^* : \mathcal{Y}^* \to \mathcal{X}^*$ of A is defined by the identity

$$\langle A^* y^*, x \rangle_{\mathcal{X}^*, \mathcal{X}} = \langle y^*, Ax \rangle_{\mathcal{Y}^*, \mathcal{Y}}$$

for all elements $y^* \in \mathcal{Y}^*, x \in \mathcal{X}$.

The Fenchel duality theorem is a very important theorem to obtain duality results. The following theorem can be obtained easily from the classical version of Fenchel Duality, stated for example in [27, Theorems 4.4.2/4.4.3].

2.2.6. Theorem (Fenchel Duality). Let $\mathcal{X}$ and $\mathcal{Y}$ be Banach spaces. Let $f : \mathcal{X} \to \mathbb{R} \cup \{+\infty\}$ and $g : \mathcal{Y} \to \mathbb{R} \cup \{\infty\}$ be convex functions and $A : \mathcal{X} \to \mathcal{Y}$ be a bounded linear map with adjoint map $A^* : \mathcal{Y}^* \to \mathcal{X}^*$. Let $f^* : \mathcal{X}^* \to \mathbb{R}$ and $g^* : \mathcal{Y}^* \to \mathbb{R}$ denote the Fenchel conjugates of f and g, respectively. Then the primal and dual values $p, d \in [-\infty, +\infty]$ defined by the problems

$$\begin{aligned} p &= \sup_{x \in \mathcal{X}} \{-f(x) - g(Ax)\} \\ d &= \inf_{y^* \in \mathcal{Y}^*} \{f^*(-A^* y^*) + g^*(y^*)\} \end{aligned} \tag{2.4}$$

satisfy the weak duality inequality $d \geq p$. Denote by $\operatorname{dom} f$ the set of points where f is finite,

$$\operatorname{dom} f := \{x \in X : f(x) < \infty\}.$$

Now, if further $A \operatorname{dom} f$ contains a point x at which g is continuous, then $p = d$.

We now want to discuss differentiability. As a reference, we use [25, §9]. There exist different notions of differentiability on Banach spaces, including the Fréchet derivative, which is a generalization of the derivative of real-valued functions in a single variable. For our purposes, we need the weaker notion of the *subdifferential*. This is defined for Banach spaces in a similar way as for real-valued functions in a single variable.

2.2.7. Definition (Subdifferential). Let $\mathcal{X}$ be a Banach space with dual $\mathcal{X}^*$. Let $f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ be a convex, proper, lower semi-continuous function. Then we say an element $x^* \in \mathcal{X}^*$ belongs to the subdifferential of f at $x \in \mathcal{X}$ if and only if

$$f(y) \geq f(x) + \langle x^*, y - x\rangle \text{ for all } y \in \mathcal{X}.$$

We write $x^* \in \partial f(x)$.

We are particularly interested in the subdifferential of Fenchel conjugates. We have the following theorem, which is a combination of [25, Prop. 9.5.1 and Thm. 9.5.1].

2.2.8. Theorem. Let $\mathcal{X}$ be a Banach space with dual $\mathcal{X}^*$. Let $f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ be a convex, proper, lower semi-continuous function. Then for $x^* \in \mathcal{X}^*$ and $x \in X$ the following three conditions are equivalent.

(i) $x^* \in \partial f(x)$

(ii) $x \in \partial f^*(x^*)$

(iii) $f(x) + f^*(x^*) - \langle x^*, x\rangle = 0$

Similarly to differentiable functions, we have the following theorem concerning minimizers of convex functions.

2.2.9. Theorem. Let $\mathcal{X}$ be a Banach space with dual $\mathcal{X}^*$. Let $f : \mathcal{X} \to \mathbb{R} \cup \{\infty\}$ be a convex, proper, lower semi-continuous function. Then $x \in \mathcal{X}$ is a minimizer of f if and only if $0 \in \partial f(x)$.

3 A Metric between Measures based on Optimal Transport Theory

This chapter treats the Wasserstein distance. We start with the definition of the Wasserstein distance and introduce optimal transport using the Kantorovich formulation. We then suggest an extension of the Wasserstein distance which makes it possible to define transport between measures with different mass. Next, we introduce the entropy-regularized version of optimal transport, which makes it possible to efficiently compute Wasserstein distances. We then discuss the properties of the entropy-regularized Wasserstein distance. For the underlying space on which measures are defined we consider a compact metric space, at times equipped with a measure.

The new contributions of this thesis consist primarily in providing rigorous proofs for results which have either been stated only in a less abstract setting, or for which no detailed proofs could be found in the literature. Section 3.1 treats standard results and definitions. Section 3.2, which introduces an extension of the Wasserstein distance for arbitrary measures, contains results which have been stated in a discrete setting, but have been proved neither in a general setting nor in the discrete setting. We contribute a formulation that is valid in the setting of a compact metric space and give detailed proofs of the results using this formulation (Proposition 3.2.2, Lemma 3.2.6, Proposition 3.2.7, Theorem 3.2.8 and Corollary 3.2.9). Section 3.3 introduces the entropy-regularized version of optimal transport and Section 3.4 examines properties of the resulting regularized distance. Various papers treating this approach are available, but many are concerned only with a discrete setting or do not include proofs. The main results in this section for which new rigorous proofs are provided are Theorem 3.3.19 and Theorem 3.3.20. We also give some new counterexamples (Example 3.3.22, Example 3.4.2). Most of the stated properties in Section 3.4 have been stated before, but without proof. Some properties had not been formulated before to our knowledge (Corollary 3.4.3, Corollary 3.4.7).

3.1 The Kantorovich Problem and the Wasserstein Distance

The goal of this section is to define a notion of similarity of measures. For this, we look to optimal transport theory.

Optimal transport is concerned with the question of how to transport several items, distributed at different starting locations, to specific target locations in an optimal way. The distribution of the items can be interpreted as a probability measure, which can be either discrete or continuous. This problem was first formulated by Monge. His formulation looked for a mapping of starting locations to target locations, with the interpretation that all items at one location are transported to the same target location. Such a map is shown in Figure 3.1. However, it is not always possible to find such a map. The problem was then reformulated by Kantorovich, allowing for items at one starting location to be split up and transported to different target locations.

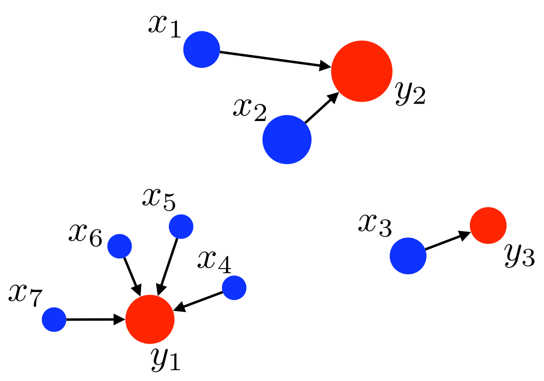


Figure 3.1: We have 7 different starting locations $x_1, \dots, x_7$, and the size of the circles representing the locations can be interpreted to be proportional to the amount of contained "items" (the "mass"). We have 3 different target locations, to which specific amounts of items should be transported.
Source: [28, §2.2]

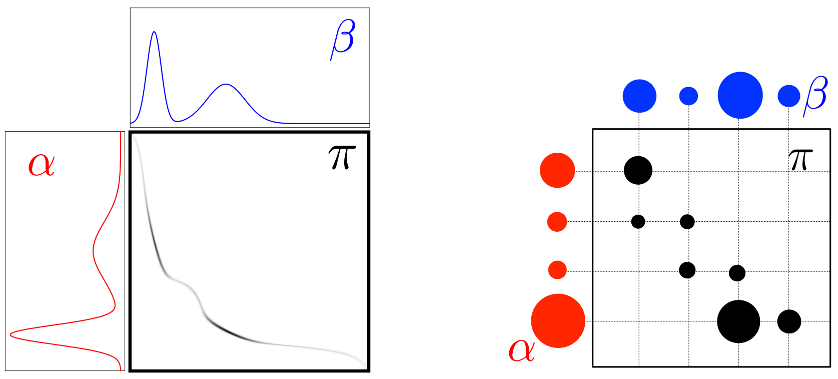


Figure 3.2: Optimal transport between two one-dimensional probabilities (left) and between two discrete measures (right). While the transport plan on the left corresponds to a map between locations, the one on the right does not. For example, the mass at the last location of α (the largest circle) is split up and distributed to two different locations.
Source: [28, §2.3]

The goal is now to find a matching, specifying not only which starting location is mapped to which target location, but also how many items are transported. This matching corresponds to a map on the Cartesian product of starting locations and target locations and it is called an *optimal coupling* or *optimal transport plan*. Instead of talking about amounts of items, we usually talk about mass. This is illustrated in Figure 3.2 for discrete locations and continuous locations. Having already in mind the application to MR images, the locations correspond in this case to voxels in a three-dimensional grid, and the mass corresponds to gray matter volume located at the voxels.

We will now formalize the above problem and prove some classical results. We first need the following definition and proposition, to be found for example in [29, §3.6]

3.1.1. Definition (Pushforward measure). Let X, Y be two measurable spaces, $T : X \to Y$ a measurable mapping and μ a measure on X. The *pushforward measure* $T^{\#}\mu$ is a measure on Y defined through

$$T^{\#}\mu(A) = \mu(T^{-1}(A))$$

for all $A \subset Y$.

This construction makes it possible to define a measure on Y for any given measure on X, using a map from X to Y. Integration of a function with respect to this new measure on Y can be expressed in terms of the original measure on X through the following change-of-variables formula.

3.1.2. Proposition. Let $\mu \in \mathcal{M}(X)$ and $T : X \to Y$. A measurable function g on Y is integrable with respect to the pushforward measure $T^{\#}\mu$ if and only if $g \circ T$ is integrable with respect to μ, and the integrals are related by

$$\int_Y g \mathrm{d}(T^{\#}\mu) = \int_X g \circ T \mathrm{d}\mu. \tag{3.1}$$

Now let Ω_1 and Ω_2 be two compact metric spaces. We will assume that they are equipped with the Borel sigma algebra, which implies that they are also measurable spaces. Let the projection maps from $\Omega_1 \times \Omega_2$ onto the respective components be given by $\mathrm{pr}_1 : \Omega_1 \times \Omega_2 \to \Omega_1, (x, y) \mapsto x$ and $\mathrm{pr}_2 : \Omega_1 \times \Omega_2 \to \Omega_2, (x, y) \mapsto y$.

3.1.3. Definition. Let Ω_1 and Ω_2 be two compact metric spaces. Given $\mu \in \mathcal{M}_+(\Omega_1), \nu \in \mathcal{M}_+(\Omega_2)$ with $\mu(\Omega_1) = \nu(\Omega_2)$, i.e. $\|\mu\| = \|\nu\|$, and a continuous function $c : \Omega_1 \times \Omega_2 \to \mathbb{R}_+$, the *Kantorovich problem* is defined as

$$\inf \left\{ W(\pi) = \int_X c(x, y) \mathrm{d}\pi(x, y) \ : \ \pi \in \Pi(\mu, \nu) \right\}, \tag{KP}$$

where $\Pi(\mu, \nu)$ is the set of so-called *transport plans*,

$$\Pi(\mu, \nu) := \left\{ \pi \in \mathcal{M}_+(\Omega_1 \times \Omega_1) \ : \ \mathrm{pr}_1^{\#} \pi = \mu, \ \mathrm{pr}_2^{\#} \pi = \nu \right\}. \tag{3.2}$$

The function c is called *cost function*, since it specifies the cost of transporting mass from location x to location y.

The entity $\pi(A, B)$ indicates how much mass is moved from a subset $A \subseteq \Omega_1$ to a subset $B \subseteq \Omega_2$. The conditions $\mathrm{pr}_1^{\#} \pi = \mu$ and $\mathrm{pr}_2^{\#} \pi = \nu$ mean that $\pi(A, Y) = \mu(A)$ and $\pi(X, B) = \nu(B)$, respectively, i.e. the total mass moved away from A by π must be exactly the mass given to A by μ, and the total mass moved to B by π must be exactly the mass given to B by ν. Figure 3.2 (left) shows the density of a transport plan π between two one-dimensional measures μ and ν, which are also visualized as densities. Without loss of generality we will from now on assume that μ and ν are probability measures, i. e. $\|\mu\| = \|\nu\| = 1$.

We now want to prove that the above problem always admits a minimizer. First, we have the following statement about the set of transport plans.

3.1.4. Lemma. Let Ω_1 and Ω_2 be compact metric spaces, $\mu \in \mathcal{P}(\Omega_1)$ and $\nu \in \mathcal{P}(\Omega_2)$. Let $\mu \otimes \nu \in \mathcal{P}(\Omega_1 \times \Omega_1)$ be the product measure, i. e. the unique measure that satisfies

$$(\mu \otimes \nu)(A, B) = \mu(A) \cdot \nu(B) \tag{3.3}$$

for all measurable sets $A \subseteq \Omega_1, B \subseteq \Omega_2$. Then the set $\Pi(\mu, \nu)$ as defined above always contains $\mu \otimes \nu$ and it is closed in the weak-* topology.

Proof. We have

$$\mathrm{pr}_1^{\#}(\mu \otimes \nu)(A) = (\mu \otimes \nu)(A \times \Omega_2) = \mu(A)\nu(\Omega_2) = \mu(A)$$

for all $A \subseteq \Omega_1$ and likewise,

$$\mathrm{pr}_2^{\#}(\mu \otimes \nu)(B) = \nu(B)$$

for all $B \subseteq \Omega_2$. Hence, $\mu \otimes \nu \in \Pi(\mu, \nu)$. Now let $(\pi_n)_n \subset \Pi(\mu, \nu)$ be a sequence with $\pi_n \xrightarrow{*} \pi^*$. We have to show that π^* is in $\Pi(\mu, \nu)$. The concatenation $f \circ \mathrm{pr}_1$ is in $C(\Omega_1 \times \Omega_2)$ for all $f \in C(\Omega_1)$

and therefore, using weak-* convergence and the change-of-variables property (3.1), we get

$$\int_{\Omega_1} f(x) \mathrm{d}\, \mathrm{pr}_1^{\#} \pi^*(x) = \int_{\Omega_1 \times \Omega_2} (f \circ \mathrm{pr}_1)(x,y) \mathrm{d}\pi^*(x,y) = \lim_n \int_{\Omega_1 \times \Omega_2} (f \circ \mathrm{pr}_1)(x,y) \mathrm{d}\pi_n(x,y)$$
$$= \lim_n \int_{\Omega_1} f(x) \mathrm{d}\, \mathrm{pr}_1^{\#} \pi_n(x) = \int_{\Omega_1} f(x) \mathrm{d}\mu(x).$$

This shows that $\mathrm{pr}_1^{\#} \pi^*$ and μ define the same functional in $C(\Omega_1)^*$. By Theorem 2.1.10 we therefore get $\mathrm{pr}_1^{\#} \pi^* = \mu$. One shows in the same way that $(\mathrm{pr}_2)_{\#}\pi^* = \nu$. Thus, $\Pi(\mu, \nu)$ is closed in the weak-* topology. ∎

Now, the existence of a minimizer of the above problem can easily be shown by the Direct Method of Calculus of Variation, as seen in the following theorem.

3.1.5. Theorem. Let Ω_1 and Ω_2 be two compact metric spaces, $\mu \in \mathscr{P}(\Omega_1)$, $\nu \in \mathscr{P}(\Omega_2)$ and $c : \Omega_1 \times \Omega_2 \to \mathbb{R}_+$ a continuous function $c \in C(\Omega_1 \times \Omega_2)$. Then KP is finite and admits a solution.

Proof. As seen in the above lemma, $\Pi(\mu, \nu)$ contains $\mu \otimes \nu$. Further c is a continuous function on a compact space and is thus bounded. In particular, $W(\mu \otimes \nu)$ is finite. For all $\pi \in \Pi(\mu, \nu)$ we have $\|\pi\| = \pi(\Omega_1 \times \Omega_2) = \mu(\Omega_1) = \nu(\Omega_2) = 1$, so $\Pi(\mu,\nu) \subseteq \mathscr{P}(\Omega_1 \times \Omega_2)$. Now let $(\pi_n)_n \subset \Pi(\mu, \nu)$ be a minimizing sequence. By Theorem 2.1.12, there exists a subsequence $(\pi_{n_k})_k$ such that $\pi_{n_k} \overset{*}{\rightharpoonup} \pi^*$ for some $\pi^* \in \mathscr{P}(\Omega_1 \times \Omega_2)$. Since $\Pi(\mu, \nu)$ is closed by the above lemma, $\pi^* \in \Pi(\mu, \nu)$. By definition, $W(\pi) = \int_{\Omega_1} c(x,y) \mathrm{d}\pi(x,y)$ is continuous with respect to weak-* convergence, because $c \in C(\Omega_1 \times \Omega_2)$. Consequently we get $W(\pi_{n_k}) \to W(\pi^*)$, so π^* is a minimizer. ∎

The above theorem however does not guarantee uniqueness of the optimal transport plan. For example, if the cost function c is constant, every transport plan is optimal.

Using the duality of measures and continuous functions, the following dual formulation of the Kantorovich problem (KP) can be derived.

3.1.6. Theorem. Let Ω_1 and Ω_2 be two compact metric spaces, let $\mu \in \mathscr{P}(\Omega_1)$ and $\nu \in \mathscr{P}(\Omega_2)$ be two probability measures and let $c : \Omega_1 \times \Omega_2 \to \mathbb{R}_+$ be a continuous cost function $c \in C(\Omega_1 \times \Omega_2)$. Then the optimal value of (KP) with cost function c is equal to the optimal value of

$$\max_{\substack{v \in C(\Omega_1), \\ w \in C(\Omega_2)}} \left\{ \int_{\Omega_1} v \mathrm{d}\mu + \int_{\Omega_2} w \mathrm{d}\nu \ : \ v \oplus w \leq c \right\}, \tag{DP}$$

where $v \oplus w : \Omega_1 \times \Omega_2 \to \mathbb{R}$, $(v \oplus w)(x,y) = v(x) + w(y)$.

This dual problem is formulated for example in [30, §1.2]. A proof of the result can be found in [30, §1.6.3].

The case where $\Omega := \Omega_1 = \Omega_2$ and the cost function $c : \Omega \times \Omega \to \mathbb{R}$ is of the form $c(x,y) = d(x,y)^p$ for $p \in [1, \infty)$ is of particular interest to us. Note that the metric d defines a continuous function $d : \Omega \times \Omega \to \mathbb{R}$. We then define the *Wasserstein Distance*.

3.1.7. Definition. Let (Ω, d) be a compact metric space. We call the metric d the *ground metric*. The *Wasserstein Distance of order p* between two measures $\mu \in \mathscr{P}(\Omega)$ and $\nu \in \mathscr{P}(\Omega)$ is defined for $p \in [1, \infty)$ through

$$\mathcal{W}_p(\mu, \nu)^p := \min_{\pi \in \Pi(\mu, \nu)} \iint_{\Omega \times \Omega} d(x,y)^p \mathrm{d}\pi(x,y).$$

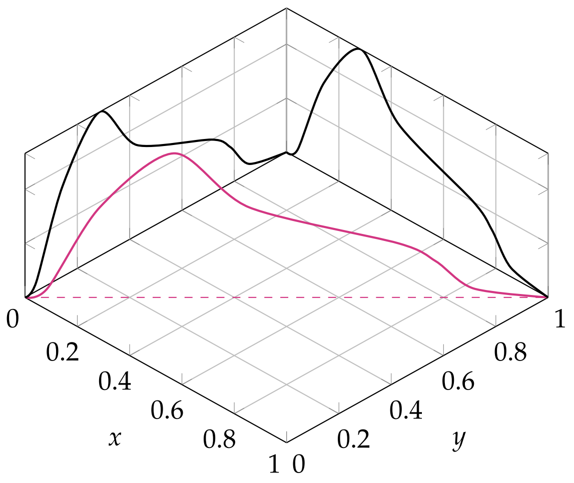


Figure 3.3: The transport plan between two identical measures is concentrated on the diagonal. Measures are visualized as densities. Note that the diagonal (dotted line) is actually a null-set with respect to the Lebesgue measure.

3.1.8. Theorem. Let Ω be a compact metric space. Then the Wasserstein Distance of order p with $p \in [1, \infty)$ defines a metric on $\mathcal{P}(\Omega)$.

While this result is stated for probability measures, it still holds true if a set of measures with arbitrary fixed volume is considered instead. A proof can be found in [31, §I.6]. As an example, we show that the Wasserstein distance between identical measures is zero.

3.1.9. Example. Let $\mu \in \mathcal{P}(\Omega)$. We want to calculate $\mathcal{W}_p(\mu, \mu)^p$. Intuitively, it should be possible to transport mass solely on the diagonal, i. e. "$\pi(\{x\} \times \{x\}) = \mu(\{x\})$" and π has no mass outside of the diagonal, see Figure 3.3. This can be formalized in the following way. Let $\Delta : \Omega \to \Omega \times \Omega,\ t \mapsto (t, t)$ be the diagonal map. Then we define $\pi_\Delta = \Delta^\# \mu \in \mathcal{M}_+(\Omega \times \Omega)$. The first marginal is $\mathrm{pr}_1^\# \pi_\Delta = \mathrm{pr}_1^\# \Delta^\# \mu = (\mathrm{pr}_1 \circ \Delta)^\# \mu = \mu$, since $\mathrm{pr}_1 \circ \Delta$ is the identity map. Analogously, $\mathrm{pr}_2^\# \pi_\Delta = \mu$, i.e. both marginals of π_Δ are indeed μ, so π_Δ is admissible. Now

$$\begin{aligned}\iint_{\Omega\times\Omega} d(x,y)^p \mathrm{d}\pi_\Delta(x,y) &= \iint_{\Omega\times\Omega} d(x,y)^p \mathrm{d}\Delta^\#\mu(x,y) = \int_\Omega d(\Delta(x))^p \mathrm{d}\mu(x)\\ &= \int_\Omega d(x,x)^p \mathrm{d}\mu(x) = 0.\end{aligned}$$

But $\iint_{\Omega\times\Omega} d(x,y)^p \mathrm{d}\pi(x,y) \geq 0$ for all $\pi \in \mathcal{M}_+(\Omega \times \Omega)$, so π_Δ is optimal.

For $p = 1$, i. e. when the cost function c is a distance function d, the dual formulation can be simplified, see [30, §3.1.1].

3.1.10. Theorem (Kantorovich-Rubinstein duality). Let (Ω, d) be a compact metric space, and let $\mu \in \mathcal{P}(\Omega)$ and $\nu \in \mathcal{P}(\Omega)$ be two probability measures. Then

$$\mathcal{W}_1(\mu, \nu) = \max_{v \in C(\Omega)} \left\{ \int_\Omega v(x) \mathrm{d}(\mu - \nu)(x) \ :\ v \text{ is 1-Lipschitz continuous} \right\}.$$

A consequence of this formula is that the 1-Wasserstein distance depends only on the difference of the two measures.

3.1.11. Corollary. Let (Ω, d) be a compact metric space, and let $\mu, \mu' \in \mathcal{P}(\Omega)$ and $\nu, \nu' \in \mathcal{P}(\Omega)$ be measures with $\mu - \nu = \mu' - \nu'$. Then

$$\mathcal{W}_1(\mu, \nu) = \mathcal{W}_1(\mu', \nu').$$

Proof. Using Theorem 3.1.10, we have

$$\begin{aligned}\mathcal{W}_1(\mu,\nu) &= \max_{v\in C(\Omega)}\left\{\int_\Omega v(x)\mathrm{d}(\mu-\nu)(x) \;:\; v \text{ is 1-Lipschitz continuous}\right\}\\ &= \max_{v\in C(\Omega)}\left\{\int_\Omega v(x)\mathrm{d}(\mu'-\nu')(x) \;:\; v \text{ is 1-Lipschitz continuous}\right\}\\ &= \mathcal{W}_1(\mu',\nu'),\end{aligned}$$

which proves the claim. ■

If $\mu-\nu = \mu'-\nu'$, this means that $\mu' = \mu+\eta$ and $\nu' = \nu+\eta$ with $\eta := \mu-\mu' = \nu-\nu'$. In terms of transport plans, one can then imagine that we get a transport plan between μ' and ν' from a transport plan between μ and ν by transporting the additional mass of η on the diagonal, which does not add any additional transport costs.

The following examples give an intuition for how the Wasserstein distance works.

3.1.12. Example. Let Ω be a compact metric space and let $a, b \in \Omega$. We want to calculate $\mathcal{W}_p(\delta_a,\delta_b)$. The only transport plan in $\Pi(\mu,\nu)$ is $\delta_a\otimes\delta_b$, which means that the mass in a is transported to b. We thus have

$$\mathcal{W}_p(\delta_a,\delta_b)^p = \iint_{\Omega\times\Omega} d^p(x,y)\mathrm{d}(\delta_a\otimes\delta_b)(x,y) = d^p(a,b).$$

3.1.13. Example. Let $\Omega = \mathbb{R}$ and let μ_1 and μ_2 be two Gaussian measures with standard deviations σ_1 and σ_2 and means m_1 and m_2. Then the 2-Wasserstein distance between μ_1 and μ_2 is

$$\mathcal{W}_2(\mu_1,\mu_2)^2 = (m_1-m_2)^2 + (\sigma_1-\sigma_2)^2,$$

see [32].

3.2 An Extension of the Wasserstein Distance for Arbitrary Measures

In the previous section we have defined the Wasserstein distance, which gives a metric on the set of probability measures. Since we want to use the Wasserstein distance as a metric between MR brain region images, whose volume can be subject to individual variation and which can thus not be interpreted as probability measures, we want to extend the Wasserstein distance to arbitrary measures. One possibility to deal with this problem is to relax the marginal constraints by replacing the equalities by inequalities, see [33] and [34]. However, this does not keep the metric properties of the Wasserstein distance. An approach that gives a metric was introduced by Kantorovich [35] and is further discussed in [36]. It is shown in [37] that this approach can be cast as a classical optimal transport problem, which means that we only need to develop tools to solve one problem. This approach is also successfully used in [38] with application to averaging neuroimaging data, which suggests that it might be applicable in our case as well. A very similar construction has also been introduced by [39] to compare SIFT histograms.

We will first introduce the approach by Kantorovich, then the reformulation into the *unbalanced mass transportation problem* introduced by [37], which is shown to be reducible to the classical mass transportation problem. Finally we will show that these formulations are equivalent.

3.2.1 The Kantorovich Extension

The extension proposed by Kantorovich is only possible for the 1-Wasserstein distance. It relies on the fact that the 1-Wasserstein distance can be extended into a norm, which can then further be extended into a norm on the whole space of measures. We will first introduce this norm, using [36] as a reference, and then the extension, using [37] as a reference.

Let $\mu_1, \mu_2 \in \mathcal{P}(\Omega)$. Our goal is to define a norm, such that $\mathcal{W}_1(\mu_1, \mu_2) = \|\mu_1 - \mu_2\|$. The condition $\mu_1(\Omega) = \mu_2(\Omega) = 1$ implies that $(\mu_1 - \mu_2)(\Omega) = 0$, thus this norm has to be defined only for measures ν with $\nu(\Omega) = 0$. Let ν be such a measure. From the Hahn decomposition (see Theorem 2.1.4) we know that $\nu = \nu^+ - \nu^-$. Similar to the Kantorovich problem, we can define the set

$$\Pi_\nu := \left\{\pi \in \mathcal{M}_+(\Omega \times \Omega) : \operatorname{pr}_1^\# \pi = \nu^+, \operatorname{pr}_2^\# \pi = \nu^-\right\} \tag{3.4}$$

as the set of transport plans between ν^+ and ν^-. The norm is the defined as follows.

3.2.1. Definition (Kantorovich Norm). Let (Ω, d) be a compact metric space. Let $\nu \in \mathcal{M}(\Omega)$ with $\nu(\Omega) = 0$. Then the *Kantorovich norm* is

$$\|\nu\|_d^0 = \inf_{\pi \in \Pi_\nu} \iint_{\Omega \times \Omega} d(x, y) \mathrm{d}\pi(x, y), \tag{3.5}$$

with Π_ν as in (3.4).

Note that this is exactly (KP) with marginals ν^+ and ν^- and d as a cost function. The existence of a minimizer thus follows from Theorem 3.1.5. That this definition yields indeed a norm is shown in [36, §1]. Note that since this is exactly (KP), one could replace d by d^p or some other cost function in the definition. We now want to show that $\|\mu - \nu\|_d^0 = \mathcal{W}_1(\mu, \nu)$. This is (almost) clear from the definition.

3.2.2. Proposition. Let (Ω, d) be a compact metric space, and let $\mu, \nu \in \mathcal{M}_+(\Omega)$ with $\|\mu\| = \|\nu\|$. Then

$$\mathcal{W}_1(\mu, \nu) = \|\mu - \nu\|_d^0.$$

Proof. We have

$$\begin{aligned} \|\mu - \nu\|_d^0 &= \inf_{\pi \in \Pi_{\mu-\nu}} \iint_{\Omega \times \Omega} d(x, y) \mathrm{d}\pi(x, y) \\ &= \inf_{\pi \in \Pi((\mu-\nu)^+, (\mu-v)^-)} \iint_{\Omega \times \Omega} d(x, y) \mathrm{d}\pi(x, y) \\ &= \mathcal{W}_1((\mu - \nu)^+, (\mu - \nu)^-). \end{aligned}$$

We now want to show that this is equal to $\mathcal{W}_1(\mu, \nu)$. We have

$$(\mu - \nu)^+ - (\mu - \nu)^- = \mu - \nu,$$

see (2.2). Now it follows from Corollary 3.1.11 that the 1-Wasserstein distance only depends on the difference of the measures, so

$$\mathcal{W}_1((\mu - \nu)^+, (\mu - \nu)^-) = \mathcal{W}_1(\mu, \nu),$$

which proves the claim. ■

We now want to extend this norm on the whole space $\mathcal{M}(X)$. We introduce a *waste penalty function* $p : \Omega \to \mathbb{R}$ satisfying the properties

$$|p(x) - p(y)| \leq d(x,y), \tag{3.6}$$

and

$$p(x) \geq \alpha \text{ for some } \alpha \geq 0. \tag{3.7}$$

We then define the extended norm.

3.2.3. Definition (K-norm). Let (Ω, d) be a compact metric space. Let $p : \Omega \to \mathbb{R}$ be a function satisfying the properties (3.6) and (3.7). For $\mu \in \mathcal{M}(\Omega)$, the *K-norm* is defined as

$$\|\mu\|_p = \inf\left\{\|\nu\|_d^0 + \int_\Omega p(x)\mathrm{d}|\mu - \nu|(x) \;:\; \nu \in \mathcal{M}(X), \nu(\Omega) = 0\right\}. \tag{3.8}$$

According to [37, Thm. 2.2], this definition yields a norm. The function p plays the role of a waste penalty function, since only the mass of ν is transferred and the mass $|\mu - \nu|$ is wasted, which is penalized by p. However, this norm is not a true extension of the Kantorovich norm in the sense that if $\mu(\Omega) = 0$, we do not necessarily have $\|\mu\|_p = \|\mu\|_d^0$. This is because the condition $p(x) \geq \alpha$ allows for cases where it is "cheaper" to waste mass instead of transferring it. This is demonstrated in the following example (cf. [37, §2.0.2]).

3.2.4. Example. Let $\Omega \subseteq \mathbb{R}^2$ and take $x, y \in \Omega$ with $d(x,y) = \|x - y\|_2 > 2$. Let $\mu = \delta_x - \delta_y$ and let $p \equiv 1$. To calculate $\|\mu\|_p$, we first consider the option of simply wasting everything by taking $\nu = 0$. This results in costs of

$$\int_\Omega 1\mathrm{d}|\delta_x - \delta_y| = \int_{\Omega\setminus\{y\}} 1\mathrm{d}|\delta_x| + \int_{\{y\}} 1\mathrm{d}|-\delta_y| = 1 + 1 = 2.$$

Another option would be to transfer everything, i. e. taking $\nu = \mu$. However, since $d(x,y) > 2$, transferring the mass from x to y costs more than 2, i. e.

$$\|\nu\|_d^0 + \int_\Omega p(x)\mathrm{d}|\mu - \nu|(x) = \|\mu\|_d^0 = d(x,y) > 2.$$

This shows that in this case $\|\mu\|_p < \|\mu\|_d^0$.

The above example shows that in order for the norms to be identical for measures with zero total mass, it should be more expensive to waste mass than to transfer it. This means that p should fulfill the condition

$$p(x) > \max_{x,y\in\Omega} d(x,y). \tag{3.9}$$

Indeed, if p satisfies (3.6) and (3.9), the K-norm is a true extension of the Kantorovich norm, i. e. $\|\mu\|_p = \|\mu\|_d^0$ for all measures μ with $\mu(\Omega) = 0$, see [37, Lemma 2.1]. This extension was originally proposed by Kantorovich (cf. [35]). If p is taken to be constant, the weaker condition

$$p > \frac{1}{2}\max_{x,y\in\Omega} d(x,y) \tag{3.10}$$

is also a sufficient condition, which follows from [37, Lemma 2.3].

Finally, a distance between two arbitrary measures $\mu, \nu \in \mathcal{M}_+(\Omega)$ can be calculated as $\|\mu - \nu\|_p$.

3.2.2 The Unbalanced Mass Transportation Problem

While the above construction is only possible for the 1-Wasserstein distance, the unbalanced mass transportation problem can be formulated for arbitrary cost functions.

3.2.5. Definition (Unbalanced Mass Transportation Problem). Let (Ω, d) be a compact measurable space. Let $p : \Omega \to \mathbb{R}$ be a continuous function satisfying the properties (3.6) and (3.7). For $\mu, \nu \in \mathcal{M}_+(\Omega)$ and a continuous cost function $c : \Omega \times \Omega \to \mathbb{R}_{\geq 0}$, the *unbalanced mass transportation problem (UMTP)* for μ and ν is defined as

$$\begin{aligned} \inf_{\pi \in \mathcal{M}_+(\Omega\times\Omega)} \iint_{\Omega\times\Omega} c(x,y)\mathrm{d}\pi(x,y) + \int_\Omega p(x)\mathrm{d}(\mu - \mathrm{pr}_1^\# \pi)(x) \\ + \int_\Omega p(x)\mathrm{d}(\nu - \mathrm{pr}_2^\# \pi)(x) \qquad \text{(UMTP)} \\ \text{subject to } \mathrm{pr}_1^\# \pi \leq \mu, \mathrm{pr}_2^\# \pi \leq \nu. \end{aligned}$$

The intuition behind this definition is the following. Since μ and ν do not have the same total mass, it is not possible to find a transport plan that has exactly the marginals μ and ν. Instead, we look for a transport plan whose marginals are smaller than the given measures μ and ν. For example, such a transport plan could be a transport plan between the minimum of the two measures. This would mean that the total mass of the difference of μ and the minimum as well as the total mass of the difference of ν and the minimum is wasted, which is penalized by the latter two terms, using the waste function p. A similar problem has been formulated in a discrete setting in [37]. They then noticed that this can be cast as a classical transportation problem. The same is true in our setting. The idea is to interpret the “wasting” of mass as transporting this mass to an additional remote point x^*, where the cost of transporting mass to the remote point is the same as wasting it. To formalize this idea, we first need to extend our domain Ω. Let

$$\tilde{\Omega} = \Omega \cup \{x^*\}.$$

We can make $\tilde{\Omega}$ into a metric space by defining

$$\tilde{d} : \tilde{\Omega} \times \tilde{\Omega} \to \mathbb{R}, \tilde{d}(x,y) = \begin{cases} d(x,y), & \text{if } x,y \in \Omega \\ C, & \text{if } x = x^* \text{ or } y = x^* \\ 0, & \text{if } x = y = x^*, \end{cases}$$

for $C > \frac{1}{2}\sup_{x,y\in\Omega} d(x,y)$. One easily checks that $(\tilde{\Omega}, \tilde{d})$ is a metric space, see the proof of Proposition 3.2.7. Next we want to define two measures $\tilde{\mu}$ and $\tilde{\nu}$ that have the same mass. Denote by $\iota : \Omega \to \tilde{\Omega}$ the inclusion map. Let M be a constant such that $M > \max(\mu(\Omega), \nu(\Omega))$. Then we define

$$\begin{aligned} \tilde{\mu} &:= \iota^\# \mu + (M - \mu(\Omega)) \cdot \delta_{x^*}, \\ \tilde{\nu} &:= \iota^\# \nu + (M - \nu(\Omega)) \cdot \delta_{x^*}, \end{aligned} \qquad (3.11)$$

which means that $\tilde{\mu}$ is μ on Ω and the point x^* has mass $M - \mu(\Omega)$, and similarly for ν.

This construction assures that $\tilde{\mu}(\tilde{\Omega}) = \tilde{\nu}(\tilde{\Omega})$, since

$$\begin{aligned}\tilde{\mu}(\tilde{\Omega}) &= \iota^{\#}\mu(\tilde{\Omega}) + (M - \mu(\Omega)) \cdot \delta_{x^*}(\tilde{\Omega}) = \mu(\iota^{-1}(\tilde{\Omega})) + (M - \mu(\Omega)) \\ &= \mu(\Omega) + M - \mu(\Omega) = M\end{aligned}$$

and in the same way

$$\tilde{\nu}(\tilde{\Omega}) = M.$$

Let further

$$\tilde{c} : \tilde{\Omega} \times \tilde{\Omega} \to \mathbb{R}, \tilde{c}(x,y) = \begin{cases} c(x,y), & \text{if } x,y \in \Omega \\ p(x), & \text{if } x = x^*, y \neq x^* \\ p(y), & \text{if } x \neq x^*, y = x^* \\ 0, & \text{if } x = y = x^*. \end{cases}$$

The set $\{x^*\}$ is open and closed, since

$$\{x^*\} = \left\{ x \in \tilde{\Omega} \,:\, d(x,x^*) \leq \frac{1}{2}C \right\} = B_{\frac{1}{2}C}^{-}(x^*) = B_{\frac{1}{2}C}(x^*).$$

Thus, Ω is closed. This means that $\Omega \times \Omega$, $\Omega \times \{x^*\}$ and $\{x^*\} \times \Omega$ are closed. Since c and p are continuous, this implies that $\tilde{c}$ is continuous.

Thus we can now consider the Kantorovich problem

$$\inf\left\{ \iint_{\tilde{\Omega}\times\tilde{\Omega}} \tilde{c}(x,y) \mathrm{d}\tilde{\pi}(x,y) \,:\, \tilde{\pi} \in \Pi(\tilde{\mu},\tilde{\nu}) \right\}. \tag{$\widetilde{\text{KP}}$}$$

The following lemma states that the Kantorovich problem using the above extension yields the same value as the unbalanced mass transportation problem.

3.2.6. Lemma. Let (Ω, d) be a compact metric space and let $\mu, \nu \in \mathcal{M}_+(\Omega)$. Then the optimal cost of ($\widetilde{\text{KP}}$) is equal to the optimal cost of (UMTP).

Proof. We show that the optimal costs are equal by showing both inequalities. For better understanding, an illustration can be seen in Figure 3.4 (measures are visualized as densities). First we show inf ($\widetilde{\text{KP}}$) $\geq$ inf (UMTP) by showing that for every $\tilde{\pi}$ which is admissible for ($\widetilde{\text{KP}}$) there exists a π admissible for (UMTP) such that the costs for both problems are equal. We claim that such a π can be constructed as $\pi := \tilde{\pi}|_{\Omega\times\Omega}$. Intuitively, this means we have to show that the mass that $\tilde{\pi}$ transports outside of $\Omega \times \Omega$, i. e. on the slices $\Omega \times \{x^*\}$ and $\{x^*\} \times \Omega$, is equal to the mass that is wasted in (UMTP), i. e. the differences $\mu - \mathrm{pr}_1^{\#}\pi$ and $\nu - \mathrm{pr}_2^{\#}\pi$, respectively. This follows from the definition of the marginal conditions on $\tilde{\pi}$. We formalize this intuition.
Let $\tilde{\pi}$ be admissible for ($\widetilde{\text{KP}}$) and let $\pi := \tilde{\pi}|_{\Omega\times\Omega}$. We first show that π is admissible for (UMTP) by checking the marginal inequality conditions. For any measurable $A \subseteq \Omega$ we have

$$\mathrm{pr}_1^{\#}\pi(A) = \pi(A \times \Omega) = \tilde{\pi}(A \times \Omega) \leq \tilde{\pi}(A \times \tilde{\Omega}) = \mathrm{pr}_1^{\#}\tilde{\pi}(A) = \tilde{\mu}(A) = \mu(A),$$

and in the same way,

$$\mathrm{pr}_1^{\#}\pi(A) \leq \nu(A).$$

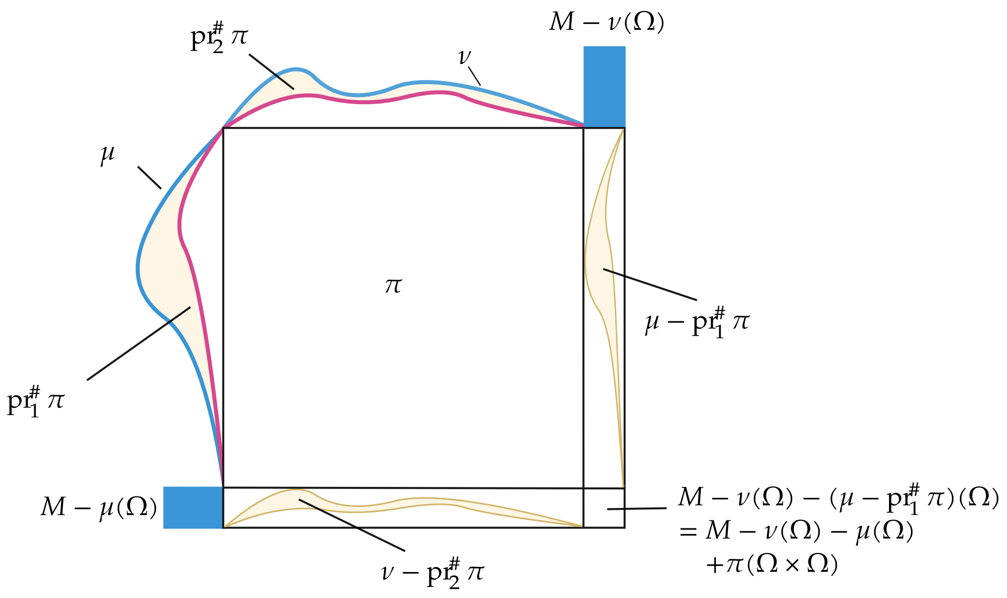

$\mathrm{pr}_2^\# \pi$
ν
$M - \nu(\Omega)$
μ
π
$\mu - \mathrm{pr}_1^\# \pi$
$\mathrm{pr}_1^\# \pi$
$M - \mu(\Omega)$
$M - \nu(\Omega) - (\mu - \mathrm{pr}_1^\# \pi)(\Omega)$
$= M - \nu(\Omega) - \mu(\Omega)$
$+\pi(\Omega \times \Omega)$
$\nu - \mathrm{pr}_2^\# \pi$

Figure 3.4: Illustration of relation between UMTP and $\widetilde{\text{KP}}$. The original marginals μ and ν are shown in blue. $\Omega \times \Omega$ corresponds to the white square. The new marginals $\tilde{\mu}$ and $\tilde{\nu}$ are constructed by adding mass to a remote point. This mass is depicted as blue bars. The marginals of π are shown in pink. The difference between the marginals of π and the target marginals is colored in beige. This difference is transported on the slices $\Omega \times \{x^*\}$ and $\{x^*\} \times \Omega$ in $\widetilde{\text{KP}}$.

Now, by splitting $\tilde{\Omega} \times \tilde{\Omega}$ into four parts, we get that the costs are

$$\begin{aligned}
\iint_{\tilde{\Omega}\times\tilde{\Omega}} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) &= \iint_{\Omega\times\Omega} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) + \iint_{\{x^*\}\times\Omega} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) \\
&\quad + \iint_{\Omega\times\{x^*\}} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) + \iint_{\{x^*\}\times\{x^*\}} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) \\
&= \iint_{\Omega\times\Omega} c(x,y)\mathrm{d}\pi(x,y) + \iint_{\{x^*\}\times\Omega} p(y)\mathrm{d}\tilde{\pi}(x,y) \\
&\quad + \iint_{\Omega\times\{x^*\}} p(x)\mathrm{d}\tilde{\pi}(x,y).
\end{aligned}$$

We have

$$\begin{aligned}
\iint_{\Omega\times\{x^*\}} p(x)\mathrm{d}\tilde{\pi}(x,y) &= \iint_{\Omega\times\tilde{\Omega}} p(x)\mathrm{d}\tilde{\pi}(x,y) - \iint_{\Omega\times\Omega} p(x)\mathrm{d}\tilde{\pi}(x,y) \\
&= \int_\Omega p(x)\mathrm{d}\,\mathrm{pr}_1^\# \tilde{\pi}(x) - \iint_{\Omega\times\Omega} p(x)\mathrm{d}\pi(x,y) \\
&= \int_\Omega p(x)\mathrm{d}\tilde{\mu}(x) - \int_\Omega p(x)\mathrm{d}\,\mathrm{pr}_1^\# \pi(x) \\
&= \int_\Omega p(x)\mathrm{d}(\mu - \mathrm{pr}_1^\# \pi)(x).
\end{aligned}$$

Analogously,

$$\iint_{\{x^*\}\times\Omega} p(y)\mathrm{d}\tilde{\pi}(x,y) = \int_\Omega p(y)\mathrm{d}(\nu - \mathrm{pr}_2^\# \pi)(y).$$

Finally we get

$$\begin{aligned}
\iint_{\tilde{\Omega}\times\tilde{\Omega}} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) = \iint_{\Omega\times\Omega} c(x,y)\mathrm{d}\pi(x,y) + \int_\Omega p(x)\mathrm{d}(\mu - \mathrm{pr}_1^\# \pi)(x) \\
+ \int_\Omega p(x)\mathrm{d}(\nu - \mathrm{pr}_2^\# \pi)(x),
\end{aligned}$$

which shows that the costs of π for (UMTP) and the costs of $\tilde{\pi}$ for ($\widetilde{\text{KP}}$) are equal.

Now for the other inequality, we construct a transport plan $\tilde{\pi}$ from π. The idea is that $\tilde{\pi}$ behaves like π on $\Omega \times \Omega$, and transports the differences $\mu - \mathrm{pr}_1^\# \pi$ and $\nu - \mathrm{pr}_2^\# \pi$ on the respective slices. Because of the marginal condition, $\tilde{\pi}$ has to transport a total mass of $M - \nu(\Omega)$ and $M - \mu(\Omega)$ on the respective slices. In order to reach this total mass, an additional mass of

$$\begin{aligned} C &:= M - \nu(\Omega) - \big(\mu(\Omega) - \mathrm{pr}_1^\# \pi(\Omega)\big) \\ &= M - \mu(\Omega) - \nu(\Omega) + \pi(\Omega \times \Omega) \\ &= M - \mu(\Omega) - \big(\nu(\Omega) - \mathrm{pr}_2^\# \pi(\Omega)\big) \end{aligned}$$

has to be transported from x^* to x^*, see also Figure 3.4. To define $\tilde{\pi}$ on the slices, let

$$\begin{aligned} \iota_1 &: \Omega \to \tilde{\Omega} \times \tilde{\Omega},\ x \mapsto (x, x^*), \\ \iota_2 &: \Omega \to \tilde{\Omega} \times \tilde{\Omega},\ x \mapsto (x^*, x), \\ \iota &: \Omega \to \tilde{\Omega},\ x \mapsto x \end{aligned}$$

Then we define

$$\tilde{\pi} = (\iota \times \iota)^\# \pi + \iota_1^\# (\mu - \mathrm{pr}_1^\# \pi) + \iota_2^\# (\nu - \mathrm{pr}_2^\# \pi) + C \cdot \delta_{(x^*, x^*)}.$$

Since $\mathrm{pr}_1^\# \pi \leq \mu$ and $\mathrm{pr}_2^\# \pi \leq \nu$, $\tilde{\pi} \geq 0$. Now we can calculate the marginals for each part. We have, using properties of pushforward,

$$\mathrm{pr}_1^\# ((\iota \times \iota)^\# \pi) = (\mathrm{pr}_1 \circ (\iota \times \iota))^\# \pi.$$

For the second term, using the identities $\iota_1 \circ \mathrm{pr}_1 = (\iota \times \iota)$ and $\mathrm{pr}_1 \circ \iota_1 = \iota$,

$$\begin{aligned} \mathrm{pr}_1^\# (\iota_1^\# (\mu - \mathrm{pr}_1^\# \pi)) &= (\mathrm{pr}_1 \circ \iota_1)^\# \mu - (\mathrm{pr}_1 \circ \iota_1 \circ \mathrm{pr}_1)^\# \pi \\ &= \iota^\# \mu - (\mathrm{pr}_1 \circ (\iota \times \iota))^\# \pi. \end{aligned}$$

For the third term, note that $\mathrm{pr}_1 \circ \iota_2$ is the map sending every $x \in \Omega$ to x^*. Thus,

$$\begin{aligned} (\mathrm{pr}_1 \circ \iota_2)^\# \eta(A) &= \begin{cases} \eta(\Omega), & \text{if } x^* \in A \\ \eta(\emptyset) = 0, & \text{else} \end{cases} \\ &= \eta(\Omega) \cdot \delta_{x^*}(A) \end{aligned}$$

for any measure $\eta \in \mathcal{M}_+(\Omega)$ and any measurable set $A \subseteq \tilde{\Omega}$. We get

$$\mathrm{pr}_1^\# (\iota_2^\# (\nu - \mathrm{pr}_2^\# \pi)) = (\mathrm{pr}_1 \circ \iota_2)^\# (\nu - \mathrm{pr}_2^\# \pi) = \nu(\Omega) \cdot \delta_{x^*} - \mathrm{pr}_2^\# \pi(\Omega) \cdot \delta_{x^*}.$$

Lastly, $\mathrm{pr}_1^\# \delta_{(x^*, x^*)} = \delta_{x^*}$. Putting everything together, we get

$$\begin{aligned} \mathrm{pr}_1^\# \tilde{\pi} &= (\mathrm{pr}_1 \circ (\iota \times \iota))^\# \pi + \iota^\# \mu - (\mathrm{pr}_1 \circ (\iota \times \iota))^\# \pi + \nu(\Omega) \cdot \delta_{x^*} - \mathrm{pr}_2^\# \pi(\Omega) \cdot \delta_{x^*} + C \cdot \delta_{x^*} \\ &= \iota^\# \mu + \nu(\Omega) \cdot \delta_{x^*} - \pi(\Omega \times \Omega) \cdot \delta_{x^*} + (\pi(\Omega \times \Omega) + M - \nu(\Omega) - \mu(\Omega)) \cdot \delta_{x^*} \\ &= \iota^\# \mu + (M - \mu(\Omega)) \cdot \delta_{x^*} \\ &= \tilde{\mu}. \end{aligned}$$

Analogously,

$$\mathrm{pr}_2^\# \tilde{\pi} = \tilde{\nu}.$$

Thus the marginal conditions are satisfied and $\tilde{\pi} \in \Pi(\tilde{\mu}, \tilde{\nu})$ is admissible. Now, for the costs we have

$$\begin{aligned}
\iint_{\tilde{\Omega}\times\tilde{\Omega}} \tilde{c}(x,y)\mathrm{d}\tilde{\pi}(x,y) = {} & \iint_{\tilde{\Omega}\times\tilde{\Omega}} \tilde{c}(x,y)\mathrm{d}\big(\ (\iota\times\iota)^\#\pi + \iota_1^\#(\mu - \mathrm{pr}_1^\#\pi) + \iota_2^\#(\nu - \mathrm{pr}_2^\#\pi) \\
& \qquad + \pi(\Omega\times\Omega)\cdot\delta_{(x^*,x^*)}\big)\ (x,y) \\
\overset{(1)}{=} {} & \iint_{\Omega\times\Omega} \tilde{c}(\iota(x),\iota(y))\mathrm{d}\pi(x,y) + \iint_{\Omega} \tilde{c}(\iota_1(x))\mathrm{d}(\mu - \mathrm{pr}_1^\#\pi)(x) \\
& \quad + \iint_{\Omega} \tilde{c}(\iota_2(x))\mathrm{d}(\nu - \mathrm{pr}_2^\#\pi)(x) + \tilde{c}(x^*,x^*)\pi(\Omega\times\Omega) \\
\overset{(2)}{=} {} & \iint_{\Omega\times\Omega} c(x,y)\mathrm{d}\pi(x,y) + \iint_{\Omega} p(x)\mathrm{d}(\mu - \mathrm{pr}_1^\#\pi)(x) \\
& + \iint_{\Omega} p(x)\mathrm{d}(\nu - \mathrm{pr}_2^\#\pi)(x),
\end{aligned}$$

where (1) follows from the change of variables formula for pushforward and (2) follows from $\tilde{c}(x^*, x^*) = 0$. We thus arrived at the costs for (UMTP). ■

Now, if the costs $\tilde{c}$ define a metric, ($\widetilde{\mathrm{KP}}$) is the 1-Wasserstein distance, which means that ($\widetilde{\mathrm{KP}}$) and thus (UMTP) defines a metric. This is the case in the following setting.

3.2.7. Proposition. Let (Ω, d) be a compact metric space and let $\mu, \nu \in \mathcal{M}_+(\Omega)$. Let $p : \Omega \to \mathbb{R}_{>0}$ be a waste function satisfying (3.6) and (3.10). Then the cost function

$$\tilde{d}(x,y) = \begin{cases} d(x,y), & \text{if } x,y \in \Omega \\ p(x), & \text{if } \neq x^*, y = x^* \\ p(y), & \text{if } x = x^*, y \neq x^* \\ 0, & \text{if } x = x^* = y \end{cases} \tag{3.12}$$

defines a metric on $\tilde{\Omega}$. and the UMTP defines a metric on $\mathcal{M}_+(\Omega)$.

Proof. First, we prove that $\tilde{d}$ defines a metric. It is immediate that $\tilde{d} \geq 0$ and $\tilde{d}(x,y) = 0$ if and only if $x = y$ for all $x, y \in \tilde{\Omega}$. The symmetry for $x, y \in \Omega$ follows because d is a metric. For $x = x^*, y \neq x^*$ we have $\tilde{d}(x^*, y) = p(y) = \tilde{d}(y, x^*)$. Finally, the triangle inequality is satisfied for $x, y, z \in \Omega$, i. e. $d(x,z) \leq d(x,y) + d(y,z)$. Otherwise, we have to consider the cases where one of x, y, z is x^*. If $z = x^*$,

$$d(x,x^*) \leq d(x,y) + d(y,x^*) \Leftrightarrow d(x,x^*) - d(y,x^*) \leq d(x,y) \Leftrightarrow p(x) - p(y) \leq d(x,y),$$

which follows from (3.6). By symmetry,

$$d(x^*,x) \leq d(x^*,y) + d(y,x)$$

holds as well. Further, if $y = x^*$, we have

$$d(x,z) \leq d(x,x^*) + d(x^*,z) \Leftrightarrow d(x,z) \leq p(x) + p(z).$$

This is true because it follows from (3.10) that

$$p(x) + p(z) \geq 2 \cdot \frac{1}{2} \sup_{y,y'} d(y,y') \geq d(x,z).$$

This shows that $\tilde{d}$ is a metric on $\tilde{\Omega}$. From Lemma 3.2.6, we know that the optimal cost for (UMTP) is equal to the optimal cost of ($\widetilde{\text{KP}}$) with cost function $\tilde{d}$. By Theorem 3.1.8, the solution of ($\widetilde{\text{KP}}$) yields the 1-Wasserstein distance, which is a metric. We thus have that the optimal cost for UMTP is equal to the 1-Wasserstein distance between $\tilde{\mu}$ and $\tilde{\nu}$ with $\tilde{d}$ as ground metric. Denoting by $\widetilde{\mathcal{W}_1}(\mu,\nu)$ the optimal cost for the UMTP between μ and ν, this means

$$\widetilde{\mathcal{W}_1}(\mu,\nu) = \mathcal{W}_1(\tilde{\mu},\tilde{\nu}).$$

We can now derive the metric axioms from this equality. Symmetry, non-negativity and identity of indiscernibles are immediate from this formula. For the triangle inequality, if $\mu,\nu,\eta \in \mathcal{M}_+(\Omega)$, we just have to choose the same M in the definition of $\tilde{\mu},\tilde{\nu}$ and $\tilde{\eta}$. For example, we could choose $M = \max(\mu(\Omega),\nu(\Omega),\eta(\Omega))$. Then the triangle inequality is also immediate from the above formula. ■

3.2.3 Connection between UMTP and K-Norm

We will now concentrate again on the case where the costs are of the form $c = d$ for a metric d, since the Kantorovich Norm is only defined for this case. We additionally assume that the function p is constant. We then want to show that the optimal cost of the UMTP between two measures μ and ν is equal to the K-norm $\|\mu-\nu\|_p$, and thus the K-norm can be cast as an optimal transport problem. This is the content of the following theorem. The goal of this section is the proof of this theorem.

For a better overview, we restate the two problems first. For $\mu,\nu \in \mathcal{M}_+(\Omega)$, the unbalaced mass transportation problem (UMTP) is

$$\begin{aligned}\inf_{\pi\in\mathcal{M}_+(\Omega\times\Omega)} \iint_{\Omega\times\Omega} c(x,y)\mathrm{d}\pi(x,y) &+ \int_\Omega p(x)\mathrm{d}(\mu - \mathrm{pr}_1^{\#}\pi)(x) \\ &+ \int_\Omega p(x)\mathrm{d}(\nu - \mathrm{pr}_2^{\#}\pi)(x) \\ \text{subject to } \mathrm{pr}_1^{\#}\pi \le \mu, &\mathrm{pr}_2^{\#}\pi \le \nu.\end{aligned} \tag{UMTP}$$

Denote by $\mathcal{M}_0(\Omega)$ the set of all measures with zero total mass,

$$\mathcal{M}_0(\Omega) := \{\mu \in \mathcal{M}(\Omega) \,:\, \mu(\Omega) = 0\}.$$

For $\mu \in \mathcal{M}_0(\Omega)$, the K-norm is

$$\begin{aligned}\|\mu\|_p &= \inf_{\eta\in\mathcal{M}_0(\Omega)} \|\eta\|_d^0 + \int_\Omega p(x)\mathrm{d}|\mu-\eta|(x) \\ &= \inf_{\eta\in\mathcal{M}_0(\Omega)} \inf_{\pi\in\Pi_\eta} \iint_{\Omega\times\Omega} d(x,y)\mathrm{d}\pi(x,y) + \int_\Omega p(x)\mathrm{d}|\mu-\eta^+ + \eta^-|(x).\end{aligned}$$

We then have the following theorem, where we consider $c = d$ in the UMTP.

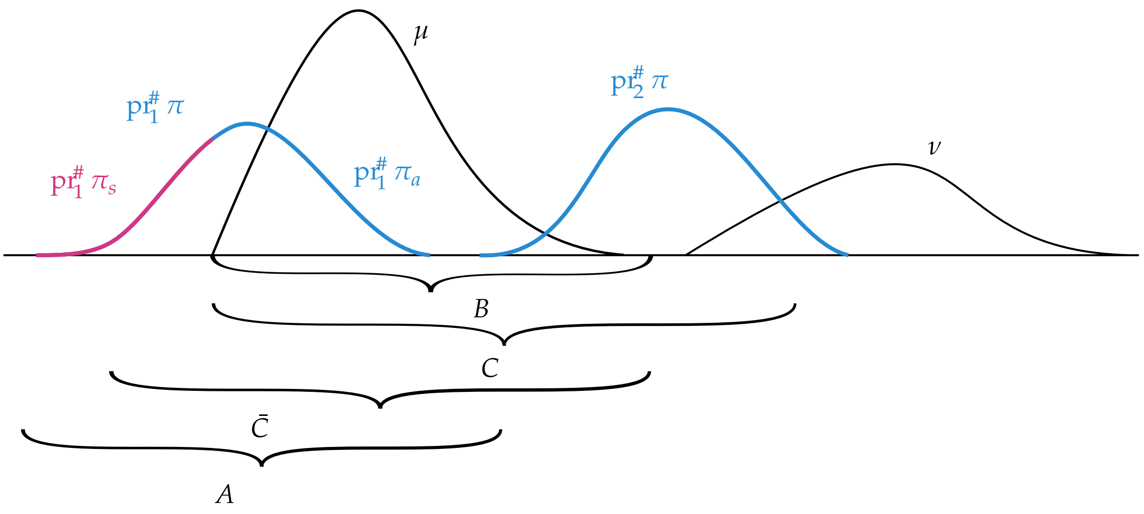


Figure 3.5: Illustration how the sets A, B, C and $\bar{C}$ could look like.

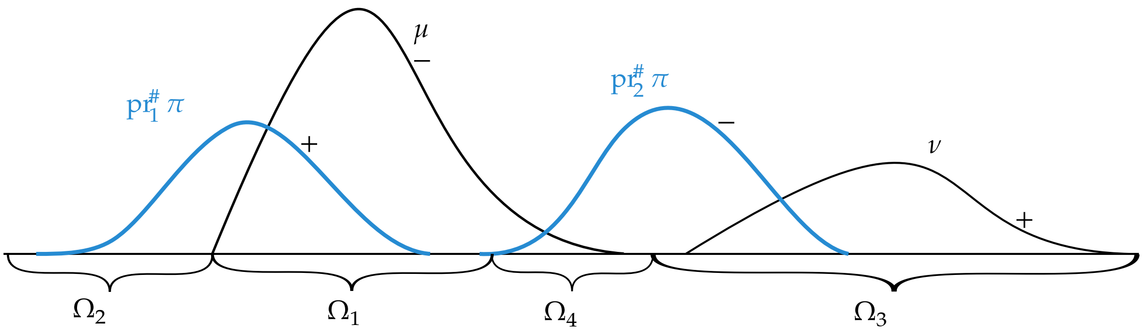


Figure 3.6: Illustration of the sets $\Omega_1, \Omega_2, \Omega_3, \Omega_4$ along with the signs with which the single densities appear in the K-norm.

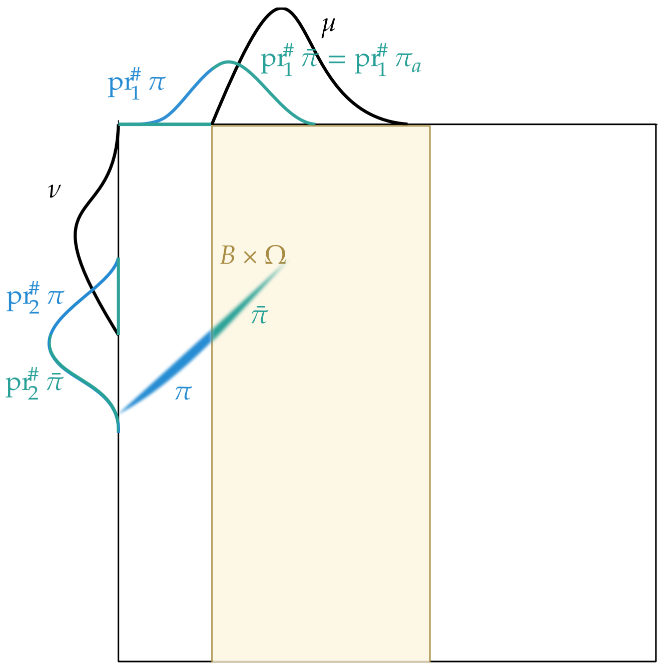


Figure 3.7: Illustration of restricting π to $B \times \Omega$. The original measures are visualized in blue, the restricted measures in turquoise.

3.2.8. Theorem. Let (Ω, d) be a compact metric space. Let $\mu, \nu \in \mathcal{M}_+(\Omega)$ be mutually singular, $\mu \perp \nu$. Let $p : \Omega \to \mathbb{R}_{>0}$ be a constant waste function. Then

$$\|\mu - \nu\|_p = \inf(\text{UMTP})$$

and further

$$\|\mu - \nu\|_p = \inf_{\tilde{\pi} \in \Pi(\tilde{\mu}, \tilde{\nu})} \iint_{\tilde{\Omega} \times \tilde{\Omega}} \tilde{c}(x, y) \mathrm{d}\tilde{\pi}(x, y)$$

with $\tilde{\mu}, \tilde{\nu}, \tilde{\Omega}$ and $\tilde{c}$ for $c = d$ as defined in Subsection 3.2.2.

Proof. We first write down the definition of $\|\mu - \nu\|_p$ as stated above. We have

$$\begin{aligned}
\|\mu - \nu\|_p &= \inf_{\eta \in \mathcal{M}_0(\Omega)} \inf_{\pi \in \Pi_\eta} \iint_{\Omega \times \Omega} d(x, y) \mathrm{d}\pi(x, y) + \int_\Omega p(x) \mathrm{d}|\mu - \nu - \eta^+ + \eta^-|(x) \\
&= \inf_{\pi \geq 0,\ \mathrm{pr}_1^\# \pi \perp \mathrm{pr}_2^\# \pi} \iint_{\Omega \times \Omega} d(x, y) \mathrm{d}\pi(x, y) + \int_\Omega p(x) \mathrm{d}\big|(\mu - \mathrm{pr}_1^\# \pi) - (\nu - \mathrm{pr}_2^\# \pi)\big|(x) \\
&=: (*),
\end{aligned}$$

where we have used the condition $\pi \in \Pi_\eta$, i. e. $\mathrm{pr}_1^\# \pi = \eta^+$ and $\mathrm{pr}_2^\# \pi = \eta^-$ and $\pi \geq 0$ and added the condition $\mathrm{pr}_1^\# \pi \perp \mathrm{pr}_2^\# \pi$ instead, since η^+ and η^- are mutually singular. This already looks very similar to the UMTP: we only need to show that the optimal π can be chosen such that $\mathrm{pr}_1^\# \pi \leq \mu$ and $\mathrm{pr}_2^\# \pi \leq \nu$. Then $\mu - \mathrm{pr}_1^\# \pi$ and $\nu - \mathrm{pr}_2^\# \pi$ are non-negative and mutually singular, since μ and ν are mutually singular, and we get

$$\int_\Omega p(x) \mathrm{d}\big|(\mu - \mathrm{pr}_1^\# \pi) - (\nu - \mathrm{pr}_2^\# \pi)\big|(x) = \int_\Omega p(x) \mathrm{d}(\mu - \mathrm{pr}_1^\# \pi)(x) + \int_\Omega p(x)(\nu - \mathrm{pr}_2^\# \pi)(x).$$

This shows the equality between the K-norm and UMTP.

We show that an optimal π can always be chosen such that $\mathrm{pr}_1^\# \pi \leq \mu$ and $\mathrm{pr}_2^\# \pi \leq \nu$. We do this in two steps. First, we show that π can be chosen such that $\mathrm{pr}_1^\# \pi \ll \mu$. Analogously, we then also get $\mathrm{pr}_2^\# \pi \ll \nu$. Making use of the absolute continuity, we then show the desired inequality in the second step.

Step 1 We show that π can be chosen such that $\mathrm{pr}_1^\# \pi \ll \mu$.

Intuitively, it makes sense that $\mathrm{pr}_1^\# \pi$ should not have any mass anywhere that μ itself lacks mass, since this would lead to additional costs.

The idea is thus to construct a coupling $\bar{\pi}$ from π such that $\mathrm{pr}_1^\# \bar{\pi}$ only has mass on a set B where μ also has mass. If we find such a set B, we can define $\bar{\pi} := \pi|_{B \times \Omega}$. We then have to show that the costs have not increased. Since we have restricted π, the transport costs in the first term have decreased. For the second term, on the part of Ω where $\mathrm{pr}_1^\# \pi$ had mass and μ did not, the costs have also decreased. However, restricting π also changes $\mathrm{pr}_2^\# \pi$, which can increase the costs. We thus need to show that the increase is not higher than the decrease.

First, we define the set B. An illustration of the construction is shown in Figure 3.5.

Let

$$\mathrm{pr}_1^\# \pi = (\mathrm{pr}_1^\# \pi)_a + (\mathrm{pr}_1^\# \pi)_s, \quad (\mathrm{pr}_1^\# \pi)_a \ll \mu,\ (\mathrm{pr}_1^\# \pi)_s \perp \mu$$

be the Radon-Nikodym decomposition of $\mathrm{pr}_1^\# \pi$ with respect to μ. We now aim to restrict π to a set such that we only keep the absolutely continuous part of the projection. First, we look for a set such that the singular part $(\mathrm{pr}_1^\# \pi)_s$ is zero on this set. By definition, since $(\mathrm{pr}_1^\# \pi)_s \perp \mu$, there exists a set $C \subseteq \Omega$ such that $(\mathrm{pr}_1^\# \pi)_s(E) = 0$ for all measurable sets $E \subseteq C$ and $\mu(E) = 0$ for all measurable sets $E \subseteq C^c$, i. e. $(\mathrm{pr}_1^\# \pi)_s = 0$ on C and $\mu = 0$ on C^c. However, μ is not guaranteed to be positive on C. We further know that $\mu \perp \nu$. We thus have a different set $\bar{C} \subset \Omega$ such that ν is zero on $\bar{C}$ and μ is zero on $\bar{C}^c$. We define $B := C \cap \bar{C}$, such that both $\nu = 0$ on $B \subseteq \bar{C}$ and $(\mathrm{pr}_1^\# \pi)_s = 0$ on $B \subseteq C$. Further we have $\mu = 0$ on $B^c = C^c \cup \bar{C}^c$, since μ is zero on C^c and well as on $\bar{C}^c$. Since also $\mathrm{pr}_1^\# \pi \perp \mathrm{pr}_2^\# \pi$, we have another decomposition into sets A and A^c. All together, we have the following relations:

$$\begin{aligned} (\mathrm{pr}_1^\# \pi)_s = \nu &= 0 \text{ on } B \\ \mu &= 0 \text{ on } B^c \\ \mathrm{pr}_2^\# \pi &= 0 \text{ on } A \\ \mathrm{pr}_1^\# \pi &= 0 \text{ on } A^c. \end{aligned} \tag{3.13}$$

Now, we define $\bar{\pi} = \pi|_{B\times\Omega} \geq 0$. An illustration of the restriction is shown in Figure 3.7. Since $\bar{\pi} \leq \pi$, the marginals of $\bar{\pi}$ are still mutually singular and they are still zero on A and A^c, respectively. Since $\bar{\pi} \leq \pi$, the first term of the cost is decreased, i. e.

$$\iint_{\Omega\times\Omega} d(x,y) \mathrm{d}\bar{\pi}(x,y) \leq \iint_{\Omega\times\Omega} d(x,y) \mathrm{d}\pi(x,y).$$

In order to show that the costs have not increased in the second term, we want to split Ω into four disjoint regions such that on each region, only two out of the four measures μ, ν, $\mathrm{pr}_1^\# \pi$ and $\mathrm{pr}_2^\# \pi$ contribute to the integral. We list these regions below, together with the measures that are zero on those regions:

$$\begin{aligned} \Omega_1 &:= A \cap B, & \mathrm{pr}_2^\# \pi &= \nu = 0 \text{ on } \Omega_1 \\ \Omega_2 &:= A \cap B^c, & \mathrm{pr}_2^\# \pi &= \mu = 0 \text{ on } \Omega_2 \\ \Omega_3 &:= A^c \cap B^c, & \mathrm{pr}_1^\# \pi &= \mu = 0 \text{ on } \Omega_3 \\ \Omega_4 &:= A^c \cap B, & \mathrm{pr}_1^\# \pi &= \nu = 0 \text{ on } \Omega_4. \end{aligned} \tag{3.14}$$

One easily sees that the sets are disjoint and that their union is Ω. An illustration of the sets is shown in Figure 3.6. We now calculate the marginals of $\bar{\pi}$. We have

$$\mathrm{pr}_1^\# \bar{\pi}(E) = \bar{\pi}(E \times \Omega) = \pi((E \cap B) \times \Omega) = \mathrm{pr}_1^\# \pi|_B(E)$$

for any measurable $E \subset \Omega$, so we have $\mathrm{pr}_1^\# \bar{\pi} = \mathrm{pr}_1^\# \pi|_B$. Further,

$$\begin{aligned} \mathrm{pr}_2^\# \bar{\pi}(E) &= \bar{\pi}(\Omega \times E) = \pi(B \times E) = \pi(\Omega \times E) - \pi(B^c \times E) \\ &= \mathrm{pr}_2^\# \pi(E) - \pi(B^c \times E) \end{aligned}$$

for any measurable $E \subset \Omega$, thus $\mathrm{pr}_2^\# \bar{\pi} = \mathrm{pr}_2^\# \pi - \pi(B^c \times (\cdot))$. We now calculate the new costs for each region Ω_i, $i = 1,2,3,4$. Since we assume the waste function to be constant, i. e. $p \equiv \alpha$ for some constant $\alpha > 0$, we have

$$\int_{\Omega_i} p(x) \mathrm{d}\big|\mathrm{pr}_1^\# \bar{\pi} - \mu - (\mathrm{pr}_2^\# \bar{\pi} - \nu)\big|(x) = \alpha \big|\mathrm{pr}_1^\# \bar{\pi} - \mu - (\mathrm{pr}_2^\# \bar{\pi} - \nu)\big|(\Omega_i)$$

for $i = 1,2,3,4$. In the following calculations, we omit the constant α. First, since $\Omega_1 \subset B$ and thus, on Ω_1, restricting $\mathrm{pr}_1^\# \pi$ to B does not change anything,

$$\begin{aligned}
\left|\mathrm{pr}_1^\# \bar\pi - \mu - (\mathrm{pr}_2^\# \bar\pi - \nu)\right|(\Omega_1) &= \left|\mathrm{pr}_1^\# \bar\pi - \mu\right|(\Omega_1) = \left|\mathrm{pr}_1^\# \pi|_B - \mu\right|(\Omega_1) \\
&= \left|\mathrm{pr}_1^\# \pi - \mu\right|(\Omega_1) \\
&= \left|\mathrm{pr}_1^\# \pi - \mu - (\mathrm{pr}_2^\# \pi - \nu)\right|(\Omega_1).
\end{aligned}$$

Second, since $\Omega_2 \subset B^c$ and thus $\mathrm{pr}_1^\# \pi|_B(\Omega_2) = 0$,

$$\begin{aligned}
\left|\mathrm{pr}_1^\# \bar\pi - \mu - (\mathrm{pr}_2^\# \bar\pi - \nu)\right|(\Omega_2) &= \left|\mathrm{pr}_1^\# \bar\pi + \nu\right|(\Omega_2) = \mathrm{pr}_1^\# \bar\pi(\Omega_2) + \nu(\Omega_2) \\
&= \mathrm{pr}_1^\# \pi|_B(\Omega_2) + \nu(\Omega_2) \\
&= \nu(\Omega_2) \\
&= \left|\mathrm{pr}_1^\# \pi - \mu - (\mathrm{pr}_2^\# \pi - \nu)\right|(\Omega_2) - \mathrm{pr}_1^\# \pi(\Omega_2).
\end{aligned}$$

We see that the costs have decreased by $\mathrm{pr}_1^\# \pi(\Omega_2)$. Third,

$$\begin{aligned}
\left|\mathrm{pr}_1^\# \bar\pi - \mu - (\mathrm{pr}_2^\# \bar\pi - \nu)\right|(\Omega_3) &= \left|\nu - \mathrm{pr}_2^\# \bar\pi\right|(\Omega_3) = \left|\nu - \mathrm{pr}_2^\# \pi + \pi(B^c \times (\cdot))\right|(\Omega_3) \\
&\leq \left|\nu - \mathrm{pr}_2^\# \pi\right|(\Omega_3) + \left|\pi(B^c \times (\cdot))\right|(\Omega_3) \\
&= \left|\mathrm{pr}_1^\# \pi - \mu - (\mathrm{pr}_2^\# \pi - \nu)\right|(\Omega_3) + \pi(B^c \times \Omega_3).
\end{aligned}$$

We see that the costs have increased by at most $\pi(B^c \times \Omega_3)$. Last,

$$\begin{aligned}
\left|\mathrm{pr}_1^\# \bar\pi - \mu - (\mathrm{pr}_2^\# \bar\pi - \nu)\right|(\Omega_4) &= \left|-\mu - \mathrm{pr}_2^\# \bar\pi\right|(\Omega_4) \\
&= \left|\mu + \mathrm{pr}_2^\# \bar\pi\right|(\Omega_4) \qquad = \mu(\Omega_4) + \mathrm{pr}_2^\# \bar\pi(\Omega_4) \\
&\leq \mu(\Omega_4) + \mathrm{pr}_2^\# \pi(\Omega_4) \\
&= \left|\mu + \mathrm{pr}_2^\# \pi\right|(\Omega_4) = \left|-\mu - \mathrm{pr}_2^\# \pi\right|(\Omega_4) \\
&= \left|\mathrm{pr}_1^\# \pi - \mu - (\mathrm{pr}_2^\# \pi - \nu)\right|(\Omega_4),
\end{aligned}$$

so the costs have decreased. Now, putting everything together, we see that we have a decrease of at least $\mathrm{pr}_1^\# \pi(\Omega_2) = \mathrm{pr}_1^\# \pi(A \cap B^c)$ and an increase of at most $\pi(B^c \times \Omega_3) \leq \pi(B^c \times \Omega) = \mathrm{pr}_1^\# \pi(B^c)$. Since $\mathrm{pr}_1^\# \pi$ is zero on A^c, we have $\mathrm{pr}_1^\# \pi(B^c) = \mathrm{pr}_1^\# \pi(A \cap B^c)$, which shows that increase and decrease cancel each other out. So we indeed have

$$\alpha\left|\mathrm{pr}_1^\# \bar\pi - \mu - (\mathrm{pr}_2^\# \bar\pi - \nu)\right|(\Omega) \leq \alpha\left|\mathrm{pr}_1^\# \pi - \mu - (\mathrm{pr}_2^\# \pi - \nu)\right|(\Omega).$$

This shows that we get $\mathrm{pr}_1^\# \bar\pi = \mathrm{pr}_1^\# \pi|_B = \mathrm{pr}_1^\# \pi_a|_B$, since $\mathrm{pr}_1^\# \pi_s$ is zero on B.

By replacing π by $\bar\pi$, we can thus assume that the projection $\mathrm{pr}_1^\# \pi$ is absolutely continuous with respect to μ for any optimal π. Similarly, we can assume $\mathrm{pr}_2^\# \pi$ to be absolutely continuous with respect to ν. In total, we have that μ is zero on B^c, and since $\mathrm{pr}_1^\# \pi \ll \mu$, so is $\mathrm{pr}_1^\# \pi$, which means that their difference $\mu - \mathrm{pr}_1^\# \pi$ is also zero on B^c. Analogously, since ν is zero on B, $\nu - \mathrm{pr}_2^\# \pi$ is zero on B. Thus, $\mu - \mathrm{pr}_1^\# \pi \perp \nu - \mathrm{pr}_2^\# \pi$ and we get

$$\begin{aligned}
(*) = \inf_{\pi \in \mathcal{M}_+(\Omega \times \Omega)} & \iint_{\Omega \times \Omega} d(x,y) \mathrm{d}\pi(x,y) + \alpha \int_\Omega \mathrm{d}\left|\mu - \mathrm{pr}_1^\# \pi\right|(x) + \alpha \int_\Omega \left|\nu - \mathrm{pr}_2^\# \pi\right|(x) \\
\text{subject to } & \mathrm{pr}_1^\# \pi \perp \mathrm{pr}_2^\# \pi \\
\text{and } & \mathrm{pr}_1^\# \pi \ll \mu, \mathrm{pr}_2^\# \pi \ll \nu \\
=: (**)&.
\end{aligned}$$

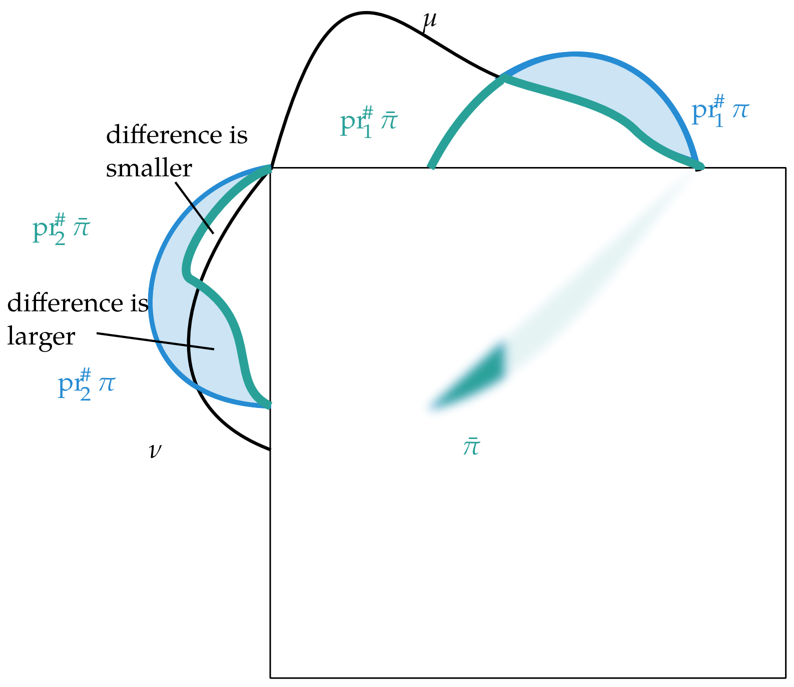


Figure 3.8: Visualization of rescaling π to $\bar\pi$. While the absolute difference between $\mathrm{pr}_2^\# \bar\pi$ and ν might locally be larger than the absolute difference between $\mathrm{pr}_2^\# \pi$ and ν, increase and decrease in the costs cancel each other out in total.

Step 2 Next, we want to show that we can even choose π such that $\mathrm{pr}_1^\# \pi \le \mu$ and $\mathrm{pr}_2^\# \pi \le \nu$, which means we can omit taking the total variation measure and we get $(**)$ = (UMTP) under the assumption $\mu \perp \nu$. Again, we aim to define a transport plan $\bar\pi$ as a modified version of π that satisfies $\mathrm{pr}_1^\# \bar\pi \le \mu$, such that the costs do not increase, and argue in the same way for the other marginal. The idea is to modify π such that $\mathrm{pr}_1^\# \bar\pi = \mu$ wherever we had $\mathrm{pr}_1^\# \pi > \mu$ before, which will lead to a decrease of costs. In order to keep π nonnegative, we reach this goal by scaling π. This will lead to smaller transport costs. However, like before, we might get an increase of costs from the second marginal. We will again show that increase and decrease cancel each other out. An illustration can be seen in Figure 3.8. In order to be able to scale π, we consider densities. Let

$$\varphi := \frac{\mathrm{d}\, \mathrm{pr}_1^\# \pi}{\mathrm{d}\mu}$$

and let

$$\Omega_+ := \{x : \ \varphi(x) \le 1 \quad \mu - a.e.\}, \qquad \Omega_- := \{x : \ \varphi(x) > 1 \quad \mu - a.e.\}.$$

For $E \subseteq \Omega_+$, we have

$$\mathrm{pr}_1^\# \pi(E) = \int_E \varphi(x) \mathrm{d}\mu(x) \le \int_E 1 \mathrm{d}\mu(x) = \mu(E),$$

so $\mu - \mathrm{pr}_1^\# \pi \ge 0$ on Ω_+. Similarly, for $E \subseteq \Omega_-$, we have $\mathrm{pr}_1^\# \pi > \mu$ and $\mu - \mathrm{pr}_1^\# \pi < 0$. We thus want to modify $\mathrm{pr}_1^\# \pi$ on Ω_-, which can be reached by modifying π on the slice $\Omega_- \times \Omega$. We define

$$\bar\pi := {}^1\!/_{\max(1, \varphi \circ \mathrm{pr}_1)} \pi.$$

Note that since the properties $\varphi \le 1$ on Ω_+ and $\varphi > 1$ on Ω_- hold $\mu - a.e.$, there exists a set $N \subseteq \Omega$ with $\mu(N) = 0$ such that the properties hold on $\Omega \setminus N$. Since $\mathrm{pr}_1^\# \pi \ll \mu$, we also have $\mathrm{pr}_1^\# \pi(N) = 0$ and thus $\pi(N \times \Omega) = \mathrm{pr}_1^\# \pi(N) = 0$, so $N \times \Omega$ is a π-null set. This assures that

${}^{1}/_{\max(1,\varphi(x))} \leq 1 \ \pi - a.e.$ and thus $\bar{\pi} \leq \pi$. Further, for $E \subseteq \Omega_+$,

$$\bar{\pi}(E \times E') = \iint_{E\times E'} {}^{1}/_{\max(1,\varphi(x))} \mathrm{d}\pi(x,y) = \iint_{E\times E'} 1 \mathrm{d}\pi(x,y) = \pi(E \times E').$$

Thus, we have not modified π on $\Omega_+ \times \Omega$, which implies $\mathrm{pr}_1^{\#}\,\bar{\pi} = \mathrm{pr}_1^{\#}\,\pi$ on Ω_+.

Let $E \subseteq \Omega_-$. Then

$$\begin{aligned}
\mathrm{pr}_1^{\#}\,\bar{\pi}(E) &= \bar{\pi}(E \times \Omega) = \iint_{E\times\Omega} {}^{1}/_{\max(1,\varphi(x))} \mathrm{d}\pi(x,y) \\
&= \iint_{E\times\Omega} {}^{1}/_{\varphi(x)} \mathrm{d}\pi(x,y) = \int_E {}^{1}/_{\varphi(x)} \int_\Omega \mathrm{d}\pi(x,y) \\
&= \int_E {}^{1}/_{\varphi(x)} \mathrm{d}\,\mathrm{pr}_1^{\#}\,\pi(x) = \int_E {}^{1}/_{\varphi(x)} \varphi(x) \mathrm{d}\mu(x) \\
&= \mu(E),
\end{aligned}$$

so $\mathrm{pr}_1^{\#}\,\bar{\pi} = \mu$ on Ω_-. Now, we show that the costs have not increased. Since $\bar{\pi} \leq \pi$, the transport costs have not increased. Let us now evaluate the terms $\alpha \int_\Omega \mathrm{d}|\mu - \mathrm{pr}_1^{\#}\,\bar{\pi}|(x)$ and $\alpha \int_\Omega |\nu - \mathrm{pr}_2^{\#}\,\bar{\pi}|(x)$. We have

$$\begin{aligned}
|\mu - \mathrm{pr}_1^{\#}\,\bar{\pi}|(\Omega) &= |\mu - \mathrm{pr}_1^{\#}\,\bar{\pi}|(\Omega_+) + |\mu - \mathrm{pr}_1^{\#}\,\bar{\pi}|(\Omega_-) \\
&= |\mu - \mathrm{pr}_1^{\#}\,\pi|(\Omega_+) \\
&= |\mu - \mathrm{pr}_1^{\#}\,\pi|(\Omega) - |\mu - \mathrm{pr}_1^{\#}\,\pi|(\Omega_-),
\end{aligned}$$

since $\mathrm{pr}_1^{\#}\,\bar{\pi} = \mathrm{pr}_1^{\#}\,\pi$ on Ω_+ and $\mathrm{pr}_1^{\#}\,\bar{\pi} = \mu$ on Ω_-. We thus have a decrease of

$$|\mu - \mathrm{pr}_1^{\#}\,\pi|(\Omega_-) = \mathrm{pr}_1^{\#}\,\pi(\Omega_-) - \mu(\Omega_-), \tag{3.15}$$

since $\mu - \mathrm{pr}_1^{\#}\,\pi < 0$ on Ω_-. Now,

$$\begin{aligned}
|\nu - \mathrm{pr}_2^{\#}\,\bar{\pi}|(\Omega) &= |\nu - \mathrm{pr}_2^{\#}\,\bar{\pi} + \mathrm{pr}_2^{\#}\,\pi - \mathrm{pr}_2^{\#}\,\pi|(\Omega) \\
&\leq |\nu - \mathrm{pr}_2^{\#}\,\pi|(\Omega) + |\mathrm{pr}_2^{\#}\,\pi - \mathrm{pr}_2^{\#}\,\bar{\pi}|(\Omega),
\end{aligned}$$

which means we have an increase of at most $|\mathrm{pr}_2^{\#}\,\pi - \mathrm{pr}_2^{\#}\,\bar{\pi}|(\Omega)$. Since $\mathrm{pr}_2^{\#}\,\bar{\pi} \leq \mathrm{pr}_2^{\#}\,\pi$, we have

$$\begin{aligned}
|\mathrm{pr}_2^{\#}\,\pi - \mathrm{pr}_2^{\#}\,\bar{\pi}|(\Omega) &= \mathrm{pr}_2^{\#}\,\pi(\Omega) - \mathrm{pr}_2^{\#}\,\bar{\pi}(\Omega) \\
&= \mathrm{pr}_1^{\#}\,\pi(\Omega) - \mathrm{pr}_1^{\#}\,\bar{\pi}(\Omega) \\
&= \mathrm{pr}_1^{\#}\,\pi(\Omega_+) + \mathrm{pr}_1^{\#}\,\pi(\Omega_-) - \mathrm{pr}_1^{\#}\,\bar{\pi}(\Omega_+) - \mathrm{pr}_1^{\#}\,\bar{\pi}(\Omega_-) \\
&= \mathrm{pr}_1^{\#}\,\pi(\Omega_+) + \mathrm{pr}_1^{\#}\,\pi(\Omega_-) - \mathrm{pr}_1^{\#}\,\pi(\Omega_+) - \mu(\Omega_-) \\
&= \mathrm{pr}_1^{\#}\,\pi(\Omega_-) - \mu(\Omega_-),
\end{aligned}$$

which is exactly the decrease (3.15). So increase and decrease indeed cancel each other out, and we have

$$\alpha \int_\Omega \mathrm{d}|\mu - \mathrm{pr}_1^{\#}\,\bar{\pi}|(x) + \alpha \int_\Omega \mathrm{d}|\nu - \mathrm{pr}_2^{\#}\,\bar{\pi}|(x) \leq \alpha \int_\Omega \mathrm{d}|\mu - \mathrm{pr}_1^{\#}\,\pi|(x) + \alpha \int_\Omega \mathrm{d}|\nu - \mathrm{pr}_2^{\#}\,\pi|(x).$$

This proves that we can always find a transport plan π such that $\mathrm{pr}_1^{\#}\,\pi \leq \mu$. With a mirrored argument for the other marginal, we achieve both $\mathrm{pr}_1^{\#}\,\pi \leq \mu$ and $\mathrm{pr}_2^{\#}\,\pi \leq \nu$ which yields $(**) =$ (UMTP) and thus $\|\mu - \nu\|_p = \mathcal{W}_1(\tilde{\mu}, \tilde{\nu})$ for mutually singular measures.

This concludes the proof. ∎

We now consider the case where the cost function $\tilde{c}$ defined through d and the waste constant is a metric.

3.2.9. Corollary. Let (Ω, d) be a compact metric space. Let $\mu, \nu \in \mathcal{M}_+(\Omega)$. Let $p : \Omega \to \mathbb{R}$ be a constant waste function satisfying $p = \alpha \geq \frac{1}{2}\sup_{x,y\in\Omega} d(x,y)$. Then

$$\|\mu - \nu\|_p = \inf (\text{UMTP})$$

and further

$$\|\mu - \nu\|_p = \inf_{\tilde{\pi}\in\Pi(\tilde{\mu},\tilde{\nu})} \iint_{\tilde{\Omega}\times\tilde{\Omega}} \tilde{d}(x,y) \mathrm{d}\tilde{\pi}(x,y) = \mathcal{W}_1(\tilde{\mu}, \tilde{\nu})$$

with $\tilde{\mu}, \tilde{\nu}, \tilde{\Omega}$ and $\tilde{d}$ as defined in Subsection 3.2.2.

Proof. Note that since the cost function $\tilde{d}$ is a metric, we can speak of the Wasserstein distance in this case. We can now remove the assumption that μ and ν are singular. Let $\mu \wedge \nu := \mu - (\mu - \nu)^+$. Then $\mu_0 = (\mu - \nu)^+$ and $\nu_0 = (\mu - \nu)^-$. As seen in the proof of Corollary 3.1.11, $\mu_0 - \nu_0 = \mu - \nu$. Further, μ_0 and ν_0 are mutually singular. Then by Theorem 3.2.8,

$$\|\mu - \nu\|_p = \|(\mu - \nu)^+ - (\mu - \nu)^-\|_p = \|\mu_0 - \nu_0\|_p = \mathcal{W}_1(\tilde{\mu}_0, \tilde{\nu}_0).$$

We have

$$\begin{aligned}
\tilde{\mu}_0 - \tilde{\nu}_0 &= \iota^\# \mu_0 + \nu_0(\Omega) \cdot \delta_{x^*} - \iota^\# \nu_0 - \mu_0(\Omega) \cdot \delta_{x^*} \\
&= \iota^\# (\mu_0 - \nu_0) + (\nu_0 - \mu_0)(\Omega) \cdot \delta_{x^*} \\
&= \iota^\# (\mu - \nu) + (\nu - \mu)(\Omega) \cdot \delta_{x^*} \\
&= \tilde{\mu} - \tilde{\nu}.
\end{aligned}$$

Since the 1-Wasserstein distance only depends on the difference by Corollary 3.1.11, we get

$$W_1(\tilde{\mu}_0, \tilde{\nu}_0) = W_1(\tilde{\mu}, \tilde{\nu}).$$

This shows $\|\mu - \nu\|_p = W_1(\tilde{\mu}, \tilde{\nu})$ for arbitrary measures $\mu, \nu \in \mathcal{M}_+(\Omega)$, if the waste constant is such that $\alpha > \frac{1}{2}\sup_{x,y\in\Omega} d(x,y)$. ■

3.3 The Entropy-Regularized Kantorovich Problem

While there has been a lot of research into efficient ways to calculate the Wasserstein distance, many algorithms only work for specific forms of the cost function or for small domains. Recently, it was suggested in [40] to add an entropic constraint on the transport plans. The resulting functional, which gives rise to a regularized variant of the Kantorovich problem, has turned out to have many nice properties and there exists a very fast algorithm to solve the corresponding minimization problem. As sources for the idea of regularized optimal transport, we use several papers, mainly [41], [42], [43], [28], [40] and [33].

3.3.1 Introduction of the Entropy-Regularized Kantorovich Problem

In this section, we define the regularized Kantorovich Problem and show that it admits a unique solution.

We define the *relative entropy* or *Kullback-Leibler divergence* of a measure. Let X be a compact metric space.

3.3.1. Definition. The *Kullback-Leibler divergence* of two measures $\mu_1, \mu_2 \in \mathcal{M}_+(X)$ is defined as

$$\mathrm{KL}(\mu_1 \parallel \mu_2) = \int_X \Big(\log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\Big) - 1\Big)\mathrm{d}\mu_1(x)$$

with the convention $\mathrm{KL}(\mu_1 \parallel \mu_2) = \infty$ if μ_1 is not absolutely continuous with respect to μ_2.

This definition differs from the classical definition (see [44]) by the additional "-1". The introduction of the "-1" is inspired by [41].

The Kullback-Leiber divergence can be formulated in different ways. We have

$$\begin{aligned}\mathrm{KL}(\mu_1 \parallel \mu_2) &= \int_X \Big(\log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\Big) - 1\Big)\mathrm{d}\mu_1(x)\\ &= \int_X \Big(\log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\Big) - 1\Big)\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\mathrm{d}\mu_2(x)\\ &= \int_X \log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\Big)\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\mathrm{d}\mu_2(x) - \mu_1(X)\end{aligned} \tag{3.16}$$

and if we assume that μ_1 and μ_2 are absolutely continuous with respect to a common measure λ, we get, letting $m_1 = \frac{\mathrm{d}\mu_1}{\mathrm{d}\lambda}, m_2 = \frac{\mathrm{d}\mu_2}{\mathrm{d}\lambda}$,

$$\begin{aligned}\mathrm{KL}(\mu_1 \parallel \mu_2) &= \int_X \Big(\log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}(x)\Big) - 1\Big)\mathrm{d}\mu_1(x)\\ &= \int_X \Big(\log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\lambda}(x)\tfrac{\mathrm{d}\lambda}{\mathrm{d}\mu_2}(x)\Big) - 1\Big)\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\lambda}(x)\mathrm{d}\lambda(x)\\ &= \int_X \left(\log\left(\frac{m_1(x)}{m_2(x)}\right) - 1\right)m_1(x)\mathrm{d}\lambda(x)\\ &= \int_X \log\left(\frac{m_1(x)}{m_2(x)}\right)m_1(x)\mathrm{d}\lambda(x) - 1.\end{aligned} \tag{3.17}$$

Now, if X is a measure space (X, λ) for some measure λ, we can consider the Kullback-Leiber divergence relative to this measure. We get the definition of the (negative) *entropy* of a measure with respect to λ, defined as

$$h(\mu) := \mathrm{KL}(\mu \parallel \lambda) = \int_X (\log\Big(\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda}(x)\Big) - 1)\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda}(x)\mathrm{d}\lambda(x). \tag{3.18}$$

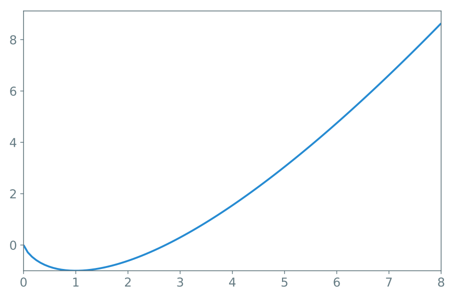


Figure 3.9: The function $\phi(r) = r \cdot (\log r - 1)$, extended continuously in 0. It can be negative, but this can be easily alleviated by adding a constant. It is strictly convex.

Now let $\phi : \mathbb{R} \to (-\infty, +\infty]$ be defined by

$$\phi(r) = \begin{cases} r(\log r - 1), & \text{if } r > 0 \\ 0, & \text{if } r = 0 \\ +\infty & \text{else.} \end{cases} \tag{3.19}$$

A plot of the function ϕ is shown in Figure 3.9. Then the negative entropy of a measure with respect to λ equals

$$h(\mu) = \int_X \phi(\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda})\mathrm{d}\lambda(x). \tag{3.20}$$

Note that ϕ is bounded from below, and thus the entropy is always bounded from below.

Let us now fix reference measures λ_i on Ω_i for $i = 1, 2$. We equip the product space $\Omega_1 \times \Omega_2$ with the product measure $\lambda_1 \otimes \lambda_2$. If we assume the marginals μ and ν and the transport plan π to be absolutely continuous with respect to the respective reference measure, i. e. $\mu \ll \lambda_1, \nu \ll \lambda_2, \pi \ll \lambda_1 \otimes \lambda_2$, we can reformulate the conditions describing $\Pi(\mu, \nu)$ in the following way. For the density of $\mathrm{pr}_1^\# \pi$ with respect to λ_1 we have

$$\begin{aligned} \mathrm{pr}_1^\# \pi(A) = \pi(A \times \Omega_2) &= \iint_{A \times \Omega_2} \frac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x, y)\mathrm{d}\lambda_1 \otimes \lambda_2(x, y) \\ &= \int_A \left(\int_{\Omega_2} \frac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x, y)\mathrm{d}\lambda_2(y) \right) \mathrm{d}\lambda_1(x) \end{aligned}$$

for all measurable sets $A \subseteq \Omega_1$. This shows

$$\frac{\mathrm{d}\,\mathrm{pr}_1^\# \pi}{\mathrm{d}\lambda_1}(x) = \int_{\Omega_2} \frac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x, y)\mathrm{d}\lambda_2(y).$$

Analogously, for the second marginal we have

$$\frac{\mathrm{d}\,\mathrm{pr}_2^\# \pi}{\mathrm{d}\lambda_2} = \int_{\Omega_1} \frac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x, y)\mathrm{d}\lambda_1(x).$$

Consequently, the marginal conditions become

$$\begin{aligned} \frac{\mathrm{d}\,\mathrm{pr}_1^\# \pi}{\mathrm{d}\lambda_1}(x) = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x) &\Leftrightarrow \int_{\Omega_2} \frac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x, y)\mathrm{d}\lambda_2(y) = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x) \text{ for all } x \in \Omega_1 \\ \frac{\mathrm{d}\,\mathrm{pr}_2^\# \pi}{\mathrm{d}\lambda_2}(y) = \frac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}(y) &\Leftrightarrow \int_{\Omega_1} \frac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x, y)\mathrm{d}\lambda_1(x) = \frac{\mathrm{d}\nu}{\mathrm{d}\lambda_1}(y) \text{ for all } y \in \Omega_2. \end{aligned} \tag{3.21}$$

We are now in a position to define the regularized Kantorovich problem.

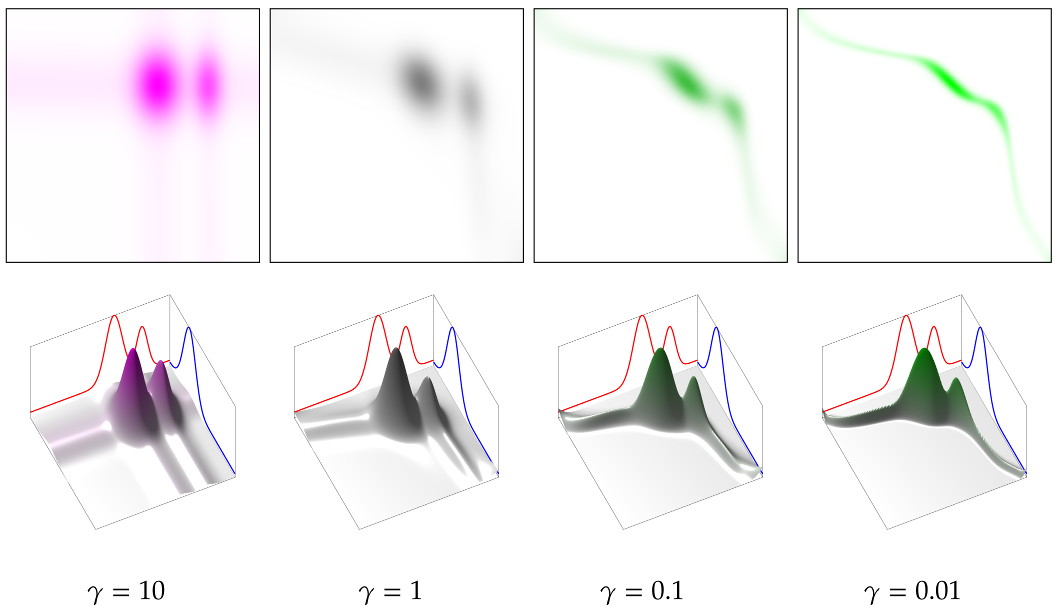


Figure 3.10: Illustration of the influence of the regularization parameter γ on the optimal transport plan between two one-dimensional densities. In the top row, the transport plans from the bottom row are shown from above. For larger values of γ, the transport plan is more spread out. For $\gamma \to 0$, it converges to the unregularized transport plan, which is a line. Source: [28, §4.1]

3.3.2. Definition. Let (Ω_1, λ_1), (Ω_2, λ_2) be two compact measure spaces and $(\Omega_1 \times \Omega_2, \lambda_1 \otimes \lambda_2)$ the product measure space. Given $\mu \in \mathcal{P}(\Omega_1)$ and $\nu \in \mathcal{P}(\Omega_2)$ such that $\mu \ll \lambda_1$ and $\nu \ll \lambda_2$ and a continuous function $c : \Omega_1 \times \Omega_2 \to \mathbb{R}_+$, the *regularized Kantorovich problem* is defined as

$$\inf \left\{ W_\gamma(\pi) \ : \ \pi \in \Pi(\mu, \nu), \pi \ll \lambda_1 \otimes \lambda_2 \right\}, \tag{KP-reg}$$

where

$$\begin{aligned} W_\gamma(\pi) &= \iint_{\Omega_1 \times \Omega_2} c(x,y) \mathrm{d}\pi(x,y) + \gamma h(\pi) \\ &= \iint_{\Omega_1 \times \Omega_2} c(x,y) + \gamma(\log\left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\right) - 1) \mathrm{d}\pi(x,y) \end{aligned} \tag{3.22}$$

for $\gamma > 0$ and $\Pi(\mu, \nu)$ is the set of transport plans as in (3.2).

The idea behind this definition is that for small γ, the solution to this problem will converge against a solution of the unregularized problem, but calculating a solution to (KP-reg) is much easier. The proof of convergence is simple for finite metric spaces and can be found in [28, Prop. 4.1]. For more general settings, it is studied in [45]. The influence of the regularization on the optimal transport plan is illustrated in Figure 3.10.

We now want to show that (KP-reg) admits a minimizer. This is not as easy as in the unregularized case and we actually need an additional assumption on μ and ν, the assumption that they both have finite entropy. Before we get to the proof, we will examine some properties of the Kullback-Leibler divergence.

As already noted, our definition of the Kullback-Leibler divergence differs slightly from classical definitions. The classical Kullback-Leibler divergence between two probability measures has the property that it is always non-negative and vanishes only if the measures are equal, see for example [44, Lemma 3.1]. In our setting, this property becomes the following.

3.3.3. Lemma. Let $\mu_1 \in \mathcal{P}(X)$ and $\mu_2 \in \mathcal{P}(X)$ be two probability measures. Then

$$\mathrm{KL}(\mu_1 \parallel \mu_2) + 1 \geq 0$$

with equality if and only if $\mu_1 = \mu_2$.

3.3.4. Lemma. Let $\mu_1 \in \mathcal{P}(X)$ be a probability measure and $\mu_2 \in \mathcal{M}_+(X)$ be an non-negative measure. Then

$$\mathrm{KL}(\mu_1 \parallel \mu_2) = \mathrm{KL}(\mu_1 \parallel \tfrac{1}{\mu_2(X)}\mu_2) - \log(\mu_2(X))$$

and

$$\mathrm{KL}(\mu_1 \parallel \mu_2) \geq -\log(\mu_2(X)) - 1$$

with equality if and only if $\mu_1 = \frac{1}{\mu_2(X)}\mu_2$.

Proof. We note that $\frac{\mathrm{d}\mu_1}{\mathrm{d}c\cdot\mu_2} = \frac{1}{c}\frac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}$ for any constant c. Now,

$$\begin{aligned}
\mathrm{KL}(\mu_1 \parallel \mu_2) &= \int_X \log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2} - 1\Big)\mathrm{d}\mu_1(x) \\
&= \int_X \log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}\tfrac{\mu_2(X)}{\mu_2(X)} - 1\Big)\mathrm{d}\mu_1(x) \\
&= \int_X \log\Big(\tfrac{1}{\mu_2(X)^{-1}}\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2}\tfrac{1}{\mu_2(X)} - 1\Big)\mathrm{d}\mu_1(x) \\
&= \int_X \log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2(X)^{-1}\mu_2} - 1\Big)\mathrm{d}\mu_1(x) - \int_X \log(\mu_2(X))\mathrm{d}\mu_1 \\
&= \int_X \log\Big(\tfrac{\mathrm{d}\mu_1}{\mathrm{d}\mu_2(X)^{-1}\mu_2} - 1\Big)\mathrm{d}\mu_1(x) - \log(\mu_2(X)),
\end{aligned}$$

which proves the first claim. Now, using that $\mu_2(X)^{-1}\mu_2$ is a probability measure, we can apply Lemma 3.3.3 and get the second claim. ■

The following properties are stated in [40] in the discrete case for the classical Kullback-Leibler divergence.

3.3.5. Lemma (Properties of the Kullback-Leibler divergence and entropy). Let $(\Omega_i, \lambda_i), i = 1, 2$ be two compact measure spaces and $(\Omega_1 \times \Omega_2, \lambda_1 \otimes \lambda_2)$ the compact measure space defined as the product measure space. Let $\mu \in \mathcal{P}(\Omega_1), \mu \ll \lambda_1$ and $\nu \in \mathcal{P}(\Omega_2), \nu \ll \lambda_2$. Then $\mu \otimes \nu$ is absolutely continuous with respect to $\lambda_1 \otimes \lambda_2$ with

$$\tfrac{\mathrm{d}\mu\otimes\nu}{\mathrm{d}\lambda_1\otimes\lambda_2}(x,y) = \tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x)\tfrac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}(y),$$

and

$$h(\mu \otimes \nu) = h(\mu) + h(\nu) + 1.$$

Further, for any $\pi \in \Pi(\mu, \nu)$ with $\pi \ll \lambda_1 \otimes \lambda_2$,

$$h(\pi) \geq h(\mu) + h(\nu) + 1.$$

Proof. First, we show that $\frac{\mathrm{d}\mu\otimes\nu}{\mathrm{d}\lambda_1\otimes\lambda_2}(x,y) = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x)\frac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}(y)$. Let $A \subseteq \Omega_1, B \subseteq \Omega_2$ be two measurable sets. Then

$$\begin{aligned}
\iint_{A\times B} \tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x)\tfrac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}(y)\mathrm{d}\lambda_1 \otimes \lambda_2(x,y) &= \int_A \tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x)\Big(\int_B \tfrac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}(y)\mathrm{d}\lambda_2(y)\Big)\mathrm{d}\lambda_1(x) \\
&= \int_A \tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x)\nu(B)\mathrm{d}\lambda_1(x) \\
&= \mu(A)\nu(B),
\end{aligned}$$

which proves the claim. Next, we prove

$$\mathrm{KL}(\pi \parallel \mu \otimes \nu) = h(\pi) - h(\mu) - h(\nu) - 2$$

and deduce the two statements. Let $m := \frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}, n := \frac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}$ and $p := \frac{\mathrm{d}\pi}{\mathrm{d}\lambda}$. We have

$$\begin{aligned}
\mathrm{KL}(\pi \parallel \mu \otimes \nu) &\overset{(3.17)}{=} \iint_{\Omega_1\times\Omega_2} \log\left(\frac{p(x,y)}{m(x)n(y)}\right)p(x,y)\mathrm{d}\lambda(x,y) - 1 \\
&= \iint_{\Omega_1\times\Omega_2} p(x,y)\log(p(x,y))\mathrm{d}\lambda(x,y) \\
&\quad - \iint_{\Omega_1\times\Omega_2} p(x,y)\log(m(x))\mathrm{d}\lambda(x,y) \\
&\quad - \iint_{\Omega_1\times\Omega_2} p(x,y)\log(n(y))\mathrm{d}\lambda(x,y) - 1 \\
&= \iint_{\Omega_1\times\Omega_2} p(x,y)\log(p(x,y))\mathrm{d}\lambda(x,y) \\
&\quad - \int_{\Omega_1} \log(m(x))\left(\int_{\Omega_2} p(x,y)\mathrm{d}\lambda_2(y)\right)\mathrm{d}\lambda_1(x) \\
&\quad - \int_{\Omega_2} \log(n(y))\left(\int_{\Omega_1} p(x,y)\mathrm{d}\lambda_1(x)\right)\mathrm{d}\lambda_2(y) - 1 \\
&\overset{(3.21)}{=} \iint_{\Omega_1\times\Omega_2} p(x,y)\log(p(x,y))\mathrm{d}\lambda(x,y) - \int_{\Omega_1} \log(m(x))m(x)\mathrm{d}\lambda_1(x) \\
&\qquad\qquad - \int_{\Omega_2} \log(n(y))n(y)\mathrm{d}\lambda_2(y) - 1 \\
&\overset{(3.16)}{=} \mathrm{KL}(\pi \parallel \lambda) + 1 - (\mathrm{KL}(\mu \parallel \lambda_1) + 1) - (\mathrm{KL}(\nu \parallel \lambda_2) + 1) - 1 \\
&= h(\pi) - h(\mu) - h(\nu) - 2,
\end{aligned}$$

as claimed. This means

$$h(\pi) = \mathrm{KL}(\pi \parallel \mu \otimes \nu) + h(\mu) + h(\nu) + 2$$

for all couplings π with marginals μ and ν. Applying Lemma 3.3.3, we see $\mathrm{KL}(\pi \parallel \mu \otimes \nu) \geq -1$ and $\mathrm{KL}(\mu \otimes \nu \parallel \mu \otimes \nu) = -1$ and it follows immediately that

$$h(\mu \otimes \nu) = h(\mu) + h(\nu) + 1 \qquad \text{and} \qquad h(\pi) \geq h(\mu) + h(\nu) + 1.$$

In order to prove existence of a minimizer for the regularized Kantorovich problem, we need some theory about the lower semicontinuity of functionals defined in terms of densities. The following theorem is helpful.

3.3.6. Theorem ([25, Thm. 13.3.1]). Let $f : \mathbb{R} \to [0, \infty]$ be a lower semicontinuous convex function and λ be a measure in $\mathcal{M}_+(X)$. Then the functional defined on $\mathcal{M}(X)$ by

$$F(\mu) = \int_X f\left(\frac{\mathrm{d}\mu}{\mathrm{d}\lambda}(x)\right)\mathrm{d}\lambda(x) + \int f^\infty\left(\frac{\mathrm{d}\mu^s}{\mathrm{d}|\mu^s|}(x)\right)\mathrm{d}|\mu^s|(x),$$

where $\mu = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda} \cdot \lambda + \mu^s$ is the Radon-Nikodym decomposition of μ with respect to λ and

$$f^\infty(a) = \lim_{t\to\infty} \frac{f(ta)}{t} \text{ for } a \in \mathbb{R},$$

is lower semicontinuous for the weak-* convergence in $\mathcal{M}(X)$.

In the following we abbreviate "almost everywhere" with a. e. .

3.3.7. Corollary. Let $f : \mathbb{R} \to [0,\infty]$ be a lower semicontinuous convex function with superlinear growth, i.e. $\lim_{t\to\infty} f(ta)/t = \infty$ for $a \neq 0$, and let λ be a measure in $\mathcal{M}_+(X)$. Let $(\mu_n)_n \subseteq \mathcal{M}(X)$ be a sequence with $\mu_n \ll \lambda$ and $\mu_n \overset{*}{\rightharpoonup} \mu$ with $\mu \in \mathcal{M}(X)$. If $\int_X f\left(\frac{d\mu_n}{d\lambda}(x)\right) d\lambda$ is bounded, then $\mu \ll \lambda$ and

$$\int_X f\left(\frac{d\mu_n}{d\lambda}(x)\right) d\lambda(x) \leq \liminf_{h\to\infty} \int_X f\left(\frac{d\mu_n}{d\lambda}(x)\right) d\lambda(x).$$

Proof. Let $F : \mathcal{M}(X) \to \mathbb{R} \cup \{\infty\}$ as in Theorem 3.3.6. Since $\mu_n \ll \lambda$, we have

$$F(\mu_n) = \int_X f\left(\frac{d\mu_n}{d\lambda}(x)\right) d\lambda(x),$$

which is bounded by some constant C by assumption. Applying the theorem we get

$$C \geq \liminf_{n\to\infty} F(\mu_n) \geq F(\mu) = \int_X f\left(\frac{d\mu}{d\lambda}(x)\right) d\lambda(x) + \int f^\infty\left(\frac{d\mu^s}{d|\mu^s|}(x)\right) d|\mu^s|(x).$$

The function f is positive, thus $\int_X f^\infty\left(\frac{d\mu^s}{d|\mu^s|}(x)\right) d|\mu^s|(x) \leq C$. But $f^\infty = \infty$ since the function f is superlinear. So necessarily $\frac{d\mu^s}{d|\mu^s|}(x) = 0$ $|\mu^s|$-a.e., which implies $\mu^s = 0$. This shows that μ admits a density w. r. t. λ and thus $\mu \ll \lambda$ and $\int_X f^\infty\left(\frac{d\mu^s}{d|\mu^s|}(x)\right) d|\mu^s|=0$. ∎

We now have all the tools to prove the existence of a minimizer of the regularized Kantorovich problem. However, we have to make the additional assumption that both measures have finite entropy.

3.3.8. Proposition. Let (Ω_1,λ_1) and (Ω_2,λ_2) be two compact measure spaces. Let $\mu \in \mathcal{P}(\Omega_1)$ and $\nu \in \mathcal{P}(\Omega_2)$ with finite entropy, i. e. $h(\mu) < \infty$ and $h(\nu) < \infty$. Then the problem

$$\inf\left\{W_\gamma(\pi) \,:\, \pi \in \Pi(\mu,\nu), \pi \ll \lambda\right\} \tag{KP-reg}$$

admits a unique solution.

Proof. First, we note that W_γ is bounded from below, since the first term is non-negative and the entropy h is bounded from below. As in the classical case, $\mu \otimes \nu$ has marginals μ and ν. It follows from Lemma 3.3.5 that $\mu \otimes \nu \ll \lambda_1 \otimes \lambda_2$ and that $h(\mu \otimes \nu) < \infty$, since both μ and ν have finite entropy. Thus, $W_\gamma(\mu \otimes \nu) < \infty$. Now, since $\mathcal{P}(X)$ is sequentially compact w. r. t. the weak-* convergence, we can find a minimizing sequence $\pi_n \overset{*}{\rightharpoonup} \pi$. Since it is a minimizing sequence, we have $\pi_n \ll \lambda_1 \otimes \lambda_2$. Let

$$\tilde{\phi} : \mathbb{R} \to [0,\infty], \tilde{\phi}(x) = \phi(x) + 1 = \begin{cases} x \cdot (\log(x) - 1) + 1, & x > 0 \\ 1, & x = 0 \\ \infty, & x < 0. \end{cases}$$

with ϕ as defined in (3.19). We note that $\tilde{\phi}$ is convex and lower semicontinuous, since it is continuous in 0 by L'Hôpital's rule, see also Figure 3.9. Further $\tilde{\phi}$ has superlinear growth, since

$$\lim_{t\to\infty} \frac{\tilde{\phi}(ta)}{t} = \lim_{t\to\infty} \frac{ta \cdot (\log(ta) - 1) + 1}{t} = \lim_{t\to\infty} a \cdot (\log(ta) - 1) + \frac{1}{t} = \infty.$$

Now consider the functional G_1 defined on $\mathcal{M}(\Omega_1 \times \Omega_2)$ by

$$\pi \mapsto \iint_{\Omega_1\times\Omega_2} \tilde{\phi}\left(\frac{d\pi}{d\lambda_1\otimes\lambda_2}(x,y)\right) d\lambda_1 \otimes \lambda_2(x,y).$$

Since π_n is a minimizing sequence of W_γ, which is bounded from below, and since an admissible transport plan exists, we can assume (by possibly taking subsequences) that

$$\iint_{\Omega_1\times\Omega_2} \tilde{\phi}(\pi_n(x,y))\mathrm{d}\lambda_1 \otimes \lambda_2(x,y)$$

is bounded. Now, Theorem 3.3.6 provides the lower semicontinuity of G_1 w. r. t. weak-* convergence and the absolute continuity of π w. r. t. $\lambda_1 \otimes \lambda_2$. As before, the functional G_2 defined on $\mathcal{M}(\Omega_1 \times \Omega_2)$ by

$$\pi \mapsto \iint_{\Omega_1\times\Omega_2} c(x,y)\mathrm{d}\pi(x,y)$$

is lower semicontinuous w. r. t. the weak-* convergence since c is a continuous function. Finally,

$$\begin{aligned}
W_\gamma(\pi) &= \iint_{\Omega_1\times\Omega_2} c(x,y)\mathrm{d}\pi(x,y) + \gamma h(\pi) \\
&\overset{(3.20)}{=} \iint_{\Omega_1\times\Omega_2} c(x,y)\mathrm{d}\pi(x,y) + \gamma \iint_{\Omega_1\times\Omega_2} \phi(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y))\mathrm{d}\lambda_1 \otimes \lambda_2(x,y) \\
&\overset{(3.20)}{=} \iint_{\Omega_1\times\Omega_2} c(x,y)\mathrm{d}\pi(x,y) + \gamma \iint_{\Omega_1\times\Omega_2} \tilde{\phi}(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)) - 1\mathrm{d}\lambda_1 \otimes \lambda_2(x,y) \\
&= G_2(\pi) + \gamma G_1(\pi) - \gamma\lambda_1 \otimes \lambda_2(X).
\end{aligned}$$

Since both G_2 and G_1 are lower semicontinuous w. r. t. weak-* convergence, W_γ is lower semicontinuous. As before, by Lemma 3.1.4, the marginal conditions are kept under weak-* convergence. This shows that π is indeed a minimizer of (KP-reg). The uniqueness follows from the strict convexity of ϕ. ■

In [41], the existence of a solution is claimed for all probability measures. The finite entropy condition is however necessary: Lemma 3.3.5 states

$$h(\pi) \geq h(\mu) + h(\nu) + 1$$

whenever $\pi \in \Pi(\mu, v)$, so if either μ or ν have infinite entropy, i. e. $h(\mu) = \infty$ or $h(\nu) = \infty$, then $h(\pi) = \infty$. Thus, $W_\gamma(\pi) = \infty$ for every admissible transport plan π.

The absolute continuity of the limit measure π is not trivial, which is demonstrated in the following two examples.

3.3.9. Example (Absolutely continuous marginals do not guarantee absolutely continuous transport plans). Let $\Omega = [0,1]$ and let λ denote the Lebesgue measure on $[0,1]$. Let $\mu \in \mathcal{P}(\Omega)$ be an absolutely continuous measure, $\mu \ll \lambda$. Let $\Delta : \Omega \to \Omega \times \Omega,\ t \mapsto (t,t)$ be the diagonal map. As in Example 3.1.9, we define the transport plan on the diagonal, $\pi := \Delta^\# \mu \in \mathcal{M}(\Omega \times \Omega)$. The image of Δ is the diagonal $\Delta(\Omega) = \{(x,x) : x \in \Omega\} \subseteq \Omega \times \Omega$. Then π is not absolutely continuous, since $\pi(\Delta(\Omega)) = \mu(\Delta^{-1}(\Delta(\Omega))) = \mu(\Omega) = 1$, but the diagonal is a null set for the Lebesgue measure on $\Omega \times \Omega$. But the marginals of π are both μ, which is absolutely continuous. An illustration can be seen in Figure 3.3.

3.3.10. Example (Absolute continuity is not preserved under weak-* convergence). Let Ω be a compact metric space, $\lambda \in \mathcal{M}(\Omega)$ a reference measure where all singleton sets are null sets, and $x_0 \in \Omega$ a fixed point. Let

$$f_n : \Omega \to \mathbb{R}, \quad f_n(x) = \begin{cases} \lambda\left(B_{\frac{1}{n}}(x_0)\right)^{-1}, & x \in B_{\frac{1}{n}}(x_0) \\ 0, & \text{else,} \end{cases}$$

and let $\mu_n = f_n\lambda$ be the measure with density f_n w. r. t. the reference measure λ. This means μ_n is a probability measure supported on a ball around x_0, whose radius decreases with n. Intuitively, this sequence should converge against the measure which only has mass in x_0. Formally, this means $\mu_n \overset{*}{\rightharpoonup} \delta_{x_0}$, , which we will now prove. We have

$$\begin{aligned} \left| \int_\Omega g(x)\mathrm{d}\mu_n(x) - \int_\Omega g(x)\mathrm{d}\delta_{x_0}(x) \right| &= \left| \int_{B_{\frac{1}{n}}(x_0)} g(x) \cdot \lambda\left(B_{\frac{1}{n}}(x_0)\right)^{-1} \mathrm{d}\lambda(x) - g(x_0) \right| \\ &= \left| \int_{B_{\frac{1}{n}}(x_0)} g(x) \cdot \lambda\left(B_{\frac{1}{n}}(x_0)\right)^{-1} \mathrm{d}\lambda(x) \right. \\ &\qquad \left. - \int_{B_{\frac{1}{n}}(x_0)} g(x_0)\lambda\left(B_{\frac{1}{n}}(x_0)\right)^{-1} \mathrm{d}\lambda(x) \right| \\ &= \left| \int_{B_{\frac{1}{n}}(x_0)} (g(x) - g(x_0)) \cdot \lambda\left(B_{\frac{1}{n}}(x_0)\right)^{-1} \mathrm{d}\lambda(x) \right| \\ &\le \int_{B_{\frac{1}{n}}(x_0)} L \cdot d(x, x_0) \cdot \lambda\left(B_{\frac{1}{n}}(x_0)\right)^{-1} \mathrm{d}\lambda(x) \\ &= L \cdot \frac{1}{n} \end{aligned}$$

for all $g \in C(X)$, where L is the Lipschitz constant of g, which exists because all continuous functions on a compact space are Lipschitz continuous. This shows $\mu_n \overset{*}{\rightharpoonup} \delta_{x_0}$. But δ_{x_0} is not absolutely continuous w. r. t. λ, since $\delta_{x_0}(\{x_0\}) = 1$, but by assumption, $\lambda(\{x_0\}) = 0$.

3.3.11. Remark (Dependence on the reference measures). In contrast to the original Kantorovich problem, the regularized Kantorovich problem depends on densities with respect to a the product measure $\lambda := \lambda_1 \otimes \lambda_2$ of the reference measures λ_1 and λ_2 on Ω_1 and Ω_2, respectively. First off, we reason that $\lambda \sim \mu \otimes \nu$ can be assumed without changing the value of W_γ. We already know that $\mu \otimes \nu \ll \lambda$, and if $\mu \otimes \nu$ is zero on any set, then so is any $\pi \in \Pi(\mu, \nu)$ because of the marginal conditions. Thus, this set would not contribute to the integral W_γ, and we can assume λ to be zero on this set as well. Now let $\lambda' = \lambda'_1 \otimes \lambda'_2$ be a second reference measure. Then $\lambda' \sim \mu \otimes \nu \sim \lambda$, and thus any π with $\pi \ll \lambda$ also fulfills $\pi \ll \lambda'$. Further, the marginal conditions do not depend on the reference measure. Thus, in both problems the same transport plans are considered. For the value of the functional with reference measure λ' we have

$$\begin{aligned} \iint_{\Omega_1 \times \Omega_2} c + \gamma\left(\log\left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda'}\right) - 1\right)\mathrm{d}\pi &= \iint_{\Omega_1 \times \Omega_2} c + \gamma\left(\log\left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}\tfrac{\mathrm{d}\lambda}{\mathrm{d}\lambda'}\right) - 1\right)\mathrm{d}\pi \\ &= \iint_{\Omega_1 \times \Omega_2} c + \gamma\left(\log\left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}\right) + \log\left(\tfrac{\mathrm{d}\lambda}{\mathrm{d}\lambda'}\right) - 1\right)\mathrm{d}\pi \\ &= W_\gamma(\pi) + \gamma \iint_{\Omega_1 \times \Omega_2} \log\left(\tfrac{\mathrm{d}\lambda}{\mathrm{d}\lambda'}\right)\mathrm{d}\pi. \end{aligned}$$

The value of the functional thus differs by $\gamma \iint_{\Omega_1 \times \Omega_2} \log\left(\frac{\mathrm{d}\lambda}{\mathrm{d}\lambda'}\right)\mathrm{d}\pi$. If $\frac{\mathrm{d}\lambda}{\mathrm{d}\lambda'}$ is constant, the value of the functional only differs by a constant, and while we get a different value, the optimal transport plan is the same. However, if $\frac{\mathrm{d}\lambda}{\mathrm{d}\lambda'} = \frac{\mathrm{d}\lambda_1}{\mathrm{d}\lambda_1'}\frac{\mathrm{d}\lambda_2}{\mathrm{d}\lambda_2'}$ is not constant, choosing a different reference measure can lead to a different optimal transport plan.

We have the following alternative expression for the functional W_γ. Let

$$K_\gamma : \Omega_1 \times \Omega_2 \to \mathbb{R}, \quad K_\gamma(x,y) = \exp\left(\frac{-c(x,y)}{\gamma}\right).$$

Then

$$W_\gamma(\pi) = \gamma \mathrm{KL}(\pi \parallel K_\gamma \lambda), \tag{3.23}$$

where we ignore for now that formally, the Kullback-Leibler divergence is only defined for probability measures. We can then rewrite the regularized Kantorovich problem as

$$\inf_{\pi \in \Pi(\mu,\nu)} \gamma \mathrm{KL}(\pi \parallel K_\gamma \lambda). \tag{3.24}$$

This means the optimal transport plan is the KL-projection of the kernel K_γ onto the set $\Pi(\mu,\nu)$. This identity can easily be seen. Indeed, we have

$$\begin{aligned}
W_\gamma(\pi) &= \iint_{\Omega_1\times\Omega_2} c(x,y) + \gamma(\log\big(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\big) - 1)\mathrm{d}\pi(x,y) \\
&= \iint_{\Omega_1\times\Omega_2} -\gamma \log\left(\exp\left(\frac{-c(x,y)}{\gamma}\right)\right) + \gamma(\log\big(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\big) - 1)\mathrm{d}\pi(x,y) \\
&= \gamma \iint_{\Omega_1\times\Omega_2} \log\big(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\big) - \log\big(K_\gamma(x,y)\big) - 1\mathrm{d}\pi(x,y) \\
&= \gamma \iint_{\Omega_1\times\Omega_2} \tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\,\Big(\log\big(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\big) - \log\big(K_\gamma(x,y)\big) - 1\Big)\,\mathrm{d}\lambda(x,y) \\
&= \gamma \iint_{\Omega_1\times\Omega_2} \tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\left(\log\left(\frac{\frac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)}{K_\gamma(x,y)}\right) - 1\right)\mathrm{d}\lambda(x,y) \\
&= \gamma \mathrm{KL}(\pi \parallel K_\gamma \lambda).
\end{aligned}$$

As a last step in this section, we want to examine the support of the optimal transport plan π. We will see that in contrast to the classical Kantorovich problem, the optimal transport plan always has "full support", i. e. the support of π is equal to the product of the supports of μ and ν. The function ϕ satisfies the following properties.

3.3.12. Lemma ([42, Lemma 2.6]). For $r \in (0,\infty)$ and $d \in \mathbb{R}$,

(i) $(\phi(r+hd) - \phi(r))/h \downarrow d \log r$ for $h \downarrow 0$

(ii) $\phi(hd)/h \downarrow -\infty$ for $h \downarrow 0$.

The following lemma is an analogon to [42, Thm. 2.7].

3.3.13. Lemma. Suppose there is a feasible solution $\hat{\pi}$ to (KP-reg) with

$$\frac{\mathrm{d}\hat{\pi}}{\mathrm{d}\lambda_1 \otimes \lambda_2} > 0 \ (\lambda_1 \otimes \lambda_2)\text{-almost everywhere, i. e. } \hat{\pi} \sim \lambda_1 \otimes \lambda_2. \tag{3.25}$$

Then the unique optimal solution π_0 to (KP-reg) satisfies $\frac{\mathrm{d}\pi_0}{\mathrm{d}\lambda_1\otimes\lambda_2} > 0$ $(\lambda_1 \otimes \lambda_2)$-almost everywhere, i. e. $\pi_0 \sim \lambda_1 \otimes \lambda_2$.

Proof. Let $u_0 := \frac{\mathrm{d}\pi_0}{\mathrm{d}\lambda_1\otimes\lambda_2}$ and $\hat{u} = \frac{\mathrm{d}\hat{\pi}}{\mathrm{d}\lambda_1\otimes\lambda_2}$. Suppose the set $N = \{(x,y) \in \Omega_1 \times \Omega_2 : u_0(x,y) = 0\}$ has positive measure $\lambda_1 \otimes \lambda_2(N) > 0$. We have

$$\gamma \iint_{\Omega_1\times\Omega_2} \phi(\hat{u}(x,y))\mathrm{d}\lambda_1 \otimes \lambda_2(x,y) = W_\gamma(\hat{\pi}) - \iint_{\Omega_1\times\Omega_2} c(x,y)\mathrm{d}\hat{\pi}(x,y) \le W_\gamma(\hat{\pi}) < \infty$$

since π_0 is feasible. Thus, $\phi(\hat{u}) \in L^1(\Omega_1 \times \Omega_2, \lambda_1 \otimes \lambda_2)$. Analogously, since π_0 is optimal, $\phi(u_0) \in L^1(\Omega_1 \times \Omega_2, \lambda_1 \otimes \lambda_2)$. By Lemma 3.3.12 and since ϕ is convex, letting $h \downarrow 0$, we have

$$\begin{aligned}\phi(\hat{u}) - \phi(u_0) &\geq h^{-1}(\phi(u_0 + h(\hat{u} - u_0) - \phi(u_0))\\ &= \begin{cases} h^{-1}(\phi(h\hat{u})) & \text{a. e. on } N \\ h^{-1}(\phi(u_0 + h(\hat{u} - u_0) - \phi(u_0)) & \text{a. e. on } N^c \end{cases}\\ &\downarrow \begin{cases} -\infty & \text{a. e. on } N \\ (\hat{u} - u_0)\log u_0 & \text{a. e. on } N^c. \end{cases}\end{aligned}$$

The sequence is decreasing as $h \to 0$, so we get from the Monotone Convergence Theorem that

$$\begin{aligned}&\iint_{\Omega_1\times\Omega_2} h^{-1}(\phi(u_0 + h(\hat{u} - u_0) - \phi(u_0))\mathrm{d}\lambda_1 \otimes \lambda_2\\ &\quad= \iint_N h^{-1}(\phi(u_0 + h(\hat{u} - u_0) - \phi(u_0))\mathrm{d}\lambda_1 \otimes \lambda_2\\ &\quad\quad+ \iint_{N^c} h^{-1}(\phi(u_0 + h(\hat{u} - u_0) - \phi(u_0))\mathrm{d}\lambda_1 \otimes \lambda_2\\ &\quad\to -\infty,\end{aligned}$$

since $\iint_{\Omega_1\times\Omega_2}(\hat{u} - u_0)\log u_0 \mathrm{d}\lambda_1 \otimes \lambda_2 \leq \iint_{\Omega_1\times\Omega_2} \phi(\hat{u}) - \phi(u_0)\mathrm{d}\lambda_1 \otimes \lambda_2 < \infty$ and since N has positive measure. This means

$$\begin{aligned}&h^{-1}\big(W_\gamma(\pi_0 + h(\hat{\pi} - \pi_0)) - W_\gamma(\pi_0)\big)\\ &\quad= h^{-1}\Big(\iint_{\Omega_1\times\Omega_2} c\cdot(u_0 + h(\hat{u} - u_0)) + \gamma\phi(u_0 + h(\hat{u} - u_0))\mathrm{d}\lambda_1 \otimes \lambda_2\\ &\quad\quad\quad - \iint_{\Omega_1\times\Omega_2} cu_0 + \gamma\phi(u_0)\mathrm{d}\lambda_1 \otimes \lambda_2\Big)\\ &\quad= h^{-1}\Big(\iint_{\Omega_1\times\Omega_2} ch(\hat{u} - u_0)\mathrm{d}\lambda_1 \otimes \lambda_2\Big)\\ &\quad\quad + h^{-1}\gamma \iint_{\Omega_1\times\Omega_2} (\phi(u_0 + h(\hat{u} - u_0)) - \phi(u_0)\mathrm{d}\lambda_1 \otimes \lambda_2)\\ &\quad= \iint_{\Omega_1\times\Omega_2} c\cdot(\hat{u} - u_0) + h^{-1}\gamma(\phi(u_0 + h(\hat{u} - u_0) - \phi(u_0))\mathrm{d}\lambda_1 \otimes \lambda_2\\ &\quad\to -\infty\end{aligned}$$

as $h \to 0$, since $\iint_{\Omega_1\times\Omega_2} c(\hat{u} - u_0) < \infty$. But the optimality and uniqueness of π_0 and the feasibility of $\hat{\pi}$ imply that the quotient is positive for all $h > 0$. This is a contradiction, and thus N has to be a $(\lambda_1 \otimes \lambda_2)$-zero set. ■

3.3.14. Remark. As already noted in Remark 3.3.11, we can always take a reference measure λ with $\lambda \sim \mu \otimes \nu$. Then condition (3.25) is always satisfied, since we can choose $\hat{\pi} = \mu \otimes \nu$. Thus, the statement of Lemma 3.3.13 states that the optimal π_0 is equivalent to $\mu \otimes \nu$, which implies $\operatorname{supp}\pi_0 = \operatorname{supp}\mu \times \operatorname{supp}\nu$.

3.3.2 A Dual Formulation

We now want to find a dual formulation for (KP-reg) using Fenchel Duality. This approach is inspired by [33] and [46]. We use the formulation stated in Theorem 2.2.6. With this notation, we have $\mathcal{X}^* = \mathcal{M}(\Omega_1 \times \Omega_2), \mathcal{Y}^* = \mathcal{M}(\Omega_1) \times \mathcal{M}(\Omega_2)$ with $\mathcal{X} = C(\Omega_1 \times \Omega_2), \mathcal{Y} = C(\Omega_1) \times C(\Omega_2)$. The problem (KP-reg) is stated in the dual space, thus has to be considered as the "dual problem" in the above setting. These considerations lead to stating (KP-reg) as

$$\inf_{\pi \in \mathcal{M}(\Omega_1 \times \Omega_2)} W_\gamma(\pi) + h(-\tilde{A}\pi), \tag{3.26}$$

where

$$h : \mathcal{M}(\Omega_1) \times \mathcal{M}(\Omega_2) \to \mathbb{R}, \quad h(\left(\begin{smallmatrix} \mu' \\ \nu' \end{smallmatrix}\right)) = \begin{cases} 0, & -\mu = \mu' \text{ and } -\nu = \nu' \\ \infty, & \text{else} \end{cases}$$

is the indicator function and

$$\tilde{A} : \mathcal{M}(\Omega_1 \times \Omega_2) \to \mathcal{M}(\Omega_1) \times \mathcal{M}(\Omega_2), \quad \tilde{A}\pi = \left(\begin{smallmatrix} \mathrm{pr}_1^\# \\ \mathrm{pr}_2^\# \end{smallmatrix}\right).$$

The term $h(-\tilde{A}\pi)$ asserts that the constraints are satisfied. Now, we need to show that all the functions involved are actually conjugate functions of some other function. We note that W_γ and h are convex and lower semicontinuous w. r. t. the weak-* convergence. For W_γ this follows from Theorem 3.3.6, as seen in the proof of Theorem 3.1.5. For the indicator function h this is apparent. Thus, $W_\gamma = ({}^*W_\gamma)^*$ and $h = ({}^*h)^*$ by Proposition 2.2.2, so W_γ is the conjugate of the preconjugate ${}^*W_\gamma$ and h is the conjugate of the preconjugate ${}^*h$. So $g = {}^*W_\gamma$ and $f = {}^*h$ are the corresponding "primal functions" in Theorem 2.2.6. We then need to show that $\tilde{A}$ is the adjoint of some operator. This is the content of the following proposition.

3.3.15. Proposition. The operator

$$\tilde{A} : \mathcal{M}(\Omega_1 \times \Omega_2) \to \mathcal{M}(\Omega_1) \times \mathcal{M}(\Omega_2), \quad \tilde{A}\pi = \left(\begin{smallmatrix} (\mathrm{pr}_1)_\# \\ (\mathrm{pr}_2)_\# \end{smallmatrix}\right)$$

is the adjoint of the operator

$$A : C(\Omega_1) \times C(\Omega_2) \to C(\Omega_1 \times \Omega_2), \quad A\left(\begin{smallmatrix} v \\ w \end{smallmatrix}\right) = v \oplus w$$

for $v \in C(\Omega_1), w \in C(\Omega_2)$, where

$$(v \oplus w)(x, y) = v(x) + w(y) \quad \text{for all } x \in \Omega_1, y \in \Omega_2.$$

Proof. Denote by $\langle \cdot, \cdot \rangle$ the dual pairing between $C(\Omega_1) \times C(\Omega_2)$ and $\mathcal{M}(\Omega_1) \times \mathcal{M}(\Omega_2)$. Then

$$\begin{aligned} \langle A\left(\begin{smallmatrix} v \\ w \end{smallmatrix}\right), \pi \rangle &= \iint_{\Omega_1 \times \Omega_2} v(x) + w(y) \mathrm{d}\pi(x, y) \\ &= \iint_{\Omega_1 \times \Omega_2} v(\mathrm{pr}_1(x, y)) \mathrm{d}\pi(x, y) + \iint_{\Omega_1 \times \Omega_2} w(\mathrm{pr}_2(x, y)) \mathrm{d}\pi(x, y) \\ &= \int_{\Omega_1} v(x) \mathrm{d}\,\mathrm{pr}_1^\# \pi(x) + \int_{\Omega_2} w(y) \mathrm{d}\,\mathrm{pr}_2 \# \pi(y) \\ &= \langle \left(\begin{smallmatrix} v \\ w \end{smallmatrix}\right), \tilde{A}\pi \rangle. \end{aligned}$$

This shows the identity $\tilde{A} = A^*$. ■

We now calculate the preconjugates ${}^*h$ and ${}^*W_\gamma$.
For the indicator function, let $v \in C(\Omega_1), w \in C(\Omega_2)$. Then we have

$$ {}^*h(\left(\begin{smallmatrix} v \\ w \end{smallmatrix}\right)) = \sup_{(\mu',\nu')} \int_X v \mathrm{d}\mu' + \int_X w \mathrm{d}\nu' - \mathbb{1}_{\left(\begin{smallmatrix} -\mu \\ -\nu \end{smallmatrix}\right)}\left(\left(\begin{smallmatrix} \mu' \\ \nu' \end{smallmatrix}\right)\right) = -\int_X v \mathrm{d}\mu - \int_X w \mathrm{d}\nu, $$

since the indicator function becomes infinity if $\mu' \neq \mu, \nu' \neq \nu$.
For W_γ we have the following proposition.

3.3.16. Proposition. The Fenchel conjugate ${}^*W_\gamma$ of the function W_γ as defined in Definition 3.3.2 is given by

$$ {}^*W_\gamma(u) = \gamma \iint_{\Omega_1\times\Omega_2} \exp\left(\tfrac{u(x,y)}{\gamma}\right) K_\gamma(x,y) \mathrm{d}\lambda_1 \otimes \lambda_2(x,y) $$

for $u \in C(\Omega_1 \times \Omega_2)$.

Proof. The Fenchel conjugate is defined as

$$ {}^*W_\gamma(u) = \sup_{\pi\in\mathcal{M}(\Omega_1\times\Omega_2)} \iint_{\Omega_1\times\Omega_2} u(x,y) \mathrm{d}\pi(x,y) - W_\gamma(\pi). $$

Since $W_\gamma(\pi) = \infty$ if π is not absolutely continuous, we can replace the supremum over measures by a supremum over densities. Let $\lambda := \lambda_1 \otimes \lambda_2$. We get, using (3.23) and (3.17),

$$ \begin{aligned} {}^*W_\gamma(u) &= \sup_{r\in L^1(\Omega_1\times\Omega_2,\ \lambda)} \iint_{\Omega_1\times\Omega_2} u(x,y) r(x,y) \mathrm{d}\lambda(x,y) - \gamma \mathrm{KL}(r\lambda \parallel K_\gamma \lambda) \\ &= \sup_{r\in L^1(\Omega_1\times\Omega_2,\ \lambda)} \iint_{\Omega_1\times\Omega_2} u(x,y) r(x,y) - \gamma\left(\log\left(\frac{r(x,y)}{K_\gamma(x,y)}\right) - 1\right) r(x,y) \mathrm{d}\lambda(x,y) \\ &= \sup_{r\in L^1(\Omega_1\times\Omega_2,\ \lambda)} \langle u, r\rangle_{\infty,1} - \iint_{\Omega_1\times\Omega_2} f((x,y), r(x,y)) \mathrm{d}\lambda(x,y) \\ &= \sup_{r\in L^1(\Omega_1\times\Omega_2,\ \lambda)} \langle u, r\rangle_{\infty,1} - I_f(r) \\ &= I_f^*(u), \end{aligned} $$

for the function $f : (\Omega_1 \times \Omega_2) \times \mathbb{R} \to \mathbb{R}$ defined through $f((x,y),t) := \gamma t(\log t - \log K_\gamma(x,y) - 1)$ for $x \in \Omega_1, y \in \Omega_2, t \in \mathbb{R}$ with the usual conventions. This function fulfills all requirements for Proposition 2.2.4, thus the equality $I_f^* = I_{f^*}$ holds, where f^* is the convex conjugate of f with respect to t. Thus we only need to solve

$$ f^*((x,y),t') = \sup_{t\in\mathbb{R}} tt' - f((x,y),t), $$

which can be done by a simple analysis of extremal points, since f is differentiable for $t > 0$. If the optimal t_0 satisfies $t_0 > 0$, we must have

$$ 0 = \tfrac{\mathrm{d}}{\mathrm{d}t}\left(tt' - f((x,y),t)\right)|_{t=t_0} \quad \Leftrightarrow \quad t_0 = \exp\left(\frac{t'}{\gamma}\right) K_{\gamma,p}(x,y), \quad x \in \Omega_1, y \in \Omega_2. $$

For $t = 0$, the expression in the supremum becomes 0, and for $t < 0$ it becomes $-\infty$. Putting the optimal t_0 for $t > 0$ into the expression and taking the maximum over all possibilities gives

$$ f^*((x,y),t') = \max\left\{-\infty, 0, \gamma \exp\left(\tfrac{t'}{\gamma}\right) K_\gamma(x,y)\right\} = \gamma \exp\left(\tfrac{t'}{\gamma}\right) K_\gamma(x,y). $$

Now,

$$I_{f^*}(u) = \gamma \iint_{X\times X} \exp\Big(\tfrac{u(x,y)}{\gamma}\Big) K_\gamma(x,y) \mathrm{d}\lambda(x,y),$$

which proves the claim. ∎

Putting everything together, we can now apply Theorem 2.2.6 to $g = {}^*W_{\gamma,p}$, $f = {}^*h$ and A as in Proposition 3.3.15. Let

$$B_\gamma(v,w) := \int_{\Omega_1} v \mathrm{d}\mu + \int_{\Omega_2} w \mathrm{d}\nu - \gamma \iint_{\Omega_1\times\Omega_2} \exp\big(\tfrac{v\oplus w}{\gamma}\big) K_\gamma \mathrm{d}\lambda_1 \otimes \lambda_2 \tag{3.27}$$

for $v \in C(\Omega_1), w \in C(\Omega_2)$. Then we get the "dual formulation", which is actually a primal formulation.

$$\sup_{v\in C(\Omega_1), w\in C(\Omega_2)} B_\gamma(v,w).$$

We see that this problem is very similar to the unregularized dual problem (DP). The constraint $v \oplus w \leq c$ has been replaced by a "soft constraint"

$$\gamma \iint_{\Omega_1\times\Omega_2} \exp\big(\tfrac{v\oplus w}{\gamma}\big) K_\gamma \mathrm{d}\lambda = \gamma \iint_{\Omega_1\times\Omega_2} \exp\big(\tfrac{v\oplus w - c}{\gamma}\big) \mathrm{d}\lambda.$$

Since we are looking for v, w that maximize $B_\gamma(v,w)$, the terms $\int_{\Omega_1} v\mathrm{d}\mu$ and $\int_{\Omega_2} w\mathrm{d}\nu$ favor large v and w. However if $v \oplus w \geq c$, the argument of the exponential function is positive and it goes faster to $-\infty$ than the linear terms go to $+\infty$.

Before we assert that strong duality actually holds, we want to examine the relation between the variables v and w and the optimal π. We need the following property of the function ϕ, stated in [42, Lemma 2.4].

3.3.17. Lemma. Let $0 < r_0 \in \mathbb{R}$. Then for all $r \in \mathbb{R}$,

$$(r - r_0)\log r_0 \leq \phi(r) - \phi(r_0).$$

We have the following characterization of the optimal π.

3.3.18. Proposition. Suppose that π_0 is feasible for (KP-reg). Suppose further that there exist measurable functions $v : \Omega_1 \to \mathbb{R}$ and $w : \Omega_2 \to \mathbb{R}$ with $v \oplus w = \gamma \log\Big(\frac{\mathrm{d}\pi_0}{\mathrm{d}\lambda_1\otimes\lambda_2}\Big) + c$. Then π_0 is optimal for (KP-reg).

Proof. Let $\lambda := \lambda_1 \otimes \lambda_2$. Let $\pi \in \mathcal{M}(\Omega_1 \times \Omega_2)$ be any feasible transport plan. Since

$$\gamma \log\Big(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\Big) = v(x) + w(y) - c(x,y) \in \mathbb{R} \text{ for all } x \in \Omega_1, y \in \Omega_2,$$

$\frac{\mathrm{d}\pi_0}{\mathrm{d}\lambda} > 0 \quad \lambda$-a. e. . Thus, by Lemma 3.3.17, we have

$$\phi\Big(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\Big) - \phi\Big(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\Big) \geq \Big(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y) - \tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\Big)\log\Big(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\Big) \quad \lambda\text{-a. e. .}$$

This means

$$\begin{aligned}
W_{\gamma,p}(\pi) - W_{\gamma,p}(\pi_0) &= \iint_{\Omega_1\times\Omega_2} c(x,y)\mathrm{d}(\pi-\pi_0)(x,y) + \gamma h\left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}\right) - \gamma h\left(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}\right)\\
&= \langle c, \pi-\pi_0\rangle + \gamma \iint_{\Omega_1\times\Omega_2} \phi\left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y)\right) - \phi\left(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\right)\mathrm{d}\lambda(x,y)\\
&\geq \langle c, \pi-\pi_0\rangle + \gamma \iint_{\Omega_1\times\Omega_2} \left(\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y) - \tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\right)\log\left(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\right)\mathrm{d}\lambda(x,y)\\
&= \langle c, \pi-\pi_0\rangle + \gamma \iint_{\Omega_1\times\Omega_2} \log\left(\tfrac{\mathrm{d}\pi_0}{\mathrm{d}\lambda}(x,y)\right)\mathrm{d}(\pi-\pi_0)(x,y)\\
&= \langle c, \pi-\pi_0\rangle + \iint_{\Omega_1\times\Omega_2} v(x) + w(y) - c(x,y)\mathrm{d}(\pi-\pi_0)(x,y)\\
&= \iint_{\Omega_1\times\Omega_2} v(x) + w(y)\mathrm{d}(\pi-\pi_0)(x,y)\\
&= \int_{\Omega_1} v(x)\mathrm{d}(\mathrm{pr}_1^\# \pi - \mathrm{pr}_1^\# \pi_0)(x) + \int_{\Omega_2} w(y)\mathrm{d}(\mathrm{pr}_2^\# \pi - \mathrm{pr}_2^\# \pi_0)(y)\\
&= 0,
\end{aligned}$$

where $\langle\cdot,\cdot\rangle$ denotes the dual pairing between $C(\Omega_1\times\Omega_2)$ and $\mathcal{M}(\Omega_1\times\Omega_2)$. This shows

$$W_{\gamma,p}(\pi) \geq W_{\gamma,p}(\pi_0)$$

for every feasible π and thus π_0 is optimal. ■

We summarize everything in the following theorem. We will see that the functions v and w from the above proposition are exactly the dual optimizers, if they exist.

3.3.19. Theorem (Dual problem). The dual problem of (KP-reg) is

$$\sup_{\substack{v\in C(\Omega_1),\\ w\in C(\Omega_2)}} \int_{\Omega_1} v\mathrm{d}\mu + \int_{\Omega_2} w\mathrm{d}\nu - \gamma\iint_{\Omega_1\times\Omega_2} \exp\left(\tfrac{v\oplus w}{\gamma}\right)K_\gamma \mathrm{d}\lambda_1\otimes\lambda_2, \tag{D-reg}$$

where $(v\oplus w)(x,y) = v(x)+w(y)$ for all $x\in\Omega_1, y\in\Omega_2$ and $K_\gamma(x,y) = \exp\left(\frac{-c(x,y)}{\gamma}\right)$. Strong duality holds, i. e. $\min(\text{KP-reg}) = \sup(\text{D-reg})$. Further, the the density $\frac{\mathrm{d}\pi}{\mathrm{d}\lambda_1\otimes\lambda_2}$ of the optimal primal variable can be recovered from the optimal dual variables v, w through

$$\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda_1\otimes\lambda_2}(x,y) = \exp\left(\tfrac{v(x)}{\gamma}\right)K_{\gamma,p}(x,y)\exp\left(\tfrac{w(y)}{\gamma}\right). \tag{3.28}$$

Conversely, if the optimal π is given (3.28), then v and w are optimal dual variables. The optimal dual variables are unique up to a constant shift, i. e. if (v_1, w_1) and (v_2, w_2) are pairs of dual optimizers, then $(v_1 - v_2, w_1 - w_2) = (C, -C)$ for some constant $C\in\mathbb{R}$.

Proof. From the above considerations, using Theorem 2.2.6, we get $\sup(\text{D-reg}) \leq \min(\text{KP-reg})$. It remains to show that the continuity condition in Theorem 2.2.6 is satisfied. The function g to consider is

$$g(u) = \gamma\iint_{\Omega_1\times\Omega_2} \exp\left(\tfrac{u(x,y)}{\gamma}\right)K_\gamma(x,y)\mathrm{d}\lambda(x,y)$$

for $u\in C(\Omega_1\times\Omega_2)$, and continuity is w. r. t. uniform convergence. Now, if $(u_n)_n \to u$ in $C(\Omega_1\times\Omega_2)$ is uniformly convergent, then so is $\exp(\frac{u_n}{\gamma})K_\gamma$, since the exponential function is continuous and K_γ is bounded. This means that we can switch the limit and the integral, so g is continuous everywhere. Thus, strong duality holds.

Now, assume that there exist optimal dual variables v and w. Define

$$u(x,y) := \exp\left(\tfrac{v(x)}{\gamma}\right)K_\gamma(x,y)\exp\left(\tfrac{w(y)}{\gamma}\right) = \exp\left(\tfrac{v(x)+w(y)-c(x,y)}{\gamma}\right).$$

Consider any smooth test function $\varphi \in C(\Omega_1)$. The optimality of v implies

$$\begin{aligned} 0 &= \tfrac{\mathrm{d}}{\mathrm{d}t}B_\gamma(v+t\varphi,w)|_{t=0} \\ &= \int_{\Omega_1}\varphi(x)\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x)\mathrm{d}\lambda_1(x) - \iint_{\Omega_1\times\Omega_2}\varphi(x)\exp\left(\tfrac{v(x)+w(y)}{\gamma}\right)K_{\gamma,p}(x,y)\mathrm{d}\lambda_1\otimes\lambda_2(x,y) \\ &= \int_{\Omega_1}\varphi(x)\left(\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x) - \int_{\Omega_2}\exp\left(\tfrac{v(x)+w(y)}{\gamma}\right)K_\gamma(x,y)\mathrm{d}\lambda_2(y)\right)\mathrm{d}\lambda_1(x). \end{aligned}$$

By the fundamental lemma of calculus of variations, this implies

$$0 = \tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x) - \int_{\Omega_2}\exp\left(\tfrac{v(x)+w(y)}{\gamma}\right)K_{\gamma,p}(x,y)\mathrm{d}\lambda_2(y)) \quad \lambda_1\text{-a. e. on } \Omega_1.$$

This means

$$\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x) = \int_{\Omega_2}\exp\left(\tfrac{v(x)+w(y)}{\gamma}\right)K_\gamma(x,y)\mathrm{d}\lambda_2(y) = \int_{\Omega_2}u(x,y)\mathrm{d}\lambda_1\otimes\lambda_2(y), \tag{3.29}$$

in terms of measures

$$\mu = \mathrm{pr}_1^\#(u\lambda).$$

Analogously, one shows

$$\nu = \mathrm{pr}_2^\#(u\lambda),$$

which shows that $u\lambda$ satisfies the marginal conditions and is thus feasible for (KP-reg). Further we have $v(x)+w(y) = \gamma\log u(x,y) + c(x,y)$ and thus by Proposition 3.3.18, $u\lambda$ is optimal. Conversely, if $u_0\lambda$ is optimal and $v_0(x)+w_0(y) = \gamma\log u_0(x,y)+c(x,y)$ for some $v_0 \in C(\Omega_1), w_0 \in C(\Omega_2)$, then

$$\tfrac{\mathrm{d}}{\mathrm{d}t}B_\gamma(v_0+t\varphi,w_0)|_{t=0} = 0,$$

since the marginal conditions are satisfied. Similarly for w_0. Thus, v_0 are w_0 are optimal. Now assume we have optimal dual variables v_1, v_2, w_1, w_2. Then, since the optimal u is unique (up to a $\lambda_1\otimes\lambda_2$-null set),

$$v_1(x)+w_1(y) = \gamma\log u(x,y)+c(x,y) = v_2(x)+w_2(y) \quad \lambda_1\otimes\lambda_2\text{-a. e. for } x\in\Omega_1, y\in\Omega_2$$

and thus

$$v_1(x)-v_2(x) = w_2(y)-w_1(y) \quad \lambda\text{-a. e. for } x\in\Omega_1, y\in\Omega_2,$$

which shows the uniqueness of the optimal dual variables up to a constant shift. ■

Since we know that the measures μ and ν are absolutely continuous with respect to λ_1 and λ_2, respectively, we can also formulate the dual problem in terms of densities, using duality between $L^1(\Omega_i,\lambda_i)$ and $L^\infty(\Omega_i,\lambda_i)$ for $i = 1,2$.

3.3.20. Theorem (Dual problem Using Densities). Let $r = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}$ and $s = \frac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}$. The dual problem of (KP-reg) in terms of r and s is

$$\sup_{\substack{v \in L^\infty(\Omega_1,\lambda_1),\\ w \in L^\infty(\Omega_2,\lambda_2)}} \int_{\Omega_1} rv \mathrm{d}\lambda_1 + \int_{\Omega_2} sw \mathrm{d}\lambda_2 - \gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v \oplus w}{\gamma}\right) K_\gamma \mathrm{d}\lambda, \tag{D-reg$'$}$$

where $(v \oplus w)(x,y) = v(x) + w(y)$ for all $x \in \Omega_1, y \in \Omega_2$ and $K_\gamma(x,y) = \exp\left(\frac{-c(x,y)}{\gamma}\right)$. Strong duality holds, i. e. $\min(\text{KP-reg}) = \sup(\text{D-reg})$.

Proof. This proof is inspired by [46]. We show that the optimal cost of (D-reg′) is equal to the optimal cost of (D-reg). Let

$$B'_\gamma(v,w) = \int_{\Omega_1} rv \mathrm{d}\lambda_1 + \int_{\Omega_2} sw \mathrm{d}\lambda_2 - \gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v \oplus w}{\gamma}\right) K_\gamma \mathrm{d}\lambda$$

for $v \in L^\infty(\Omega_1), w \in L^\infty(\Omega_2)$. Since $C(\Omega_i) \subseteq L^\infty(\Omega_i)$ for $i = 1, 2$, we have

$$\begin{aligned}
\sup_{\substack{v \in L^\infty(\Omega_1),\\ w \in L^\infty(\Omega_1)}} B'_\gamma(v,w) &= \sup_{\substack{v \in L^\infty(\Omega_1),\\ w \in L^\infty(\Omega_1)}} \int_{\Omega_1} rv \mathrm{d}\lambda_1 + \int_{\Omega_2} sw \mathrm{d}\lambda_2 - \gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v \oplus w}{\gamma}\right) K_\gamma \mathrm{d}\lambda \\
&\geq \sup_{\substack{v \in C(\Omega_1),\\ w \in C(\Omega_2)}} \int_{\Omega_1} rv \mathrm{d}\lambda_1 + \int_{\Omega_2} sw \mathrm{d}\lambda_2 - \gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v \oplus w}{\gamma}\right) K_\gamma \mathrm{d}\lambda \\
&= \sup_{\substack{v \in C(\Omega_1),\\ w \in C(\Omega_2)}} \int_{\Omega_1} v \mathrm{d}\mu + \int_{\Omega_2} w \mathrm{d}\nu - \gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v \oplus w}{\gamma}\right) K_\gamma \mathrm{d}\lambda \\
&= \sup_{\substack{v \in C(\Omega_1),\\ w \in C(\Omega_2)}} B_\gamma(v,w)
\end{aligned}$$

We need to show that the optimal cost of (D-reg′) can be approximated by continuous functions. For this we approximate functions $v \in L^\infty(\Omega_1), w \in L^\infty(\Omega_2)$ by continuous functions. Since $C(\Omega_1)$ and $C(\Omega_2)$ are dense in $L^1(\Omega_1)$ and $L^1(\Omega_2)$, respectively (see [22, Thm. 3.14]), we have sequences $(v_n)_n \subseteq C(\Omega_1), (w_n)_n \subseteq C(\Omega_2)$ such that $v_n \to v$ in $L^1(\Omega_1)$ and $w_n \to w$ in $L^1(\Omega_2)$. Since the limits v, w are in $L^\infty(\Omega_1)$ and $L^\infty(\Omega_2)$, respectively, they are bounded. We can thus assume $(v_n)_n$ and $(w_n)_n$ to be uniformly bounded by some constant $C < \infty$, by cutting off too large values if necessary. By the sequential Banach-Alaoglu theorem, we can then assume (by restricting to a subsequence) that $v_n \overset{*}{\rightharpoonup} v$ in $L^\infty(\Omega_1) = L^1(\Omega_1)^*$ and $w_n \overset{*}{\rightharpoonup} w$ in $L^\infty(\Omega_2) = L^1(\Omega_2)^*$. We get

$$\int_{\Omega_1} rv_n \mathrm{d}\lambda_1 + \int_{\Omega_2} sw_n \mathrm{d}\lambda_2 \to \int_{\Omega_1} rv \mathrm{d}\lambda_1 + \int_{\Omega_2} sw \mathrm{d}\lambda_2$$

by the weak-* convergence. We can further assume (by again restricting to a subsequence), that $v_n \to v$ pointwise almost everywhere and $w_n \to w$ pointwise almost everywhere, see [22, Thm. 3.12]. Since the exponential function is continuous, $\exp\left(\frac{v_n \oplus w_n}{\gamma}\right) \to \exp\left(\frac{v \oplus w}{\gamma}\right)$ pointwise almost everywhere. Further, v_n and w_n are uniformly bounded, and thus $\exp\left(\frac{v_n \oplus w_n}{\gamma}\right)$ is uniformly bounded. By the Dominated Convergence Theorem ([22, Thm. 1.34]), we then get

$$\gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v_n \oplus w_n}{\gamma}\right) K_\gamma \mathrm{d}\lambda_1 \otimes \lambda_2 \to \gamma \iint_{\Omega_1 \times \Omega_2} \exp\left(\tfrac{v \oplus w}{\gamma}\right) K_\gamma \mathrm{d}\lambda_1 \otimes \lambda_2.$$

In total, this shows

$$B'_\gamma(v_n, w_n) \to B'_\gamma(v,w) = B_\gamma(v,w),$$

which concludes the proof. ■

The next step will be to examine whether the supremum in the dual problem is attained. We have the following result.

3.3.21. Proposition (Existence of dual optimizers). Let $\mu \in \mathcal{M}_+(\Omega_1)$ and $\nu \in \mathcal{M}_+(\Omega_2)$ with finite entropy. Then there exist measurable functions $a : \Omega_1 \to \mathbb{R}$ and $b : \Omega_2 \to \mathbb{R}$ such that the density of the optimal π for (KP) is given as

$$\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x,y) = a(x)K_\gamma(x,y)b(x) \quad (\lambda_1 \otimes \lambda_2)\text{-a. e.}\,.$$

If further (3.25) is satisfied, $v = \gamma \log a$ and $w = \gamma \log b$ satisfy

$$\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda}(x,y) = \exp\big(\tfrac{v(x)}{\gamma}\big)K_\gamma(x,y)\exp\big(\tfrac{w(y)}{\gamma}\big) \quad (\lambda_1 \otimes \lambda_2)\text{-a. e.}\,.$$

Proof. If we consider the the Kantorovich problem (KP) as in (3.23) as the KL-projection of the kernel function K_γ on the set of measures with fixed marginals, the existence of a and b follows from [47, Thm. 3]. Now if (3.25) is satisfied, Lemma 3.3.13 assures that

$$\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x,y) = a(x)K_\gamma(x,y)b(y) > 0 \; \lambda_1 \otimes \lambda_2\text{-a. e.}\,,$$

thus $a > 0$ and $b > 0$ and $\log a$ and $\log b$ are finite $\lambda_1 \otimes \lambda_2$-a. e. . ■

Now, together with Proposition 3.3.18, we have that π is optimal if and only if

$$\tfrac{\mathrm{d}\pi}{\mathrm{d}\lambda_1 \otimes \lambda_2}(x,y) = \exp\big(\tfrac{v(x)}{\gamma}\big)K_\gamma(x,y)\exp\big(\tfrac{w(y)}{\gamma}\big) \; (\lambda_1 \otimes \lambda_2)\text{-a. e.}$$

for some measurable functions $v : \Omega_1 \to \mathbb{R}$ and $w : \Omega_2 \to \mathbb{R}$.

We can now ask ourselves if those functions are continuous or bounded. This is not the case in general, which can be seen in the following example.

3.3.22. Example (Dual problem has no solution). We will now consider two examples where the dual problem admits no solution. Assume that the dual problem for some marginals μ, ν admits a solution $v \in C(\Omega_1), w \in C(\Omega_2)$. Then the optimality condition (3.29) for v is

$$\begin{aligned}\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda_1}(x) &= \int_{\Omega_2} \exp\big(\tfrac{v(x)+w(y)}{\gamma}\big)K_{\gamma,p}(x,y)\mathrm{d}\lambda_2(y)\\ &= \exp\big(\tfrac{v(x)}{\gamma}\big)\int_{\Omega_2} \exp\big(\tfrac{w(y)}{\gamma}\big)K_{\gamma,p}(x,y)\mathrm{d}\lambda_2(y).\end{aligned}$$

Let $z(x) = \int_{\Omega_2} \exp\big(\tfrac{w(y)}{\gamma}\big)K_{\gamma,p}(x,y)\mathrm{d}\lambda_2(y)$. Now, let $\Omega_1 = \Omega_2 = [0,1], \lambda = \lambda_1 = \lambda_2$ the Lebesgue measure and let μ be defined by the density

$$\tfrac{\mathrm{d}\mu}{\mathrm{d}\lambda}(x) = \begin{cases} 0.5, & x \leq 0.5 \\ 1.5, & x > 0.5. \end{cases}$$

We have $h(\mu) \approx -0.87$, so a solution to the primal problem exists. We get

$$v(x) = \gamma \log\big(\tfrac{\mathrm{d}\mu_2}{\mathrm{d}\lambda}(x)\big)z(x)^{-1} = \begin{cases} \gamma\frac{\log 0.5}{z(x)}, & x \leq 0.5 \\ \gamma\frac{\log 1.5}{z(x)}, & x > 0.5. \end{cases}$$

But since $z(x) > 0$ for all $x \in [0,1]$ and $\log(0.5) < 0$ while $\log(1.5) > 0$, we see $v(x) < 0$ for $x \leq 0.5$ and $v(x) > 0$ for $x > 0.5$, so v is not a continuous function. Thus, (D-reg) does not admit an optimizer. Moreover, existence does not fail just because of the continuity, which we can see in the next example. Let μ be defined by the density $\frac{\mathrm{d}\mu}{\mathrm{d}\lambda}(x) = \exp\left(\frac{-1}{\sqrt{x}}\right)$. Again, μ has finite entropy $h(\mu) \approx -0.52$. If we assume $w \in L^\infty([0,1])$, then z is bounded. Now,

$$v(x) = \gamma \log\left(\frac{\mathrm{d}\mu}{\mathrm{d}\lambda}(x)\right) z(x)^{-1} = \gamma \frac{-1}{\sqrt{x} z(x)}$$

and thus

$$v(x) \to -\infty \text{ as } x \to 0.$$

Thus, v cannot be in $L^\infty([0,1])$ and (D-reg′) does not admit an optimizer either.

However, if Ω_1 and Ω_2 are finite, all measurable functions are bounded by the maximal attained value, which can be selected as a maximum over a finite set. Thus, the dual problem (D-reg′) has a solution in the finite case because of Proposition 3.3.21.

This lies the basis for an algorithm to compute the optimal transport plan.

3.3.3 Solving the Regularized Kantorovich Problem using Alternate Projections

In this section we will state an algorithm for solving the regularized Kantorovich problem defined in Definition 3.3.2, using as a reference [48]. It relies on the expression of the density u of the optimal transport plan shown in Theorem 3.3.19 and the marginal conditions. We will first restate these results.

Denote the density of the optimal transport plan by u and the densities of the marginals μ and ν by r and s, respectively, i. e. $\pi = u(\lambda_1 \otimes \lambda_2), \mu = r\lambda_1, \nu = s\lambda_2$. Remember that the density of the optimal transport plan can be expressed as

$$u(x,y) = \exp\left(\frac{v(x)}{\gamma}\right) K_{\gamma,p}(x,y) \exp\left(\frac{w(y)}{\gamma}\right) \tag{3.28}$$

for some measurable functions v, w. Now, let $a = \exp\left(\frac{v}{\gamma}\right)$ and $b = \exp\left(\frac{w}{\gamma}\right)$. Then

$$u(x,y) = a(x) K_{\gamma,p}(x,y) b(y). \tag{3.30}$$

The marginal conditions, or optimality conditions for v, w in the dual problem, then are

$$\begin{aligned} a(x) \int_{\Omega_2} K_\gamma(x,y) b(y) \mathrm{d}\lambda_2(y) &= r(x) \quad \lambda_1\text{-a. e.} \\ b(y) \int_{\Omega_1} K_\gamma(x,y) a(x) \mathrm{d}\lambda_1(x) &= s(y) \quad \lambda_2\text{-a. e.}, \end{aligned} \tag{3.31}$$

see (3.21). These equations are often called *Schrödinger equations* in the literature.

The idea of the algorithm is to iteratively reestimate a and b by enforcing alternately the constraint on the first and the second marginal of the density associated to a and b through u. This means we construct a sequence $u^{(0)}, u^{(1)}, u^{(2)}, \ldots$ such that for the even elements of the sequence, the first marginal density is r, and for the odd elements, the second marginal density is s. This can be done by alternately solving the equations in (3.31) for a and b.

The resulting algorithm is known in many forms and can be found under the names "Iterative Proportional Fitting Procedure (IPFP)", "Sinkhorn-Knopp algorithm" (in the discrete case) or "Iterative Bregman Projections", where the last comes from the interpretation of projecting (with respect to the KL-divergence) the transport plan alternately on the sets of probability measures with one fixed marginal. Explicitly, the algorithm consists of the following recursion:

$$\begin{aligned} b_0(y) &= 1, \\ a_0(x) &= \frac{r(x)}{\int_{\Omega_2} K_\gamma(x,y)\mathrm{d}\lambda_2(y)}, \\ b_1(y) &= \frac{s(y)}{\int_{\Omega_1} K_\gamma(x,y)a_0(x)\mathrm{d}\lambda_1(x)}, \\ a_1(x) &= \frac{r(x)}{\int_{\Omega_2} K_\gamma(x,y)b_1(y)\mathrm{d}\lambda_2(y)}, \end{aligned} \tag{3.32}$$

and for arbitrary n,

$$\begin{aligned} b_n(y) &= \frac{s(y)}{\int_{\Omega_1} K_\gamma(x,y)a_{n-1}(x)\mathrm{d}\lambda_1(x)}, \\ a_n(x) &= \frac{r(x)}{\int_{\Omega_2} K_\gamma(x,y)b_n(y)\mathrm{d}\lambda_2(y)}. \end{aligned} \tag{3.33}$$

Note that for all n, $a_n > 0$ λ_1-a. e. and $b_n > 0$ λ_2-a. e. , since $K_\gamma > 0$. Thus, the quotients are well defined. The corresponding sequence of transport plan densities is defined by

$$\begin{aligned} u^{(2n)}(x,y) &= a_n(x)K_{\gamma,p}(x,y)b_n(y), \\ u^{(2n+1)}(x,y) &= a_n(x)K_{\gamma,p}b_{n+1}(y), \quad n \geq 0. \end{aligned} \tag{3.34}$$

As described above, we have for the marginal densities

$$\begin{aligned} \int_{\Omega_2} u^{(2n)}(x,y)\mathrm{d}\lambda_2(y) &= \int_{\Omega_2} a_n(x)K_\gamma(x,y)b_n(y)\mathrm{d}\lambda_2(y) \\ &= a_n(x)\int_{\Omega_2} K_\gamma(x,y)b_n(y)\mathrm{d}\lambda_2(y) \\ &= \frac{r(x)}{\int_{\Omega_2} K_\gamma(x,y)b_n(y)\mathrm{d}\lambda_2(y)}\int_{\Omega_2} K_\gamma(x,y)b_n(y)\mathrm{d}\lambda_2(y) \\ &= r(x), \end{aligned}$$

which shows $\mathrm{pr}_1^\#\big(u^{(2n)}\lambda_1 \otimes \lambda_2\big) = r\lambda_1$, see (3.21). Analogously,

$$\int_{\Omega_1} u^{(2n+1)}(x,y)\mathrm{d}\lambda_1(x) = s(y).$$

The convergence of this scheme is proved in [48] under very mild assumptions.

3.3.23. Theorem (Convergence of IPFP). If s is bounded, then the sequence $(u^{(n)}\lambda_1 \otimes \lambda_2)_n$ converges to an optimal transport plan $u\lambda_1 \otimes \lambda_2$ in total variation.

Proof. This follows from [48, Thm. 3.5] and because K_γ is continuous and thus bounded. ■

3.3.24. Remark. Formally, the measure with respect to which the KL projection is considered has to be a probability measure, which is not the case for $K_\gamma\lambda$. This can be alleviated by rescaling, which only changes the value of the functional W_γ by a constant, as seen in Lemma 3.3.4.

This algorithm can now be used to calculate a solution to the regularized Kantorovich problem (KP-reg). Since it is possible to apply the same regularization to the Kantorovich problem for unbalanced mass transport ($\widetilde{\mathrm{KP}}$), the algorithm can also be applied for unbalanced mass transport.

3.4 The Entropy-Regularized Wasserstein Distance and its Properties

In the previous section, we have considered the regularized Kantorovich problem. We showed that this problem admits a unique solution and stated an algorithm to solve it. Now we can apply this regularization to approximate the Wasserstein distance by using considering a ground metric to a specific power as the cost function in the Kantorovich problem. We formally define the *entropy-regularized Wasserstein distance* and examine some of its properties, including metric properties as well as some properties that make it suitable as a loss function, notably differentiability and convexity.

3.4.1. Definition (Entropy-Regularized Wasserstein Distance). Let (Ω, d) be a compact metric space. Let λ_1, λ_2 be two measures on Ω. The *entropy-regularized Wasserstein distance (ERWD) of order p* between two probability measures μ and ν is defined as

$$\mathcal{W}^p_{\gamma,p}(\mu,\nu) := \inf\left\{W_{\gamma,p} \ : \ \pi \in \Pi(\mu,\nu), \pi \ll \lambda_1 \otimes \lambda_2\right\} \tag{3.35}$$

where

$$W_{\gamma,p} := \iint_{X\times X} d(x,y)^p \mathrm{d}\pi(x,y) + \gamma h(\pi).$$

First, we want to examine if the ERWD actually deserves the name "distance". For several reasons, the answer to this question is negative. The first reason is that the value $\mathcal{W}^p_{\gamma,p}(\mu,\nu)$ defined above is not actually guaranteed to be positive for all γ. This is because

$$\iint_{\Omega\times\Omega} d(x,y)^p \mathrm{d}\pi(x,y) \leq \operatorname{diam}(\Omega)^p \pi(\Omega\times\Omega) = \operatorname{diam}(\Omega)^p,$$

but if $h(\pi)$ is negative, γ can be such that $-\gamma h(\pi) \geq \operatorname{diam}(\Omega)^p$. A concrete example follows.

3.4.2. Example (ERWD can be negative). Let Ω be a finite metric space with $\lambda_1 = \lambda_2$ the counting measure. Let $a, b \in \Omega$ and $\mu = \delta_a, \nu = \delta_b$. Then the only transport plan is given by $\pi = \delta_{(a,b)}$. The density of the Dirac measure $\delta_{(a,b)}$ with respect to the counting measure on the product space is the function

$$f_{\delta_x}(x') = \begin{cases} 1, & x' = x \\ 0, & \text{else.} \end{cases}$$

Thus,

$$h(\pi) = \iint_{\Omega\times\Omega} f_{\delta_x}(x,y)(\log\left(f_{\delta_x}\right)(x,y) - 1)\mathrm{d}\lambda_1 \otimes \lambda_2(x,y) = -1$$

and

$$W_\gamma(\pi) = d(a,b)^p - \gamma,$$

which is negative for $\gamma > d(a,b)^p$.

The problem of negative values can easily be bypassed by adding a constant, since the entropy function φ is bounded from below by -1. A more accurate way to assure non-negativity is to subtract the minimum of the functional $W_{\gamma,p}$. This minimum can be calculated by making use of the properties of the Kullback-Leibler divergence stated in Section 3.3.

3.4.3. Corollary. For any two probability measures $\mu, \nu \in \mathcal{P}(\Omega)$ we have

$$\mathcal{W}_{\gamma,p}(\mu,\nu) \geq -\gamma\Big(1 + \log\Big(\mathrm{vol}_{K_{\gamma,p}}\Big)\Big),$$

where

$$\mathrm{vol}_{K_{\gamma,p}} = (K_{\gamma,p}\lambda_1 \otimes \lambda_2)(\Omega \times \Omega) = \iint_{\Omega\times\Omega} K_{\gamma,p}(x,y)\mathrm{d}\lambda_1 \otimes \lambda_2(x,y)$$

with equality if and only if

$$\frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1} = \frac{1}{\mathrm{vol}_{K_{\gamma,p}}} \int_\Omega K_{\gamma,p}(\cdot,y)\mathrm{d}\lambda_2(y) \quad \text{and} \quad \frac{\mathrm{d}\nu}{\mathrm{d}\lambda_2} = \frac{1}{\mathrm{vol}_{K_{\gamma,p}}} \int_\Omega K_{\gamma,p}(x,\cdot)\mathrm{d}\lambda_1(x).$$

Proof. Let π be the optimal transport plan between μ and ν. From (3.23) we know that

$$\mathcal{W}_{\gamma,p}(\mu,\nu) = W_{\gamma,p}(\pi) = \gamma \mathrm{KL}(\pi \parallel K_{\gamma,p}\lambda_1 \otimes \lambda_2)$$

and from Lemma 3.3.4 we know that

$$\mathrm{KL}(\pi \parallel K_{\gamma,p}\lambda_1 \otimes \lambda_2) \geq -1 - \log\Big((K_{\gamma,p}\lambda_1 \otimes \lambda_2)(\Omega\times\Omega)\Big) = -1 - \log\Big(\mathrm{vol}_{K_{\gamma,p}}\Big)$$

with equality if and only if

$$\pi = \frac{1}{\mathrm{vol}_{K_{\gamma,p}}} K_{\gamma,p}\lambda_1 \otimes \lambda_2.$$

This is the case if and only if μ and ν are the marginals of $\frac{1}{\mathrm{vol}_{K_{\gamma,p}}} K_{\gamma,p}\lambda_1 \otimes \lambda_2$, which is the case if

$$\frac{\mathrm{d}\mu}{\mathrm{d}\lambda_1} = \frac{1}{\mathrm{vol}_{K_{\gamma,p}}} \int_\Omega K_{\gamma,p}(\cdot,y)\mathrm{d}\lambda_2(y),$$

and analogously for $\frac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}$. Further, π is indeed optimal in this case, since $W_{\gamma,p}$ assumes its minimum. ■

This suggests that the ERWD between two identical measures might not be zero. Indeed, we will see later that the ERWD is not even minimal for two identical measures, i. e. there exists for every $\mu \in \mathcal{P}(\Omega)$ a $\tilde{\mu} \in \mathcal{P}(\Omega)$ such that

$$\mathcal{W}_{\gamma,p}(\mu,\tilde{\mu}) < \mathcal{W}_{\gamma,p}(\mu,\mu).$$

A second reason why the ERWD does not really deserve the name distance is that the triangle inequality is only fulfilled approximately, as noted in [41, §4]. The symmetry however holds.

3.4.4. Proposition. If $\lambda_1 = \lambda_2$, the regularized Wasserstein distance is symmetric. For all $\mu, \nu \in \mathcal{P}(\Omega)$ with finite entropy, all $\gamma > 0$ and all $1 \leq p < \infty$,

$$\mathcal{W}_{\gamma,p}(\mu,\nu) = \mathcal{W}_{\gamma,p}(\nu,\mu).$$

Proof. Let $\pi \in \Pi(\mu,\nu)$ be the optimal transport plan between μ and ν. The idea is to "transpose" π. Let $\lambda := \lambda_1 = \lambda_2$. Let $u := \frac{\mathrm{d}\pi}{\mathrm{d}\lambda\otimes\lambda}$. Define $u^t(x,y) := u(y,x)$ and $\pi^t = u^(t\lambda \otimes \lambda)$. Then

$$\int_\Omega u^t(x,y)\mathrm{d}\lambda(y) = \int_\Omega u(y,x)\mathrm{d}\lambda(y) = \frac{\mathrm{d}\nu}{\mathrm{d}\lambda}(x)$$

and

$$\int_\Omega u^t(x,y)\mathrm{d}\lambda(x) = \int_\Omega u(y,x)\mathrm{d}\lambda(x) = \frac{\mathrm{d}\mu}{\mathrm{d}\lambda}(y),$$

so $\pi^t \in \Pi(\nu, \mu)$. Now,

$$\begin{aligned}W_{\gamma,p}(\pi^t) &= \iint_{\Omega\times\Omega} d(x,y)^p u^t(x,y) + \gamma\big(\log\big(u^t(x,y)\big) - 1\big)u^t(x,y)\mathrm{d}\lambda \otimes \lambda(x,y)\\ &= \iint_{\Omega\times\Omega} d(y,x)^p u(y,x) + \gamma(\log(u(y,x)) - 1)u(y,x)\mathrm{d}\lambda(x)\mathrm{d}\lambda(y)\\ &= \iint_{\Omega\times\Omega} d(y,x)^p u(y,x) + \gamma(\log(u(y,x)) - 1)u(y,x)\mathrm{d}\lambda(y)\mathrm{d}\lambda(x)\\ &= W_{\gamma,p}(\pi).\end{aligned}$$

This shows $\mathcal{W}_{\gamma,p}(\nu,\mu) \le \mathcal{W}_{\gamma,p}(\mu,\nu)$. By swapping the roles of μ and ν, we also have $\mathcal{W}_{\gamma,p}(\mu,\nu) \le \mathcal{W}_{\gamma,p}(\nu,\mu)$, and thus $\mathcal{W}_{\gamma,p}(\nu,\mu) = \mathcal{W}_{\gamma,p}(\mu,\nu)$. ∎

3.4.5. Proposition (Convexity). Let $\mu \in \mathcal{P}(\Omega)$. Then the functions

$$\mathcal{W}^{\mu}_{\gamma,p} : \mathcal{P}(\Omega) \to \mathbb{R}, \nu \mapsto \mathcal{W}_{\gamma,p}(\mu,\nu) \quad \text{and} \quad \mathcal{W}^{\nu}_{\gamma,p} : \mathcal{P}(\Omega) \to \mathbb{R}, \nu \mapsto \mathcal{W}_{\gamma,p}(\nu,\mu)$$

are convex functions.

Proof. Since

$$\mathcal{W}_{\gamma,p}(\mu,\nu) = \sup_{v,w\in C(\Omega)} \int_\Omega v\mathrm{d}\mu + \int_\Omega w\mathrm{d}\nu - \gamma \iint_{\Omega\times\Omega} \exp\left(\tfrac{v\oplus w}{\gamma}\right),$$

the function $\mathcal{W}_{\gamma,p}$ is convex as a supremum of affine functions. This implies the convexity of $\mathcal{W}^{\mu}_{\gamma,p}$ and $\mathcal{W}^{\nu}_{\gamma,p}$. ∎

A very nice property of the ERWD is its differentiability.

3.4.6. Corollary. Let $\mu,\nu \in \mathcal{P}(\Omega)$. Then

$$\partial\mathcal{W}_{\gamma,p}(\mu,\nu) = \{(v,w) \in C(\Omega)\times C(\Omega) \;:\; (v,w) \text{ are optimizers of (D-reg)}\}.$$

The elements in the subdifferential only differ by a constant shift, i. e. for each (v_1,w_1) and (v_2,w_2) in the subdifferential, $(v_1 - v_2, w_1 - w_2) = (C, -C)$ for some constant $C \in \mathbb{R}$.

Proof. The dual problem

$$\begin{aligned}&\sup_{\substack{v\in C(\Omega),\\ w\in C(\Omega)}} \int_\Omega v\mathrm{d}\mu + \int_\Omega w\mathrm{d}\nu - \gamma \iint_{\Omega\times\Omega} \exp\left(\tfrac{v\oplus w}{\gamma}\right)K_{\gamma,p}\mathrm{d}\lambda_1 \otimes \lambda_2\\ &= \sup_{\substack{v\in C(\Omega),\\ w\in C(\Omega)}} \langle \left(\begin{smallmatrix}\mu\\ \nu\end{smallmatrix}\right), \left(\begin{smallmatrix}v\\ w\end{smallmatrix}\right)\rangle - \gamma \iint_{\Omega\times\Omega} \exp\left(\tfrac{v\oplus w}{\gamma}\right)K_{\gamma,p}\mathrm{d}\lambda_1 \otimes \lambda_2\end{aligned}$$

can be interpreted as the Fenchel conjugate F^* of the function

$$F : C(\Omega)\times C(\Omega) \to \mathbb{R}\cup\{+\infty\}, \quad (v,w) \mapsto \gamma \iint_{\Omega\times\Omega} \exp\left(\tfrac{v\oplus w}{\gamma}\right)K_{\gamma,p}\mathrm{d}\lambda_1 \otimes \lambda_2.$$

Thus we have $\partial\mathcal{W}_{\gamma,p}(\mu,\nu) = \partial F^*(\mu,\nu)$. Now, if (v,w) are dual optimizers, we have

$$F^*\left(\left(\begin{smallmatrix}\mu\\ \nu\end{smallmatrix}\right)\right) = \langle \left(\begin{smallmatrix}\mu\\ \nu\end{smallmatrix}\right), \left(\begin{smallmatrix}v\\ w\end{smallmatrix}\right)\rangle - F\left(\left(\begin{smallmatrix}v\\ w\end{smallmatrix}\right)\right).$$

The function F is continuous with respect to uniform convergence, thus also lower semi-continuous (see also the proof of Theorem 3.3.19). It is further convex and proper. We can thus apply Theorem 2.2.8 and get $\left(\begin{smallmatrix}v\\ w\end{smallmatrix}\right) \in \partial F^*\left(\left(\begin{smallmatrix}\mu\\ \nu\end{smallmatrix}\right)\right)$. The rest follows from Theorem 3.3.19. ∎

This implies that

$$\partial \mathcal{W}^{\mu}_{\gamma,p}(\nu) = \{w \in C(\Omega) \,:\, (v,w) \text{ are optimizers of (D-reg) for some } v \in C(\Omega)\}$$

and

$$\partial \mathcal{W}^{\nu}_{\gamma,p}(\mu) = \{v \in C(\Omega) \,:\, (v,w) \text{ are optimizers of (D-reg) for some } w \in C(\Omega)\}\,.$$

While this does not actually mean that the function $\mathcal{W}^{\mu}_{\gamma,p}$ is differentiable, since more than one element is contained in the subdifferential, we will see later that there is a natural choice for the gradient if we want to use it in a gradient descent scheme.

We can now examine for which measure $\mathcal{W}^{\nu}_{\gamma,p}$ is minimized and we will see as already indicated that it is not minimized by ν. An illustration is depicted in Figure 4.1.

3.4.7. Corollary. The function $\mathcal{W}^{\nu}_{\gamma,p}$ is minimized for $\tilde{\nu}$ defined by the density

$$\tfrac{\mathrm{d}\tilde{\nu}}{\mathrm{d}\lambda_1}(x) = \int_{\Omega} K_{\gamma,p}(x,y) \left(\int_{\Omega} K_{\gamma,p}(z,y)\mathrm{d}\lambda_1(z)\right)^{-1} \mathrm{d}\nu(y).$$

Proof. Since $\mathcal{W}^{\nu}_{\gamma,p}$ is convex, it is minimized by $\tilde{\nu} \in \mathcal{P}(\Omega)$ if and only if $0 \in \partial \mathcal{W}^{\nu}_{\gamma,p}$, as stated in Theorem 2.2.9. Thus, we need to evaluate the condition

$$0 \in \partial \mathcal{W}^{\nu}_{\gamma,p}(\tilde{\nu}) = \{v \in C(\Omega) \,:\, (v,w) \text{ are optimizers of (D-reg) for some } w \in C(\Omega)\}\,.$$

This means that 0 is a dual optimizer. By (3.28), the density u of the primal optimizer then is

$$u(x,y) = K_{\gamma,p}(x,y) \exp\left(\tfrac{w(y)}{\gamma}\right)$$

for some $w \in C(\Omega)$. Because of the marginal conditions, the first marginal has to be $\tilde{\nu}$, thus

$$\tfrac{\mathrm{d}\tilde{\nu}}{\mathrm{d}\lambda_1}(x) = \int_{\Omega} u(x,y)\mathrm{d}\lambda_2(y) = \int_{\Omega} K_{\gamma,p}(x,y) \exp\left(\tfrac{w(y)}{\gamma}\right)\mathrm{d}\lambda_2(y)$$

and the second marginal is

$$\begin{aligned}\tfrac{\mathrm{d}\nu}{\mathrm{d}\lambda_2}(y) &= \int_{\Omega} u(x,y)\mathrm{d}\lambda_1(x) = \int_{\Omega} K_{\gamma,p}(x,y) \exp\left(\tfrac{w(y)}{\gamma}\right)\mathrm{d}\lambda_1(x)\\ &= \exp\left(\tfrac{w(y)}{\gamma}\right) \int_{\Omega} K_{\gamma,p}(x,y)\mathrm{d}\lambda_1(x)\end{aligned}$$

Combining these equations we get

$$\tfrac{\mathrm{d}\tilde{\nu}}{\mathrm{d}\lambda_1}(x) = \int_{X} K_{\gamma,p}(x,y) \left(\int_{\Omega} K_{\gamma,p}(z,y)\mathrm{d}\lambda_1(z)\right)^{-1} \mathrm{d}\nu(y),$$

which is the desired expression. ■

The term $\left(\int K_{\gamma,p}(z,y)\mathrm{d}\lambda(z)\right)^{-1}$ in the above expression is a "normalization term", providing that $\tilde{\nu} \in \mathcal{P}(X)$. The minimizer is thus basically attained by convolving the reference ν with the kernel $K_{\gamma,p}$. Since for $\gamma \to 0$, the density of $K_{\gamma,p}(x,y)$ decreases continually for $d(x,y) > 0$, i. e. $K_{\gamma,p}(x,y) \to 0$ for $x \neq y$, the minimizer $\tilde{\nu}$ converges to ν. Thus, for small values of γ, the minimizer resembles the reference ν. For larger values of γ, the minimizer is very diffuse, and for $\gamma \to \infty$, $K_{\gamma,p}(x,y) \to 1$, and the minimizer converges to $\tilde{\nu} \equiv \frac{1}{\lambda_1(\Omega)}\lambda_1$. This implies that for large values of γ, all measures seem more and more similar to the measure $\frac{1}{\lambda_1(\Omega)}\lambda_1$, so they seem more and more alike.

4 Application to MR images

In this chapter we will discuss how the theory developed in the previous chapter can be applied to MR images. We will first examine the more general case of measures on a finite space Ω, since MR images, understood as measures on a three-dimensional finite grid, also fall into this category, but other applications exist. One example is the case where Ω is constructed from a compact set $\Omega \subseteq \mathbb{R}^n$ via discretization. This gives rise to a metric on Ω defined through the Euclidean metric on $\mathbb{R}^n$. Images are defined in this way, using a subset of $\mathbb{R}^2$. It is also possible to use Riemannian manifolds and triangle meshes, see [41]. Other examples for discrete metric spaces that do not arise from discretization schemes include the case where Ω is a set of strings, and d is a string edit distance, for example the Hamming distance or the Levenshtein distance, and in [49], Ω is a set of words and d is chosen such that it represents semantic discrepancy between the words. We thus see that there is a broad field of applications. In the case of a finite space, measures as well as densities can be represented as vectors. Rewriting the formulas from Chapter 3 using sums, vectors and matrices makes it easier to implement them later.

4.1 A Summary of the Results for Finite Spaces

We will now give a summary of the most important concepts and results, reformulated in the setting of finite spaces. In this section, we denote by $\otimes$ and $\oslash$ the componentwise multiplication and division. We first redefine the Wasserstein distance. Let $\Omega = \{x_1, \dots x_n\}$. Measures or densities on Ω are vectors in $\mathbb{R}^n$, and integrals become sums. We assume that the metric on Ω is given as a symmetric matrix $d \in \mathbb{R}^{n\times n}_{\geq 0}$. The Wasserstein distance between two vectors $r, s \in \mathbb{R}^n_{\geq 0}$ with $\|r\|_1 = \|s\|_1$ is

$$\begin{aligned} \mathcal{W}_p(r,s)^p &= \min_{u \in \mathbb{R}^{n\times n}_{\geq 0}} \sum_{ij} u_{ij} d_{ij}^p \\ &\text{subject to} \sum_i u_{ji} = r_j \text{ and } \sum_i u_{ij} = s_j \text{ for } 1 \leq j \leq n. \end{aligned} \tag{4.1}$$

This definition yields a metric. However, it is only defined for vectors with the same L^1-norm, since the constraints cannot be fulfilled otherwise. To circumvent this problem, we introduced the *Unbalanced Mass Transport Problem (UMTP)* in Subsection 3.2.2 and showed that it can be cast in the same setting as above. We repeat this construction in the discrete setting. First, we need a *waste penalty vector* $p \in \mathbb{R}^n_{>0}$ that satisfies $|p_i - p_j| \leq d_{ij}$ for all $1 \leq i, j \leq n$. Then the UMTP between r and s is

$$\begin{aligned} &\min_{u \in \mathbb{R}^{n\times n}_{\geq 0}} \sum_{ij} u_{ij} d_{ij}^p + \sum_i p_i\Big(r_i - \sum_j u_{ij}\Big) + \sum_i p_i\Big(s_i - \sum_j u_{ji}\Big) \\ &\text{subject to} \sum_j u_{ij} \leq r_i, \sum_j u_{ji} \leq s_i \text{ for all } 1 \leq i \leq n. \end{aligned} \tag{4.2}$$

This means we replace the usual constraints by inequality constraints and additionally penalize the deviation from the desired marginal with the waste penalty vector p.

In Subsection 3.2.3, we showed that this is equivalent to a different construction, which becomes the following in the discrete setting. Let $\tilde{\Omega} = \Omega \cup \{x_{n+1}\} = \{x_1, \dots x_{n+1}\}$. We define a new $(n+1) \times (n+1)$ distance matrix $\tilde{d}$ as

$$\tilde{d} = \begin{pmatrix} d_{11} & \dots & d_{1n} & p_1 \\ \vdots & \dots & \vdots & \vdots \\ d_{n1} & \dots & d_{nn} & p_n \\ p_1 & \dots & p_n & 0 \end{pmatrix} \tag{4.3}$$

Denote by $\mathbb{1}$ the vector of ones. Then the L^1-norm of r can be calculated as

$$\sum_{i=1}^{n} r_i = \mathbb{1}^T r,$$

and analogously for s. We choose a constant M dominating the total masses of r and s, i. e. $M \geq \max(\mathbb{1}^T r, \mathbb{1}^T s)$ and define new marginals as

$$\tilde{r} = \begin{pmatrix} r_1 \\ \vdots \\ r_n \\ M - \mathbb{1}^T r \end{pmatrix} \text{ and } \tilde{s} = \begin{pmatrix} s_1 \\ \vdots \\ s_n \\ M - \mathbb{1}^T s \end{pmatrix}. \tag{4.4}$$

This means we added an additional point x_{n+1} that serves as mass difference compensation: additional mass can be transported to and from this remote point. For the construction of new marginals $\tilde{r}$ and $\tilde{s}$, additional mass is put on x_{n+1} to assure that both vectors have the same mass M. They can then be rescaled to have unit mass. With this construction, solving the UMTP is equivalent to solving the transportation problem

$$\begin{aligned} &\min_{u \in \mathbb{R}_{\geq 0}^{(n+1)\times(n+1)}} \sum_{ij} u_{ij} \tilde{d}_{ij}^{p} \\ &\text{subject to} \sum_i u_{ji} = \tilde{r}_j \text{ and } \sum_i u_{ij} = \tilde{s}_j \text{ for } 1 \leq j \leq n+1. \end{aligned} \tag{4.5}$$

Further, if the vector p satisfies $p_i \geq \frac{1}{2} \max_{i,j} d_{ij}$, this construction yields a metric, since $\tilde{d}$ is a metric in this case.

In Section 3.3, we suggested entropic regularization for efficient computation. We introduced a “reference measure”, with respect to which we considered densities. In the finite case this corresponds to “area weights”, see [41].

Let α be the vector of area weights. The vectors $r, s \in \mathbb{R}^n$ and the matrix $u \in \mathbb{R}^{n \times n}$ are now considered as densities, the corresponding measures are $r \otimes \alpha, s \otimes \alpha$ and $u \otimes \alpha\alpha^T$. They are assumed to be normalized such that $\alpha^T r = \alpha^T s = 1$. We define the entropy of a density $u \in \mathbb{R}^n$ with respect to α as

$$h(u) = \sum_{i,j} (\log u_{ij} - 1) u_{ij} \alpha_i \alpha_j.$$

Then we consider the discrete entropy-regularized Wasserstein distance for some $\gamma > 0$ as

$$\begin{aligned} \mathcal{W}^p_{\gamma,p}(r,s) = \min_{u \in \mathbb{R}^{n\times n}_{\geq 0}} \sum_{ij} u_{ij} d^p_{ij} \alpha_i \alpha_j + \gamma h(u) \\ \text{subject to} \sum_i u_{ji} = r_j \text{ and } \sum_i u_{ij} = s_j \text{ for } 1 \leq j \leq n. \end{aligned} \tag{4.6}$$

This regularization converges to the classical Wasserstein distance for $\gamma \to 0$. The same regularization works for the unbalanced problem by using $\tilde{r}, \tilde{s}$ and $\tilde{d}$ instead. We also need an area weight for the new point x_{n+1}. This could be chosen as the average of α.

We further derived a dual formulation for this problem in Subsection 3.3.2. This dual formulation now reads

$$\max_{v,w \in \mathbb{R}^n} \sum_i v_i r_i \alpha_i + \sum_i w_i s_i \alpha_i - \gamma \sum_{i,j} \exp\left(\frac{v_i}{\gamma}\right) K^{\gamma,p}_{ij} \exp\left(\frac{w_j}{\gamma}\right) \alpha_i \alpha_j,$$

where

$$K^{\gamma,p} \in \mathbb{R}^{n\times n}, K^{\gamma,p}_{ij} = \exp\left(\frac{-d^p_{ij}}{\gamma}\right) \text{ for } 1 \leq i,j \leq n.$$

We have seen that the optimal coupling u is related to the dual optimizers v and w through

$$u_{ij} = \exp\left(\frac{v_i}{\gamma}\right) K^{\gamma,p}_{ij} exp \frac{w_j}{\gamma} \text{ for } 1 \leq i,j \leq n.$$

This means we can represent u using $2n$ instead of n^2 variables. Additionally, u is always optimal if it can be expressed in this way. This expression can be used to find the optimal u by starting with arbitrary vectors v and w and alternatingly adapting them in such a way that the marginal conditions are satisfied. Letting $a = \exp(\frac{v}{\gamma})$ and $b = \exp(\frac{w}{\gamma})$, these read

$$\begin{aligned} r_j &= \sum_i u_{ji} \alpha_i = \sum_i a_j K^{\gamma,p}_{ji} b_i \alpha_i = a_j K^{\gamma,p} (\alpha \otimes b)_j, \\ s_j &= \sum_i u_{ij} \alpha_i = \sum_i a_i K^{\gamma,p}_{ij} b_j \alpha_i = b_j K^{\gamma,p} (\alpha \otimes a)_j, \end{aligned}$$

for $1 \leq j \leq n$. In terms of vectors, this can be stated as

$$\begin{aligned} r &= a \otimes K^{\gamma,p} (\alpha \otimes b), \\ s &= b \otimes K^{\gamma,p} (\alpha \otimes a), \end{aligned} \tag{4.7}$$

The algorithm proposed in Subsection 3.3.3 becomes now the "area weighted version of Sinkhorn's algorithm" stated in [41]. It is also known as RAS algorithm and it is known to converge for arbitrary matrices $K^{\gamma,p}$ with positive entries ([50, Theorem 3]. In Corollary 3.4.6, we calculated the subdifferential of the ERWD with respect to one fixed marginal. For fixed s, we have for any r_0,

$$\partial_r \mathcal{W}^p_{\gamma,p}(r_0, s) = \left\{ v : (v,w) \in \mathbb{R}^{2n} \text{ are dual optimizers} \right\}.$$

If we want to use this gradient in a gradient descent scheme, we want the total mass to stay the same. We can thus use a "canonical gradient" v_0 with $\alpha^T v_0 = 0$, asserting $r_0 - t v_0$ has the same total mass as r_0. We can get such a v_0 from an arbitrary dual optimizer v by setting

$$v_0 = v - \alpha^T v. \tag{4.8}$$

By setting $w_0 = w + \alpha^T w$, (v_0, w_0) stays a valid pair of dual optimizers. This was also proposed in [34]. The complete algorithm is stated in Algorithm 4.1. This same algorithm can be applied for standard Wasserstein distances as well as for unbalanced mass transport by using $\tilde{r}$ and $\tilde{s}$ instead of r and s and $\tilde{d}$ instead of d in the definition of $K_{\gamma,p}$.

Algorithm 4.1 Algorithm to calculate the dual variables that define the optimal transport plan.

```
1: $b \leftarrow 1$
2: // projection iterations
3: for $i = 1, 2, 3, \dots$ do
4:     $a \leftarrow r \oslash K(\alpha \otimes b)$
5:     $b \leftarrow s \oslash K(\alpha \otimes a)$
6: end for
7: // transform
8: $v \leftarrow \gamma \ln a$
9: $w \leftarrow \gamma \ln b$
10: // shift to get canonical gradient in r
11: $v \leftarrow v - \alpha^T v$
12: $w \leftarrow w + \alpha^T v$
13: return $v, w$
```

To get the final value $\mathcal{W}_{\gamma,p}(\mu, \nu)$, one has to plug the result of the algorithm into the dual formulation (D-reg). If we assume that v and w are indeed optimal, then

$$\sum_{i,j} \exp\left(\frac{v_i}{\gamma}\right) K_{ij}^{\gamma,p} \exp\left(\frac{w_j}{\gamma}\right) \alpha_i \alpha_j = \sum_{i,j} u_{ij} \alpha_i \alpha_j = \sum_j s_j \alpha_j = 1.$$

Consequently, we get

$$\mathcal{W}_{\gamma,p}(r, s) = \alpha^T (v \otimes r + w \otimes s) - \gamma. \tag{4.9}$$

As noted in Section 3.4, this value can be negative. We propose to deal with this problem by subtracting the minimal value. This is useful if the ERWD is used as a loss function. As seen in Corollary 3.4.3, this minimal value is given by

$$-\gamma\left(1 + \log\left(\alpha^T K^{\gamma,p} \alpha\right)\right). \tag{4.10}$$

We have further seen that the ERWD with one marginal fixed is in general not minimized by that fixed marginal, i. e. the function

$$s \mapsto \mathcal{W}_{\gamma,p}(r, s)$$

is not minimized by r. Instead, it is minimized by

$$\tilde{r} = (K^{\gamma,p}(r \otimes \alpha)) \oslash (K^{\gamma,p} \alpha). \tag{4.11}$$

An illustration can be seen in Figure 4.1.

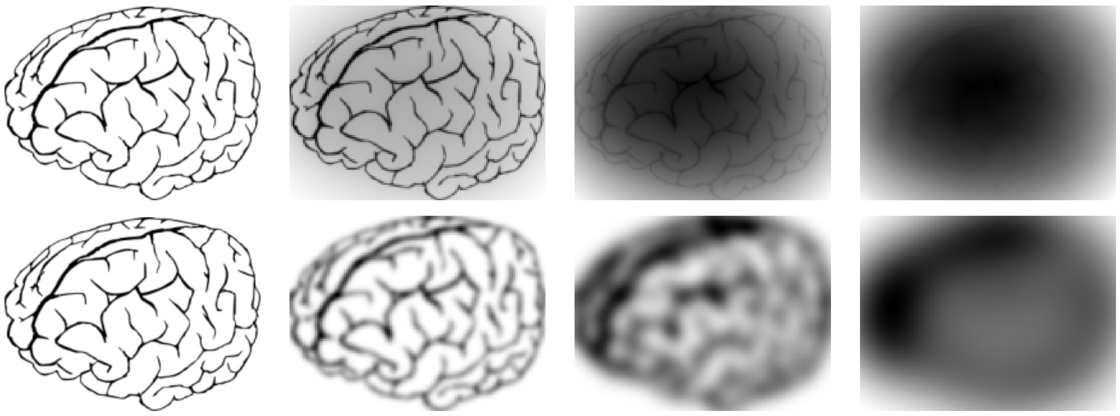


Figure 4.1: Minimizers for different values of p and γ. On the left, the original is pictured. It can be interpreted as a measure, where mass is represented in black. The measures that minimize the ERWD to the original image are depicted next to it for different values of γ . Upper row: $p = 0.1$, bottom row: $p = 2$. For $p = 0.1$ the cost function does not increase significantly for far away points, and the entropy function favors a spread out transport plan. Since the costs are high for far away points for $p = 2$, the blur only occurs locally.

4.2 The Setting of MR Images and Numerical Improvements

Now we want to consider the case of MR images of the brain (or brain regions). As already noted in Chapter 2, an MR image is defined on a grid of three-dimensional voxels. Such a voxel has three coordinates $(i, j, k) \in \mathbb{N}^3$. In Definition 1.1.1, we defined the concept of a tensor, which is basically a multidimensional array, understood as a function from a set of coordinates into $\mathbb{R}$. An MR image is then a rank three tensor of shape $(n_1, n_2, n_3) \in \mathbb{N}^3$. Thus we will now describe the setting where Ω is a voxel grid, which encompasses not only MR images.

Let $h \in (0, \infty)$ be a step size. Let $x^i := hi$ for $i \in \mathbb{Z}^3$. This gives a discretization of $\mathbb{R}^3$ as a voxel grid. Next, let

$$[n] := \left\{ 0, \dots, n-1 \right\} \text{ for } n \in \mathbb{N},$$

$$I := [n_1] \times [n_2] \times [n_3] = \left\{ i = (i_1, i_2, i_3) \ : \ i_j \in [n_j] \text{ for } 1 \leq j \leq 3 \right\}$$

as in Definition 1.1.1. Further let

$$\Omega = \left\{ x^i = (x^i_1, x^i_2, x^i_3) \in \mathbb{R}^3 \ : \ i \in I,\ x^i = hi = (hi_1, hi_2, hi_3) \right\}.$$

The distance between voxels x^i and x^j is their Euclidean distance. MR images are then functions on Ω, which is the same as a function on I, which means in turn that they are tensors.

We can equip Ω with a reference measure λ or area weights α by taking into account the step size. Let $\alpha_i = h^3$, and $\lambda(\{x^i\}) = \alpha_i$. In the previous section, we have assumed that Ω is ordered. To achieve this, we have to relabel the elements of I to sort them linearly. One option would be to sort them lexicographically. This is however not compatible with how multidimensional arrays are converted to one-dimensional arrays in `Python` and `C`. Instead, they reorder colexicographically, which corresponds to mapping

$$(i_1, i_2, i_3) \mapsto i_1 \cdot (n_2 \cdot n_3) + i_2 \cdot n_3 + i_3,$$

see Table 4.1.

We can then calculate $K^{\gamma,p}$ as a matrix and store it. Let $n := n_1 \cdot n_2 \cdot n_3$. Then $K^{\gamma,p}$ is an $n \times n$ matrix. This can however be very memory intensive: for example, if $n_1 = n_2 = n_3 = 30$, then

input		output
$(0,0,0)$	$\mapsto$	0
$(0,0,1)$	$\mapsto$	1
$\vdots$		$\vdots$
$(0,0,n_3-1)$	$\mapsto$	n_3-1
$(0,1,0)$	$\mapsto$	n_3
$(0,1,1)$	$\mapsto$	n_3+1
$\vdots$		$\vdots$
(n_1-1,n_2-1,n_3-1)	$\mapsto$	$n_1\cdot n_2\cdot n_3-1$

Table 4.1: Ordering 3-tuples linearly. The order is colexicographic, i. e. lexicographic when reading sequences from right to left.

$n = 30^3 = 27\,000$, thus the $n\times n$-matrix $K^{\gamma,p}$ has over 700 million entries. Storing this matrix in single precision takes up almost 3GB, and double that amount for double precision. Since the Euclidean metric is translation invariant, we are storing a lot of redundant information. This can be avoided using *convolutions* or more precisely *cross-correlation*.

We have $d(x,y) = \|x-y\|$ for $x,y\in\Omega$, which implies the translation invariance. We can thus write

$$K_{\gamma,p}(x,y) = g_{\gamma,p}(x-y) \quad \text{with} \quad g_{\gamma,p}:\mathbb{R}^3\to\mathbb{R},\; g_{\gamma,p}(x) = \exp\Big(-\tfrac{\|x\|^p}{\gamma}\Big). \tag{4.12}$$

Let $i\in I$ be fixed. Let f be a function on Ω. Using index shift and the above observation, we get on the voxel grid

$$\begin{aligned}(*) &:= \int_\Omega f(y)K_{\gamma,p}(x^i,y)\mathrm{d}\lambda(y) = \sum_{j\in I} f(x^j)g_{\gamma,p}(x^j-x^i)\lambda(\{x^j\}) = \sum_{j\in I} f(x^j)g_{\gamma,p}(x^{j-i})\lambda(\{x^j\})\\ &= \sum_{j\in I-i} f(x^{i+j})g_{\gamma,p}(x^j)\lambda(\{x^{i+j}\}),\end{aligned}$$

where $I-i = \{(j_1-i_1,j_2-i_2,j_3-i_3) \,:\, (i_1,i_2,i_3)\in I\}$. The index set $I-i$ is now dependent on i, which is impractical. Considering all i, the values that j can take are in the set

$$\hat{I} := \{-n_1+1,\dots,n_1-1\}\times\{-n_2+1,\dots,n_2-1\}\times\{-n_3+1,\dots,n_3-1\}\,.$$

This means that the tensor corresponding to $g_{\gamma,p}$ needs to have shape $(2n_1-1)\times(2n_2-1)\times(2n_3-1)$. If we now consider the set $\hat{I}$ instead of I and define $\hat{f}(xi) = 0$ if $i\in\hat{I}\backslash I$, we finally get, interpreting f and $g_{\gamma,p}$ as tensors,

$$\begin{aligned}(*) &= \sum_{j\in\hat{I}} \hat{f}(x^{i+j})g_{\gamma,p}(x^j)\lambda(\{x^{i+j}\}\\ &= \sum_{j_1=-n_1+1}^{n_1-1}\sum_{j_2=-n_2+1}^{n_2-1}\sum_{j_3=-n_3+1}^{n_3-1} (\hat{f}\otimes\alpha)[i_1+j_1,i_2+j_2,i_3+j_3]g_{\gamma,p}[j_1,j_2,j_3].\end{aligned}$$

The formula on the right is known in the literature as *cross-correlation* and is often understood when talking of *convolution*, see for example the implementation in [3]. Defining $\hat{f}$ corresponds to "padding" f with zeros. An illustration in 2D in the case where the image has shape 3×3 and the kernel has shape $(2\cdot3-1)\times(2\cdot3-1) = 5\times5$ is shown in Figure 4.2.

This means that we now only have to store a filter kernel of size $(2n_1-1)\times(2n_2-1)\times(2n_3-1)\lessapprox 8n_1n_2n_3$. If again $n_1 = n_2 = n_3 = 30$, then our filter kernel has have approximately 200 000 entries, which only amounts to 1 MB of memory usage.

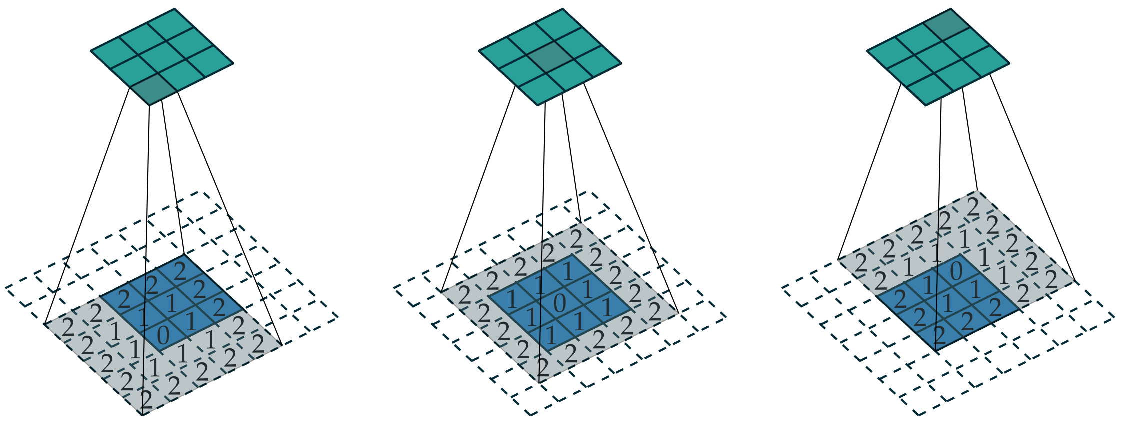


Figure 4.2: An illustration of calculating the integral $(*)$ via convolution. The function f is a matrix of shape 3×3 and is visualized in blue. The kernel $K^{\gamma,p}$ has shape 5×5 and is visualized in gray. The printed values are not the values of the kernel, but of the underlying distance, which is in this case the maximum metric for simplification. The output is visualized in green and the highlighted pixel corresponds to the fixed i in $(*)$. We see that choosing a 5×5 filter, the integral over the whole image f is computed.

In the special case where $p = 2$, we can further simplify the calculation, since

$$g_{\gamma,2}(x) = \exp\left(-\frac{\|x\|^2}{\gamma}\right) = \exp\left(-\frac{x_1^2 + x_2^2 + x_3^2}{\gamma}\right) = \exp\left(-\frac{x_1^2}{\gamma}\right)\exp\left(-\frac{x_2^2}{\gamma}\right)\exp\left(-\frac{x_3^2}{\gamma}\right).$$

Let $g_\gamma : \mathbb{R} \to \mathbb{R}, g_\gamma(x) = \exp\left(-\frac{x^2}{\gamma}\right)$ with corresponding discretization on the grid $\mathbb{Z}h$.

Now using the previous calculations for $p = 2$, we have

$$\begin{aligned}
(*) &= \sum_{j_1=-n_1+1}^{n_1-1} \sum_{j_2=-n_2+1}^{n_2-1} \sum_{j_3=-n_3+1}^{n_3-1} (\hat{f} \otimes \alpha)[i_1 + j_1, i_2 + j_2, i_3 + j_3] g_{\gamma,2}[j_1, j_2, j_3] \\
&= \sum_{j_1=-n_1+1}^{n_1-1} \sum_{j_2=-n_2+1}^{n_2-1} \sum_{j_3=-n_3+1}^{n_3-1} (\hat{f} \otimes \alpha)[i_1 + j_1, i_2 + j_2, i_3 + j_3] g_\gamma[j_1] g_\gamma[j_2] g_\gamma[j_3] \\
&= \sum_{j_1=-n_1+1}^{n_1-1} g_\gamma[j_1] \sum_{j_2=-n_2+1}^{n_2-1} g_\gamma[j_2] \sum_{j_3=-n_3+1}^{n_3-1} g_\gamma[j_3] (\hat{f} \otimes \alpha)[i_1 + j_1, i_2 + j_2, i_3 + j_3].
\end{aligned}$$

This corresponds to applying a one-dimensional convolution three times. By further zero-padding $\hat{f}$, we can take $n_{\max} := \max\{n_1, n_2, n_3\}$ in every sum instead of n_1, n_2 and n_3 and thus only one filter is needed, which has $2n_{\max} - 1$ entries.

Another issue to consider is numerical precision. The values of $K^{\gamma,p}$ can be very small. Let δ be the smallest nonzero representable number. Then

$$\begin{aligned}
K^{\gamma,p}(x^i, x^j) > \delta &\Leftrightarrow \exp\left(-\frac{\|x^i - x^j\|^p}{\gamma}\right) > \delta \\
&\Leftrightarrow \|x^i - x^j\|^p < -\gamma \log \delta \\
&\Leftrightarrow h^{\frac{p}{2}} \|i - j\|^p < -\gamma \log \delta \qquad (4.13) \\
&\Leftrightarrow \gamma > h^{\frac{p}{2}} \|i - j\|^p (-\log \delta)^{-1}. \qquad (4.14)
\end{aligned}$$

So for fixed γ, we can calculate for which points the value of the kernel is even representable, using (4.13). Or, we can see how small γ is allowed to be, using (4.14) (note that since δ is small, $-\log\delta$ will be positive). However, choosing

$$\gamma > h^{\frac{p}{2}} \max_{i,j}(\|i-j\|^p)(-\log\delta)^{-1}$$

is often too restrictive. We thus decide to allow the kernel to assume zero values, which corresponds to a "truncation" of the kernel. This is discussed in [43, §3.3]. Formally, for a subset $N \subseteq \Omega \times \Omega$, the truncated kernel is defined as

$$\hat{K}_{\gamma,p}(x,y) = \begin{cases} K_{\gamma,p}(x,y) \text{ if } (x,y) \in N \\ 0, \text{ else.} \end{cases}$$

This corresponds to setting $d(x,y) = \infty$ if $(x,y) \notin N$. This does not define a metric. Further, it does not define a continuous function, so the results about existence of minimizers / maximizers are no longer applicable. However, it is sufficient for us if the algorithm converges. So if we know of a subset $N \subseteq X \times X$ on which most of the mass of the optimal transport plans is concentrated, it is enough to solve the problem restricted to N and thus using the truncated kernel. Since MR images of brains or brain regions are very similar to each other, we can assume that transport plans are not going to transport mass "very far", and thus take N to be the set of all points that are not further apart than some threshold, i. e.

$$N = \{(x,y) \in \Omega \times \Omega : d(x,y) \leq \theta\} \tag{4.15}$$

for some $\theta > 0$. This corresponds to taking a smaller filter kernel instead of the whole one. In terms of the matrix $K_{\gamma,p}$, this leads to sparsity. We further elaborate on choosing neighborhoods in Subsection 7.2.1.

More numerical issues arise during the iterations of the algorithm. As we have seen in Example 3.3.22, the dual variables are not necessarily bounded, which can lead to over- and underflows in lines 4 and 5 in Algorithm 4.1. This problem is treated in Section 6.2.

5 Connecting the Wasserstein Distance and Machine Learning

In this chapter we describe how the Wasserstein distance can be used in machine learning. The first possibility is to use the Wasserstein distance in a kernel, used to implicitly perform calculations in a hidden feature space, see Section 1.3 and Section 1.4. The second presented possibility in this chapter is the usage of the Wasserstein distance as a loss function. In the context of computer vision, the Wasserstein distance has further been used for image retrieval [51] and SIFT matching [39].

5.1 Using the Wasserstein Distance for the Construction of Kernels

As presented in Section 1.3 and Section 1.4, kernel functions can be used to perform Support Vector Classification and Principal Component Analysis in a hidden feature space. Using the Wasserstein distance in kernels for Support Vectors Classification has been done for example in [40] to classify MNIST images, and in [52] the Wasserstein distance has been applied to histograms of local descriptors and used for texture and object classification. The following definitions are based on [53]. There are several ways to define kernels from metrics, the most popular being the radial basis function (RBF) kernel, defined as

$$k_f^{\mathrm{rbf}}(x,y) = \exp(-\varepsilon f(x,y)) \text{ for } x,y \in \mathcal{Z},\ \varepsilon \in \mathbb{R}_{>0}, \tag{5.1}$$

where $\mathcal{Z}$ is some set and f is a symmetric, non-negative, zero-diagonal function. For example, if a metric d is defined on $\mathcal{Z}$, we get the kernel

$$k^{\mathrm{rbf}}(x,y) = \exp(-\varepsilon d(x,y)^q) \text{ for } x,y \in \mathcal{Z},\ \varepsilon, q \in \mathbb{R}_{>0}. \tag{5.2}$$

Another kernel is the *linear distance substitution kernel*, defined as

$$k_d^{\mathrm{lin}}(x,y) = -\frac{1}{2}\left(d(x,y)^2 - d(x,O)^2 - d(y,O)^2\right) \tag{5.3}$$

for $x,y \in \mathcal{Z}$ and some origin $O \in \mathcal{Z}$, see [11, §2]. The idea behind this definition is that if the metric d is induced by an inner product, as is the case for the Euclidean distance, and if $O = 0$, $k_d^{\mathrm{lin}}(x,y)$ is equal to the inner product $\langle x,y\rangle$. The advantage is that the definition k_d^{lin} only depends on distances, which can be advantageous if the inner product is unknown or no inner product inducing the metric exists.

A special property of the linear distance substitution kernel is that it is distance-preserving, which is stated in the following proposition, see also [12].

5.1.1. Proposition. Let d be a metric function on $\mathcal{Z}$ which is negative definite. Let $\mathcal{H}$ be the RKHS associated to k_d^{lin}. Let $\phi : \mathcal{Z} \to \mathcal{H}$ be the corresponding feature map. Then the distance between feature vectors in the RKHS $\mathcal{H}$ is equal to the distance of the corresponding elements in $\mathcal{Z}$, i. e. for any $x, y \in \mathcal{Z}$,

$$\|\phi(x) - \phi(y)\|_{\mathcal{H}} = d(x, y).$$

Proof. We have

$$\begin{aligned}
\|\phi(x) - \phi(y)\|_{\mathcal{H}}^2 &= \langle \varphi(x) - \varphi(y), \varphi(x) - \varphi(y) \rangle = \langle \varphi(x), \varphi(x) \rangle + \langle \varphi(y), \varphi(y) \rangle - 2\langle \varphi(x), \varphi(y) \rangle \\
&= k_d^{\text{lin}}(x,x) + k_d^{\text{lin}}(y,y) - 2k_d^{\text{lin}}(x,y) \\
&= \ -\tfrac{1}{2}\big(d(x,x)^2 - d(x,O)^2 - d(x,O)^2\big) - \tfrac{1}{2}\big(d(y,y)^2 - d(y,O)^2 - d(y,O)^2\big) \\
&\qquad\qquad\qquad\qquad\qquad\qquad + \big(d(x,y)^2 - d(x,O)^2 - d(y,O)^2\big) \\
&= d(x,O)^2 + d(y,O)^2 + d(x,y)^2 - d(x,O)^2 - d(y,O)^2 \\
&= d(x,y)^2,
\end{aligned}$$

which proves the claim. ■

We now want to discuss the definiteness of such kernels. We define what it means for a kernel to be negative definite.

5.1.2. Definition (Negative Definite Kernel). Let $\mathcal{Z}$ be a nonempty set. A function $f : \mathcal{Z} \times \mathcal{Z} \to \mathbb{R}$ is called *negative definite* if f is symmetric and for any $m \geq 0$, any family $x^{(1)}, \dots, x^{(m)} \in \mathcal{Z}$ and any sequence $c_1, \dots, c_m \in \mathbb{R}$ with $\sum_{i=1}^m c_i = 0$ the inequality

$$\sum_{i,j=1}^{m} c_i c_j f(x^{(i)}, x^{(j)}) \leq 0$$

holds.

Concerning definiteness of the above kernels, the following theorem holds.

5.1.3. Theorem (Definiteness of Kernels). The following statements are equivalent for any symmetric, non-negative, zero-diagonal distance function f.

(i) There exists a Hilbert space $\mathcal{H}$ and an embedding $\phi : \mathcal{Z} \to \mathcal{H}$, such that $f(x,y) = \|\phi(x) - \phi(y)\|^2$.

(ii) The kernel k_f^{rbf} is positive definite for all ε.

(iii) The kernel k_f^{lin} is positive definite.

(iv) The function f is negative definite.

Proof. The equivalence between (i) and (iv) is stated, for example, in [53, Thm. 4.3]. The equivalence between (i) and (ii) is stated in [28, Prop. 8.1]. The equivalence between (i) and (iii) is stated in [11, Prop. 1]. ■

The kernels k_f^{rbf} and k_f^{lin} being positive definite means that the corresponding hidden feature space is a Reproducing Kernel Hilbert Space (RKHS), see Section 1.3. This means that in order to check if the hidden feature space is indeed a Hilbert space, it suffices to check if the function f is negative definite. Unfortunately, we have the following result for $f = \mathcal{W}_{\gamma,p}^{pq}$.

5.1.4. Proposition ([28, Prop. 8.2]). For $\mathcal{Z} = \mathbb{R}^3$ and $d(x,y) = \|x-y\|_2$, the p-Wasserstein distance is not negative definite for $p = 1, 2$.

This means that the hidden feature space corresponding to Wasserstein kernels is not guaranteed to be a Hilbert space. However, as already noted in Section 1.3, the positive definiteness of the kernel is not necessary in order to perform support vector classification and principal component analysis. It is also still possible that k_f^{rbf} is positive definite for some $\varepsilon > 0$.

We want to conclude with a toy example, illustrating why using Wasserstein distances in SVM can be useful. Consider as data a set of probability measures with mutually disjoint support, split into two classes. Then to any distance function that depends only on "pointwise" differences, all these measures will look the same. The Wasserstein distance, on the other hand, takes into account the ground metric, and can thus distinguish the measures from each other. For example, let $\Omega = [0,1]^2$ and

$$X_1 = \left\{\delta_{(x,y)} \,:\, x < y\right\}, \quad X_2 = \left\{\delta_{(x,y)} \,:\, x > y\right\},$$

so X_1 consists of Dirac measures concentrated above the diagonal, and X_2 consists of Dirac measures concentrated below the diagonal. Let $X = X_1 \cup X_2$. Now if we consider the total variation distances between these measures, we get

$$d_{TV}(\delta_{(x,y)}, \delta(x',y')) = \|\delta_{(x,y)} - \delta(x',y')\| = \begin{cases} 0, & \text{if } x = x' \text{ and } y = y' \\ 2, & \text{else.} \end{cases}$$

Intuitively, this cannot lead to a good classification result. But if we consider Wasserstein distances, we have

$$\mathcal{W}_p(\delta_{(x,y)}, \delta(x',y')) = \|(x - x', y - y')\|_2.$$

This means that the Wasserstein distance between elements in X is the same as the distance between the corresponding points in $\mathbb{R}^2$, where the classes are trivially linearly separable by the diagonal. To test this in practice, we discretize the Dirac measures as can be seen in Figure 5.1. We can now consider the 1-distance and 2-distance or Euclidean distance between the discretized images, interpreted as vectors. Since the supports of the discretized measures are no longer disjoint, classification using 1- or 2-distances might not be completely impossible, but should still perform worse.

To test this, we use a set of 800 generated images, with 400 images for each class. We keep out a test set of 40 images for each class and train a classifier on the rest. As kernel, we consider the respective kernels k^{rbf} and k^{lin}. We test the parameters $C = \{0.001, 0.01, 0.1, 1, 10\}$ and $\varepsilon = \left\{10^k \,:\, k \in \{-3, -2.5, -2, \ldots, 2\}\right\}$. An accuracy of $\approx 98\%$ could be achieved using both 1- and 2-Wasserstein distances and an accuracy of $\approx 80\%$ could be achieved using both 1- and 2-distances. The accuracy was independent of the used kernel. The samples that were misclassified by the Wasserstein distance kernels were samples that had nonzero values on the diagonal due to discretization and were thus hard to classify. They can be seen in Figure 5.2.

The superiority of the Wasserstein distance in this case is even more apparent when performing PCA. Since we know that the data has been constructed from points in the plane, it should be easy to perform a dimensionality reduction to 2 dimensions. We apply PCA using the negative distance kernel and compare the performance of 1- and 2-Wasserstein distances and 1- and 2-distances. As with the SVM, we use 80 images as a test set. The results are visualized in Figure 5.3. It is apparent that the Wasserstein kernels work as expected, while 1- and 2-distances fail to capture the structure of the dataset.

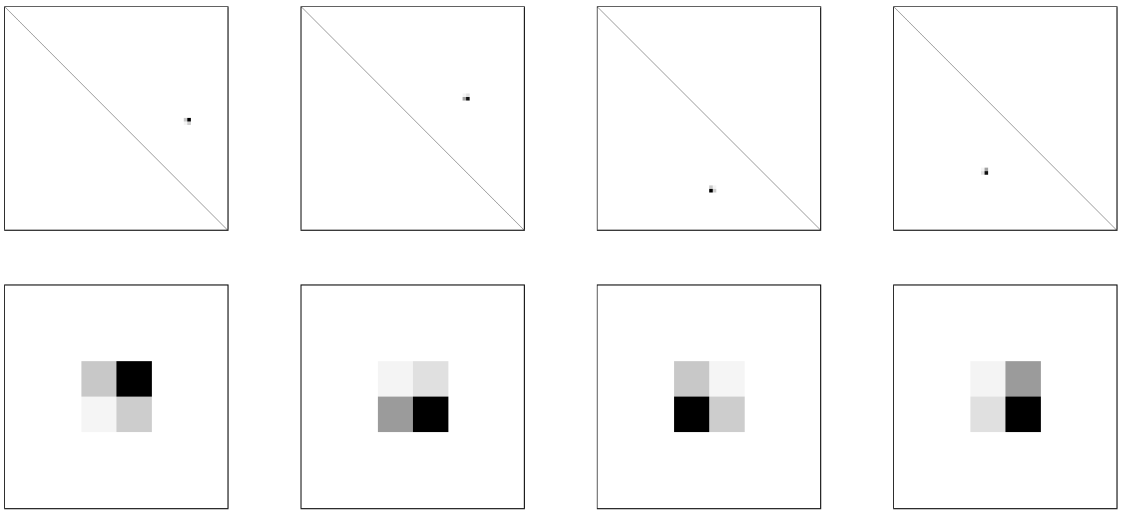


Figure 5.1: Discretization of Dirac measures. In the bottom row a zoom in can be seen.

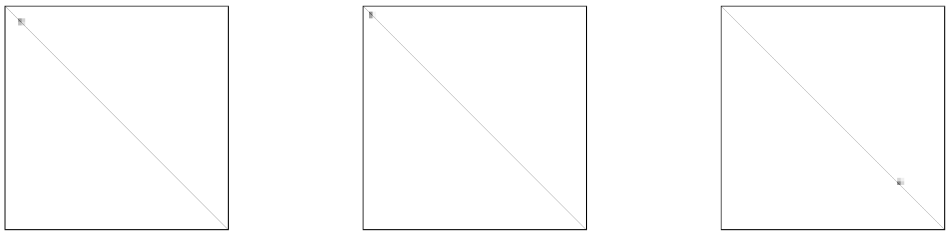


Figure 5.2: Discretized Dirac measures that were misclassified by Wasserstein kernel SVM.

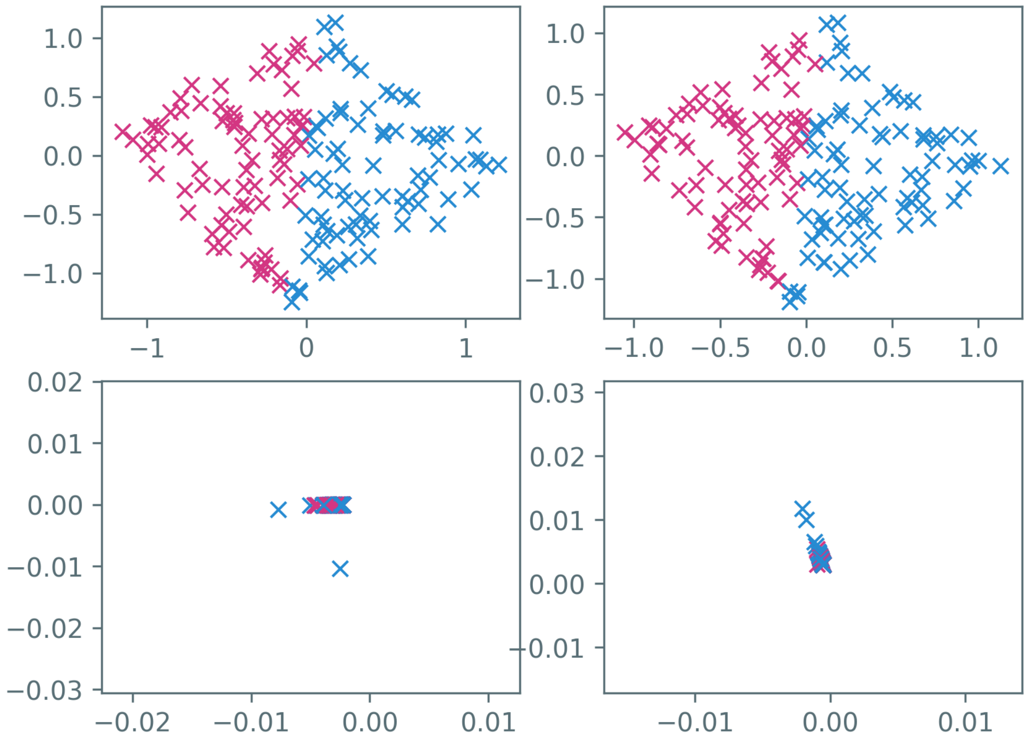


Figure 5.3: Projection of the Dirac measures onto the two main principal components. The colors represent the two classes. Top row: 1-Wasserstein and 2-Wasserstein kernel, bottom row: L^1 and L^2 kernel.

5.2 Using the Wasserstein Distance as a Loss Function

Another possibility to use the Wasserstein distance in machine learning is to use it as a loss function for neural networks. In 2015, this has been proposed in [34] for multiclass learning. They consider networks that outputs a vector of scores or probabilities for the classes. These vectors are then compared with the Wasserstein distance. The Wasserstein distance has become popular when it was applied as a loss function for Generative Adversarial Networks in [54] in 2017. These two papers use two different methods to compute the Wasserstein distance. [34] use the entropic regularization scheme presented in this thesis, and [54] use the Kantorovich-Rubinstein duality Theorem 3.1.10, which works only for the 1-Wasserstein distance.

Using the Wasserstein distance as a loss function means, in the notation of Section 1.1, if we have a neural net $f(\cdot;\theta)$ where $\theta \in \mathbb{R}^n$ represents the parameters to learn, we want to minimize

$$\sum_{i=1}^{k} \mathcal{W}_{\gamma,p}(f(x^{(i)};\theta), y_i),$$

where $\{x^{(1)}, \dots, x^{(k)}\}$ is a set of samples with corresponding ground truth labels $y_i, i = 1, \dots, k, k \in \mathbb{N}$. Let us consider the example of multiclass learning. In multiclass learning, the ground truth labels are usually encoded as binary vectors ("one hot encoding" or "1-of-K encoding"). For example, if we have l classes, and $x^{(i)}$ is of class $j \in \{1, \dots, l\}$, then y_i is the j-th unit vector in $\mathbb{R}^l$. This can be interpreted as the probability that $x^{(i)}$ belongs to class j being 1. The output of f will then also be a probability vector, representing with which probability $x^{(i)}$ belongs to which class. These probability vectors can be compared with the Wasserstein distance. The advantage is that similarity of classes is taken into account. The loss could then give feedback that classifying a muffin as a chihuahua is very bad, while classifying a muffin as a cookie or cake is more or less okay. This works if the distance between muffin and cookie in the ground metric is much lower than the distance between muffin and chihuahua.

Another application, which is the application we propose in this thesis, is the application to unsupervised learning with an autoencoder. To our knowledge, this has not been tried before. In this case the samples $x^{(i)}$ are themselves interpreted as measures and the ground truth labels are equal to the input. Now the main idea why the Wasserstein distance could be a good loss function is that it takes into account the ground metric. This is especially useful if the distributions to be compared have a non-overlapping support, since other loss functions like the 2- or 1-distances are often constant in such cases. An illustration can be seen in Figure 5.4.

More formally, let $\theta \in [0,1]$ and let f_θ be a function like in Figure 5.4 with a peak at θ and f_θ is zero outside of $(\theta - h, \theta + h)$ for some $h \in (0,1)$. Assume we have a network $g(x;\theta)$ with

$$g(x;\theta) = f_\theta(x) \text{ for } x, \theta \in [0,1]$$

and we want to learn f_{θ_0} for some $\theta_0 \in [0,1]$ using gradient descent and an L^2 loss function. The function f_{θ_0} corresponds to the red function in the left plot in Figure 5.4. The loss as a function in θ is calculated as $\|f_{\theta_0} - g(\theta)\|_2^2$. This corresponds to the red function in the right plot. Now we have

$$\|f_{\theta_0} - g(\theta)\|_2^2 = c$$

for some constant $c \in \mathbb{R}$ for all θ with $|\theta - \theta_0| \geq 2h$, which means that

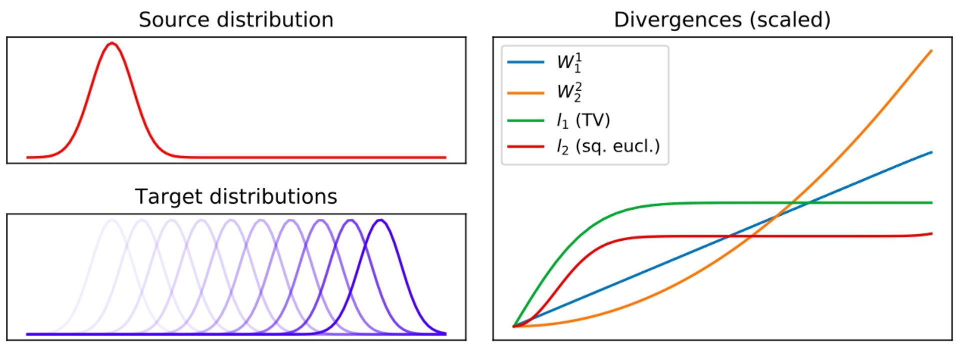


Figure 5.4: Different distances between probability densities with one peak. If two densities have disjoint support, their 1- and 2- distances are constant, while the Wasserstein distances do not exhibit this problem. Since the plot shows the squared 2- and $\mathcal{W}_2$ distances, they are technically only divergences.

Source: [55]

$$\frac{\mathrm{d}}{\mathrm{d}\theta}\|f_{\theta_0} - g(\theta)\|_2^2 = 0$$

for all θ with $|\theta - \theta_0| \geq 2h$. Thus, if g_θ is initialized badly, it is impossible to learn the correct parameter θ_0 because of the vanishing gradient. However, as can be seen in the right plot in Figure 5.4, the gradient of the 1-Wasserstein and 2-Wasserstein distances do not vanish. A similar example is also given in [54] to motivate the usage of Wasserstein distances for Generative Adversarial Networks.

6 Implementation

6.1 Libraries

To implement the algorithm, we use the programming language Python, which is standardly used in the field of machine learning. In this section we want to give a short overview of some important Python libraries that have been used and why these libraries have been chosen.

`Scikit-Learn` `Scikit-Learn` is a free software library for machine learning in Python. It includes standard machine learning algorithms such as support vector machines and principal component analysis. It allows the possibility to define classification pipelines consisting for example of data reduction and classification. It further includes many useful tools to evaluate the performance of a model, including cross-validation and hyperparameter optimization.

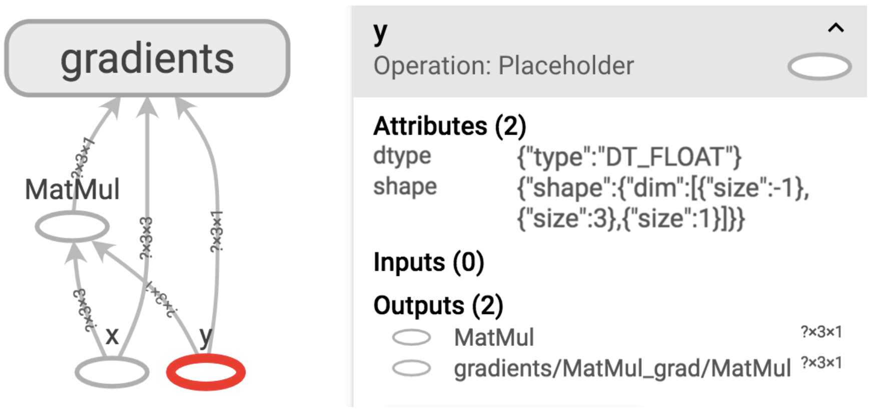
`TensorFlow` `TensorFlow` is an open-source software library designed for distributed numerical computation. It is mainly used for programming neural networks, but can also be used for other mathematical problems. A big advantage of `TensorFlow` is its support of multiple CPUs and GPUs as well as the support of different platforms. It models all operations and their dependencies in a dataflow graph. Such a graph consists of nodes, which represent the operations, and edges, which represent the data that flows between the nodes, thus modeling dependencies. The data is saved in multidimensional arrays, which are called tensors, as seen in Section 1.1. This explains the name "`TensorFlow`". When programming in `TensorFlow`, the first step is always to define a computation graph. This is a symbolic model, and in order to perform actual calculations, a *session* has to be started, during which different chunks of data can be "fed" into the computation graph. A simple `TensorFlow` program could look like the following:

```python
import tensorflow as tf

# define two placeholders
x = tf.placeholder(dtype=tf.float32, shape=[None, 3, 3], name='x')
y = tf.placeholder(dtype=tf.float32, shape=[None, 3, 1], name='y')

# multiply placeholders
z = tf.matmul(x, y)
```

First, two "placeholders" are defined. The first one represents a list of matrices of arbitrary length, which is why the first dimension is "None". The second represents a list of vectors. Next, a tensor `z` is defined, which is to be the matrix-vector product of `x` and `y`. Now if we want to calculate the matrix-vector product of some 3×3 matrix and 3×1 vector, we have to do the following:



y
Operation: Placeholder

Attributes (2)

dtype	{"type":"DT_FLOAT"}
shape	{"shape":{"dim":[{"size":-1},{"size":3},{"size":1}]}}

Inputs (0)

Outputs (2)

MatMul	?×3×1
gradients/MatMul_grad/MatMul	?×3×1

Figure 6.1: A `TensorFlow` computation graph for matrix-vector multiplication as visualized using TensorBoard.

```
# evaluate
with tf.Session() as sess:
    print(sess.run(z, feed_dict={x: [numpy.full((3, 3), fill_value=1)],
                                 y: [numpy.full((3, 1), fill_value=1)]}))
```

Here we used lists of length 1 as input in the `feed_dict`. `TensorFlow` also supports *autodifferentiation*. This means that it is possible to calculate gradients for all operations that are defined in `TensorFlow`. For example, we can do the following:

```
# define gradient
grad = tf.gradients(z, x)

# evaluate gradient
with tf.Session() as sess:
    print(sess.run(grad, feed_dict={x: [numpy.full((3, 3), fill_value=1)],
                                    y: [numpy.full((3, 1), fill_value=1)]}))
```

This calculates the sum of the partial derivatives $\frac{dz_i}{dx_j}$ for z_i in z for each direction x_j. In this example, the result would be the vector $(3,3,3)^T$.

`TensorFlow` also contains visualization tools in the suite called `TensorBoard`. This makes it possible to visualize the training loss and validation loss as well as other metrics during and after training a network. It is further possible to visualize the computation graph. A visualization of the above graph in `TensorBoard` can be seen in Figure 6.1.

We use `TensorFlow` to implement the calculation of the entropy-regularized Wasserstein distances. `TensorFlow` is a natural choice because of its native support of GPUs, which can be very well exploited by the algorithm presented in Chapter 4. Further, it can later be used as a loss function, since `TensorFlow` supports autodifferentiation.

`Keras` `Keras` is an open source high-level machine learning software library. It provides a convenient way to rapidly prototype almost any neural network. This is done by supplying building blocks that can easily be combined to create neural networks. `Keras` does not handle any low-level computations, but relies instead on a *backend engine*. One example for such a backend engine is `TensorFlow`. Functionalities provided by `Keras` include training and building networks with little code, evaluating the loss and validation loss across the training epochs and automatically saving the best model. A simple neural networks can be defined like this ([1, §3.2.1]):

```
from keras import models, layers, optimizers

# define model
model = models.Sequential()
model.add(layers.Dense(32, activation='relu', input_shape=(784,)))
model.add(layers.Dense(10, activation='softmax'))

# compile the model by specifying which optimizer, loss function
# and validation metrics to use
model.compile(optimizer=optimizers.RMSprop(lr=0.001),
              loss='mse',
              metrics=['accuracy'])

# fit the model on some training data
model.fit(input_tensor, target_tensor, batch_size=128, epochs=10)
```

The parameter `batch_size` specifies how many samples are passed through the network at once. After each batch is passed through the network, the loss is evaluated on this batch and the model is updated using the optimizer. A full training cycle on the complete training set is called an epoch. For example, if our training set consisted of 1280 samples and the batch size was 128, an epoch consists of 10 updates of the neural network weights. The number of epochs can be specified with the parameter `epochs`. We use `Keras` for the development of autoencoders. When using `TensorFlow` as backend, the entropy-regularized Wasserstein distance implemented in `TensorFlow` can be incorporated into `Keras` as a loss function.

Additional libraries include `numpy, scipy` and `matplotlib`, which we do not introduce further.

6.2 Implementation of the Wasserstein Distance Algorithm

This section treats the implementation of the Wasserstein distance algorithm in `Tensorflow`. We address the specifics of the implementation that are due to the usage of `Tensorflow` and its graph model, some numerical considerations and how to use the Wasserstein distance as a loss function in `Keras`.

The implementation is split into a Python module containing low-level methods including the algorithm and a class that can be initialized with all relevant parameters, which contains high-level methods for easy computation of Wasserstein distances.

The module contains the following methods:

- `calculate_dual_variables(dist0, dist1, kernel_mat, niter=1000, evaluate_convergence=None, reference_measure=None)`
- `calculate_dual_variables_convolutional(dist0, dist1, convolutional_filter, reference_measure, niter=1000, evaluate_convergence=None)`
- `calculate_dual_variables_convolutional_separable(dist0, dist1, convolutional_filter, dim, niter=1000, reference_measure=None)`
- `calculate_min(dist0, kernel_mat, reference_measure=None)`
- `calculate_min_gamma(neighborhood_size, num_dims, np_dtype, exponent):`

- `conv_distance(v, w, dist0, dist1, gamma, reference_measure=None)`
- `conv_distance_original(u, v, cost_mat, kernel_mat, reference_product_measure=None)`
- `conv_distance_primal(u, v, gamma, cost_mat, kernel_mat, reference_product_measure=None)`
- `convolutional_filter(dim, gamma, exponent=2, size=None, cut=True, tf_dtype=tf.float32, scale_factor=None)`
- `convolutional_filter_separable(dim, gamma, radius=None, tf_dtype=tf.float32, scale_factor=None)`
- `convolutional_kernel_separable(conv_filter, x, dim)`
- `convolve(x, filter)`
- `coordinates(dim)`
- `distance_matrix(dim, metric='sqeuclidean', dtype=np.float32, scale_factor=None, neighborhood_size=None)`
- `entropy(measure, reference_measure=None)`
- `calc_coupling(v, w, kernel_mat)`
- `calc_tilde_distributions(dist0, dist1, max_volume, reference_measure=None)`
- `calc_tilde_distance(distance_mat, remote_point_distance=None)`
- `kernel_matrix(dim, gamma, p=2, scale_factor=None)`
- `kernel_matrix_precomputed(dist_mat, gamma, p=2)`
- `normalize(data, flatten=False, reference_measure=None, add_eps=True)`

The method `calculate_dual_variables` contains the algorithm stated in Algorithm 4.1 in Chapter 4, where the kernel is stored as a matrix. In Section 4.2, we presented the possibility to use convolutions instead of saving the kernel as a matrix.

An implementation of Algorithm 4.1 using separable convolutions and standard convolutions can be found in the methods `calculate_dual_variables_convolutional_separable` and `calculate_dual_variables_convolutional`, respectively. While we have seen in Section 4.2 that storing the whole kernel as a matrix takes up a lot more memory than using convolutions, the computation time using convolutions is only lower if the convolutional filter is very small. Using a separable convolutional filter has not led to any speed up compared to using standard convolutions. The reason for these phenomena is probably due to the efficient implementation of matrix-vector multiplication on GPUs.

The method `calculate_min_gamma` calculates the minimal value for γ such that all kernel values in a specified neighborhood are representable, see Equation 4.15 and Subsection 7.2.1.

To apply the kernel to vectors using convolution, the four methods `convolutional_filter`, `convolve`, `convolutional_filter_separable` and `convolutional_kernel_separable` are used. To save the kernel as a matrix, the methods `kernel_matrix` and `kernel_matrix_precomputed` can be used. The kernel matrix is calculated from the ground distance matrix, which is calculated using the methods `coordinates` and `distance_matrix`.

The method `calculate_min` calculates the minimal value that can be assumed by the ERWD using the formula (4.10).

The methods `conv_distance`, `conv_distance_original` and `conv_distance_primal` calculate different costs from the dual variables. The "dual cost" (4.9) is calculated directly from the dual variables using `conv_distance`. The method `conv_distance_primal` calculates the "primal cost" from the optimal coupling, which can be calculated from the dual variables using the method `calc_coupling`, and from the entropy of the optimal coupling, which is returned by the method `entropy`. The transport cost of the coupling using the original, unregularized functional can be calculated using the method `conv_distance_original`.

The two methods `calc_tilde_distributions` and `calc_tilde_distance` make it possible to calculate the ERWD between distributions with different mass by using the transformations stated in (4.3) and (4.4).

Finally, the method `normalize` is used to normalize the data with respect to a given reference measure. Further, a small value ε is added, such that the data is everywhere positive and conditions (3.25) is satisfied. The value ε is chosen as the smallest representable positive number such that $1 + \varepsilon \neq 1$. This depends on the datatype of the given data (single or double precision).

We now want to address some specifics concerning the implementation of the algorithm by the method `calculate_dual_variables`. As input, it takes two tensors of shape (m, n), representing lists containing the marginal distributions. Here, n is the total amount of voxels/pixels in the underlying space, and m is the amount of samples that are being processed at the same time (the length of the list). It further takes the kernel saved as a matrix, the number of iterations and the reference measure as inputs. Lastly, it can be specified if the convergence should be evaluated. If an integer k is given, the convergence is evaluated every k iterations. Evaluation is done by storing the 1-norm (TV-norm) between the marginal of the current coupling and the desired marginal.

The kernel can be either stored in the computation graph as a constant, or a placeholder can be used. In this case, the values for the kernel matrix have to be fed into the placeholder at runtime. Since the latter option is more complicated, we use the first one. However, it is not always possible to save the kernel in the graph, because the graph definition is not allowed to be larger than 2GB. This limit is exceeded when we save the kernel in double precision. In this case, we have to use placeholders. The algorithm further consists of matrix multiplications and divisions, which can be easily implemented using the `TensorFlow` functions `matmul` and `div`.

The trickiest part is the `for loop`. Using a standard Python `for loop` would lead to the operations defined in the body of the loop being added to the computation graph once for each iteration. This would lead to a very large graph and should be avoided. Instead of Python loops, we thus use the `TensorFlow` `while_loop`. A simple `TensorFlow` `while loop` program, adding up numbers from 1 to 10 would look like this:

```
import tensorflow as tf

r0 = tf.constant(0)
condition = lambda i, r: tf.less(i, 10)
body = lambda i, r: [tf.add(i, 1), tf.add(r, i)]
result = tf.while_loop(condition, body, [r0, r0])

with tf.Session() as sess:
    print(result.eval())
```

`Condition` and `body` are functions that take the loop variables `r` and `i` as arguments. The function condition returns a boolean depending on the loop variables. The function body calculates something using the loop variables and returns the new values for the loop variables, to be used in the next iteration. These two functions are then given to the function while_loop, together with the starting values for the loop variables. In our case, the loop variables are a counter i and the dual variables v and w. Additionally, to evaluate the convergence, lists diff0 and diff1 containing the differences between current and desired marginal are added to the loop variables. The projection steps are contained in the body function. They look like this:

```
# project on constraint on dist1
Kv = tf.matmul(kernel_mat, a * v)
Kv = tf.clip_by_value(Kv, min_val, max_val)
w = tf.div(dist1, Kv)
w = tf.clip_by_value(w, min_val, max_val)
```

Between the matrix multiplication and division we have added a clipping step. This takes care of over- and underflows by clipping any values smaller than min_val and larger than max_val. These two values are chosen depending on the data type.

The correctness of the implementation is hard to test. We use some unit test to test some obvious requirements. We test the non-negativity of the optimal coupling and the non-negativity of the value of the original functional. Further, for random target marginals, we test the implementation of the algorithm by comparing the marginals of the optimal coupling to the target marginals, and by asserting the approximate equality of primal and dual cost. We test the algorithm for unnormalized measures by asserting that it returns approximately the same cost as the standard algorithm when using normalized marginals as input. For a test concerning the actual data, some test data was kindly provided by Bernhard Schmitzer, who used his implementation (which does not use TensorFlow) to calculate some distances. Comparing both results showed that they were approximately the same, where the difference was smaller when using double precision.

The two use-cases of Wasserstein distances are calculating a pairwise distance matrix to use it in RBF kernels for SVM or PCA and using it as a loss function. For this purpose, there exists a class WassersteinCalculator. This class is initialized once with the parameters

- spatial_input_dim: the dimension of the underlying grid,
- neighborhood_size: size of the neighborhood where the distance is finite (for kernel truncation), default is whole grid, see also Subsection 7.2.1,
- gamma: the regularization parameter, if 'auto', gamma assumes the smallest value such that all kernel values are representable in the neighborhood specified in neighborhood_size,
- niter: the number of iterations
- exponent: the exponent used to construct the cost matrix from the Euclidean distance matrix,
- conv_size: the size of the convolutional filter to use, can be None,
- unnormalized: boolean to determine if the extension for unnormalized measures should be used,
- max_volume: maximal volume of measures, needed if unnormalized is True,

- `remote_point_distance`: distance of the additional point to all other points in unnormalized extension, given relative to maximum distance, standard is 0.5,
- `dtype`: data type to use, float32 or float64,
- `step_size`: step size h to use for the underlying grid, needed to calculate the ground metric,
- `reference_measure`: the reference measure, can be a scalar, a vector or `None`,
- `variant`: 'reg' or 'noreg', whether to return regularized or unregularized costs,
- `subtract_min_constant`: whether to subtract the minimum value assumed by ERWD,

This class supplies a method `calc_wasserstein_distance`, which returns a tensor containing the distance if evaluated. It additionally returns the lists containing the convergence values, if specified. Further, there exists a method `wloss`, which acts as a wrapper for `Keras`. It takes the parameters `y_true` and `y_pred` and calls `calc_wasserstein_distance`. At this point we need to consider how we want `TensorFlow` to differentiate this loss function. One possibility is to use autodifferentiation. This is suggested in [28, §9.1.3], but no example application is given. It turned out that in our case, this is numerically instable if γ is small or if we make many iterations. Instead we use the theoretical gradient, which we normalize as stated in (4.8). This approach is also used by [34]. The normalizing of the gradient is done in the method `conv_distance`. Now, we do not want the autodifferentiation to further differentiate the dual variables with respect to the input marginals. To implement this, we use the TensorFlow function `stop_gradient`. So the call to `conv_distance` looks like this:

```
distance = conv_distance(tf.stop_gradient(v), tf.stop_gradient(w), ...).
```

To facilitate the computation of pairwise distances between a set of measures, the class further contains a method `calculate_pairwise_distances_batch`, which builds a computation graph using `calculate_wasserstein_distance` and then feeds the data into this graph in batches. It returns a complete distance matrix.

6.3 Implementation of SVM, PCA and Autoencoder

In this section we describe the implementation of SVM, PCA and autoencoder. In particular, we discuss how SVM and PCA can be implemented with precomputed kernel matrices. We further describe the structure of an autoencoder and how the Wasserstein loss is incorporated.

6.3.1 SVM and PCA

We start with SVM and PCA. Both are implemented in `sklearn`. Support vector classification is implemented in `sklearn.svm.SVC`, see [56]. Using precomputed kernel matrices is supported. However, we want to be able to use precomputed kernel matrices with different parameters ε and q, see Equation 5.2. These parameters should be optimized using a hyperparameter optimization strategy, for example grid search. For this reason, we need a wrapper class for the class `SVC` implemented in `sklearn` that takes those parameters as arguments and computes the respective kernels from a given distance matrix. This class is then compatible with `sklearn`. A similar wrapper class is needed for PCA for the same reasons. PCA using kernels is implemented in `sklearn` in `sklearn.decomposition.KernelPCA`, see [57]. Compared to standard PCA, it is not possible to invert the transformation for precomputed kernels. Note that the implementation of PCA is independent of the definiteness of the kernel. The implementation of SVM in `sklearn` is based on LIBSVM, which also works for indefinite kernels, since it is not assumed that the kernel matrix is positive semidefinite, see [58, §4.1].

We implement two wrapper classes SVCPrecomputed and PCAPrecomputed as subclasses of SVC and KernelPCA, respectively. We implement functions fit, predict, decision_function and score for the SVM. First, one needs to decide how to feed the train and test data. One option would be to number the whole data consecutively, feed indices and implement a kernel function that looks up the corresponding values in the precomputed matrix. However, this option leads to a high run time, since the values are looked up one by one. We thus choose to feed the distance matrix instead. As training input, the square matrix containing all pairwise distances between training instances is used. Similarly, the matrix containing all pairwise distances between training instances and test instances has to be used as testing input. Since sklearn already supports the usage of precomputed matrices, the splitting and feeding can be handled automatically by the class KFold or StratifiedKFold that handle k-fold cross-validation.

Now in order to use RBF and linear distance substitution kernels as introduced in Section 5.1, we implement a method transform_kernel that calculates linear distance substitution kernel and RBF kernel with specified parameters. Instead of implementing the kernel

$$k_d^{\text{lin}}(x,y) = -\frac{1}{2}\big(d(x,y)^2 - d(x,O)^2 - d(y,O)^2\big)$$

for some origin O, we implement instead the *negative distance kernel*

$$k_d^{\text{nd}}(x,y) := -\frac{1}{2}\big(d(x,y)^2\big).$$

These are equivalent for SVMs as stated in [11]. Since the kernel matrix is centered for PCA, the choice of origin is irrelevant, see [59]. Choosing as origin the mean of the data, k_d^{lin} and k_d^{nd} only differ by a constant, which leads to the same eigenvectors in (1.20). Thus, these kernels are also equivalent for PCA.

Implementing fit, predict, decision_function and score is then done by first transforming the passed distance matrix into a kernel matrix and then calling the corresponding method of the superclass. The implementation of PCAPrecomputed works in a similar manner. One only needs to pay attention in which methods this transformation is done, since some methods call other methods (fit_transform calls fit and score calls predict), and thus transforming the kernel in both methods would lead to applying the transformation twice. To make sure that the implementations are correct, we use unit tests to test that SVCPrecomputed gives the same result as SVC when using Euclidean distances. As a data set, we take the Dirac image set we also used in Section 5.1. Similarly, we test that PCAPrecomputed gives the same result as KernelPCA for the RBF kernel. It turns out that the reductions using PCAPrecomputed and PCA when using the negative distance kernel are mirrored. This is probably due to some implementation details. We further test that GridSearch returns the same best parameters and best scores using SVC and SVCPrecomputed, respectively.

The following is an extract of the class SVCPrecomputed:

```python
from sklearn.svm import SVC
import numpy as np


class SVCPrecomputed(SVC):
    def __init__(self, C=1.0, gamma='auto', exponent=1, kernel_type='rbf', ...):
        super().__init__(C=C, kernel='precomputed', ...)
        self.gamma = gamma
        self.exponent = exponent
        self.kernel_type = kernel_type
        ...

    def _transform_kernel(self, X):
        if self.kernel_type == 'rbf':
            return np.exp(-self.gamma * X ** self.exponent)
        else:
            return -0.5 * X ** 2

    def fit(self, X, y, sample_weight=None):
        kernel = self._transform_kernel(X)
        return super().fit(X=kernel, y=y, sample_weight=sample_weight)
```

6.3.2 Autoencoder

We now give a short description of the implementation of the autoencoders. Since we do not implement overly complicated autoencoders, the implementation can be done completely in `Keras`.

We use three levels of abstraction: an abstract class `Autoencoder` specifying an interface consisting of methods `encode`, `decode`, `transform`, `reconstruct` and `train`. The class also already holds placeholders for input, hidden representation and output, which are linked through the methods. The second level is an abstract class `StandardAutoencoder`, which supplies some functionality that is specific for the type of autoencoder we want to test. It additionally contains functionality to transform the autoencoder into a classifier, see Section 1.2 and Figures 1.5 and 1.6. This includes methods `classify`, `train_classifier` and `evaluate_classifier`. A method `evaluate` is also added for the autoencoder. The methods `train` and `train_classifier` are implemented. Training of the autoencoder supports Wasserstein loss and Euclidean loss. This is done by initializing the loss function at class instantiation:

```
if loss == 'wasserstein':
    # initialize wasserstein loss calculator
    self.loss_func = wloss.WassersteinCalculator(spatial_input_dim=self.
        spatial_input_dim, gamma=gamma, niter=niter,
        exponent=exponent, neighborhood_size=neighborhood_size,
        reference_measure=1, step_size=1).wloss
elif loss == 'sqeuclidean':
    self.loss_func = 'mean_squared_error'
```

Note that the function `wloss` has exactly the signature required for a `Keras` loss function. For training, we then compile the model with the loss function:

```
model.compile(optimizer=optimizer, loss=self.loss_func)
```

As optimizer, we use the optimizer Adam, `keras.optimizers.adam`, where we specify the learning rate, but leave all other hyperparameters fixed. The fitting method further uses a concept

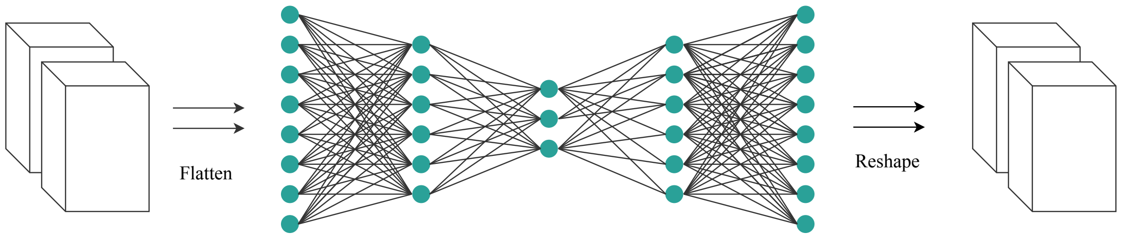


Figure 6.2: An autoencoder using dense layers with 3-dimensional input.

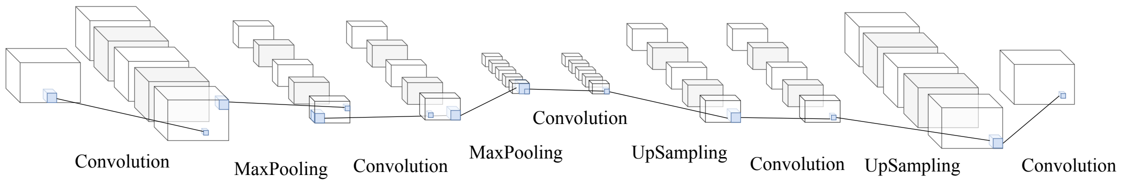


Figure 6.3: An autoencoder using convolutional layers. For reduction, MaxPooling is used. For decoding, UpSampling is used. In this autoencoder, 5 different convolutional filters are used in every layer except in the last one, where only 1 is used.

introduced in `Keras` called *callbacks*. A callback is a set of functions that will be automatically applied at given stages of the training procedure, for example after every epoch. We use the callbacks `Tensorboard`, `ModelCheckpoint`, `EarlyStopping` and a custom callback saving some example reconstructions at different iterations. `Tensorboard` is used for visualizing the train- and validation loss and accuracy during and after training. The class `ModelCheckpoint` supplies the automatic storage of the best model weights. It can be specified what "best" means: criteria could be validation loss, which is the default setting, or other validation metrics such as validation accuracy. `EarlyStopping` monitors the improvement of the validation loss. If it has not improved for more than a specified amount of epochs, the training procedure is stopped.

In total, the class `StandardAutoencoder` only the leaves the encoding, decoding and classification methods to be implemented. We derive two different subclasses: one which uses only standard dense layers for encoding and decoding, and one that only uses 3D convolutions. The architecture of the convolutional autoencoder is inspired by the convolutional autoencoder in [60]. In both cases, the classification is done by a single dense layer, mapping to a single output value in $[0, 1]$. This is achieved using a sigmoid loss function. As loss, we use *binary crossentropy*, which is a standard loss used for two-class classification. A schematic diagram of the dense autoencoder is shown in Figure 6.2, and a schematic diagram of the convolutional autoencoder is shown in Figure 6.3.

7 Experiments

In this chapter we present some experiments using the Wasserstein distance as a metric between brain regions. The chapter opens with a description of the data set. The first part of experiments serve as an evaluation of how the Wasserstein distance as a metric compares to the Euclidean distance. The second part of experiments concerns training autoencoders using the Wasserstein distance as a loss function.

7.1 Description of the Data Set

The considered data set of MR brain scans originates from various studies conducted by different compartments of the *Universitätsklinikum Münster*. The two main contributors to the data set are the *BiDirect Study*[1] and the study *FOR2107*[2].

For classification tasks we only use the data originating from the BiDirect study, since all scans have been performed by the same 3-Tesla MRI scanner [61], which means the influence of the particular scanner on the scan is eliminated. For this thesis, 993 T1-weighted (gray matter) MR scans of 993 different individuals, originating from the BiDirect study, were available. Out of these 993 individuals, 558 were female and 435 were male. Furthermore, 443 individuals were healthy controls aged between 35 and 66 years, randomly sampled from the population register of the city Münster. The other 560 individuals were diagnosed with major depressive disorder and recruited from different psychiatric and psychosomatic hospitals in the region [62].

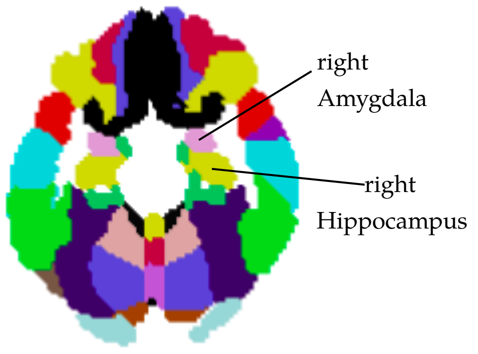


Figure 7.1: An axial slice of the AAL atlas. The left and right Amygdala is depicted in light pink in the upper half. The left and right Hippocampus can be seen in yellow next to the Amygdala. The image was obtained through `MRIcron`.

The MR scans were then processed using voxel based morphometry, which includes fitting the scan to a template and correcting for total brain volume. Since the resolution of the whole brain scan is too large to consider for Wasserstein distances, we restrict ourselves to the right Amygdala and the right Hippocampus. These are regions which have been shown to be associated with depression, see for example [63]. The position of the regions within the brain are determined using the AAL atlas[3], and an interface for extraction was supplied by the software `Photon`[4], which was developed by the machine learning team of the Department of Psychiatry of the University of Münster. The regions are embedded into a cuboid. After extraction, the region

[1]`https://www.medizin.uni-muenster.de/epi/probanden/bidirect-studie/`
[2]`http://for2107.de/`
[3]`http://neuro.imm.dtu.dk/wiki/Automated_Anatomical_Labeling`
[4]https://www.photon-ai.com/

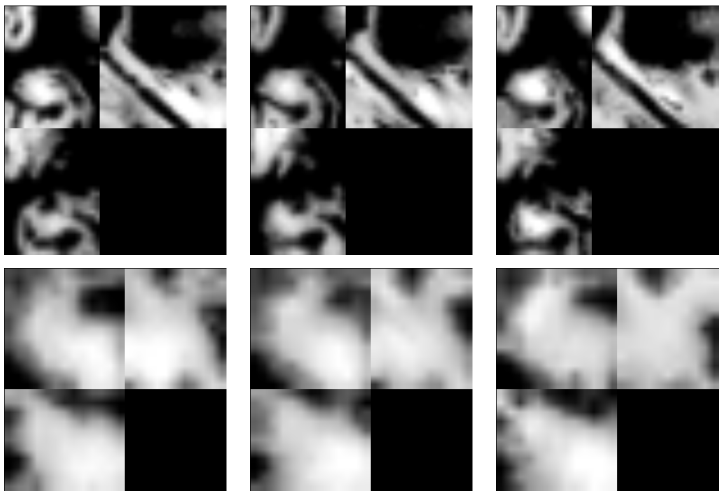


Figure 7.2: Visualization of coronal, sagittal and axial slices (yz-, xz- and xy-planes) of the regions Hippocampus (top row) and Amygdala (bottom row) for three different individuals.

Amygdala has shape $13 \times 11 \times 13$ and consists of a total of 1859 voxels. The region Hippocampus is slightly larger with a size of $21 \times 28 \times 27$ and consists of a total of 15876 voxels. The extracted regions can be visualized with the software `MRIcron`[5]. Figure 7.2 shows such a visualization. A visualization of the AAL atlas that divides the brain template into different regions is shown in Figure 7.1.

7.2 Evaluation of the Wasserstein Distance as a Metric Between MR Images

In this section we want to describe some experiments to evaluate if the Wasserstein distance is suitable as a metric between brain region MR images. For lack of alternatives, we try to do this by comparing how well classification tasks can be performed using the Wasserstein distance instead of the Euclidean distance. This can be done by replacing the Euclidean distance in classification pipelines. It is necessary to choose parameters for the Wasserstein distance. These parameters include parameters arising from the algorithm used to approximate the real Wasserstein distance, and a parameter concerning the ground metric. We treat the choice of parameters in the first subsection, and apply the different resulting configurations to classification pipelines in the following subsection.

7.2.1 Calculating the Wasserstein Distance with Different Parameters

In this section we discuss the different parameters that need to be chosen to calculate Wasserstein distances. While some of these parameters can be fixed without loss of generality, there are several possibilities for others, which leads to different configurations.

The entropy-regularized Wasserstein distance (ERWD) depends on the following parameters:

- the exponent p of the ground distance,
- the side length h of the voxels,
- the regularization parameter γ,

[5]`https://www.nitrc.org/projects/mricron`

- the number of iterations performed in Algorithm 4.1,
- the reference measure λ in the general setting corresponding to the area weights α in the discrete setting,
- the numerical precision (single or double),
- the remote point distance, if using the Kantorovich extension for unnormalized measures.

For the choice of the reference measure, we note that all voxels have the same size, thus it makes sense to define constant area weights α^i. The voxels further have a given volume defined through their side length h. A natural choice would thus be to define $\alpha^i = h^3$ for $i \in I$ with I as in Definition 1.1.1. As noted in Remark 3.3.11, choosing any other constant area weights α only results in a constant shift of the resulting distance values, thus the particular choice of the value of α^i is not really relevant. Further, the choice of h is linked to the choice of γ: only the relation $\frac{h^p}{\gamma}$ is relevant, since the distance appears only in the kernel $K_{\gamma,p} = \exp\left(-\frac{d^p}{\gamma}\right)$. We can thus fix $h = 1$ and $\alpha^i = 1$ in the following experiments. Now for choosing feasible values for γ, it makes sense to consider that the values of the kernel $K_{\gamma,p}$ should not be too small for numerical precision. We have seen in Equation 4.14, that

$$K_{\gamma,p}(x^i, x^j) > \delta \Leftrightarrow \gamma > h^{\frac{p}{2}} \|i - j\|^p (-\log \delta)^{-1}$$

for $i, j \in I$ and for any small constant δ. Letting $h = 1$, we should thus choose

$$\gamma > (-\log \delta)^{-1}$$

to have at least some values greater than δ (for points with $\|i - j\| \leq 1$). For single precision, the smallest representable positive number is approximately 10^{-37} and for double precision it is 10^{-308}. This results in $\gamma > 0.01$ and $\gamma > 0.001$, respectively. However, just having some representable numbers is not enough, but we choose instead to consider a neighborhood on which all values should be representable. Let r be the "radius" of the neighborhood, i. e. we consider for each voxel x^i a neighborhood centered at this voxel in the shape of a cube of side length $2r + 1$. The maximal distance of x^i to any point in this cube is $\sqrt{3r^2}$. We thus have to choose

$$\gamma > (3r^2)^{\frac{p}{2}} (-\log \delta)^{-1}.$$

However, this is not sufficient to guarantee the convergence of the algorithm, since the problem of potentially unbounded dual variables remains. In order to evaluate how many representable values are necessary to obtain sensible results, we first calculate a value for γ such that all values are representable. This γ depends on the size of the region. We then truncate the kernel to different cube-shaped neighborhoods and consider the difference between the distances resulting from a truncated kernel and the distances resulting from the full kernel. We take a mean over 10 computed distances for each region. The resulting plots for different exponents are depicted in Figure 7.4 for the Amygdala. We conclude from the plot that choosing a radius of 5 is sufficient for all exponents. A similar plot for the Hippocampus suggests a radius of 7.

The exponent p is chosen from the set $\{0.1, 0.5, 1, 2\}$. Using an exponent smaller than 1 was proposed in [40] for classification of MNIST images using Wasserstein kernels. The exponent $p = 0.1$ has to be interpreted as the 1-Wasserstein distance with the ground metric $d(x, y) = \|x - y\|_2^{0.1}$. An illustration of how the different exponents influence the costs is shown in Figure 7.3. It can be seen that for the exponent 0.1, the most variance is within a local neighborhood.

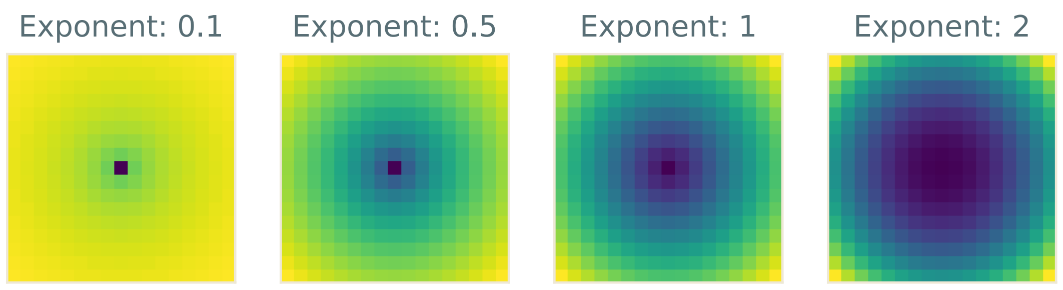


Figure 7.3: Distances of the central point for different exponents in two dimensions. Each plot has its own color scale for visualization purposes. Darker colors correspond to lower values.

The remote point distance is chosen dependent on the maximum ground distance. We consider factors of 0.5 and 1.1, which both make the extended distance a metric.

In order to choose a number of iterations, we have to evaluate the convergence of the algorithm. Remember that the algorithm calculates "dual variables" a and b defining the transport plan as $u = D_a K_{\gamma,p} D_b$, where D_a denotes the matrix whose diagonal is a. These variables a and b define the optimal solution if the marginal conditions

$$\begin{aligned} a \otimes K_{\gamma,p}(\alpha \otimes b) &= r \\ b \otimes K_{\gamma,p}(\alpha \otimes a) &= s. \end{aligned} \tag{7.1}$$

as stated in Equation 4.7 are fulfilled. Remember further that the algorithm works by alternatingly enforcing one of the conditions. Thus after each iteration, one of the conditions should always be fulfilled if no numerical issues have arisen. We propose to evaluate the convergence by looking at

$$\begin{aligned} &\left\| a \otimes K_{\gamma,p}(\alpha \otimes b) - r \right\|_1, \\ &\left\| b \otimes K_{\gamma,p}(\alpha \otimes a) - s \right\|_1. \end{aligned} \tag{7.2}$$

We evaluate the convergence for different parameters using 10 samples for each region and plotting the average. Table 7.1a lists the considered configurations.

We do not treat the unnormalized variant, since it became clear after a few experiments that

	p	γ	truncate	iterations
(1)	0.1	0.015	no	1000
(2)	0.1	0.014	5	2000
(3)	0.5	0.054	no	1000
(4)	0.5	0.034	5	1000
(5)	1	0.258	no	500
(6)	1	0.099	5	1000
(7)	2	5.805	no	150
(8)	2	0.859	5	200

(a) Configurations using automatically calculated γ values.

	p	γ	truncate	iterations
(9)	0.1	0.01	no	3000[6]
(10)	0.1	0.1	no	100
(11)	0.1	1	no	3

(b) Additional configurations.

Table 7.1: List of configurations for the Amygdala. For truncation, the neighborhood size was chosen as per Figure 7.4.

[6]With double precision

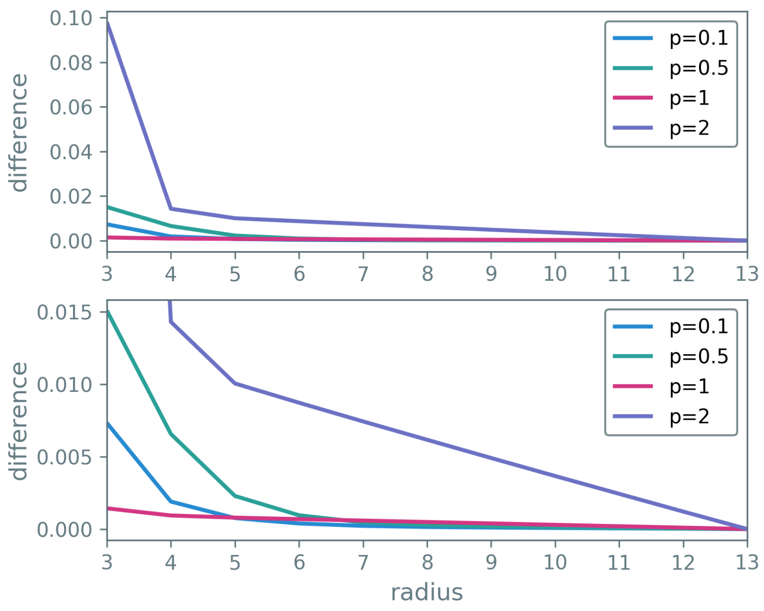


Figure 7.4: Differences of Amygdala distances after constraining transport to a neighborhood of a specific radius compared to using the full grid (radius = 13). For $p = 2$ the difference is more distinctive. This is reasonable since the costs for far transport increase fast for $p = 2$.

	(1)	(2)	(2a)	(10)	(10a)	(10b)	(10c)	(11)
(1)	0.0	0.58	10.46	21.06	19.84	19.77	21.06	52.07
(2)	0.58	0.0	10.24	21.31	19.57	19.51	21.31	51.82
(2a)	10.46	10.24	0.0	19.92	9.44	9.81	19.92	42.48
(10)	21.06	21.31	19.92	0.0	29.5	29.02	0.0	60.92
(10a)	19.84	19.57	9.44	29.5	0.0	0.76	29.5	33.92
(10b)	19.77	19.51	9.81	29.02	0.76	0.0	29.02	33.2
(10c)	21.06	21.31	19.92	0.0	29.5	29.02	0.0	60.92
(11)	52.07	51.82	42.48	60.92	33.92	33.2	60.92	0.0

Table 7.3: Average deviation between distances resulting from two different configurations in percent, relative to the maximum value, rounded to two decimals. The considered region was the Amygdala.

the convergence behavior does not change significantly.

The convergence of the algorithm depends on γ. The larger γ is, the longer it takes for the algorithm to converge. As an example, we consider additionally the configurations listed in 7.1b for the region Amygdala using the exponent 0.1. The approximate numbers of iterations needed for convergence of the respective configurations are listed in Table 7.1. If not stated otherwise, the calculations were always made with single precision. Some selected convergence plots are shown in Figure 7.5.

Out of the configurations listed in Table 7.1, we test the those which use the exponent $p = 0.1$ as well as the following modified configurations:

- (2a) using UMTP and a factor of 0.5 (relative to the neighborhood),
- (10a) using UMTP and a factor of 0.5,
- (10b) using UMTP and a factor of 1.1,
- (10c) using double precision.

[7]Using a convolutional filter of shape $(7, 7, 7)$

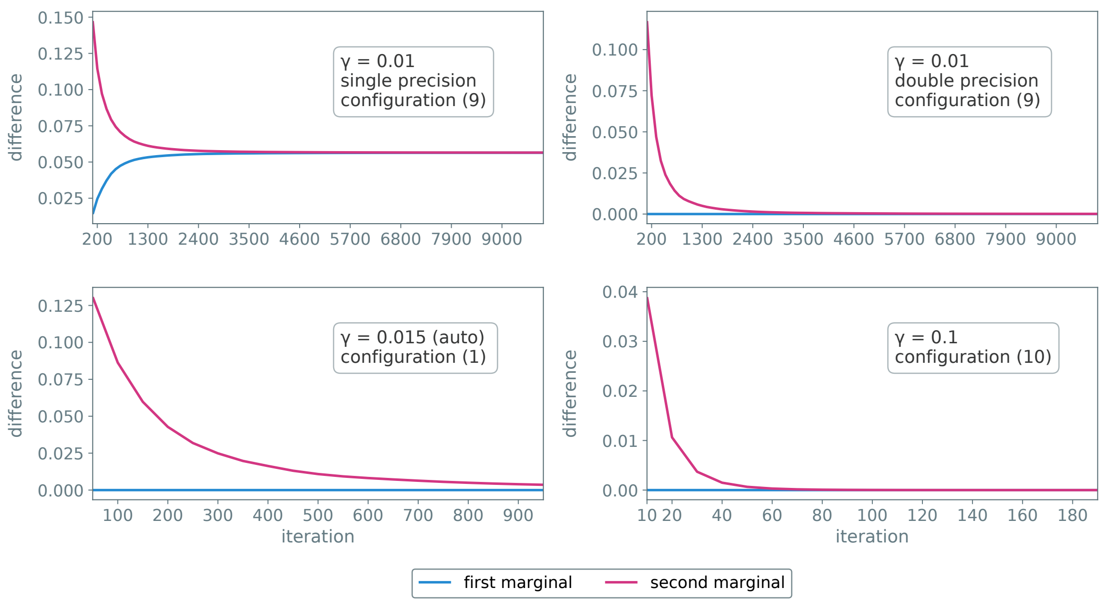


Figure 7.5: Convergence plots for different configurations for the region Amygdala with exponent $p = 0.1$. To put the values into context, note that the marginals have total mass 1.

We exclude configuration (9), because the runtime was too long. All calculations were performed on a server made available by the Department of Psychiatry, which is equipped with NVIDIA Titan Xp and NVIDIA GeForce GTX 1080 graphic cards. To evaluate the influence of the different parameters, we consider the average absolute difference between the pairwise distance matrices resulting from two different configurations. To obtain relative values, we divide these differences by the largest distance value of the two configurations and further multiply by 100 to get percentage values. The results are listed in Table 7.3. As already suggested by Figure 7.4, the difference between configurations (1) and (2) is small. Similarly, the difference between configurations (10a) and (10b) is small, since they are the same except for the remote point distance. There is however a rather large difference between using UMTP and not using UMTP, see the differences between (10) and (10a) and between (2) and (2a). The largest differences occur for configuration (11). Configurations (10) and (10c) yield practically identical distances, so using only single precision did not make a difference in this case. Table 7.2 lists the runtimes and iterations for each configuration.

For comparison, we have also included a run using convolution, as proposed in Section 4.2. We used a filter of shape $(7, 7, 7)$, which corresponds to a neighborhood of radius 3. It is remarkable that it took much longer than using matrix multiplication. This effect is also visible when using CPU instead of GPU: calculating 100 distances using convolutions took about 60 times as long as calculating the same 100 distances using matrices. We further see that using double precision instead of single precision had a large impact on the runtime. All these effects are probably due to implementation details of `TensorFlow`.

We conclude for further experiments that using convolutions is not advisable. We also choose not to further consider double precision calculations: The runtime when using double precision is dramatically higher than for single precision, yet we observed, by comparing configurations (10) and (10c), that there does not seem to be a large difference between single and double precision when both lead to convergence. However, double precision has to be considered when using single precision is not sufficient, as is the case for smaller values of γ, for example in configuration (9). We saw that for smaller values of γ, the amount of iterations needed for convergence increases, which leads to yet longer runtime.

A comparison between configurations (10a) and (10b) also showed that the influence of the remote point distance seems to be little. We thus fix a remote point distance of 0.5.

	iterations	runtime
(1)	1000	46m 50s
(2)	2000	57m 29s
(2a)	2000	57m 15s
(10)	300	13m 33s
(10^7)	300	5h54min58s
(10a)	300	9m 20s
(10b)	300	13m 28s
(10c)	300	4h 47m 59s
(11)	300	8m 51s

Table 7.2: Runtimes of the different configurations for the Amygdala.

7.2.2 Classification

In this subsection, all experiments are performed using the following setup. We consider two tasks for classification: classification regarding gender and classification regarding diagnosis, where the diagnosis can be one of "major depressive disorder (MDD)" and "healthy control (HC)". As a scoring metric we use accuracy, where

$$\text{accuracy} = \frac{\text{number of correctly classified samples}}{\text{number of total classified samples}}.$$

This is only informative if both classes are balanced, i. e. if the set to classify consists of samples of both classes in equal shares.

For validation, we use *4-fold stratified cross-validation*, meaning that the data set is split into four parts, where each part contains roughly the same amount of samples of the first class as of the second. The model is then trained on three of the four parts, while the other part is used for testing. This is repeated for each combination, such that each part is used once as a test set. We repeat this procedure 6 times, each time shuffling the data. We then report the average test accuracy score. This is implemented using `RepeatedStratifiedKFold` from the package `model_selection` of `scikit-learn`.

We use *grid search* as a hyperparameter optimization strategy, meaning we test every combination of hyperparameters in a set range. Grid search is implemented in `scikit-learn` in the package `model_selection` as `GridSearchCV` and it is possible to use it with `RepeatedStratifiedKFold`.

Wasserstein SVC

As explained in Section 5.1, we can define a kernel function from the Wasserstein distance. We now want to examine how well support vector classification with Wasserstein RBF kernels performs in comparison to support vector classification with Euclidean RBF kernels for different parameters of ε.

As a reminder, the RBF kernel is defined as

$$k_f^{\text{rbf}}(x,y) = \exp(-\varepsilon f(x,y)) \text{ for } x,y \in \mathcal{Z},\ \varepsilon \in \mathbb{R}_{>0},$$

for some function f that is symmetric, positive and has zero-diagonal, where $\mathcal{Z}$ is the set of MR images of brain regions. We now want to compare the usage of the Euclidean distance, i. e. $f(x,y) = \|x-y\|_2^2$, to the usage of the p-Wasserstein distance for various parameters p and q, i. e. $f(x,y) = \mathcal{W}_{\gamma,p}^{pq}$. In order for the influence of the ε value to be comparable for Wasserstein and Euclidean distances, we divide the distances by the 50th percentile of the respective precomputed

	γ	score
(1)	0.015	75.9
(2)	0.014	75.9
(2a)	0.014	73.9
(10)	0.1	75.3
(10a)	0.1	73.8
(10b)	0.1	73.9
(10c)	0.1	75.3
(11)	1	61.3
Euclidean		74.0

(a) Scores for $p = 0.1$.

	p	score
(2)	0.1	75.9
(2a)	0.1	73.9
(4)	0.5	74.3
(4a)	0.5	72.2
(6)	1	71.9
(6a)	1	69.9
(8)	2	72.0
(8a)	2	66.1
Euclidean		74.0

(b) Scores for different exponents.

Table 7.4: Accuracy scores in percent obtained for gender classification using different exponents on the region Amygdala.

distance matrix. We consider $\varepsilon \in \{10^{-3}, 10^{-2}, 10^{-1}, 1, 10\}$.

An additional hyperparameter is given by the misclassification penalty parameter C, as described in Section 1.3. We consider $C \in \{0.1, 1, 10, 100\}$.

As exponent q for the Wasserstein kernel, we consider the values $q = 0.5$ and $q = 1$ for $p = 2$, and $q = 1$ and $q = 2$ for $p = 1$ (remember that $p = 0.1$ actually means the 1-Wasserstein distance with a different ground metric). As already noted in Chapter 5 in Proposition 5.1.4, these kernels are not guaranteed to be positive definite. We can however evaluate how many negative eigenvalues appear in the empirical, finite kernel matrices. It turns out that there are almost always negative eigenvalues present for $q = 2$, and that all kernels constructed using configuration (11), which has a high regularization parameter, have negative eigenvalues.

Since the classification accuracy for the diagnosis using support vector machines on the region Amygdala is only marginally above 50%, we consider only gender classification on this region. The results using the configurations from the previous section are shown in Table 7.4a. The performance is not significantly better, however the best result using Wasserstein kernels is 2 percentage points above the best result using Euclidean kernels. Both were obtained using RBF kernels. It is surprising that the configurations using the UMTP (2a and 10a,b) do not give better results than the configurations where the brain region volume is normalized. It is however not surprising that the configuration (11) performs badly, since the regularization parameter was chosen very high, which means that subtle differences cannot be detected well. Since the configurations (1) and (2), the ones using an automatic γ-value, gave the best results, we consider for the other exponents only the corresponding configurations and the UMTP-variant. The results, including the corresponding configurations for $p = 0.1$, are shown in Table 7.4b.

To test the influence of negative eigenvalues, we take a closer look of the classification results of configuration (2). It had negative eigenvalues for $q = 2$. The performance of the specific RBF kernels is depicted in Table 7.5.

We repeat some of the experiments for the larger region Hippocampus. We consider the configurations listed in Table 7.6. The iterations and runtime for the whole data set are included. It is apparent that the size of the region has a large impact on the runtime, which is why we do not consider any larger regions. However, this would not prevent an application in practice, since to classify a new sample one only has to calculate the distance of one rather than of 993 samples to all other samples. It is actually even enough to calculate the distance to all support vectors

q	ε	score	std. score	neg. ev.
1	0.001	67.45	3.26	0
1	0.01	70.40	2.68	0
1	0.1	75.34	2.49	0
1	1.0	75.94	2.38	0
1	10.0	63.47	3.36	0
2	0.001	66.02	3.21	405
2	0.01	66.02	3.21	405
2	0.1	71.84	2.78	405
2	1.0	74.71	2.56	402
2	10.0	75.73	2.74	369

Table 7.5: Mean classification scores for different kernels, using configuration (2) on the region Amygdala. Additionally to the mean score, we also list the standard deviation of the scores. The misclassification parameter C is always 10, since this always gave the best results. The last row lists the amount of negative eigenvalues of the RBF kernel. No relation between negative eigenvalues and score is apparent.

when using SVMs. This can be done in about a minute. The classification results are shown in 7.7a. Differently to the previous region, the configuration (1), which uses a higher γ performs a little better than the configuration (2), which uses the smallest possible γ for a neighborhood. Again the results are only marginally better than those obtained using the Euclidean distance.

	p	γ	truncate	UMTP	iterations	runtime
(1)	0.1	0.1	no	no	300	14h18min48s
(1a)	0.1	0.1	no	yes	300	18h3min23s
(2)	0.1	0.015	7	no	1000	1d22h56min
(3)	0.5	0.1	no	yes	1000	1d23h32min

Table 7.6: Considered configurations for the region Hippocampus, including the number of iterations made and the runtimes for the whole data set.

	p	γ	score gender	score diagnosis
(1)	0.1	0.1	84.9	60.2
(1a)	0.1	0.1	79.2	60.1
(2)	0.1	0.015	83.3	58.6
(3)	0.5	0.1	78.1	59.9
Euclidean			83.3	58.4

(a) Raw MR images

	γ	score gender	score diagnosis
(1)	0.1	84.1	58.7
(1a)	0.1	81.2	57.3
(2a)	0.015	83.8	58.1
(1)	0.1	83.4	58.4

(b) Gradient images. Top: full gradients, bottom: 50th percentile.

Table 7.7: Accuracy scores for classification with respect to gender and diagnosis on the region Hippocampus.

For both regions, we have seen that the Wasserstein distance does not seem to perform better than the Euclidean distance. A cause for this might be the fact that the images have a large overlapping support. As we have seen in Chapter 5, the Wasserstein distances lead to better results than the Euclidean distance if the samples have little overlapping support. Further, the Wasserstein distance might perform badly because the regions contain homogeneous parts and we might be more interested in contours. For this reason, we generate contour images by taking the gradient magnitude, i. e. the norm of the gradient. The gradient can be computed using the Sobel filter, see Section 1.1. A corresponding function is implemented in `scipy.ndimage`. We further consider a variant where values in the 50th percentile are set to zero. Illustrations can be seen in Figure 7.6, which shows that the images still have a large common support. No improvement in classification accuracy could be achieved, as can be seen in 7.7b.

[8] `https://www.nitrc.org/projects/mricron`

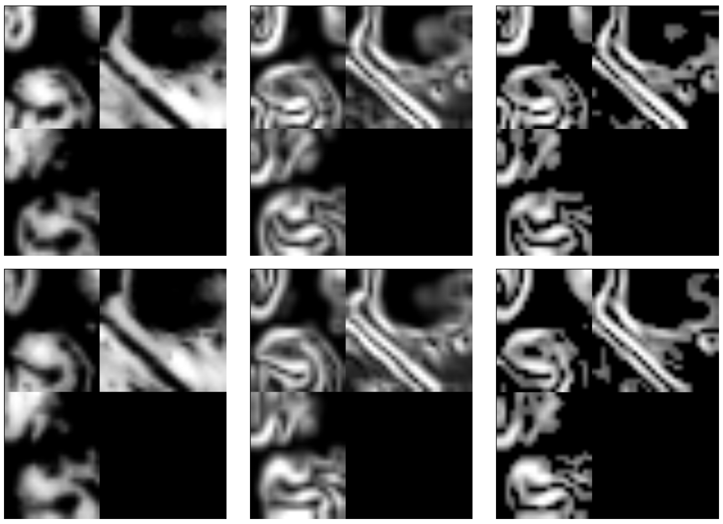


Figure 7.6: Slices from the region Hippocampus for two different samples. Left: original, center: gradient magnitude, right: gradient magnitude using only the upper 50th percentile. Visualized using `MRIcron`.[8]White pixels correspond to large values.

Wasserstein PCA

The Wasserstein kernel can not only be used for support vector classification, but also in kernel principal component analysis, which was introduced in Section 1.4. An evaluation of the resulting reduction is again possible by classification. For classification we use Euclidean RBF SVM and Linear SVM, where the hyperparameters ε, C and q are chosen in the same range as before.

As in the previous subsection, we start with the classification with respect to gender on the region Amygdala. As reduction size we consider the extreme reduction to only 2 components, the reduction to approximately 10% of the components, which amounts to 185 components and the reduction to approximately 2% of the components, which amount to 32 components. We consider the configurations (2), (2a), (4), (4a), (6), (6a), (8), (8a), (10), (10a) and (11) as in Table 7.4. We only consider a Wasserstein RBF kernel, since it seemed to perform better than the linear kernel. The results are shown in Table 7.8a.

It is surprising that configuration (11) yields results almost as good as the other configurations, while the classification using SVM performed noticeably worse. Again, the results are comparable to those obtained using the Euclidean distance in an RBF kernel. We see that a classification using only 2 components is not possible. However, a reduction to 185 components performs similarly to using all components, where the best classification accuracy was 75.9%, see 7.4a. The classification using only 32 components performs a little worse.

The same experiment can be repeated with the region Hippocampus. We test a reduction to 128 components, which corresponds to a reduction to approximately 1% of the components. The results are shown in 7.8b. In total, the diagnosis can be predicted with similar accuracy on a basis of 128 components as on a basis of all components, where the best achieved accuracy was 60.2%, see Table 7.7a. For gender reduction, the accuracy is a little lower using a basis of 128 components, since the accuracy was almost 85% before, see Table 7.7a. We only compared the exponents $p = 0.1$ and $p = 0.5$, since the other exponents did not perform well for the Amygdala. We observe that $p = 0.1$ performs considerably better than $p = 0.5$.

	p	2 cp.	32 cp.	185 cp.
(2)	0.1	58.6	73.1	75.4
(2a)	0.1	55.7	73.1	73.7
(4)	0.5	59.0	72.6	74.4
(4a)	0.5	58.7	70.3	72.9
(6)	1	59.7	71.4	72.1
(6a)	1	58.3	67.9	71.7
(8)	2	60.4	70.9	71.9
(8a)	2	54.6	66.4	66.5
(10)	0.1	60.7	73.1	75.7
(10a)	0.1	55.1	73.0	74.5
(11)	0.1	58.7	72.6	73.8
Euclidean		60.7	72.7	75.3

(a) Gender classification on Amygdala.

	p	γ	score gender	score diagnosis
(1)	0.1	0.1	81.0	58.7
(1a)	0.1	0.1	78.2	59.4
(2)	0.1	0.015	81.1	56.9
(3)	0.5	0.1	73.9	58.2
Euclidean			79.9	60.0

(b) Classification on Hippocampus for 128 components.

Table 7.8: Classification results on the regions Amygdala and Hippocampus after a reduction to different amounts of components using kernel PCA with different kernels.

7.2.3 Summary

To summarize, we have seen that the Wasserstein distance performs marginally better than the Euclidean distance over an average of 24 train and test splits. However, the results have not been cross-validated in a nested fashion, which means that the results tend to be optimistic. It would thus be incautious to leap to any conclusions starting from such small differences.

We have observed that the exponent $p = 0.1$ performed noticeably better in almost all experiments. An explanation for this might be that the most relevant differences in the images are local: the cost function corresponding to $p = 0.1$ still penalizes higher transport distances, but the increase of costs is slowed for higher distances, while for $p = 1$ the increase remains constant, and for $p = 2$ it gets faster with higher distances (the derivatives of d^p with respect to the distance d are strictly monotonically decreasing, constant and strictly monotonically increasing, respectively). This means that local changes carry much weight for $p = 0.1$, while they carry little weight for $p = 2$. Compare also Figure 7.3. We have further observed that the UMTP variant for unnormalized measures did not perform better. This was unexpected, since we assumed that we lose information by normalizing the MR images. It remains an open question why this loss of information seems to be irrelevant.

Since the Wasserstein distance did not perform worse than the Euclidean distance, it still makes sense to test it as a loss function in autoencoders. For this, we aim to choose parameters on the basis of the previous results. Thus, we choose the exponent $p = 0.1$. For the Amygdala, we choose a neighborhood size of 5 with the corresponding γ value. For the Hippocampus, the configuration with a larger value of $\gamma = 0.1$ seemed to perform better and additionally requires fewer iterations.

However, we cannot put too much trust into these choices, since these parameters were only tested on images of actual brain regions, which are already quite similar to each other. When using the Wasserstein distance in an autoencoder, we compare these images to artificial images that do not look like brain regions in the first epochs. But since we cannot try all configurations, we stick with this choice.

7.3 Using the Wasserstein Distance as a Loss Function in Autoencoders

In this section we will present the results obtained by using the Wasserstein distance in an autoencoder. We conduct experiments using the regions Amygdala and Hippocampus. For training the autoencoder, we use the total data made available by the Department of Psychiatry, obtained through different studies. We then discard the decoder of the autoencoder and replace it by a very simple classifier. This classifier is then trained using only the data from the BiDirect study, which was also used in the previous experiments. No hyperparameter optimization has been made and the presented models have been found by trial and error.

7.3.1 Fully Connected Autoencoder for Amygdala

The first architecture we present is a fully connected architecture as shown as a schematic diagram in Figure 6.2. We use two layers for encoding and two layers for decoding, as in the diagram. For the region Amygdala, we consider a reduction to a hidden representation of size 32, which is approximately 2% of the original size. The intermediate layers have size 1024. We then try two activation functions: a sigmoid activation function and a ReLu activation function. An illustration of the Amygdala autoencoder can be seen in Figure 7.7.

If a Wasserstein loss is used, the data is normalized before training and the last activation function is replaced by a softmax function to assure that the output is also normalized. As parameters for the Wasserstein distance, we use $p = 0.1$, a neighborhood of size 5 with the γ as the smallest value that asserts representable numbers in the neighborhood, and 1000 iterations. The same architectures are tested using a Euclidean loss, with the only difference that the data is not normalized and the last activation function is not a softmax function. As learning rate we have chosen a small rate of 10^{-5}, since we found that this prevented overfitting. We stop training if the validation loss has not improved for 100 epochs when using a Euclidean loss and for 50 epochs when using a Wasserstein loss. We choose a lower number for the Wasserstein loss because training takes much longer: while training with a Euclidean loss took about 0.5s per epoch, training with a Wasserstein loss took about 15s per epoch.

A plot of the loss is shown in Figure 7.8. It can be seen that none of the configurations suffers from overfitting, since both validation loss and train loss go down. The Euclidean loss comes closer to zero, but this does not necessarily have to mean anything: while we have subtracted the theoretical minimum from the Wasserstein distance, asserting that the smallest assumable value is indeed zero (which is in general not the case because of the regularization), the minimum distance is not achieved between any two identical measures, see Corollary 3.4.3.

For classification, we replace the decoder with a classifier, which adds a fully connected layer with size 1, followed by a sigmoid activation function, which outputs a value between 0 and

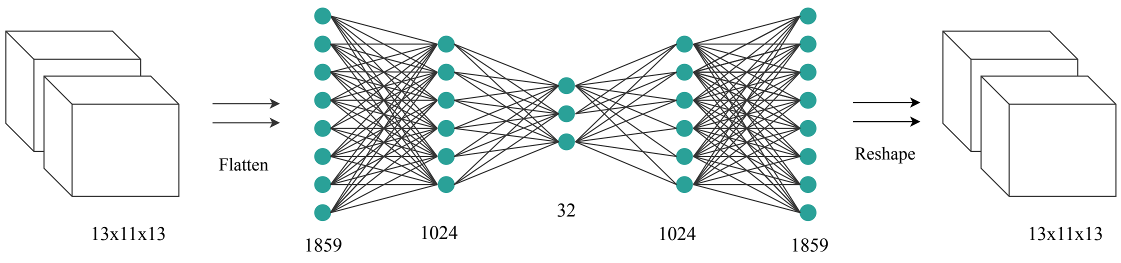


Figure 7.7: Architecture of the dense Amygdala autoencoder.

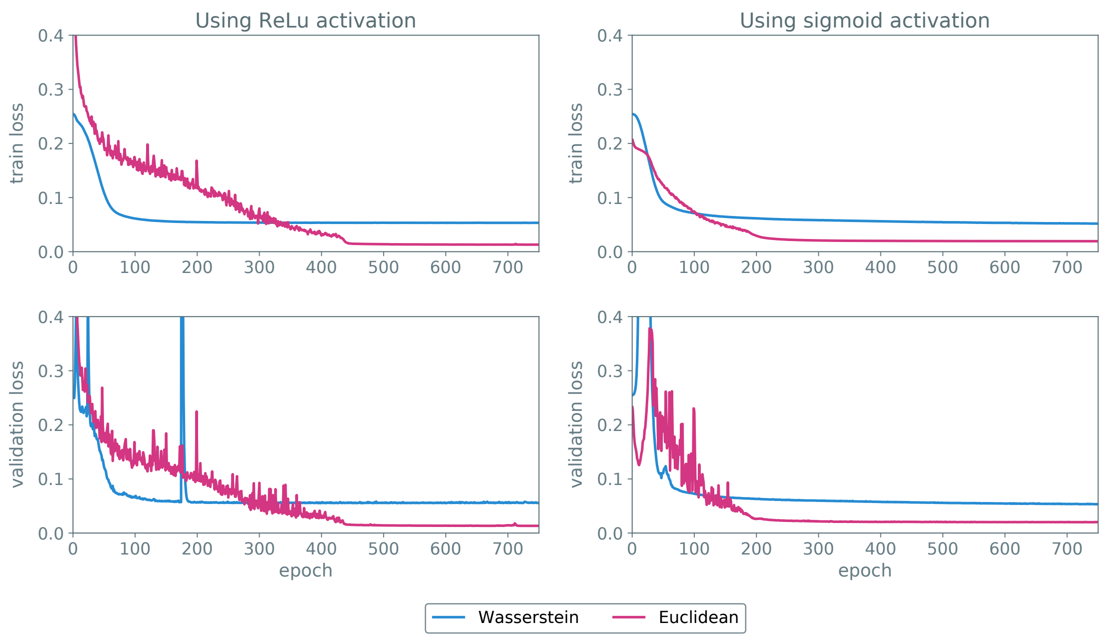


Figure 7.8: Train and validation loss for different configurations for the dense autoencoder trained on the region Amygdala.

	sigmoid	ReLu	PCA	no pretraining
Wasserstein	66.4	73.3	73.1	72.2
Euclidean	71.5	70.7	72.7	

Table 7.9: Gender classification results using a classifier constructed from the autoencoder in Figure 7.7 trained on the region Amygdala using Wasserstein and Euclidean loss, respectively.

1. This value is then rounded to get a class prediction of either 0 or 1. To prevent overfitting, a dropout layer is added, which consists of randomly setting a fraction of input units to 0. We have experimentally chosen a dropout rate of 60%. To evaluate the performance of the classifier, we have applied the same repeated cross-validation scheme as before, notably a 4-fold cross validation repeated 6 times. This was possible since the network to train and the training set were not very large. The results are presented in Table 7.9. We see that the average results are relatively similar for all methods. We also test training the classifier without pretraining the lower layers with an autoencoder. This leads to a slightly worse result. Note that while the results using autoencoders instead of principal component analysis are only marginally better, there exists further potential in hyperparameter and architecture optimization, while there are no additional parameters for principal component analysis.

A visualization of the learning process of the different configurations is shown in Figure 7.11. It can be seen that the configurations using a Wasserstein loss have already learned a good reconstruction after 100 epochs in contrast to the configurations using a Euclidean loss. This is also mirrored in the train and validation loss shown in Figure 7.8. It further looks like the configuration with Euclidean loss and ReLu activations learned the mean of the images, since the final reconstructions are not distinguishable. All other configurations produced distinguishable reconstructions.

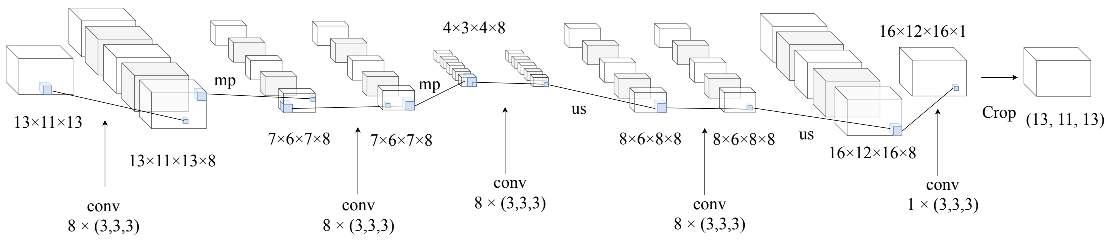


Figure 7.9: Architecture of the convolutional Amygdala autoencoder. MaxPooling layers and UpSampling layers are abbreviated with "mp" and "us", respectively. Since the dimensions of the Amygdala are not powers of two, the output has to be cropped.

7.3.2 Convolutional Autoencoder for Amygdala

The second architecture is a convolutional autoencoder. The architecture is shown in Figure 7.9. We use 8 convolutional filters of size (3,3,3) for each of the two convolutional layers. Between those layers, we use MaxPooling layers with a pooling size of (2,2,2) and strides of (2,2,2), resulting in a reduction by a factor of 8. Since we apply MaxPooling two times, we get a total reduction by a factor of 64, which amounts to a reduction to approximately 1.5% of the input size. After summing up the results from the 8 different filters, which is done implicitly by the following convolutional layers in the decoder, the hidden representation has shape $4 \times 3 \times 4$ (size 48). A plot of the loss for different configurations is shown in Figure 7.10. For a Euclidean loss, no difference between ReLu and sigmoid activation is apparent. For the Wasserstein loss, sigmoid activation functions lead to more oscillation and no improvement in the validation loss, which is why training broke off after approximately 350 iterations. We notice that here, the Euclidean validation loss is smoother, while for the dense autoencoder, the Wasserstein validation loss was smoother. One could hypothesize that the Wasserstein loss behaves better than the Euclidean loss for the dense autoencoder because it already incorporates spatial information, while for the Euclidean loss it is necessary to incorporate the spatial structure by using convolutions. The advantage of using spatial information is further suggested by the reconstructions shown in Figure 7.12: the Euclidean reconstructions after 100 epochs (see Figures 7.12a and 7.12c) already look much better than the reconstructions after 100 epochs using the dense autoencoder, see Figures 7.11a and 7.11c. However, a more detailed analysis would be necessary to support this hypothesis. Figure 7.12 further shows that the reconstructions are less accurate than with the dense autoencoder, which is mirrored by the respective loss values: for both Euclidean and Wasserstein loss, train and validation loss are lower for the dense architecture. We also trained classifiers using the convolutional autoencoder, but the results were similarly inconclusive as with the dense autoencoder.

7.3.3 Convolutional Autoencoder and Fully Connected Autoencoder for Hippocampus

We further tried the same techniques on the region Hippocampus. For the Wasserstein loss, we decided to use the parameters $p = 0.1$ and $\gamma = 0.1$ with 300 iterations (configuration (1) from Table 7.6). For the dense autoencoder, we used a hidden layer of size 2048 and a reduction size of 128. For the convolutional autoencoder, we used the same size and number of filters, resulting now in a reduced representation of shape $6 \times 7 \times 7$ (size 294). Since this region is larger than the Amygdala, training times were longer: Training using a Euclidean loss took about 2s per epoch for both autoencoders. Training using a Wasserstein loss took about 150s per epoch for the dense autoencoder and 280s per epoch for the convolutional autoencoder. This means that

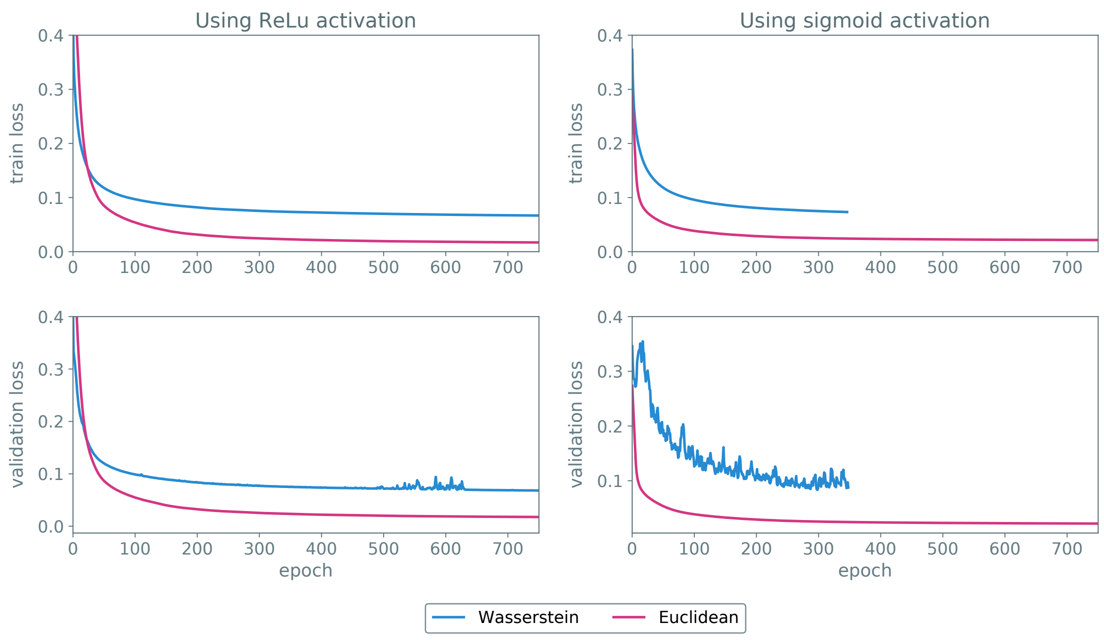


Figure 7.10: Train and validation loss for different configurations of the convolutional autoencoder trained on the region Amygdala.

training the dense autoencoder for 1000 epochs with a Wasserstein loss would take over 3 days. Again, we stopped training after no improvement in the validation loss for 50 and 100 epochs, respectively.

Plots of the losses can be seen in Figures 7.13 and 7.14. Both show that using a sigmoid activation function with a Wasserstein loss did not give good results. As on the Amygdala, the Euclidean loss is smoother for the convolutional autoencoder. When using a ReLu activation function, the Wasserstein training loss is smooth, and the validation loss is smooth with one exception after about 100 iterations. Thus, the losses behave similarly as before.

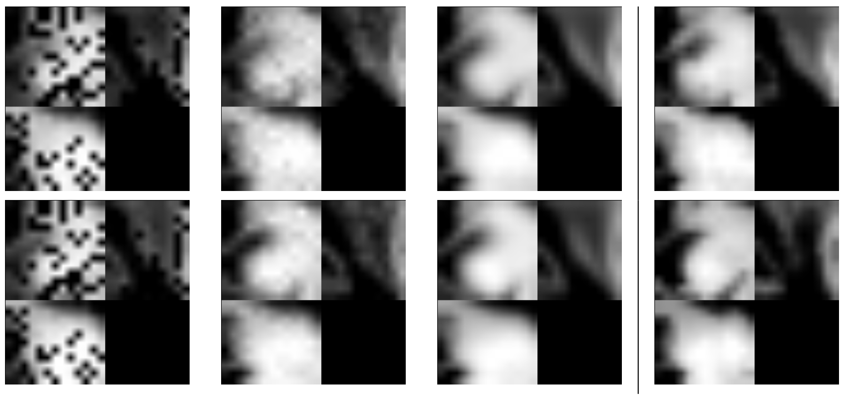
For the more promising ReLu configurations, we also take a look at the reconstructions after 100 iterations, as well as at the final reconstructions, see Figure 7.16. We can see that all final reconstructions look good, except for the one resulting from using a Wasserstein loss with a convolutional autoencoder. We also see that the dense autoencoder with Wasserstein loss already gives very good reconstructions after 100 epochs, while the Euclidean loss reconstruction still has a lot of artifacts. Again, we also tried classification, but without any noteworthy results.

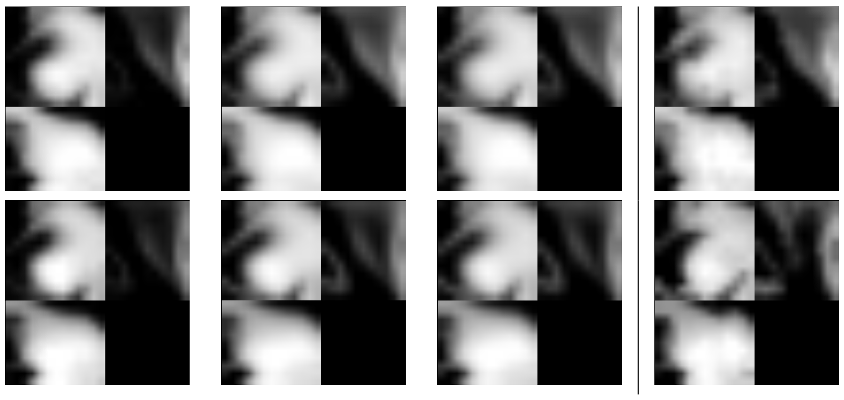
7.3.4 Summary

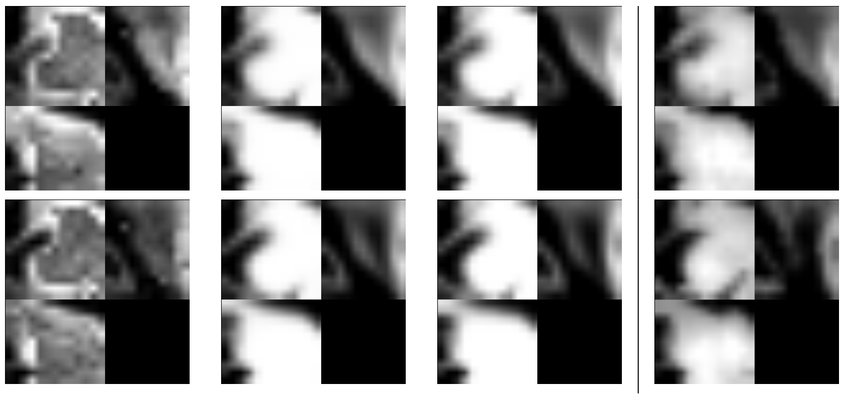
To summarize, we have shown that training an autoencoder with a Wasserstein loss is possible.

We observed that training with a Wasserstein loss is especially advantageous over training with a Euclidean loss when using dense autoencoders and hypothesized that this might be due to the spatial structure being incorporated into the Wasserstein loss. While not shown in this thesis, it also became obvious that a very small learning rate has to be used. However, this might be due to the relatively small training set. We further observed that using a Wasserstein loss led to better reconstructions after fewer epochs. Nevertheless, it cannot be said that training with a Wasserstein loss is “faster”, since it takes about 10 times as long as training with a Euclidean loss (which is still rather remarkable, considering the complexity of the Kantorovich problem).

We did not achieve any noteworthy classification results, but this could probably not be expected since we used rather small and simple autoencoders and classification networks.


(a) Euclidean with ReLu activations.


(b) Wasserstein with ReLu activations.


(c) Euclidean with sigmoid activations.

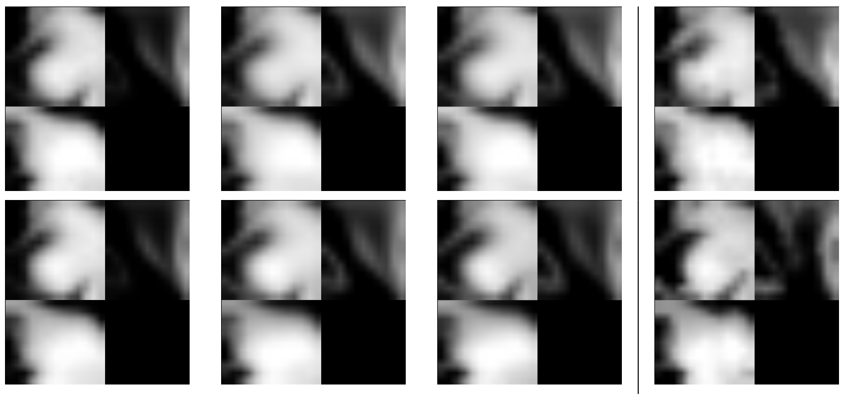
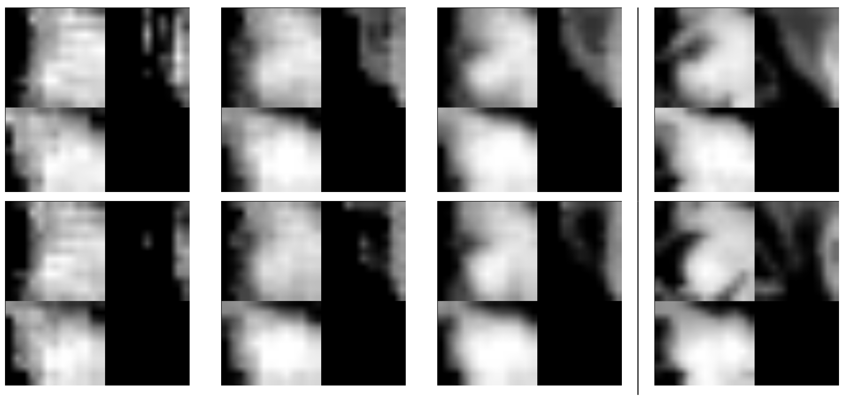
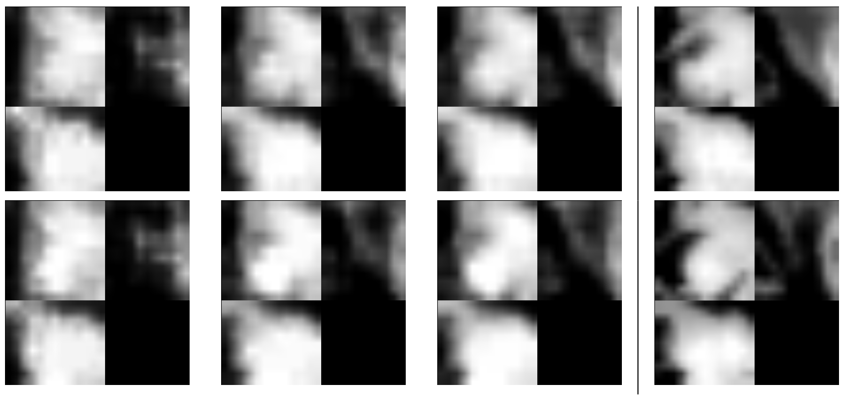
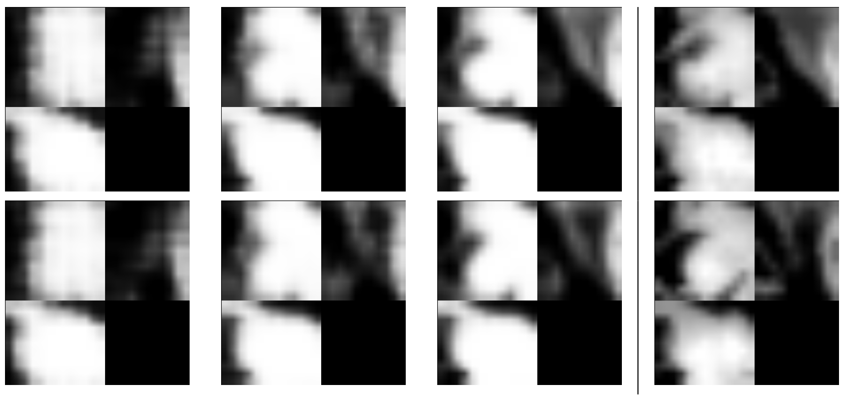

(d) Wasserstein with sigmoid activations.

Figure 7.11: Reconstructions of two different samples of the Amygdala with a dense autoencoder: after 100 epochs, after 300 epochs and after the final epoch. The original is shown in the right column.


(a) Euclidean with ReLu activations.


(b) Wasserstein with ReLu activations.


(c) Euclidean with sigmoid activations.

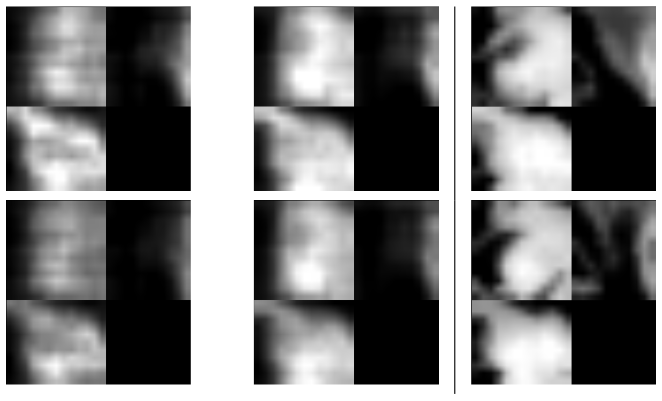

(d) Wasserstein with sigmoid activations.

Figure 7.12: Reconstructions of two different samples of the Amygdala with a convolutional autoencoder: after 100 epochs, after 300 epochs and after the final epoch. The original is shown in the right column.

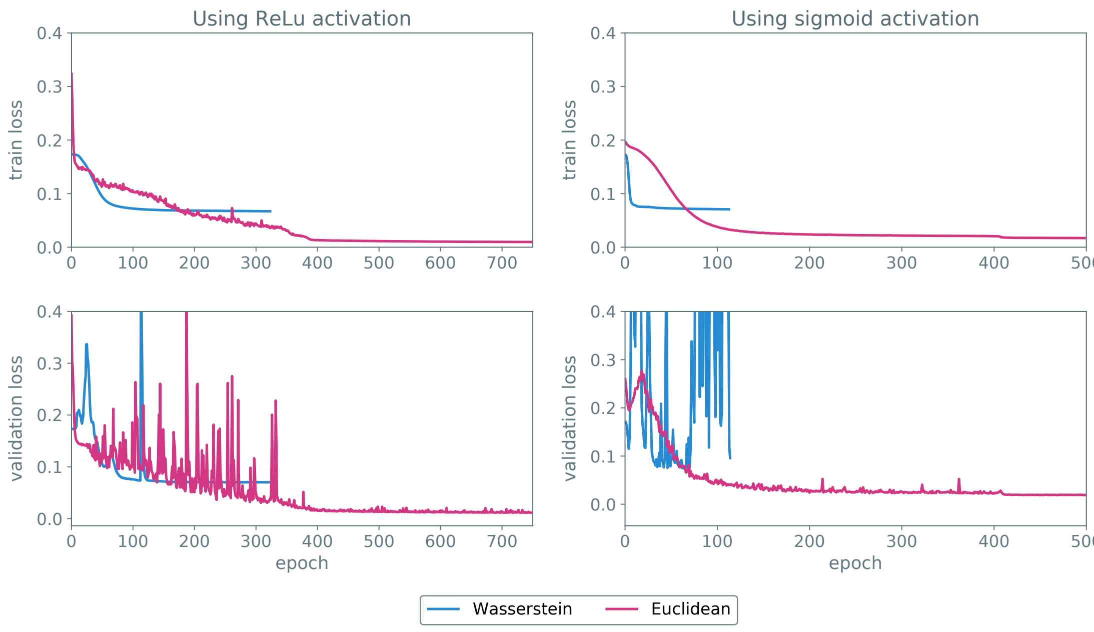


Figure 7.13: Train and validation loss for different configurations for the dense autoencoder trained on the region Hippocampus.

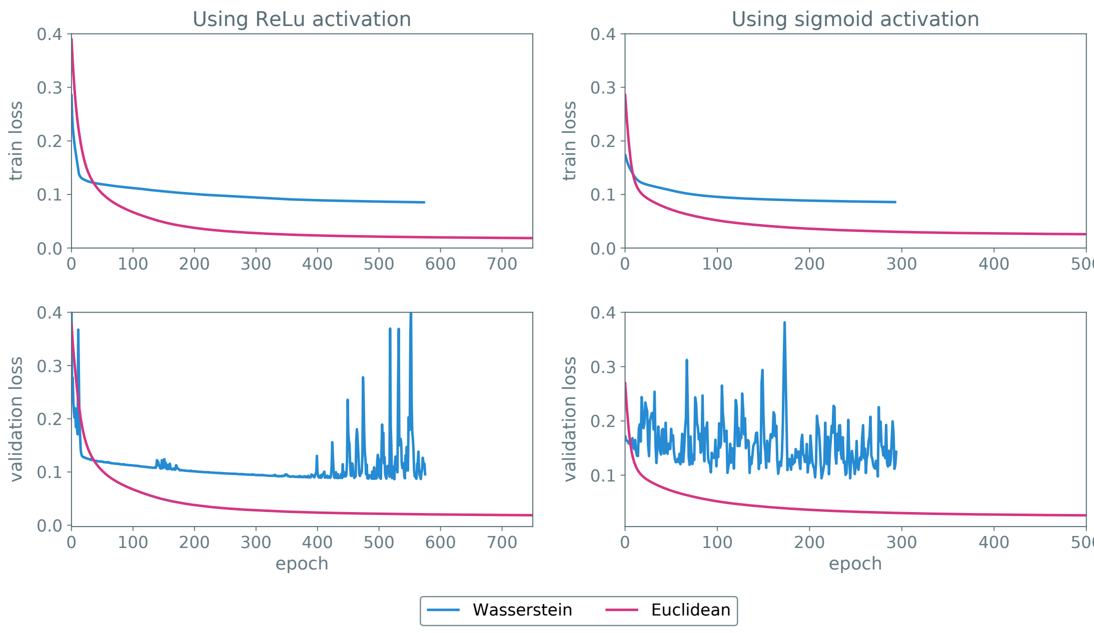


Figure 7.14: Train and validation loss for different configurations for the convolutional auto-encoder trained on the region Hippocampus.

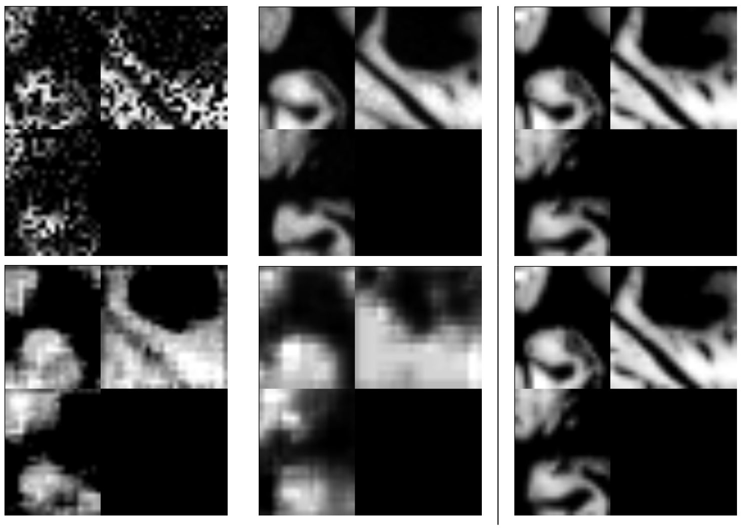


Figure 7.15: Reconstructions of Hippocampus after 100 epochs. Top: Dense autoencoder with Euclidean and Wasserstein loss, and original. Bottom: Convolutional autoencoder with Euclidean and Wasserstein loss, and original.

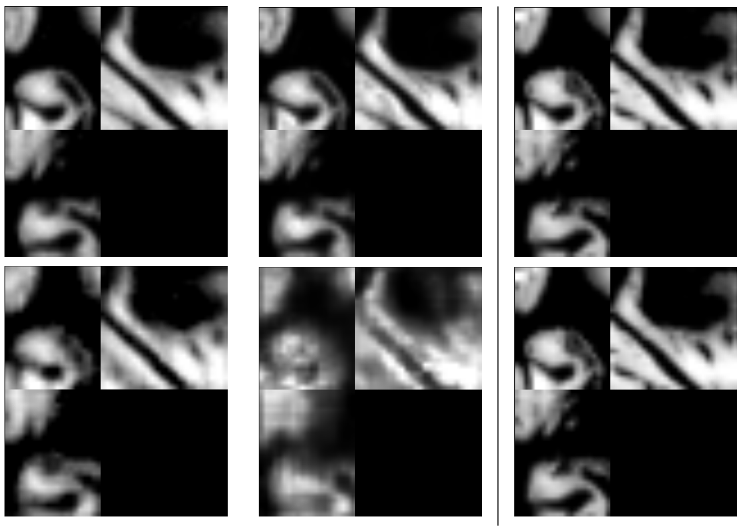


Figure 7.16: Reconstructions of Hippocampus after final epoch. Top: Dense autoencoder with Euclidean and Wasserstein loss, and original. Bottom: Convolutional autoencoder with Euclidean and Wasserstein loss, and original.

8 Conclusion and Outlook

In this thesis we have examined the usability of the Wasserstein distance as a metric between structural magnetic resonance images of the brain with the goal of using it for machine learning based post-processing. More specifically, we aimed to use the Wasserstein distance as a loss function in a specific kind of neural network called autoencoder to perform dimensionality reduction. An autoencoder, consisting of a decoder and an encoder, is trained to replicate its input. Replication, in this context, has to be understood in terms of a similarity measure: we aimed to use the Wasserstein distance as such. This goal was achieved.

The main challenge in achieving this goal was to find a way to apply the Wasserstein distance in a machine learning setting. Traditional algorithms are limited to specific settings which do not include three-dimensional data, have far too high a runtime to be considered in a machine learning setting, where a large amount of data has to be processed, or do not supply a way to calculate gradients for backpropagation. We thus proposed to use the entropy-regularized Wasserstein distance, which shares many of the properties of the exact Wasserstein distance, while its computation can be performed much more efficiently. We introduced this regularization in Section 3.3 and derived the algorithm with mathematical rigor in the setting of probability measures on a compact metric space. Because the computation algorithm is very compatible with GPUs, it suits machine learning well. While not being differentiable in the strict sense, the regularized distance possesses a canonical gradient, which is also rigorously proven in Section 3.3. This gradient occurs as a side product of the algorithm and can be employed for backpropagation. With these prerequisites it was possible to implement the algorithm in the machine learning framework `TensorFlow` and to further provide an interface allowing integration with the high-level framework `Keras`.

To evaluate the usability of the Wasserstein distance, we aimed to employ it for classification of brain images with respect to gender and diagnosis. Despite the significant speedup resulting from the regularization, processing whole brain images is still computationally difficult. We thus extracted the brain regions Amygdala and Hippocampus from each brain image. These regions had previously been shown to be connected to depression. We then classified these regions using support vector machines with radial basis function kernels. These radial basis function kernels depend on a distance function, for which both the Wasserstein distance and the Euclidean distance can be substituted. However, a major drawback of the Wasserstein distance is its restriction to measures with fixed volume, which requires a normalization of gray matter volume. We proposed to deal with this drawback by introducing an extension of the Wasserstein distance on the whole space of measures in Section 3.2. We provided a detailed proof of the equivalence of this extension to a formulation in terms of a standard optimal transport problem to which entropic regularization can be applied, allowing us to use the same algorithm as before.

Despite successfully having dealt with this drawback, the classification results were rather underwhelming. The Wasserstein distance with previous normalization outperformed the extension allowing for a volume change, and performed only marginally better than the Euclidean distance. Since the results have not been validated with nested cross-validation, they tend to be optimistic and are thus not significant. We have further employed the same kernel trick with principal component analysis with similarly underwhelming results.

It remains unclear why the inclusion of the ground metric did not lead to much improvement. We have demonstrated the superiority of the Wasserstein example on a toy data set in Chapter 5. Yet even here the superiority was not as outright as expected: In theory, classification without any information on the ground metric should have been impossible due to disjoint supports of the considered measures. However, as a consequence of discretization, some of the measures had a minimal overlap. This was already enough to achieve a classification accuracy of 80% using the Euclidean distance. This suggests that the additional information on the ground metric is only instrumental if the considered measures have little to no common support. As this is not the case for MR images, this might be an explanation as to why the Euclidean distance worked similarly well. However, since the Wasserstein distance also did not perform worse than the Euclidean distance, we still tried to use it in autoencoders.

We tried two different architectures of autoencoders and used a data set of 1800 samples to train it. While this worked well, the training time was very long due to the higher computation time of Wasserstein distances compared to the Euclidean distance. This was a problem especially for the larger region Hippocampus, where training for 150 epochs already took a whole day, yet not having converged. This will only be amplified by an eventual growth of the data set, which poses a big problem, since one advantage of an autoencoder is the possibility to include large amounts of unlabeled data. It also makes hyperparameter optimization even more tedious than it already is without a complex loss function. Since the Wasserstein distance could not be shown to be better suited as a distance than the Euclidean distance, we conclude that training autoencoders with a Wasserstein loss is not worth the computational effort.

A few options that could not be treated in this thesis, however, remain to be tried out, and might be addressed in future work. They are outlined as follows:

- The Wasserstein distance could be used to pretrain an autoencoder for a few epochs, continuing fine-tuning with the Euclidean distance. This could also be combined with different kinds of autoencoders such as denoising autoencoders or contractive autoencoders, which were shortly presented in Section 1.2.
- While we focused on distance values resulting from a converged algorithm, another option could be to fix a much smaller number of iterations to make the computation quicker and use this approximate value instead. The proposed method to obtain a gradient is then no longer valid, since the variable proposed as gradient only corresponds to a gradient upon convergence. Instead, one could use autodifferentiation to calculate the gradient.
- We further tried to approximate the true Wasserstein distance as well as possible by choosing a small regularization parameter, which, however, leads to a slower convergence behavior. We have seen in Subsection 7.2.1 that the convergence is very fast for larger parameters. While the classification results with larger parameters were worse, it might still be possible that an autoencoder is pushed in the right direction, even with a larger regularization parameter. It could then be fine-tuned.

- Another possible disadvantage of the Wasserstein distance is that the images are normalized before feeding them into the autoencoder, possibly leading to loss of information. Alternative possibilities to deal with this problem could be examined, for example to feed the total mass as an extra input variable either into the autoencoder or into the classifier. Furthermore, the Wasserstein distance really only needs input and output to have the same mass. So instead of normalizing all inputs and outputs to have mass 1, the inputs could be left untouched and the outputs could be normalized to have the same mass as the corresponding input. This would require a few changes in the implementation.
- It has been proposed in [64] to learn the ground metric on which the Wasserstein distance depends. While a natural ground metric arising from the three-dimensional structure of the voxel grid was already available, it turned out that not the Euclidean distance, but the distance to a power of 0.1 gave the best results. It might thus be possible that yet other ground metrics could be better suited to the structure of a brain.
- In this thesis, we considered the representation of MR images as images on a voxel grid. It is also possible to obtain surface data on a triangulation grid, between which the Wasserstein distance could also be defined. Similar experiments as presented in this thesis could then be constructed with this representation.
- The Wasserstein distance could further be used on functional MRI data, which has an additional time axis. The ground metric would then have to be extended along this time axis.
- The entropy-regularized variant of the Wasserstein distance further makes it possible to interpolate between data efficiently. This could be interesting for follow-up studies or for functional MRI data.

Bibliography

[1] Francois Chollet. *Deep Learning with Python*. 1st. Greenwich, CT, USA: Manning Publications Co., 2017.

[2] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[3] *Documentation of Convolution in Tensorflow*. URL: `https://www.tensorflow.org/api_docs/python/tf/nn/convolution`.

[4] Vincent Dumoulin and Francesco Visin. "A guide to convolution arithmetic for deep learning". In: *ArXiv e-prints* (Mar. 2016). eprint: `1603.07285`.

[5] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. `http://www.deeplearningbook.org`. MIT Press, 2016.

[6] Dumitru Erhan et al. "Why Does Unsupervised Pre-training Help Deep Learning?" In: *J. Mach. Learn. Res.* 11 (Mar. 2010), pp. 625–660.

[7] E. Hosseini-Asl, R. Keynton, and A. El-Baz. "Alzheimer's disease diagnostics by adaptation of 3D convolutional network". In: *2016 IEEE International Conference on Image Processing (ICIP)*. Sept. 2016, pp. 126–130.

[8] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. "A Training Algorithm for Optimal Margin Classifiers". In: *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*. COLT '92. Pittsburgh, Pennsylvania, USA: ACM, 1992, pp. 144–152.

[9] Corinna Cortes and Vladimir Vapnik. "Support-Vector Networks". In: *Mach. Learn.* 20.3 (Sept. 1995), pp. 273–297.

[10] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. Cambridge, MA, USA: MIT Press, 2001.

[11] Bernard Haasdonk and Claus Bahlmann. "Learning with Distance Substitution Kernels". In: *Pattern Recognition*. Ed. by Carl Edward Rasmussen et al. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 220–227.

[12] Michel Neuhaus and Horst Bunke. "Edit Distance-based Kernel Functions for Structural Pattern Classification". In: *Pattern Recogn.* 39.10 (Oct. 2006), pp. 1852–1863.

[13] O. Chapelle, P. Haffner, and V. N. Vapnik. "Support Vector Machines for Histogram-based Image Classification". In: *Trans. Neur. Netw.* 10.5 (Sept. 1999), pp. 1055–1064.

[14] Cheng Soon Ong et al. "Learning with Non-positive Kernels". In: *Proceedings of the Twenty-first International Conference on Machine Learning*. ICML '04. Banff, Alberta, Canada: ACM, 2004, pp. 81–.

[15] B. Haasdonk. "Feature space interpretation of SVMs with indefinite kernels". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 27.4 (Apr. 2005), pp. 482–492.

[16] K. Pearson. "On Lines and Planes of Closest Fit to Systems of Points in Space". In: *Philosophical Magazine* 2 (6 1901), pp. 559–572.

[17] Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. "Nonlinear Component Analysis as a Kernel Eigenvalue Problem". In: *Neural Computation* 10.5 (1998), pp. 1299–1319.

[18] Sebastian Raschka. *Kernel tricks and nonlinear dimensionality reduction via RBF kernel PCA*. 2014. URL: `https://sebastianraschka.com/Articles/2014_kernel_pca.html` (visited on 09/11/2018).

[19] Sebastian Mika et al. "Kernel PCA and De-Noising in Feature Spaces". In: *ADVANCES IN NEURAL INFORMATION PROCESSING SYSTEMS 11*. MIT Press, 1999, pp. 536–542.

[20] GH. Bakir, J. Weston, and B. Schölkopf. "Learning to Find Pre-Images". In: *Advances in Neural Information Processing Systems 16*. Cambridge, MA, USA: MIT Press, June 2004, pp. 449–456.

[21] Xiaolin Huang et al. "Indefinite kernels in least squares support vector machines and principal component analysis". In: *Applied and Computational Harmonic Analysis* 43.1 (2017), pp. 162–172.

[22] Walter Rudin. *Real and Complex Analysis, 3rd Ed.* New York, NY, USA: McGraw-Hill, Inc., 1987.

[23] Krishna B. Athreya and Soumen N. Lahiri. *Measure Theory and Probability Theory (Springer Texts in Statistics)*. Berlin, Heidelberg: Springer-Verlag, 2006.

[24] M.E. Taylor. *Measure Theory and Integration*. Vol. 76. Graduate studies in mathematics. American Mathematical Soc., 2006.

[25] Hedy Attouch, Giuseppe Buttazzo, and Grard Michaille. *Variational Analysis in Sobolev and BV Spaces: Applications to PDEs and Optimization, Second Edition*. Philadelphia, PA, USA: Society for Industrial and Applied Mathematics, 2014.

[26] R. T. Rockafellar. "Integrals which are convex functionals." In: *Pacific J. Math.* 24.3 (1968), pp. 525–539.

[27] J.M. Borwein and Q.J. Zhu. *Techniques of Variational Analysis*. CMS Books in Mathematics. Springer, 2005.

[28] G. Peyré and M. Cuturi. "Computational Optimal Transport". In: *ArXiv e-prints* (Mar. 2018). arXiv: `1803.00567 [stat.ML]`.

[29] V.I. Bogachev. *Measure Theory*. Measure Theory Vol. 1. Springer Berlin Heidelberg, 2007.

[30] F. Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Progress in Nonlinear Differential Equations and Their Applications. Springer International Publishing, 2015.

[31] C. Villani. *Optimal Transport: Old and New*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2008.

[32] D.C Dowson and B.V Landau. "The Fréchet distance between multivariate normal distributions". In: *Journal of Multivariate Analysis* 12.3 (1982), pp. 450–455.

[33] "Scaling algorithms for unbalanced optimal transport problems". In: *Math. Comp.* 87 (2018), pp. 2563–2609.

[34] Charlie Frogner et al. "Learning with a Wasserstein Loss". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'15. Montreal, Canada: MIT Press, 2015, pp. 2053–2061.

[35] L.V. Kantorovich and G.S. Rubinstein. "On a space of completely additive functions (Russian)". In: *Vestnik Leningrad Univ.* 7 (13 1958), pp. 52–59.

[36] Leonid G. Hanin. "An extension of the Kantorovich norm". In: *Contemp. Math.* (226 1999): *Monge Ampère Equation: Applications to Geometry and Optimization (Deerfield Beach, FL, 1997)*, pp. 113–130.

[37] Kevin Guittet. *Extended Kantorovich norms : a tool for optimization*. Research Report RR-4402. INRIA, 2002.

[38] A. Gramfort, G. Peyré, and M. Cuturi. "Fast Optimal Transport Averaging of Neuroimaging Data". In: *Information Processing in Medical Imaging*. Ed. by Sebastien Ourselin et al. Cham: Springer International Publishing, 2015, pp. 261–272.

[39] Ofir Pele and Michael Werman. "A Linear Time Histogram Metric for Improved SIFT Matching". In: *Proceedings of the 10th European Conference on Computer Vision: Part III*. ECCV '08. Marseille, France: Springer-Verlag, 2008, pp. 495–508.

[40] Marco Cuturi. "Sinkhorn Distances: Lightspeed Computation of Optimal Transport". In: *Advances in Neural Information Processing Systems 26*. Ed. by C. J. C. Burges et al. Curran Associates, Inc., 2013, pp. 2292–2300.

[41] Justin Solomon et al. "Convolutional Wasserstein Distances: Efficient Optimal Transportation on Geometric Domains". In: *ACM Trans. Graph.* 34.4 (July 2015), 66:1–66:11.

[42] J.M. Borwein, A.S. Lewis, and R.D. Nussbaum. "Entropy Minimization, DAD Problems, and Doubly Stochastic Kernels". In: *Journal of Functional Analysis* 123.2 (1994), pp. 264–307.

[43] Bernhard Schmitzer. "Stabilized Sparse Scaling Algorithms for Entropy Regularized Transport Problems". In: *ArXiv pre-print* (Oct. 2016). arXiv: `1610.06519 [math.OC]`.

[44] S. Kullback and R. A. Leibler. "On Information and Sufficiency". In: *Ann. Math. Statist.* 22.1 (Mar. 1951), pp. 79–86.

[45] Christian Léonard. "From the Schrödinger problem to the Monge–Kantorovich problem". In: *Journal of Functional Analysis* 262.4 (2012), pp. 1879–1920.

[46] "Continuous entropic regularization and Sinkhorn-type algorithms". work in progress.

[47] L. Rüschendorf and W. Thomsen. "Note on the Schrödinger equation and I-projections". In: *Statistics & Probability Letters* 17.5 (1993), pp. 369–375.

[48] Ludger Rüschendorf. "Convergence of the Iterative Proportional Fitting Procedure". In: *Ann. Statist.* 23.4 (Aug. 1995), pp. 1160–1174.

[49] Antoine Rolet, Marco Cuturi, and Gabriel Peyré. "Fast Dictionary Learning with a Smoothed Wasserstein Loss". In: *Proceedings of the 19th International Conference on Artificial Intelligence and Statistics*. Ed. by Arthur Gretton and Christian C. Robert. Vol. 51. Proceedings of Machine Learning Research. Cadiz, Spain: PMLR, Sept. 2016, pp. 630–638.

[50] Michael Bacharach. "Estimating Nonnegative Matrices from Marginal Data". In: *International Economic Review* 6.3 (1965), pp. 294–310.

[51] Yossi Rubner, Carlo Tomasi, and Leonidas J. Guibas. "The Earth Mover's Distance As a Metric for Image Retrieval". In: *Int. J. Comput. Vision* 40.2 (Nov. 2000), pp. 99–121.

[52] J. Zhang et al. "Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study". In: *International Journal of Computer Vision* 73.2 (June 2007), pp. 213–238.

[53] Sadeep Jayasumana et al. “Kernel Methods on the Riemannian Manifold of Symmetric Positive Definite Matrices”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2013.

[54] Martin Arjovsky, Soumith Chintala, and Léon Bottou. “Wasserstein Generative Adversarial Networks”. In: *Proceedings of the 34th International Conference on Machine Learning*. Ed. by Doina Precup and Yee Whye Teh. Vol. 70. Proceedings of Machine Learning Research. International Convention Centre, Sydney, Australia: PMLR, June 2017, pp. 214–223.

[55] Rémi Flamary. *Optimal transport for machine learning*. 2017. URL: `https://remi.flamary.com/pres/OTML_ISIS_2017.pdf` (visited on 09/11/2018).

[56] *Documentation of sklearn.svm.SVC*. URL: `http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`.

[57] *Documentation of sklearn.decomposition.KernelPCA*. URL: `http://scikit-learn.org/stable/modules/generated/sklearn.decomposition.KernelPCA.html`.

[58] Chih-Chung Chang and Chih-Jen Lin. “LIBSVM: A Library for Support Vector Machines”. In: *ACM Trans. Intell. Syst. Technol.* 2.3 (May 2011), 27:1–27:27.

[59] Bernhard Schölkopf. “The Kernel Trick for Distances”. In: *Proceedings of the 13th International Conference on Neural Information Processing Systems*. NIPS’00. Denver, CO: MIT Press, 2000, pp. 283–289.

[60] Francois Chollet. *Building Autoencoders in Keras*. URL: `https://blog.keras.io/building-autoencoders-in-keras.html` (visited on 09/17/2018).

[61] H. Wersching and K. Berger. “Neue Kohorten”. In: *Bundesgesundheitsblatt - Gesundheitsforschung - Gesundheitsschutz* 55.6 (June 2012), pp. 822–823.

[62] Henning Teismann et al. “Establishing the bidirectional relationship between depression and subclinical arteriosclerosis – rationale, design, and characteristics of the BiDirect Study”. In: *BMC Psychiatry* 14.1 (June 2014), p. 174.

[63] Ronny Redlich et al. “Brain Morphometric Biomarkers Distinguishing Unipolar and Bipolar Depression: A Voxel-Based Morphometry–Pattern Classification Approach”. In: *JAMA psychiatry* 71.11 (2014), pp. 1222–1230. eprint: `/data/journals/psych/931006/yoi140054.pdf`.

[64] M. Cuturi and D. Avis. “Ground Metric Learning”. In: *ArXiv e-prints* (Oct. 2011). arXiv: `1110.2306 [stat.ML]`.