LEUPHANA
UNIVERSITÄT LÜNEBURG

Computing Efficient Data Summaries

Von der Fakultät Wirtschaftswissenschaften
der Leuphana Universität Lüneburg
zur Erlangung des Grades

Doktor der Naturwissenschaften
- Dr. rer. nat. -

genehmigte Dissertation von
Sebastian Mair

geboren am 7. September 1988
in Dillingen an der Donau

Eingereicht am: 20.05.2021

Mündliche Verteidigung (Disputation) am: 24.09.2021

Erstbetreuer und Erstgutachter: Prof. Dr. Ulf Brefeld

Zweitgutachter: Prof. Dr. Stephan Günnemann

Drittgutachter: Prof. Dr. Søren Hauberg

Die einzelnen Beiträge des kumulativen Dissertationsvorhabens sind oder werden wie folgt veröffentlicht:

Mair, S., Boubekki, A., and Brefeld, U. (2017). Frame-based data factorizations. In *International Conference on Machine Learning*, pages 2305–2313

Mair, S., Rudolph, Y., Closius, V., and Brefeld, U. (2018). Frame-based optimal design. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 447–463. Springer, Cham

Mair, S. and Brefeld, U. (2019). Coresets for archetypal analysis. In *Advances in Neural Information Processing Systems*, pages 7247–7255

Brefeld, U., Lasek, J., and Mair, S. (2019). Probabilistic movement models and zones of control. *Machine Learning*, 108(1):127–147

Fadel, S. G., Mair, S., da Silva Torres, R., and Brefeld, U. (2021a). Contextual movement models based on normalizing flows. *AStA Advances in Statistical Analysis*, pages 1–22

Fadel, S. G., Mair, S., da Silva Torres, R., and Brefeld, U. (2021b). Principled interpolation in normalizing flows. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 116–131. Springer

Veröffentlichungsjahr: 2021

Veröffentlicht im Onlineangebot der Universitätsbibliothek unter der URL: `http://www.leuphana.de/ub`

Abstract

Extracting meaningful representations of data is a fundamental problem in machine learning. Those representations can be viewed from two different perspectives. First, there is the representation of data in terms of the number of data points. Representative subsets that compactly summarize the data without superfluous redundancies help to reduce the data size. Those subsets allow for scaling existing learning algorithms up without approximating their solution. Second, there is the representation of every individual data point in terms of its dimensions. Often, not all dimensions carry meaningful information for the learning task, or the information is implicitly embedded in a low-dimensional subspace. A change of representation can also simplify important learning tasks such as density estimation and data generation.

This thesis deals with the aforementioned views on data representation and contributes to them. We first focus on computing representative subsets for a matrix factorization technique called archetypal analysis and the setting of optimal experimental design. For these problems, we motivate and investigate the usability of the data boundary as a representative subset. We also present novel methods to efficiently compute the data boundary, even in kernel-induced feature spaces. Based on the coreset principle, we derive another representative subset for archetypal analysis, which provides additional theoretical guarantees on the approximation error. Empirical results confirm that all compact representations of data derived in this thesis perform significantly better than uniform subsets of data.

In the second part of the thesis, we are concerned with efficient data representations for density estimation. We analyze spatio-temporal problems, which arise, for example, in sports analytics, and demonstrate how to learn (contextual) probabilistic movement models of objects using trajectory data. Furthermore, we highlight issues of interpolating data in normalizing flows, a technique that changes the representation of data to follow a specific distribution. We show how to solve this issue and obtain more natural transitions on the example of image data.

Zusammenfassung

Das Extrahieren sinnvoller Repräsentationen von Daten ist ein grundlegendes Problem im maschinellen Lernen. Diese Repräsentationen können aus zwei verschiedenen Perspektiven betrachtet werden. Zum einen gibt es die Repräsentation von Daten in Bezug auf die Anzahl der Datenpunkte. Repräsentative Teilmengen helfen große Datenbestände kompakt zusammenzufassen. Dazu werden beispielsweise überflüssige Redundanzen weggelassen. Diese Teilmengen erlauben es, bestehende Lernalgorithmen hochzuskalieren, ohne deren Lösung zu approximieren. Zum anderen gibt es die Repräsentation jedes einzelnen Datenpunktes in Bezug auf seine Dimensionen. Oft tragen nicht alle Dimensionen sinnvolle Informationen, oder Informationen sind implizit in einem niedrigdimensionalen Unterraum eingebettet. Ein Wechsel der Repräsentation kann auch wichtige Verfahren wie die Dichteschätzung und die Datengenerierung vereinfachen.

Diese Arbeit beschäftigt sich mit den oben genannten Perspektiven zur Datenrepräsentation und leistet einen Beitrag dazu. Wir konzentrieren uns zunächst auf die Berechnung repräsentativer Teilmengen für die Archetypenanalyse und auf das Setting der optimalen Versuchsplanung. Für diese Probleme motivieren und untersuchen wir die Brauchbarkeit der Punkte am Rand der Daten repräsentative Teilmenge. Außerdem stellen wir neuartige Methoden zur effizienten Berechnung dieser Randpunkte vor. Basierend auf dem Coreset-Prinzip leiten wir eine weitere repräsentative Teilmenge für die Archetypenanalyse her, welche zusätzliche theoretische Garantien bietet. Empirische Ergebnisse bestätigen, dass alle kompakten Repräsentationen von Daten, die in dieser Arbeit vorgestellt werden, deutlich besser abschneiden als zufällige Untermengen.

Im zweiten Teil der Arbeit beschäftigen wir uns mit effizienten Datenrepräsentationen für die Dichteschätzung. Wir analysieren raum-zeitliche Probleme, die z.B. in der Sportanalytik auftreten, und zeigen, wie man (kontextuelle) probabilistische Bewegungsmodelle von Objekten anhand von Trajektoriendaten lernt. Darüber hinaus untersuchen wir Probleme der Interpolation von Daten bei Normalizing Flows, einem Verfahren, das die Darstellung von Daten so verändert, dass sie einer vorgegebenen Wahrscheinlichkeitsverteilung folgen. Wir zeigen am Beispiel von Bilddaten, wie man dieses Problem löst und natürlichere Interpolationsübergänge erhält.

Contents

List of Figures

List of Tables

List of Algorithms

List of Acronyms

AA archetypal analysis 5–8, 15, 17–20, 27, 28, 35–41, 57, 60–62, 64, 121, 122

B-NAF block neural autoregressive flow 92

BPD bits per dimension 112, 114–116, 123, 151

CN conditioning network 148

DPP determinantal point process 12, 54

ELBO evidence lower bound 92

ESP elementary symmetric polynomials 54

FID Fréchet inception distance 105, 108, 112–116, 120, 123, 151

GAN generative adversarial network 92, 118, 119

GMM Gaussian mixture model 20

GP Gaussian process 11, 13, 74

KDE kernel density estimate . 6, 79–82, 90, 91, 94, 96, 97, 99–103, 122, 123

KID kernel inception distance 112–116, 120, 123, 151

LASSO least absolute shrinkage and selection operator . 5, 41, 48, 49, 55

lerp linear interpolation 106, 108

LP linear programming 27, 33, 34, 39

MF matrix factorization 16, 17, 19, 32, 38

MMD maximum mean discrepancy 112

MSE mean squared error 50, 51

NAF neural autoregressive flow 92

nclerp norm-corrected linear interpolation 107, 108, 113

NF normalizing flow . . 6, 7, 9, 14, 15, 21, 22, 91–95, 98, 104, 105, 119, 120

NICE non-linear independent component estimation 92

NMF non-negative matrix factorization 17, 35–37

NN neural network 4, 23, 91, 148

NNLS non-negative least squares 5, 8, 30, 32, 36, 39, 45, 121

OED optimal experimental design . 5, 7, 8, 41, 42, 44, 49, 53–55, 121, 122

PCA principal component analysis 4, 12, 13

QP quadratic program 27, 30

RBF radial basis function 38, 46

RealNVP real-valued non-volume preserving 22, 92

RSS residual sum of squares 18–20

SELU self-scaling exponential linear unit 98

slerp spherical linear interpolation 111

SVM support vector machine 11, 12, 38

VAE variational autoencoder 4, 13, 14, 19, 92, 118, 119

vMF von Mises-Fisher 14, 110–116, 118–120, 123, 150–152

Chapter 1

Introduction

The field of machine learning provides a set of algorithms for automatically inferring patterns and relationships in data. An algorithm learns such patterns and relationships from experience, which is commonly a set of data points that forms the input data set. Typically, each data point is characterized by a number of features (or attributes) in a vectorized form. This data set is often represented by a so-called *design matrix* consisting of the stacked data points. Thus, its size is the number of data points times the number of features.

The most common learning setting is *supervised learning*, where each data point is associated with a label, denoting the output (or target). The goal is to infer a mapping from input to output data by minimizing a loss function, typically within the framework of empirical risk minimization (Vapnik, 1991). Typical examples are, *e.g.*, regression and classification. While having access to labels renders supervised learning easier to evaluate, unsupervised learning is especially appealing as it does not rely on (possibly) expensive labeling. *Unsupervised learning* is concerned with discovering structures and/or patterns in data. Canonical examples include anomaly detection, clustering, density estimation, dimensionality reduction, and latent variable models.

The representation of data, *i.e.*, the data set or design matrix, is often sub-optimal and crucial for the quality of a learning task (Bengio et al., 2013). For example, the data set might be too large, include a large amount of redundant information, or the attributes lack predictive features. Thus, a change of representation is frequently necessary. This change of representation can be done as a pre-processing step or as a part of a learning algorithm. In general, there are two views on the data on which a change can happen.

1.1 Two Views on Representation Learning

Representation learning is about finding another representation of data, usually represented as a design matrix, which is better suited for a specific purpose. Changing the representation can be seen in two different ways. The first one is about the number of data points and thus affects the sample size of the data set. The second one is concerned with a change of the representation in terms of the features.

The Sample Size View

In some domains, data is available in large quantities and might carry a lot of redundant information. In other words, the number of data points may be too large to run specific algorithms. Hence, efficient inference becomes difficult. This problem setting is also referred to as *large-scale learning*. To scale machine learning models to large data sets, the options are usually to either approximate the model at hand or the training data. Apart from that, there could also be memory and computational budgets that force us to use an approximation. Henceforth, we focus on approximating the data set.

A small subset of data that approximates its key features well is called a *representative subset* or *data summary*, as it aims to summarize the data. A straightforward idea is to use a uniform subset of data, where points are sampled independently and uniformly at random. This rather ad-hoc approach often yields an unbiased estimator of the objective function that is being optimized.

More sophisticated ways to sample points need to exist that are (i) more efficient and (ii) lead to a better representations. Here, (i) refers to even smaller subsets that yield competitive results as compared to uniform subsets. As a result, those subsets are more sample efficient.

The goal of representative subsets is as follows. First, a learning algorithm should perform similarly on the subset as on the original data. Thus, the subset is usually tailored to a specific algorithm. In addition, those subsets often also provide an individual weight per data point. That way, many similar data points might be represented by only one data point that has a higher weight. Second, it is advantageous if the construction of a representative subset also comes with a theoretical guarantee on the approximation error. Third, a standard requirement of subset selection methods is that they can be efficiently computed. Often, those selection approaches are unsupervised, meaning they only consider input data (the design matrix) and no labels.

For example, in k-means clustering (Lloyd, 1982) state-of-the-art representative subsets (Bachem et al., 2018a) employ importance sampling strategies and can be constructed with just two passes over the data.

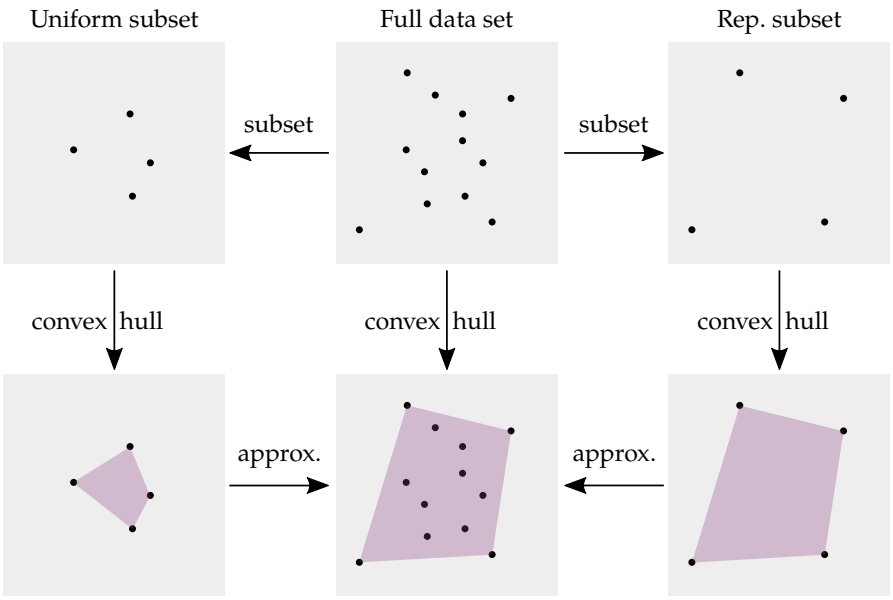

Figure 1.1: An example of learning on subsets. The full data set (top center) is represented by uniform subset (top left) and a representative subset (top right). An algorithm, here the computation of a convex hull, executed on the representative subset (bottom right) performs better than a uniform subset (bottom left) when compared to all data (bottom center).

Figure 1.1 depicts an example for the computation of a convex hull. A small uniform subset covers a smaller area than the desired convex hull. This is due to the fact, that the most important points for this problem are located on the boundary of data. In comparison, the representative subset on the right side of the figure includes more relevant points and thus provides a better summary of the full data. As a result, it yields a much better approximation while being of the same size as the uniform subset.

The Dimension View

Traditionally, representation learning is understood as a change of representation in terms of the dimensions of data. Often, not all dimensions carry meaningful information for the learning task, the essential information is implicitly embedded in a low-dimensional subspace, or the given features are not all predictive. In the latter case, a change of representation is often done as a part of a learning algorithm.

For example, a feature mapping is frequently used to transform the input data into another space that is beneficial for the learning task. An often-used approach is to use feature mappings that are implicitly defined

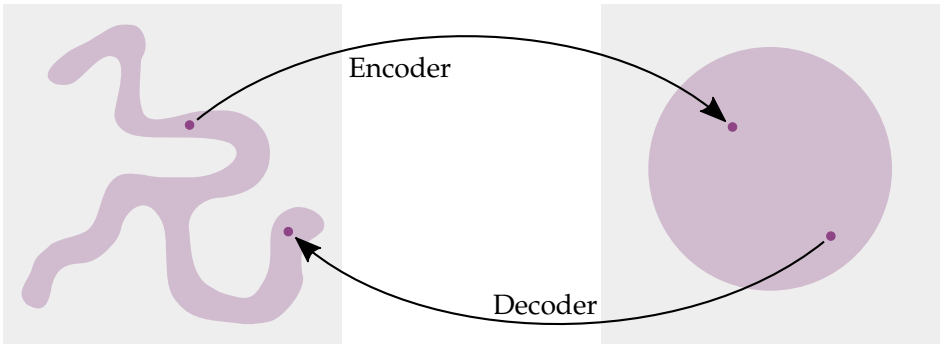

Figure 1.2: An example of a data transformation. Left: an input space that has a complex distribution. Right: a representation of the same data having a simpler distribution. Transformations allow to move points from one space to another.

by kernels (Schölkopf and Smola, 2002). This allows us to leverage non-linear relations in data. An example would be kernel principal component analysis (Schölkopf et al., 1998), which serves as a non-linear extension to principal component analysis (PCA) (Pearson, 1901). Neural networks (NNs) (LeCun et al., 2015; Goodfellow et al., 2016) also serve as an example of feature transformations. Consider, *e.g.*, a simple feed-forward neural network for a classification task. All layers except the last can be seen as a non-linear feature extraction, whereas the last layer is performing, *e.g.*, a logistic regression on the extracted features.

Autoencoders also pursue a non-linear feature transformation (Rumelhart et al., 1986; LeCun, 1987). This unsupervised method consists of two parts. The first one is called an encoder and embeds the input data into a so-called *latent space* or *latent representation*. The second part, called decoder, then reconstructs the input data from the latent representation. Hence, the decoder is (approximately) inverting the transformation of the encoder. The latent representation is often of lower dimensionality and hence forces the autoencoder to extract useful properties of the data.

To facilitate operations such as data generation in the framework of generative modeling, it is advantageous to enforce a specific structure on the latent space. A variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) allows us to specify a desired prior distribution on the latent representation. This allows us to easily sample from the prior and generate new synthetic data points by decoding them back into data space. Thus, the encoder can be seen as a transformation that changes the distribution of data. An example is depicted in Figure 1.2. A complex data distribution is transformed into an easier distribution. This is not only advantageous for generative modeling but also for density estimation.

1.2 Contributions

This thesis contributes to the area of unsupervised representation learning and has a focus on computing efficient data summaries. The following chapters contain ideas and contributions which belong to either of the previously introduced views on representation learning. In this thesis, the central research questions are:

(i) How to efficiently obtain representative subsets for learning tasks that allow learning the same from fewer data in a more efficient way?

(ii) How to obtain data representations that facilitate the usage of specific operations such as density estimation and interpolation?

The specific contributions are as follows.

The Frame as a Representative Subset

We introduce the concept of the frame, which consists of the data points on the boundary of a data set. In other words, the frame is the intersection of data with the boundary of the convex hull of data. The frame is then used as a representative subset for specific learning tasks. The hypothesis is that the data boundary should already contain most of the data set information for linear models. We show this on the example of archetypal analysis (AA) (Cutler and Breiman, 1994), an interpretable matrix factorization with additional convexity constraints. Restricting archetypal analysis to the frame yields very competitive results compared to computing the factorization on the full data set. Besides, using the frame as a representative subset allows for a much faster computation of the factorization. We also derive a novel way to efficiently compute the frame of a data set and reveal a connection to a well-known optimization algorithm called non-negative least squares (NNLS) (Lawson and Hanson, 1995).

The idea of the frame as a representative subset can also be used in optimal experimental design (OED) (Fedorov, 1972), where the input data set describes a set of experiments. Assuming a linear dependence between input and output data, the goal is to select some experiments to execute and obtain their labels under a computational budget. Once again, we show that competitive results can be obtained much faster using the frame. For OED, we also consider non-linear designs via kernels, derive how to compute the frame in feature space, and show theoretically that the frame is equal to the input set under specific assumptions. This implies, the expressive power of some non-linear functions is so immense that every data point counts. Furthermore, we show that computing the frame can be seen as a transposed version of LASSO (Tibshirani, 1996) that selects data points instead of features.

Coresets for Archetypal Analysis

Unfortunately, no theoretical analysis on the induced approximation error of the frame for archetypal analysis can be done. As a remedy, we consider coresets, which offer an elegant way of compactly representing large data sets by weighted subsets. On such coresets, which can be constructed in linear time, methods perform provably competitive compared to the full data sets. Approximating the data set often means neglecting redundant information and allows us to run well-known and understood methods on large data sets to which those methods would not scale in their standard formulation. We conduct the missing theoretical analysis, and we are the first to derive coresets for archetypal analysis.

Probabilistic Movement Models

Changing the representation is also often beneficial in practical applications. We show this on the example of movement models of soccer players. Via a change of representation, from a global to a local coordinate system, we can express the distribution of a soccer player's next position using kernel density estimates (KDEs). Besides, this allows us to model location-invariance, meaning that the general ability to move should be independent of a player's current position. We encode contextual information such as current velocity and time horizon for the prediction via a bag of models. We further turn these probabilistic movement models into *zones of control*, describing which part of the pitch is controlled by which player.

Due to a scalability issue of KDEs, we extend this work by modeling the density via *normalizing flows*, a state-of-the-art framework for density learning using invertible neural networks. We also incorporate contextual information by considering a conditional distribution to maintain a single contextual model instead of a bag of models. Our findings show that our approach outperforms the state of the art while being orders of magnitude more efficient with respect to computation time and memory requirements.

Interpolation in Generative Modeling

A normalizing flow consists of a chain of parametrized bijective functions that transforms the data into a representation that follows a specific distribution, usually a Gaussian. As mentioned earlier, a change of representation, here to a latent space, facilitates certain operations. We consider the generative aspect of normalizing flows and question the ubiquitous use of the Gaussian distribution in latent space, especially when the interpolation of data is of interest. Specifically, we show that interpolating Gaussian sam-

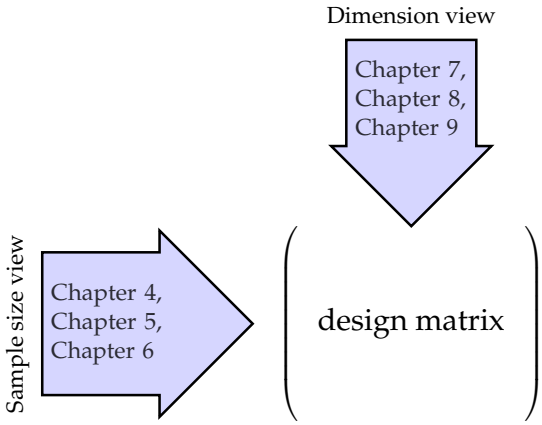

Figure 1.3: The two views of representation learning shown on a $n \times d$ design matrix $\mathbf{X}$. Here, n denotes the amount of d-dimensional data points.

ples moves away from the implied data manifold and propose using fixed p-norm spaces and distributions on them. Our experimental results show superior performance in various metrics for interpolating samples while maintaining the same sampling quality.

1.3 Outline

This thesis is organized as follows. We present prior work relevant to this thesis in Chapter 2. More specific related work is discussed in the respective chapters. In Chapter 3, we present the necessary notation and background information, such as an introduction to archetypal analysis and normalizing flows. Those concepts are then used in the remaining chapters. We introduce the frame as a representative subset for archetypal analysis in Chapter 4. In addition, we provide a novel and efficient way of computing the frame. In Chapter 5, we extend the frame idea to kernel-induced feature spaces and show that the frame also serves as a representative subset for optimal experimental design. Chapter 6 improves upon the work in Chapter 4 as it introduces a representative subset for archetypal analysis that comes with theoretical guarantees. In Chapter 7, we derive probabilistic models for describing movements by a change of representation. Chapter 8 improves upon this approach and introduces a unified contextual movement model based on normalizing flows. In Chapter 9, we use normalizing flows for changing the representation of data and consider the problem of interpolation. We highlight the problems of existing approaches and offer a solution. Finally, Chapter 10 concludes this thesis. Figure 1.3 shows which chapter addresses which of the two previously mentioned views on representation learning.

1.4 Previously Published Work

Most chapters of this thesis are based on articles that emerged from collaborations. Some of those articles have already been published as conference papers and journal articles. Other articles are submitted and currently under review. The following list provides a brief overview and my contributions to those articles.

[1] Mair, S., Boubekki, A., and Brefeld, U. (2017). Frame-based data factorizations. In *International Conference on Machine Learning*, pages 2305–2313

Together with Ahcène Boubekki I worked on the scalability of archetypal analysis from which the concept of using the frame emerged from. I had the idea of using the NNLS algorithm to efficiently compute the frame, which was then used as a representative subset for archetypal analysis. In addition, I developed the theoretical analysis of the proposed approach while the implementation and the empirical evaluation were done together with Ahcène Boubekki.

[2] Mair, S., Rudolph, Y., Closius, V., and Brefeld, U. (2018). Frame-based optimal design. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 447–463. Springer, Cham

This paper builds upon the frame idea which was introduced in the last paper [1] and extends it to kernel-induced feature spaces. Just as in the last paper, the frame served as a representative subset, this time for optimal experimental design. I developed the theoretical foundations while Yannick Rudolph and Vanessa Closius took care of the implementation and carried out the experiments under my supervision.

[3] Mair, S. and Brefeld, U. (2019). Coresets for archetypal analysis. In *Advances in Neural Information Processing Systems*, pages 7247–7255

In this paper, I proposed another representative subset for large-scale learning of archetypal analysis. I derived a coreset for archetypal analysis and showed a connection to k-means clustering. The coreset resembles a representative subset that comes with theoretical guarantees. Those guarantees were missing in the first frame paper [1]. I did all the theoretical and experimental analyses myself.

[4] Brefeld, U., Lasek, J., and Mair, S. (2019). Probabilistic movement models and zones of control. *Machine Learning*, 108(1):127–147

Ulf Brefeld had the idea of deriving data-driven probabilistic movement models that serve as a foundation for so-called zones of control. Together

with Jan Lasek, I worked out the details of the proposed approach. In addition, we implemented a prototype and conducted the experiments together. Parts of this paper have also been published as a book chapter (Brefeld et al., 2020).

[5] Fadel, S. G., Mair, S., da Silva Torres, R., and Brefeld, U. (2021a). Contextual movement models based on normalizing flows. *AStA Advances in Statistical Analysis*, pages 1–22

[6] Fadel, S. G., Mair, S., da Silva Torres, R., and Brefeld, U. (2021b). Principled interpolation in normalizing flows. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 116–131. Springer

Together with Samuel Fadel, I worked on density estimation and generative modeling using normalizing flows. In the first paper [5], we improved the movement models proposed in [4] in terms of accuracy and scalability. In addition, we derived a unified model which includes contextual information such as a time horizon, current velocity, and the positions of the other players. For the second paper [6], we looked into the problem of interpolation in generative modeling using normalizing flows. For both papers, Samuel Fadel and I contributed equally in terms of theory and experimental evaluation.

Chapter 2

Related Work

This chapter discusses related literature relevant to this thesis. In particular, the two aforementioned views on representation learning are considered. Other individual related works are discussed in the respective chapters.

The Sample Size View

Large-scale learning is ubiquitous in machine learning. To deal with large data under computational constraints in space and/or time, one has to either approximate the method at hand or the large data itself. Selecting a representative subset of data is thus a fundamental problem in machine learning. DuMouchel et al. (1999) introduced a method which is scaling down a large data set and called the method *data squashing*. The goal is to obtain a representative subset via a lossy compression of data that leads to approximately the same results as analyses conducted on the original data set. To be efficient, the approach is required to perform better than a uniform subset.

In general, strategies for representative subset selection can be divided into three classes. The first deals with unsupervised data subset selection, which has no access to any labels. The second class deals with supervised data subset selection, where the selection algorithm has access to a labeled data set. Finally, the third class uses an active learning approach which uses queries to obtain labels as feedback for the subset selection process. All approaches proposed in the first part of this thesis belong to the first class of unsupervised data subset selection. Hence, we put our focus on this setting in the remainder.

Training kernel methods such as support vector machines (SVMs) (Boser et al., 1992; Cortes and Vapnik, 1995) or Gaussian processes (GPs) (Rasmussen and Williams, 2006) at large scales is prohibitive as training scales cubically and storage scales quadratically in the number of training examples. To remedy this, low-rank approximations of the kernel matrix have

been proposed. Most frequently used are random Fourier features (Rahimi and Recht, 2007) and the Nyström approximation (Williams and Seeger, 2001). The latter's quality strongly depends on the chosen subset of data on which the approximation is built. The points in this subset are also called landmark points. In the standard setting, a uniform subset of data is chosen. However, other approaches such as sampling based on the entries of the kernel matrix (Drineas and Mahoney, 2005), an adaptive sampling technique (Deshpande et al., 2006), greedy sampling (Farahat et al., 2011), and a recursive sampling approach (Musco and Musco, 2017) have been shown to produce better results.

The idea of using the frame that is the subset of minimum cardinality that yields the same convex hull as the entire data set is motivated by geometry. Coresets and the previously mentioned landmarks also originate from computational geometry and usually refer to subsets that approximate the shape, extent, or other geometrical properties of data. Such geometric properties include the diameter, width, the smallest bounding box, and the smallest enclosing ball, among others (Agarwal et al., 2004). *Coresets* are also often used in machine learning. They summarize large data sets by small, possibly weighted, subsets on which machine learning models perform provably competitive compared to the performance of the model trained on all data. Coresets have been proposed for a variety of settings, *e.g.*, for SVMs (Tsang et al., 2005; Tukan et al., 2020b,a), logistic regression (Munteanu et al., 2018; Tukan et al., 2020b), kernel density estimates (Phillips and Tai, 2020), principal component analysis (PCA) (Feldman et al., 2016, 2020), and clustering approaches such as k-means (Har-Peled and Mazumdar, 2004; Har-Peled and Kushal, 2007; Chen, 2009; Feldman and Langberg, 2011; Lucic et al., 2016; Bachem et al., 2018a,b; Feldman et al., 2020). In addition, Bayesian coresets have been proposed for efficient Bayesian inference on small weighted subsets of data (Huggins et al., 2016; Campbell and Broderick, 2018, 2019; Campbell and Beronov, 2019; Manousakas et al., 2020; Zhang et al., 2020). Comprehensive surveys about coresets are provided by Phillips (2016) and Feldman (2020).

Another direction that is frequently followed is to select *diverse subsets* for data summarization and as representative subsets. A well-known approach that enforces diversity is to use determinantal point processes (DPPs), which were originally introduced by Macchi (1975) and brought to the machine learning community by Kulesza et al. (2012). DPPs are commonly used for diverse subset selection, *i.e.*, in document summarization (Chao et al., 2015), image search tasks (Kulesza and Taskar, 2011), pose estimation (Kulesza and Taskar, 2010), recommender systems (Zhou et al., 2010), and as landmarks for a Nyström approximation (Schreurs et al., 2019).

All approaches mentioned so far use subsets of data as representative subsets. However, the generation of new, *synthetic* data points was also explored. For example, in GP regression, sparse inducing points are used for scaling up GP inference (Snelson and Ghahramani, 2005; Titsias, 2009). Recently, Manousakas et al. (2020) proposed Bayesian pseudocoresets. Just as Bayesian coresets, they aim to make computationally demanding Bayesian inference problems scalable by restricting the computation to a small (weighted) subset of data. However, the difference to vanilla Bayesian coresets is that pseudocoresets consist of synthetic points to achieve differential privacy (Dwork, 2008).

The Dimension View

The previous part was concerned with an efficient representation of data with respect to the number of data points. However, finding an efficient representation of data in terms of its dimensions, hence their features, is of equal importance. The goal of representation learning is to find a good representation of data, but what *good* is depends on the problem at hand. Examples include representations that are disentangled, interpretable, low-dimensional, sparse, yield better generalizations or useful representations for downstream tasks, or representations that facilitate specific operations, *i.e.*, the generation of new data.

One class of important representation learning techniques is autoencoders (Rumelhart et al., 1986; LeCun, 1987). They can be seen as non-linear extensions of linear methods such as the PCA (Pearson, 1901). The idea of an autoencoder is to have a non-linear transformation into a latent space that is usually of lower dimensionality than the input signal. This transformation is realized by an encoder network. In the case of a low-dimensional latent space, the data is being compressed, and the latent representation should preserve important properties of data. The decoder network then transforms the latent representation back into the original data space. Hence, the decoder can be seen as an approximately inverse operation of the encoder.

A variational autoencoder (VAE) (Kingma and Welling, 2014; Rezende et al., 2014) imposes the latent space to follow a given prior distribution. This allows the generation of new synthetic data. Since the VAE enforces a specific distribution on the latent space, it is possible to first sample from this distribution and then transform those samples from the latent space back to data space using the decoder. Hence, VAEs are probabilistic models that allow for efficient sampling. However, their ability as density estimators is limited since the data likelihood can only be approximated.

Nevertheless, VAEs have been shown to work well even for sequential, symbolic data such as text. For example, Bowman et al. (2016) use a VAE

for language modeling, embedding sentences in a continuous space, and generating of new sentences. A similar approach is made by Miao et al. (2016), which use the VAE for document modeling. Here, it is worthwhile to mention that the structure of how data is represented in the latent space is also influenced by the specific choice of the prior distribution.

Davidson et al. (2018) enforce a hyperspherical latent space in a VAE and use a von Mises-Fisher (vMF) distribution as a prior. They show that the change of prior is beneficial over the standard choice of a Gaussian prior, especially for semi-supervised classification tasks and for link prediction on graphs. In addition, Xu and Durrett (2018) also employ a vMF distribution on a hypersphere as a prior of a VAE for document modeling and obtain better results than using a Gaussian prior.

Changing the prior is also beneficial when working with hierarchically structured data such as graphs. Embedding such data in Euclidean spaces might be disadvantageous and hyperbolic spaces have proven to be a suitable alternative. For example, Mathieu et al. (2019) propose a Poincaré VAE that enforces a hyperbolic latent space. As prior distributions, they consider a Riemannian normal distribution and a wrapped normal distribution.

A related approach to VAEs are normalizing flows (NFs) (Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rippel and Adams, 2013; Dinh et al., 2015; Rezende and Mohamed, 2015). Like variational autoencoders, normalizing flows are probabilistic and generative models. New data can be efficiently generated by first sampling from a given prior distribution and then transforming the sample using the decoder. However, instead of having two separate networks that are approximately inverse to each other as in the VAE, a normalizing flow uses an invertible neural network and hence a bijective transformation as an encoder. Thus, the decoder is simply given by the inverse of the encoder. A consequence of the bijectivity is that the latent space in normalizing flows is of the same dimensionality as the input space. However, this also means that there is no reconstruction error. Normalizing flows can be trained by minimizing the exact negative log-likelihood. We provide more details on normalizing flows in the next chapter.

Chapter 3

Background

In this chapter, we introduce concepts that are used in the chapters that follow. For example, we take a look at the concept of the frame which is used in Chapters 4 and 5, archetypal analysis as we use it in Chapters 4 and 6, and at normalizing flows, which are used in Chapters 8 and 9.

3.1 Preliminaries

Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}_{i=1}^n$ be a discrete data set consisting of n data points in d dimensions, *i.e.*, $\mathbf{x}_i \in \mathbb{R}^d$ for $i = 1, 2, \ldots, n$. The convex hull $\text{conv}(\mathcal{X})$ of $\mathcal{X}$ is the intersection of all convex sets containing $\mathcal{X}$. Furthermore, $\text{conv}(\mathcal{X})$ is the set of all convex combinations of points in $\mathcal{X}$. We now introduce the concept of the *frame* as follows.

Definition 3.1. *Let* $\text{conv}(\mathcal{X})$ *be the convex hull of a discrete data set* $\mathcal{X} \subset \mathbb{R}^d$. *The frame* $\mathcal{F} \subseteq \mathcal{X}$ *of* $\mathcal{X}$ *consists of points on the boundary of* $\text{conv}(\mathcal{X})$, *i.e.*, $\mathcal{F} = \partial\,\text{conv}(\mathcal{X}) \cap \mathcal{X}$.

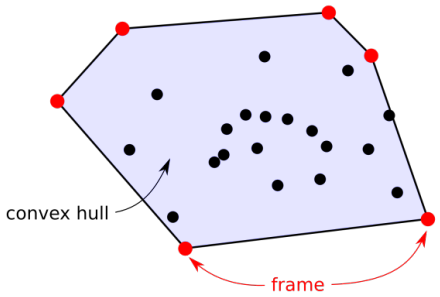

Figure 3.1: An illustration of the frame. The frame consists of points on the boundary of the convex hull of a data set.

Hence, the frame consists of the extreme points of $\mathcal{X}$. Those points cannot be represented as convex combinations of other points rather than themselves. An example of the frame is depicted in Figure 3.1. Note that the frame $\mathcal{F}$ and the data set $\mathcal{X}$ yield the same convex hull, *i.e.*, $\text{conv}(\mathcal{F}) = \text{conv}(\mathcal{X})$. By $q = |\mathcal{F}|$ we refer to the size of the frame and we call the portion of points in $\mathcal{X}$ belonging to the frame $\mathcal{F}$, *i.e.*, q/n, the *frame density*. We propose an efficient and novel way of computing the frame in Chapter 4 and show how to compute the frame in kernel-induced feature spaces in Chapter 5.

There are some straightforward consequences that we use in later chapters. When the inner product between two points is maximized, one of the points is an extreme point and thus belongs to the frame.

Lemma 3.1. *Let $\mathcal{X}$ be a finite set of discrete points, then*

$$\forall \mathbf{x} \in \mathcal{X} : \quad \arg\max_{\mathbf{x}' \in \mathcal{X}} \mathbf{x}^\top \mathbf{x}' \in \mathcal{F}.$$

Proof. Linearity and convexity of the inner product imply that its maximum is realized by an extreme point of the domain. Since the domain is $\mathcal{X}$, the maximum belongs to its frame $\mathcal{F}$. □

Note that the same holds not only for any $\mathbf{x} \in \mathcal{X}$, but for any fix vector in general. In addition, every point of the domain lies in the span of some points on the frame.

Proposition 3.1. *Every point $\mathbf{x}$ of $\mathcal{X}$ can be written as a convex combination of at most $d + 1$ points from the frame $\mathcal{F}$ of $\mathcal{X}$.*

Proof. For the proof, we refer to Brondsted (2012). □

3.2 Archetypal Analysis for Matrix Factorization

In unsupervised learning, matrix factorizations (MFs) are used in many different learning tasks, including clustering, classification, recommendation, text and social network analysis, image denoising, and hyperspectral imaging. Consequently, a lot of approaches to factor a matrix into two or more matrices have been proposed. The general setting is as follows. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}_{i=1}^n$ be a data set consisting of n data points in d dimensions and $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the design matrix. The goal is to decompose the design matrix and represent it as a matrix product $\mathbf{X} = \mathbf{AZ}$, consisting of a weight matrix $\mathbf{A} \in \mathbb{R}^{n \times k}$ and a factor matrix $\mathbf{Z} \in \mathbb{R}^{k \times d}$. Here, $k \in \mathbb{N}$ denotes the latent dimensionality. Matrix factorizations come in many variants that usually differ in the loss function which is used to quantify the reconstruction error and their set of constraints, which are imposed on the factorization.

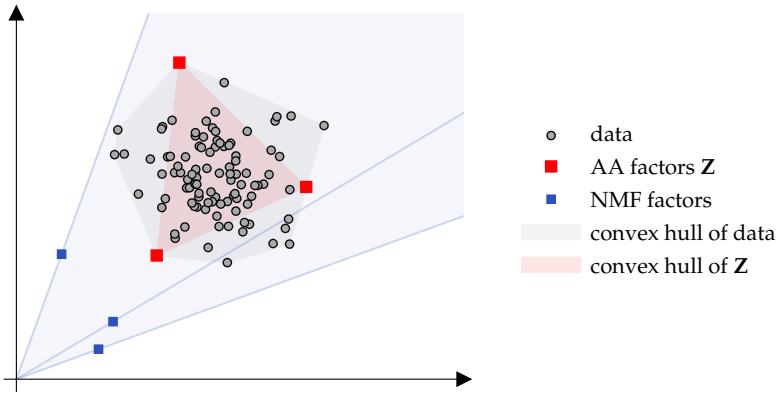

Figure 3.2: A comparison of factorizations computed with NMF (blue squares) and AA (red squares). While NMF may place the factors far away from data, AA uses data points on the boundary of the convex hull and yields an intuitive solution.

A prominent example is the non-negative matrix factorization (NMF) (Paatero and Tapper, 1994; Lee and Seung, 1999, 2000), where all matrices $\mathbf{X}, \mathbf{A}$, and $\mathbf{Z}$ are assumed to be non-negative, and the error is typically measured in terms of the Frobenius norm. A common limitation of matrix factorization techniques, including NMF, is a lack of interpretability of the resulting factors. The learned factors may lie far from other data points and do not necessarily contribute to interpretability, as depicted in Figure 3.2. Thus, the focus is on interpretable factors which lead to stochastic NMFs for clusterings (Arora et al., 2011, 2013) and convexity constraints (Ding et al., 2010).

However, computing interpretable factors is actually an "old" problem: archetypal analysis (Cutler and Breiman, 1994) is an unsupervised learning method that represents every data point as a convex combination of prototypes, the so-called archetypes. Every data point is represented as a convex mixture of (a subset of) archetypes and, due to the convexity, these mixtures are often interpreted probabilistically. A key property of archetypal analysis is that the archetypes are themselves convex mixtures of data points. Consequently, archetypes lie on the boundary of the convex hull of the data. Hence, archetypal analysis approximates the convex hull with a given number of vertices. It follows that this approximation is equivalent to a matrix factorization of the design matrix. Due to the convexity constraints, archetypal-based factorizations are not only better interpretable but unfortunately also much more expensive than regular matrix factorization techniques, which hinders usage at even moderate scales. An example of archetypal analysis is depicted in Figure 3.3 for various numbers ($k = 2, 3, 5$) of archetypes.

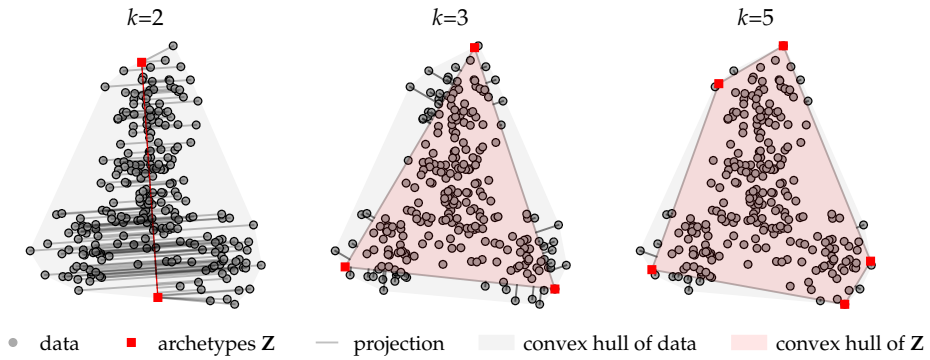

Figure 3.3: An example of archetypal analysis in two dimensions for various numbers ($k = 2, 3, 5$) of archetypes $\mathbf{Z}$. The archetypes are placed on the boundary of the convex hull of data. The convex hull of archetypes $\mathbf{Z}$ approximates the convex hull of data with a given number of corner points. All points within the convex hull of $\mathbf{Z}$ are represented in a lossless way. All points outside are being represented by points of the convex hull of $\mathbf{Z}$. Thus, those points are being projected.

We now formally introduce archetypal analysis. In archetypal analysis, every data point $\mathbf{x}_i$ is represented as a convex combination of k archetypes $\mathbf{z}_1, \ldots, \mathbf{z}_k$, *i.e.*,

$$\mathbf{x}_i = \mathbf{Z}^\top \mathbf{a}_i, \quad \sum_{j=1}^{d} (\mathbf{a}_i)_j = 1, \quad (\mathbf{a}_i)_j \geq 0,$$

where $\mathbf{a}_i \in \mathbb{R}^k$ is the weight vector of the i-th data point $\mathbf{x}_i$, and $\mathbf{Z} \in \mathbb{R}^{k \times d}$ is the matrix of stacked archetypes. The archetypes $\mathbf{z}_j$ $(j = 1, \ldots, k)$ themselves are also represented as convex combinations of data points, *i.e.*,

$$\mathbf{z}_j = \mathbf{X}^\top \mathbf{b}_j, \quad \sum_{i=1}^{n} (\mathbf{b}_j)_i = 1, \quad (\mathbf{b}_j)_i \geq 0,$$

where $\mathbf{b}_j \in \mathbb{R}^n$ is the weight vector of the j-th archetype. Let $\mathbf{A} \in \mathbb{R}^{n \times k}$ and $\mathbf{B} \in \mathbb{R}^{k \times n}$ be the matrices consisting of the weights $\mathbf{a}_i$ $(i = 1, \ldots, n)$ and $\mathbf{b}_j$ $(j = 1, \ldots, k)$, respectively. Then, archetypal analysis yields a factorization of the design matrix as follows,

$$\mathbf{X} = \mathbf{ABX} = \mathbf{AZ}, \tag{3.1}$$

where $\mathbf{Z} = \mathbf{BX} \in \mathbb{R}^{k \times d}$ is the matrix of archetypes. Due to the convexity constraints, the weight matrices $\mathbf{A}$ and $\mathbf{B}$ are row-stochastic. By minimizing the residual sum of squares (RSS), given by

$$\mathrm{RSS}(k) = \|\mathbf{X} - \mathbf{ABX}\|_F^2, \tag{3.2}$$

Algorithm 3.1 Archetypal analysis (AA)

Input: data matrix $\mathbf{X}$, number of archetypes k
Output: factor matrices $\mathbf{A}$ and $\mathbf{Z}$, where $\mathbf{Z} = \mathbf{B}\mathbf{X}$ and $\mathbf{A}\mathbf{B}\mathbf{X} \approx \mathbf{X}$
$\mathbf{Z} \leftarrow$ initialization of the archetypes
while not converged **do**
  for $i = 1, 2, \ldots, n$ **do**
    $\mathbf{a}_i = \underset{\|\mathbf{a}_i\|_1=1,\ \mathbf{a}_i \geq \mathbf{0}}{\arg\min} \|\mathbf{Z}^\top \mathbf{a}_i - \mathbf{x}_i\|_2^2$
  end for
  $\mathbf{Z} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{X}$
  for $j = 1, 2, \ldots, k$ **do**
    $\mathbf{b}_j = \underset{\|\mathbf{b}_j\|_1=1,\ \mathbf{b}_j \geq \mathbf{0}}{\arg\min} \|\mathbf{X}^\top \mathbf{b}_j - \mathbf{z}_j\|_2^2$
  end for
  $\mathbf{Z} = \mathbf{B}\mathbf{X}$
end while

the optimal weight matrices $\mathbf{A}$ and $\mathbf{B}$ can be found. Here, $\|\cdot\|_F$ denotes the Frobenius norm. Optimizing the RSS is a non-convex problem in $\mathbf{A}$ and $\mathbf{B}$, but it is convex in $\mathbf{A}$ for a fixed $\mathbf{B}$ and vice versa. After initializing the archetypes, the optimization is typically done by iteratively solving for $\mathbf{A}$ and $\mathbf{B}$ as outlined in Algorithm 3.1.

The archetypes can be initialized using a random subset of data. However, a better way of initializing the archetypes is given by the `FurthestSum` procedure of Mørup and Hansen (2010, 2012).

Due to the convexity constraints, archetypal-based factorizations are not only better interpretable but unfortunately also much more expensive than regular matrix factorization techniques, which hinders usage at even moderate scales. Several approaches have been proposed to remedy the edacious nature of archetypal analysis, proposing, *e.g.*, efficient active-set quadratic programming (Chen et al., 2014), projected gradients (Mørup and Hansen, 2010, 2012), or Frank-Wolfe techniques (Bauckhage et al., 2015) for optimization. Although these approaches are useful contributions, they do not mitigate the inherent complexity of archetypal analysis nor provide theoretical guarantees on the approximation quality.

Besides, there are also non-linear extensions to archetypal analysis. For example, Mørup and Hansen (2010, 2012) propose a kernelized variant of archetypal analysis. Another non-linear variant called *AAnet* based on autoencoders is proposed by van Dijk et al. (2019). A probabilistic alternative based on variational autoencoders (VAEs) is provided by Keller et al. (2019, 2020).

The Geometry of Archetypal Analysis

From a geometrical point of view, archetypal analysis yields an approximation of the convex hull with just k vertices. Every point inside the induced polytope can be reconstructed in a lossless way, and every point outside will be projected onto it. This can be seen in Figure 3.3, where the lines orthogonal to the red polytope denote the projections. The optimization of the RSS will minimize the reconstruction error. The objective function can be rewritten as a sum of projections of the data points to the archetype-induced convex hull as follows,

$$\|\mathbf{X} - \mathbf{AZ}\|_F^2 = \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{q} \in \text{conv}(\{\mathbf{z}_1, \ldots, \mathbf{z}_k\})} \|\mathbf{x} - \mathbf{q}\|_2^2,$$

where $\text{conv}(\cdot)$ refers to the convex hull of a set. Apart from this geometrical interpretation, there is another fact that connects archetypal analysis to the convex hull and thus the frame, as outlined in the preliminaries. It can be shown that, in general, the archetypes $\mathbf{z}_1, \ldots, \mathbf{z}_k$ lie on the boundary $\partial\,\text{conv}(\mathcal{X})$ of the convex hull of the data $\mathcal{X}$.

Theorem 3.1 (Cutler and Breiman (1994))**.** *Let $\mathcal{X} \subset \mathbb{R}^d$ be a discrete data set, $\text{conv}(\mathcal{X})$ be its convex hull and $\mu \in \mathbb{R}^d$ be the mean of $\mathcal{X}$. Furthermore, let $k \in \mathbb{N}$ be the number of archetypes and $\mathcal{F}$ be the frame of $\mathcal{X}$ with cardinality q.*

(i) If $k = 1$, choosing $\mathbf{z}_1 = \mu$ minimizes the RSS;

(ii) if $1 < k < q$, there is a set of archetypes $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\}$ on the boundary of $\text{conv}(\mathcal{X})$ that minimizes the RSS;

(iii) if $k = q$, choosing $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\} = \mathcal{F}$ results in a RSS of zero.

3.3 Normalizing Flows for Density Estimation

Another common problem in unsupervised learning is density estimation. Let $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\} \subset \mathbb{R}^d$ be a set of iid instances which are assumed to be drawn from an unknown distribution $p_x(\mathbf{x})$. A straightforward parametric way of estimating a density of $\mathcal{X}$ is to assume a known distribution class $\hat{p}_x(\mathbf{x})$, parametrized by θ, and to estimate the optimal parameters θ. This approach is known as maximum likelihood. It can be seen as moving the distribution by changing θ to fit the static data set $\mathcal{X}$ best. If the assumed distribution class $\hat{p}_x(\mathbf{x})$ underfits the data set, we typically increase its complexity. One way would be to use a mixture model such as the Gaussian mixture model (GMM) and estimate its parameters using the expectation maximization algorithm (Dempster et al., 1977).

Instead of increasing the complexity of the distribution and moving it to fit the data best, we could also fix a simple base distribution and

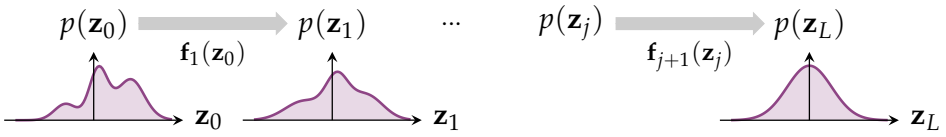

Figure 3.4: An example of a one-dimensional normalizing flow. The leftmost plot shows an unknown data distribution $p_x(\mathbf{x})$ which is successively transformed by a chain of transformations $\mathbf{f} = \mathbf{f}_L \circ \mathbf{f}_{L-1} \circ \cdots \circ \mathbf{f}_1$ to follow a simple transformation $p_z(\mathbf{z})$, that is depicted in the rightmost plot.

instead move the data points via a powerful parametrized transformation. This orthogonal approach to density estimation is provided by so-called normalizing flows (Tabak and Vanden-Eijnden, 2010; Tabak and Turner, 2013; Rippel and Adams, 2013; Dinh et al., 2015; Rezende and Mohamed, 2015), which aim to learn an accurate model of $p_x(\mathbf{x})$. The main building blocks of normalizing flows are bijective transformations and the change of variable theorem, which is given as follows.

Theorem 3.2 (Billingsley (2008))**.** *Let Z be a multivariate continuous random variable with probability density function $p_z(\mathbf{z})$ and $\mathbf{f}^{-1} : \mathbb{R}^d \to \mathbb{R}^d, \mathbf{z} \mapsto \mathbf{x}$ a bijective and differentiable function within the domain of $\mathbf{z}$, then the probability density of $X = \mathbf{f}^{-1}(Z)$ is given by*

$$p_x(\mathbf{x}) = p_z\left(\mathbf{f}(\mathbf{x})\right) \cdot \left|\det J_{\mathbf{f}}(\mathbf{x})\right|,$$

where $J_{\mathbf{f}}(\mathbf{x}) = \frac{\partial}{\partial \mathbf{x}} \mathbf{f}(\mathbf{x})$ is the Jacobian matrix of the transformation $\mathbf{f}$.

Let $\mathbf{f}^{(\theta)} : \mathbb{R}^d \to \mathbb{R}^d$ be a bijective transformation parametrized by θ. The idea of a normalizing flow is to introduce $\mathbf{z} = \mathbf{f}^{(\theta)}(\mathbf{x})$ and using the change of variable theorem to express the unknown $p_x(\mathbf{x})$ by a (simpler) distribution $p_z(\mathbf{z})$, defined on $\mathbf{z} \in \mathbb{R}^d$, given by

$$p_x(\mathbf{x}) = p_z(\mathbf{f}^{(\theta)}(\mathbf{x})) \cdot |\det J_{\mathbf{f}^{(\theta)}}(\mathbf{x})|, \tag{3.3}$$

where $J_{\mathbf{f}}^{(\theta)}(\mathbf{x})$ is the Jacobian matrix of the bijective transformation $\mathbf{f}^{(\theta)}$. We refer to $p_z(\mathbf{z})$ as the base distribution for which typically an isotropic Gaussian is used. We can learn the transformation $\mathbf{f}^{(\theta)}$ from the data to its base representation by optimizing the parameters θ that minimize the negative log-likelihood

$$\min_{\theta} \; - \sum_{\mathbf{x} \in \mathcal{X}} \log p_x(\mathbf{x}).$$

Increasing the expressive complexity of the density estimation now corresponds to increasing the complexity of the transformation $\mathbf{f}^{(\theta)}$. Instead of having just a single transformation, we use a chain of L parametrized

bijective functions, *i.e.*, $\mathbf{f}^{(\theta)} = \mathbf{f}_L^{(\theta)} \circ \mathbf{f}_{L-1}^{(\theta)} \circ \cdots \circ \mathbf{f}_1^{(\theta)}$, that maps observations $\mathbf{x}$ into representations $\mathbf{z}$ that are governed by the base distribution $p_z(\mathbf{z})$. Let $\mathbf{z}_0 = \mathbf{x}$ be the input data point and $\mathbf{z}_L = \mathbf{z}$ be the corresponding output of the chain, where every intermediate variable is given by $\mathbf{z}_j = \mathbf{f}_j(\mathbf{z}_{j-1})$, for $j = 1, \ldots, L$. This is illustrated in Figure 3.4. The log-likelihood can then be expressed as the log-likelihood of the base distribution and each log-determinant of the Jacobians of each transformation as

$$\log p_x(\mathbf{x}) = \log p_z(\mathbf{z}) + \sum_{j=1}^{L} \log \left| \det J_{\mathbf{f}_j^{(\theta)}}(\mathbf{z}_{j-1}) \right|. \tag{3.4}$$

In the remainder, we drop the subscripts of the distribution $p(\cdot)$ and the superscripts of the transformation $\mathbf{f}^{(\theta)}$, whenever it is clear from context.

Normalizing Flows as Generative Models

As introduced above and shown in Figure 3.4, a normalizing flow transforms a sample $\mathbf{x}$ using the bijective transformation $\mathbf{f}$ into a base representation $\mathbf{z}$, which follows a specified base distribution $p(\mathbf{z})$. To create a new sample $\mathbf{x}$, the inverse direction of this approach can be used. First, a sample $\mathbf{z}$ is being drawn from the base distribution $p(\mathbf{z})$ and then transformed into $\mathbf{x}$ using the inverse chain of transformations $\mathbf{f}^{-1}$. The $\mathbf{x}$ generated in this way follows the data distribution $p(\mathbf{x})$.

Types of Transformations

Flow-based generative models can be categorized by how the Jacobian structure of each transformation $\mathbf{f}_j$ is designed since computing its determinant is crucial to its computational efficiency. The Jacobians either have a lower triangular structure, such as autoregressive flows (Kingma et al., 2016), or a structured sparsity, such as coupling layers in RealNVP (Dinh et al., 2017) and Glow (Kingma and Dhariwal, 2018). Chen et al. (2019) introduce transformations with free-form Jacobians, which allow much higher expressibility, by replacing the computation of the determinant with another estimator for the log-density. For more information regarding flow-based generative models, we refer the reader to Papamakarios et al. (2021).

As an architecture for the flow models in this thesis, we use Glow (Kingma and Dhariwal, 2018), which is itself based on RealNVP (Dinh et al., 2017). Glow is built from three main transformations: activation normalization (actnorm), 1×1 invertible convolution, and affine coupling. Those transformations are employed in a multi-scale architecture, reshaping the image tensors to have fewer pixels with more channels, referred to as

squeezing. The channels are then split, and further operations are only performed on half of them. This squeezing and splitting scheme is performed several times for scalability. Squeezing, however, is not suitable for vectorial data. We first detail on the transformations for vectorial data before we introduce their variants for tensor data.

Transformations for Vector Data

We begin by introducing the actnorm and affine coupling transformations. For both of them, let $\mathbf{z}, \mathbf{u} \in \mathbb{R}^d$ be d-dimensional vectors, where $\mathbf{z}$ is the input of the current transformation, and $\mathbf{u}$ is its output, being directly fed into the next transformation in the chain.

Actnorm. Let $\mathbf{a} \in \mathbb{R}^d$ be a scaling vector and $\mathbf{b} \in \mathbb{R}^d$ an offset vector. The actnorm transformation (Kingma and Dhariwal, 2018), its inverse, and log-determinant of its Jacobian are given by

$$\begin{aligned} \mathbf{u} &= \mathbf{f}(\mathbf{z}) = \mathbf{a} \odot \mathbf{z} + \mathbf{b}, \\ \mathbf{z} &= \mathbf{f}^{-1}(\mathbf{u}) = (\mathbf{u} - \mathbf{b})/\mathbf{a}, \\ \log|\det J_{\mathbf{f}}(\mathbf{z})| &= \sum_{j=1}^{d} \log|a_j|, \end{aligned} \tag{3.5}$$

respectively, where $\odot$ denotes the Hadamard product. Particularly, $\mathbf{a}$ and $\mathbf{b}$ are learned as part of the transformation and are initialized with the first batch of data. This initialization is such that the mean and standard deviation of $\mathbf{u}$ are zero and one, respectively.

Affine coupling. This transformation (Dinh et al., 2017) is slightly more involved. The input $\mathbf{z} \in \mathbb{R}^d$ is initially split into two parts $(\mathbf{z}_1, \mathbf{z}_2)$, each of which is d'-dimensional, where $d' = d/2$. Then, the second part is simply copied, $\mathbf{u}_2 = \mathbf{z}_2$, while $\mathbf{z}_1$ is transformed based on information from $\mathbf{z}_2$ as

$$\begin{aligned} (\log \mathbf{a}, \mathbf{b}) &= \mathrm{NN}_{\mathrm{ac}}(\mathbf{z}_2), \\ \mathbf{u}_1 &= \exp(\log \mathbf{a}) \odot \mathbf{z}_1 + \mathbf{b}, \end{aligned} \tag{3.6}$$

where $\mathrm{NN}_{\mathrm{ac}} : \mathbb{R}^{d'} \to \mathbb{R}^d$ is a non-linear mapping, generally realized by a neural network (NN). The output of the network can be split into the two parameter vectors $\log \mathbf{a} \in \mathbb{R}^{d'}$ and $\mathbf{b} \in \mathbb{R}^{d'}$. The output of the transformation is then given by $\mathbf{u} = \mathrm{concat}(\mathbf{u}_1, \mathbf{u}_2)$. The inverse transformation can be computed by first splitting $\mathbf{u}$ into $(\mathbf{u}_1, \mathbf{u}_2)$, copying the second part as before, *i.e.*, $\mathbf{z}_2 = \mathbf{u}_2$, computing the log-scale and offset parameters using $\mathrm{NN}_{\mathrm{ac}}(\cdot)$ as

in Equation (3.6), and computing the inverse affine linear transformation via

$$\mathbf{z}_1 = (\mathbf{u}_1 - \mathbf{b}) / \exp(\log \mathbf{a}).$$

Finally, the two parts are merged with $\mathbf{z} = \text{concat}(\mathbf{z}_1, \mathbf{z}_2)$. The log-determinant of the Jacobian of this transformation is

$$\log |\det J_{\mathbf{f}}(\mathbf{z})| = \sum_{j=1}^{d'} \log |a_j|.$$

Permutation. This type of transformation permutes the dimensions and thus changes their order. The idea is as follows. Since the affine coupling transformation only changes one-half of the data, the other half remains unchanged. Hence, some permutation ensures that every dimension is affected. Originally, this transformation simply reversed the order of dimensions (Dinh et al., 2015, 2017). However, any fixed random permutation can be used. This transformation obviously has an inverse transformation, and the log-determinant of the Jacobian is zero.

Transformations for Tensor Data

We are now concerned with tensors, which occur, *i.e.*, in image data. Hence, let $\mathbf{z}, \mathbf{u} \in \mathbb{R}^{d_1 \times d_2 \times c}$, where $c \in \mathbb{N}$ denotes the number of channels, $\mathbf{z}$ is the input, and $\mathbf{u}$ is the output of the current transformation.

Actnorm. Let $\mathbf{a} \in \mathbb{R}^c$ and $\mathbf{b} \in \mathbb{R}^c$ be a scaling vector and an offset vector, respectively. The actnorm transformation for tensor data and its inverse are the same as introduced before. The log-determinant of its Jacobian is given by

$$\log |\det J_{\mathbf{f}}(\mathbf{z})| = d_1 d_2 \sum_{j=1}^{c} \log |a_j|.$$

Note that this transformation now acts on the channels and treats every pixel in the tensor equally. As before, $\mathbf{a}$ and $\mathbf{b}$ are initialized using the first batch of data such that the mean and standard deviation of the output $\mathbf{u}$ are zero and one, respectively.

Squeezing. This transformation can be used to *squeeze* some dimensions. Consider, for example, an image of width d_1, height d_2, and c channels. The squeezing transformation can turn this tensor into a tensor of width $d_1/2$, height $d_2/2$, and $4c$ channels. Note that this can be beneficial when using the affine coupling transformation with a splitting at channel level, where an even number of channels is then needed.

Affine coupling. The input $\mathbf{z} \in \mathbb{R}^{d_1 \times d_2 \times c}$ is split into two parts $(\mathbf{z}_1, \mathbf{z}_2)$, each of which is $d_1 \times d_2 \times c'$-dimensional, where $c' = c/2$. Hence, the split is done with respect to the channels of the image tensor. The rest of the transformation is done in an analogous way as introduced earlier.

1×1 **convolutions.** A generalization of the permutation transformation is the invertible 1×1 convolution as proposed by Kingma and Dhariwal (2018). The transformation, its inverse, and log-determinant are given by

$$\begin{aligned} \mathbf{u}_{i,j} &= [\mathbf{f}(\mathbf{z})]_{i,j} = \mathbf{W}\mathbf{z}_{i,j}, \\ \mathbf{z}_{i,j} &= [\mathbf{f}^{-1}(\mathbf{u})]_{i,j} = \mathbf{W}^{-1}\mathbf{u}_{i,j}, \\ \log|\det J_{\mathbf{f}}(\mathbf{z})| &= d_1 d_2 \log|\det \mathbf{W}|, \end{aligned}$$

respectively, where $\mathbf{W} \in \mathbb{R}^{c \times c}$ is a $c \times c$ weight matrix and i, j are indices of the tensor. The weight matrix $\mathbf{W}$ is initialized by a random rotation matrix.

Chapter 4

Frame-based Archetypal Analysis

In this chapter, we examine the idea of using the frame as a representative subset for the problem of archetypal analysis (AA). We introduce Frame-AA as a more efficient version of archetypal analysis, and leverage the observation made in Chapter 3 that archetypal analysis yields an approximation of the convex hull of data with a predefined number of vertices. This makes the frame a natural choice for a representative subset. Given the frame, computing the archetypes reduces to a search within the frame. We propose to go one step further and approximate AA by restricting the whole optimization to the frame: firstly, we compute the frame, and secondly, we compute the archetypes on the frame (and just on the frame). Naturally, the result is a factorization of the frame itself. The factorization can then be extended to all data points by recomputing the weights in hindsight.

Nonetheless, computing the frame is not trivial. In general, the frame can be computed by any convex hull algorithm that works in arbitrary dimensions, such as QuickHull (Barber et al., 1996). However, its application becomes quickly infeasible for dimensionalities larger than three because of dispensable triangulations. Discarding the triangulations leads to solutions based on linear programming (LP) (Dulá and Helgason, 1996; Ottmann et al., 2001; Dulá and López, 2012), which test each individual point whether it is part of the frame or not. These approaches require adequate preprocessing as duplicates may cause false negatives. Hence, we introduce a novel method on how to compute the frame efficiently.

In summary, the key contributions of this chapter are as follows: we (i) show that the frame can be computed efficiently by a quadratic program (QP), (ii) provide theoretical and empirical justifications for the developed method, (iii) propose an efficient approximation of archetypal analysis which uses the frame as a representative subset, and (iv) show the efficiency and competitiveness of our proposed approach empirically on several data sets.

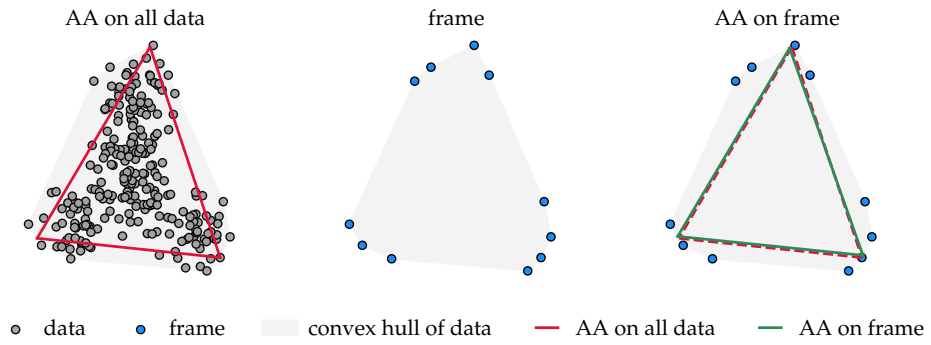

Figure 4.1: The frame as a representative subset for archetypal analysis. Left: AA on a data set; center: the frame of the full data set; and right: AA computed on the frame (green) compared to AA computed on the full data set (red).

4.1 Frame-based Factorizations

The idea of the proposal is as follows. Let $\mathcal{X}$ be the data set as previously introduced and $\mathcal{F}$ be its frame. Due to the fact that the archetypes lie on the boundary of the convex hull (*cf.* Theorem 3.1), it is possible to restrict the search of the archetypes to the frame. However, we intend to go one step further. Since the frame $\mathcal{F}$ and the data $\mathcal{X}$ yield the same convex hull, we restrict the entire factorization in Equation (3.1) to the matrix $\mathbf{H} \in \mathbb{R}^{q \times d}$ that consists of stacked points of the frame, *i.e.*, the factorization becomes

$$\mathbf{H} = \mathbf{ABH} = \mathbf{AZ} \quad \text{with} \quad \mathbf{Z} = \mathbf{BH}.$$

The idea is depicted in Figure 4.1. Although archetypal analysis is only computed on the frame, it often yields almost identical archetypes as AA computed on the whole data set, as illustrated in the right subplot of Figure 4.1. The reduction of points yields an accelerated computation.

Assuming a low frame density, *i.e.*, $q \ll n$, a sufficient approximation of the frame, and therefore the convex hull, will also cover most of the points in the interior of its induced polytope. On the other hand, if the frame density is high, the problem will reduce to standard AA in the limit $q \to n$, and the speed-up will vanish. Based on the nature of the problem, we claim that AA makes no sense in scenarios of high frame densities since almost all points will be projected unless k is chosen very high, which is usually not the case.

After computing the factorization by AA, we have to recompute the weight matrix $\mathbf{A} \in \mathbb{R}^{q \times k}$ for all data points using the optimized archetypes $\mathbf{Z}$. This procedure is called *Frame-AA* and is presented in Algorithm 4.1. Note that the idea does not only apply for standard archetypal analysis as presented in Cutler and Breiman (1994) but also all variants thereof (Mørup and Hansen, 2010; Chen et al., 2014; Bauckhage et al., 2015).

Algorithm 4.1 Frame-AA

Input: data set $\mathcal{X}$, number of archetypes $k \in \mathbb{N}$
Output: factor matrices $\mathbf{A}, \mathbf{Z} = \mathbf{BH}$ ($\mathbf{ABH} \approx \mathbf{X}$)
$\mathcal{F} = \text{Frame}(\mathcal{X})$
$\mathbf{H} =$ frame matrix
$\mathbf{A}, \mathbf{Z} = \text{ArchetypalAnalysis}(\mathbf{H}, k)$
$\mathbf{A} = \mathbf{0} \in \mathbb{R}^{n \times k}$
for $i = 1, 2, \ldots, n$ **do**
  $\mathbf{a}_i = \underset{\|\mathbf{a}_i\|_1 = 1, a_{ij} \geq 0}{\arg\min} \|\mathbf{Z}^\top \mathbf{a}_i - \mathbf{x}_i\|_2^2$
end for

Until now, we assumed that the frame $\mathcal{F}$ or equivalently the frame matrix $\mathbf{H} \in \mathbb{R}^{q \times d}$, as required in the first line of the algorithm, is already given. In the following section, we present a novel algorithm for efficiently computing the frame of a discrete data set.

4.1.1 Representing a Single Point

Let $\mathcal{F}$ be the frame of $\mathcal{X}$ and $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the design matrix of $\mathcal{X}$. The idea is as follows. For every $\mathbf{x} \in \mathcal{X}$ we aim to solve the linear system $\mathbf{Xs} = \mathbf{x}$ subject to the constraints that the solution $\mathbf{s}$ is non-negative, sums up to one, and uses only points from the frame, *i.e.*,

$$\begin{aligned} &\underset{\mathbf{s}}{\text{solve}} \quad \mathbf{Xs} = \mathbf{x} \\ &\text{s.t.} \quad s_i \geq 0 \ \wedge \ \mathbf{1}^\top \mathbf{s} = 1 \ \wedge \ s_i \neq 0 \Rightarrow \mathbf{x}_i \in \mathcal{F}. \end{aligned} \tag{4.1}$$

Proposition 4.1 reduces this to a least-squares approach.

Proposition 4.1. *Let $\mathcal{F}$ be the frame of $\mathcal{X}$, $\mathbf{X} \in \mathbb{R}^{d \times n}$ be the design matrix of $\mathcal{X}$, $\mathbf{x} \in \mathcal{X}$ and $\bar{\mathbf{X}} = \left[\mathbf{X}^\top, \mathbf{1}\right]^\top \in \mathbb{R}^{(d+1) \times n}$ as well as $\bar{\mathbf{x}} = (\mathbf{x}^\top, 1)^\top \in \mathbb{R}^{d+1}$ be the augmented versions of $\mathbf{X}$ and $\mathbf{x}$. Then, the following problems have the same solutions.*

(i) solve $\mathbf{Xs} = \mathbf{x}$ s.t. $s_i \geq 0 \ \wedge \ \mathbf{1}^\top \mathbf{s} = 1 \ \wedge \ s_i \neq 0 \Rightarrow \mathbf{x}_i \in \mathcal{F}$.

(ii) solve $\bar{\mathbf{X}}\mathbf{s} = \bar{\mathbf{x}}$ s.t. $s_i \geq 0 \ \wedge \ s_i \neq 0 \Rightarrow \mathbf{x}_i \in \mathcal{F}$.

(iii) $\arg\min_{\mathbf{s} \geq 0} \frac{1}{2}\|\bar{\mathbf{X}}\mathbf{s} - \bar{\mathbf{x}}\|_2^2$ s.t. $s_i \neq 0 \Rightarrow \mathbf{x}_i \in \mathcal{F}$.

Proof. Statement (i) is equivalent to (ii) as it integrates the constraint $\mathbf{1}^\top \mathbf{s} = 1$ into the system of linear equations. Proposition 3.1 assures that (i) has a solution. Hence, the minimum of $\frac{1}{2}\|\bar{\mathbf{X}}\mathbf{s} - \bar{\mathbf{x}}\|_2^2$ is always zero and a solution of (iii) is also a solution to (ii) and (i). □

Let $f(\mathbf{s}) = \frac{1}{2}\|\bar{\mathbf{X}}\mathbf{s} - \bar{\mathbf{x}}\|_2^2$. If the condition that the positive positions of $\mathbf{s}$ refer to points on the frame is dropped, then the problem

$$\mathbf{s} = \arg\min_{\mathbf{s}\geq 0} f(\mathbf{s}) \tag{4.2}$$

is equivalent to the non-negative least squares (NNLS) problem, which is a special case of a QP. The active-set method from Lawson and Hanson (1995) is proven to yield a least-squares estimate of Equation (4.2). As the minimum is zero, it also gives a solution to the linear problem up to the condition that the points contributing to the solution belong to the frame. The method of Lawson and Hanson (1995) called NNLS is outlined in Algorithm 4.2.

Algorithm 4.2 Lawson and Hanson's active set algorithm

Input: design matrix $\bar{\mathbf{X}}$ and right hand side $\bar{\mathbf{x}}$
Output: solution $\mathbf{s}$ with $\mathbf{s}_i \geq 0 \ \ \forall i$
$\mathcal{P} = \emptyset$
$\mathcal{Z} = \{1, 2, \ldots, n\}$
$\mathbf{s} = \mathbf{0}$
$\mathbf{w} = -\nabla f(\mathbf{s}) = \bar{\mathbf{X}}^\top(\bar{\mathbf{x}} - \bar{\mathbf{X}}\mathbf{s})$
while $\mathcal{Z} \neq \emptyset$ and $\exists\, \mathbf{w}[\mathcal{Z}] \geq \varepsilon$ **do**
  $k = \arg\max_j\{\, w_j \mid j \in \mathcal{Z} \,\}$
  $\mathcal{Z} = \mathcal{Z} \setminus \{k\}$
  $\mathcal{P} = \mathcal{P} \cup \{k\}$
  $\bar{\mathbf{X}}_\mathcal{P} = \bar{\mathbf{X}}[:, \mathcal{P}] \in \mathbb{R}^{(d+1)\times|\mathcal{P}|}$
  $\mathbf{z} = \arg\min_\mathbf{z} \|\bar{\mathbf{X}}_\mathcal{P}\mathbf{z} - \bar{\mathbf{x}}\|_2^2$
  $\mathbf{z}[\mathcal{Z}] = 0$
  $\mathcal{J} = \{\, j \in \mathcal{P} \mid \mathbf{z}_j \leq \varepsilon \,\}$
  while $|\mathcal{J}| > 0$ **do**
    $\alpha = \min\{\, \frac{s_j}{s_j - z_j} \mid j \in \mathcal{J} \,\}$
    $\mathbf{s} = \mathbf{s} + \alpha \cdot (\mathbf{z} - \mathbf{s})$
    for $i = 1, 2, \ldots, n$ **do**
      if $i \in \mathcal{P}$ and $|s_i| \leq \varepsilon$ **then**
        $\mathcal{P} = \mathcal{P} \setminus \{i\}$
        $\mathcal{Z} = \mathcal{Z} \cup \{i\}$
        $\bar{\mathbf{X}}_\mathcal{P}[:, i] = \mathbf{0}$
      end if
    end for
    $\mathbf{z} = \arg\min_\mathbf{z} \|\bar{\mathbf{X}}_\mathcal{P}\mathbf{z} - \bar{\mathbf{x}}\|_2^2$
    $\mathbf{z}[\mathcal{Z}] = 0$
  end while
  $\mathbf{x} = \mathbf{z}$
  $\mathbf{w} = -\nabla f(\mathbf{s}) = \bar{\mathbf{X}}^\top(\bar{\mathbf{x}} - \bar{\mathbf{X}}\mathbf{s})$
end while

However, until now, it remains unclear whether the positive elements of the solution refer to points on the frame. The following theorem shows that this is actually the case.

Theorem 4.1. *Algorithm 4.2 from Lawson and Hanson (1995) solves the problem in Equation* (4.1).

Proof of Theorem 4.1. Let Algorithm 4.2 solve the problem in Equation (4.2). Denote by $\mathbf{w} = -\nabla f(\mathbf{s}) = \bar{\mathbf{X}}^\top(\bar{\mathbf{x}} - \bar{\mathbf{X}}\mathbf{s})$ the negative gradient with respect to $\mathbf{s}$. The points $\{\ \bar{\mathbf{x}}_i \mid i \in \mathcal{P}\ \}$ involved in the solution are selected via

$$k = \arg\max_j \left\{\ w_j \mid j \in \mathcal{Z}\ \right\} = \arg\max_j \left\{\ \left[\bar{\mathbf{X}}^\top(\bar{\mathbf{x}} - \bar{\mathbf{X}}\mathbf{s})\right]_j \mid j \in \mathcal{Z}\ \right\}$$
$$= \arg\max_j \left\{\ \bar{\mathbf{x}}_j^\top \bar{\mathbf{x}} - \sum_{i=1}^{n} s_i \cdot \bar{\mathbf{x}}_i^\top \bar{\mathbf{x}}_j \mid j \in \mathcal{Z}\ \right\}.$$

Lemma 3.1 assures that those points belong to the frame. Since there is no other way of selecting points, all of them belong to the frame. Hence, the condition $s_i \neq 0 \Rightarrow \mathbf{x}_i \in \mathcal{F}$ is satisfied. The claim follows with Proposition 4.1. □

The following proposition from Lawson and Hanson (1995) characterizes the solution of this method.

Proposition 4.2 (Kuhn-Tucker conditions)**.** *A vector $\mathbf{s} \in \mathbb{R}^n$ is a solution for $f(\mathbf{s}) = \frac{1}{2}\|\bar{\mathbf{X}}\mathbf{s} - \bar{\mathbf{x}}\|_2^2$ if and only if there exists a vector $\mathbf{w} \in \mathbb{R}^n$ and a partitioning of the integers from 1 to $d+1$ into subsets $\mathcal{Z}$ and $\mathcal{P}$ such that with $\mathbf{w} = -\nabla f(\mathbf{s}) = \bar{\mathbf{X}}^\top(\mathbf{x} - \bar{\mathbf{X}}\mathbf{s})$ it holds that*

$$s_i = 0 \quad \text{for } i \in \mathcal{Z}, \qquad s_i > 0 \quad \text{for } i \in \mathcal{P},$$
$$w_i \leq 0 \ \text{for } i \in \mathcal{Z}, \qquad w_i = 0 \ \text{for } i \in \mathcal{P}.$$

Proposition 3.1 states that no more than $d+1$ extreme points are needed to define a point. In addition, Algorithm 4.2 chooses iteratively one extreme point after another. It is guaranteed to lower the error in each iteration (Lawson and Hanson, 1995) it terminates with at most $d+1$ non-negative positions and yields a sparse convex combination.

4.1.2 Computing the Frame

Until now, we represented a single point as a sparse convex combination of points on the frame. By doing this for every point $\mathbf{x}_i$ ($i = 1, 2, \ldots, n$) in the data set, we obtain the frame $\mathcal{F}$ of $\mathcal{X}$. Algorithm 4.3 summarizes this procedure, called *NNLS-Frame*, and Corollary 4.1 proves this claim.

Corollary 4.1. *Algorithm 4.3 yields the frame $\mathcal{F}$ of $\mathcal{X}$.*

Algorithm 4.3 NNLS-Frame

Input: design matrix $\bar{\mathbf{X}}$
Output: indices of ext. points $\mathcal{E}$ and weight matrix $\mathbf{W}$
$\mathcal{E} = \emptyset$
$\mathbf{W} = \mathbf{0} \in \mathbb{R}^{n \times n}$
for $i = 1, 2, \ldots, n$ **do**
  $\mathbf{s}_i = \text{NNLS}(\bar{\mathbf{X}}, \bar{\mathbf{x}}_i)$
  $\mathcal{P}_i = \{\, j \in \{1, 2, \ldots, n\} \mid (\mathbf{s}_i)_j > 0 \,\}$
  $\mathcal{E} = \mathcal{E} \cup \mathcal{P}_k$
  $\mathbf{W}[i, :] = \mathbf{s}_k$
end for

Proof. Theorem 4.1 ensures that the positive positions of every solution s_i ($i = 1, 2, \ldots, n$) refers to points on the frame. Taking the union of those positions recovers the frame indices $\mathcal{E}$ since every extreme point defines itself, and therefore its index is part of the union. □

Note that Algorithm 4.3 already realizes a matrix factorization for the special case $k = q$. This is also stated in the following corollary.

Corollary 4.2. *Let q be the frame size, $k = q$ and $\mathcal{E}$ and $\mathbf{W}$ be the result of Algorithm 4.3. Then it holds that*

(i) $\mathbf{X} = \mathbf{X}\mathbf{W} = \mathbf{X}[:, \mathcal{E}]\mathbf{W}[\mathcal{E}, :]$ *is a factorization of the design matrix $\mathbf{X}$ with $\mathbf{W}[\mathcal{E}, :] \in \mathbb{R}^{k \times n}$ being a non-negative stochastic matrix and $\mathbf{X}[:, \mathcal{E}] \in \mathbb{R}^{d \times k}$ being the submatrix of $\mathbf{X}$ containing only the frame of $\mathcal{X}$.*

(ii) the factorization above is lossless, i.e.,

$$\|\mathbf{X} - \mathbf{X}[:, \mathcal{E}]\mathbf{W}[\mathcal{E}, :]\|_F^2 = 0.$$

4.1.3 Complexity Analysis

There are two main computations within the NNLS method in Algorithm 4.2. First, the negative gradient is being computed, which has a complexity of $\mathcal{O}(n(d+1))$. Second, an unconstrained least-squares problem $\mathbf{z} = \arg\min_{\mathbf{z}} \|\bar{\mathbf{X}}_{\mathcal{P}}\mathbf{z} - \bar{\mathbf{x}}\|_2^2$ is being solved. The latter can be rewritten as $\mathbf{z}[\mathcal{P}] = (\bar{\mathbf{X}}_{\mathcal{P}}^\top \bar{\mathbf{X}}_{\mathcal{P}})^{-1} \bar{\mathbf{X}}_{\mathcal{P}}^\top \bar{\mathbf{x}}$ and has a complexity of $\mathcal{O}((d+1) \cdot |\mathcal{P}|^2)$, where the average size of $\mathcal{P}$ is $\frac{1}{2}(d+1)$. Since no more than $(d+1)$ points are sufficient for the solution (compare Theorem 4.1), the `while`-loop is being executed at most $(d+1)$ times. Therefore, the complexity of the NNLS method is $\mathcal{O}(\text{NNLS}) = \mathcal{O}((d+1)[\frac{1}{4}(d+1)^3 + n(d+1)])$ on average. The complexity of NNLS-Frame presented in Algorithm 4.3 is $\mathcal{O}(n \cdot \text{NNLS}) = \mathcal{O}(\frac{n}{4}(d+1)^4 + n^2(d+1)^2)$. The multiplier is less than n since points that are already known to be extreme can be omitted.

Algorithm 4.4 Divide and conquer strategy of NNLS-Frame

Input: data $\mathcal{X}$, number of splits $I \in \mathbb{N}$
Output: indices of extreme points $\mathcal{E}$
$\mathcal{X} = \bigcup_{\iota=1}^{I} \mathcal{X}^{(\iota)}$ with $\mathcal{X}^{(i)} \cap \mathcal{X}^{(j)} = \emptyset$ for $i \neq j$
$\mathcal{E} = \emptyset$
for $\iota = 1, 2, \ldots, I$ **do**
  $\mathcal{E}_k = \text{NNLS-Frame}(\mathcal{X}^{(\iota)})$
  $\mathcal{E} = \mathcal{E} \cup \mathcal{E}_\iota$
end for
$\mathcal{E} = \text{NNLS-Frame}(\mathcal{X}_\mathcal{E})$

In contrast, the fastest LP-based approach (Dulá and López, 2012) so far scales in $\mathcal{O}(n \cdot \text{LP}_{d+1,|\mathcal{F}|} + dn|\mathcal{F}|)$, where $\text{LP}_{r,t}$ is the runtime of a LP with r equality constraints and t non-negative variables.

4.1.4 Computing the Frame Efficiently

One way to speed up the frame computation is a divide and conquer approach. The underlying principle is stated in the following lemma.

Lemma 4.1. *Let $\mathcal{A}$ and $\mathcal{B}$ be non-empty discrete sets, then*

$$\text{conv}(\mathcal{A} \cup \mathcal{B}) = \text{conv}\big(\text{conv}(\mathcal{A}) \cup \text{conv}(\mathcal{B})\big).$$

The idea is as follows. Let $\mathcal{X}^{(1)} \cup \ldots \cup \mathcal{X}^{(I)}$ be a random partition of $\mathcal{X}$. The size n_ι of every subset $\mathcal{X}^{(\iota)}$ should be significantly smaller than the size of $\mathcal{X}$, *i.e.*, $n_\iota \ll n$. The assumption of having a pairwise disjunction is not necessary but reasonable. Instead of the whole data set $\mathcal{X}$, Algorithm 4.3 is now executed on every subset $\mathcal{X}^{(\iota)}$ for $\iota = 1, 2, \ldots, I$. Finally, the frame of $\mathcal{X}$ is obtained by merging the frames of every subset and run the proposed approach again on it. The procedure is summarized in Algorithm 4.4. Besides, the `for`-loops of Algorithms 4.1, 4.3 and 4.4 can be trivially parallelized.

4.2 Experiments

4.2.1 Computing the Frame

For the experiments in this section, we want to control the frame density. Hence, use the same synthetic data[1] as in Dulá and López (2012), which was generated according to a procedure described in Lopez (2005). The data consists of $n = 2500, 5000, 7500, 10000$ data points in $d = 5, 10, 15, 20$ dimensions with a frame density of $1, 15, 25, 50, 75$ percent respectively.

[1] `http://www.people.vcu.edu/~jdula/FramesAlgorithms/`

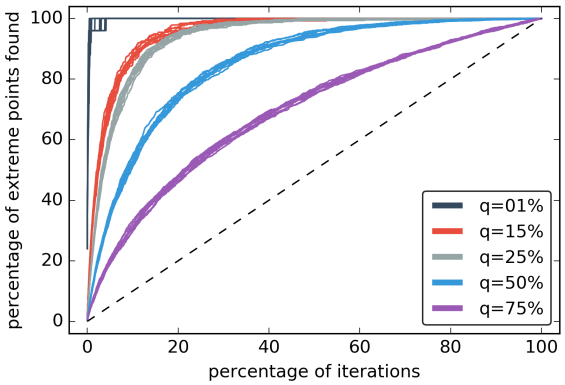

Figure 4.2: Evolution of the number of discovered frame points. The lower the frame density the faster the frame is being discovered.

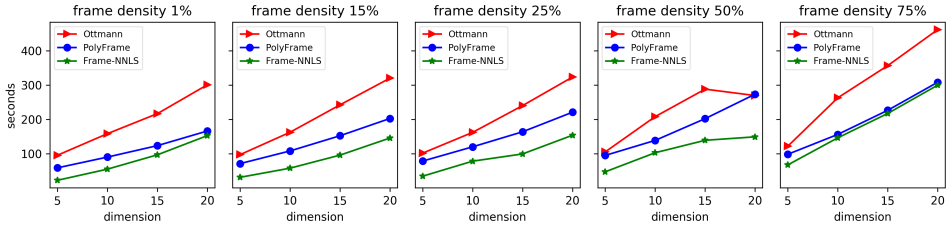

Figure 4.3: Timing results of computing the frame for various frame densities.

The first experiment is about the discovery of extreme points. In contrast to LP-based approaches (Dulá and Helgason, 1996; Ottmann et al., 2001), which discover up to one extreme point per iteration, NNLS-Frame finds up to $d + 1$ extreme points. This is a result of Theorem 4.1 and illustrated in Figure 4.2. The graph shows the percentage of discovered extreme points against the percentage of iterations conducted on a synthetic data set with $n = 2500$ points in 5 dimensions with various frame densities. The lower the frame density, the faster the frame is being discovered. Even for a fairly high frame density of 75%, the discovery is faster than approximately linear as one could expect from an LP-based approach. This is depicted as a dashed line. Note that if an approximation of the frame is sufficient, the computation could be stopped earlier.

In our second experiment, we show that finding the frame is faster than using LP-based approaches. The two baselines are taken from Ottmann et al. (2001) and Dulá and Helgason (1996). We use the implementation by Dulá and López (2012) and implement our approach in the same programming language for compatibility. We test on data sets of sizes $n = 2500, 5000, 7500, 10000$ and of dimensionality $d = 5, 10, 15, 20$. Figure 4.3 shows timing results for $n = 10000$. The figure shows that our approach is always faster than the baselines, irrespectively of the configuration.

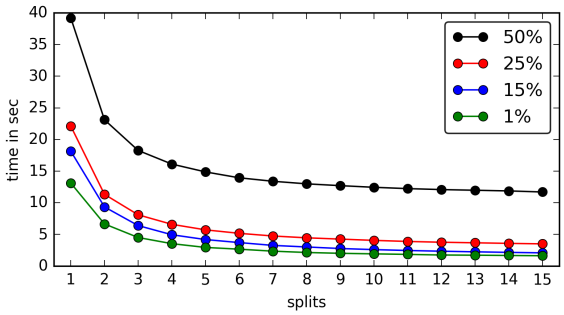

Figure 4.4: Timing results for the divide and conquer approach.

The third experiment is about the divide and conquer approach introduced in Section 4.1.4. As shown in Figure 4.4, the runtime is halved for every configuration using only three partitions. The take-home message is that even a small number of partitions can lead to a substantial speed-up. Note that those partitions can be processed in parallel.

Table 4.1: A description of the data sets and their properties including their frame sizes q, and frame densities q/n.

Data set		n	d	q	Frame density
Banking2	(Dulá and López, 2012)	12456	8	715	5.74%
Banking1	(Dulá and López, 2012)	4971	7	345	6.94%
USAFSurvey	(Epifanio et al., 2013)	2420	6	368	15.21%
yeast	(Horton and Nakai, 1996)	1484	8	242	16.31%
Banking3	(Dulá and López, 2012)	19939	11	4960	24.88%
SpanishSurvey	(Vinué, 2017)	600	5	150	25.00%
swiss-heads	(Cutler and Breiman, 1994)	200	6	115	57.50%
skel2	(Heinz et al., 2003)	507	10	431	85.01%
ozone	(Cutler and Breiman, 1994)	330	10	308	93.33%

4.2.2 Matrix Factorization

We now compare the frame-based archetypal analysis, Frame-AA (F-AA), to several baselines, including standard archetypal analysis (Cutler and Breiman, 1994) (AA), ConvexHull-NMF (Thurau et al., 2011) (CH-NMF), Convex-NMF (Ding et al., 2010) (C-NMF) and standard NMF (Lee and Seung, 1999). The first two methods are implemented in Python. For CH-NMF and C-NMF we use *pymf*[2], and for NMF we use *scikit-learn* (Pedregosa et al., 2011). Table 4.1 depicts the data sets used for this experiment. The

[2] `http://pypi.python.org/pypi/PyMF`

Table 4.2: Average Frobenius norm reported on 36 repetitions with random initializations for $k = 6$ archetypes. The symbol "-" indicates that the method could not be executed due to negative data.

Data set	F-AA	AA	CH-NMF	C-NMF	NMF
Banking2	90.08	72.35	-	-	-
Banking1	67.20	58.97	-	-	-
USAFSurvey	904.22	902.07	1239.67	2192.30	688.35
yeast	5.43	5.02	9.18	7.04	3.52
Banking3	134.30	131.81	-	-	-
SpanishSurvey	94.84	93.51	117.91	254.92	32.14
swiss-heads	75.05	74.67	87.06	116.07	47.16
skel2	64.84	64.87	77.78	101.73	51.75
ozone	1532.12	1669.70	-	-	-

frame sizes and densities are computed with our proposed NNLS-Frame method.

Table 4.2 reports on the results for the choice of $k = 6$ in terms of reconstruction error measured with the Frobenius norm. We obtain similar results for $k = 8, 10, 12$. The number of iterations executed per algorithm is fixed to 100 to obtain fair results. We use random initializations for all algorithms and report on averages over 36 repetitions. As seen in Table 4.2, F-AA yields similar errors as AA. The lowest errors are achieved with NMF. The baseline CH-NMF, which approximates the frame instead of computing it exactly, yields a higher error of approximately 20%, while C-NMF is even worse. In summary, the error of F-AA is similar to standard AA and much better than CH-NMF and C-NMF.

Usually, it is a-priori not known how to choose the latent dimensionality k. A standard approach is to choose k with respect to the so-called elbow criterion, which requires several runs for different values of k to be executed. In such a scenario, our approximative approach is particularly beneficial. Frame-AA requires the computation of the frame $\mathcal{F}$ before archetypes can be located. However, once the frame is complete, it can be used for finding any number of archetypes. Hence, it is interesting to see the cumulative time taken by the methods when evaluating several configurations, say $k = 4, 6, 8, \ldots, 16$.

This is depicted in Figure 4.5 for the USAFSurvey data. We report again on averages as well as standard errors on 36 repetitions executed on random initializations. Frame-AA turns out fastest at the first evaluation of $k = 4$ despite the computation of the frame. Since the frame is static for a data set, it can be reused for all remaining computations, and Frame-AA clearly outperforms its peers. Note that the divide and conquer strategy proposed

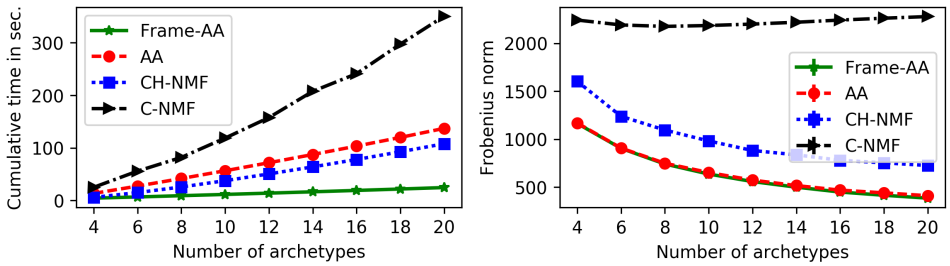

Figure 4.5: Results on the USAFSurvey data. Cumulative time for computing the frame and the factorization (left) and the reconstruction error in terms of the Frobenius norm (right) for various numbers of archetypes k.

in Algorithm 4.4 leads to an additional speed-up that is not shown here. The reconstruction error is shown on the right-hand side of Figure 4.5. Once again, Frame-AA yields a very similar error as standard AA, computed on the whole data set, while the C-NMF and CH-NMF baselines perform much worse.

4.3 Autoencoders

We consider a prototypical application of Frame-AA as an autoencoder similar to the approach of Bauckhage et al. (2015). However, instead of focusing on the reconstruction, we are interested in the quality of the learned embedding given by the weights of the convex combinations.

For this exemplary task, we use the MNIST data set (LeCun et al., 2010). It consists of 60,000 training and 10,000 test images of handwritten digits (0-9), which are of size 28×28, hence rendering the problem 784-dimensional. To lower the frame density, we split every image into 4×4 sub-images yielding 49 patches per image. An example is depicted in Figure 4.6. Those patches are stacked up to form a new design matrix. For simplicity, we focus on a subtask and aim to separate digit 1 from digit 7. The task is thus phrased as a standard binary classification task. We sample 500 images per class. The final design matrix is of size 49000×16, and its frame density is approximately 14%. The design matrix is then factorized as described in Section 4.1. After the factorization, every patch is described by k weights contained within the matrix $\mathbf{A}$. The weights of the convex combinations are then vectorized, yielding an embedding of size $m = 49k$ per image.

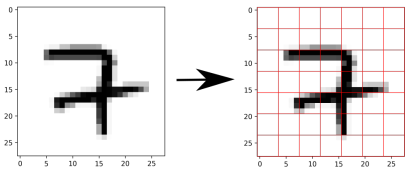

Figure 4.6: An example of the subdivision of MNIST data into 49 patches.

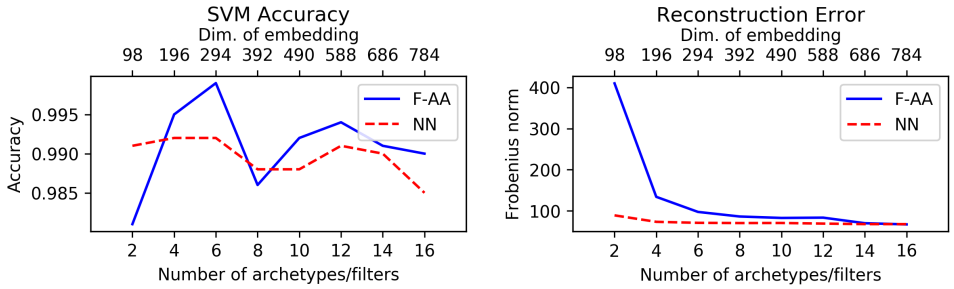

Figure 4.7: Predictive performance in terms of accuracy (left) and the reconstruction error in terms of the Frobenius norm (right) for various numbers of archetypes / embedding sizes.

As a baseline, we employ an autoencoder that uses a convolutional layer (LeCun et al., 1989) to learn intermediate representations of images. To perform a fair comparison, the convolutional layer learns k filters of the same size as Frame-AA, that is, 4×4 and stride 4×4. This layer has a sigmoid activation function followed by an upsampling layer that recovers the original image size. The last layer is another convolutional layer with one filter of size 4×4, stride one, and sigmoid activation. We optimize the network using a squared loss as in the matrix factorization. We compute an embedding analogously to Frame-AA by representing an image by its output of the first convolutional layer, yielding the same embedding size of $m = 49k$.

For every obtained embedding, a support vector machine (SVM) (Boser et al., 1992; Cortes and Vapnik, 1995) is trained on $k = 2, 4, \ldots, 16$ filters (archetypes), meaning that every image is now represented by a vector of size $m = 98, 196, \ldots, 784$. Note that the last one is identical to the original image size. We deploy a radial basis function (RBF) kernel, and the parameters C and γ are optimized on a $2^{-7}, \ldots, 2^{7}$ grid, respectively. For the hyperparameter search, we use a hold-out set consisting of another 500 images per class that are disjoint from the training and test sets.

To evaluate the predictive performance, we sample another 500 test images per class and obtain their embeddings using the trained approaches described above. The results are depicted in Figure 4.7. The right part of the figure depicts the reconstruction errors in terms of the Frobenius norm. The curve of Frame-AA has a typical elbow structure, as one would expect, while the error of the neural network is almost static. Although the autoencoder achieves smaller reconstruction errors, the left side of Figure 4.7 shows that the SVM effectively leverages the representation learned by F-AA. The archetype-based embedding leads to more accurate classification models.

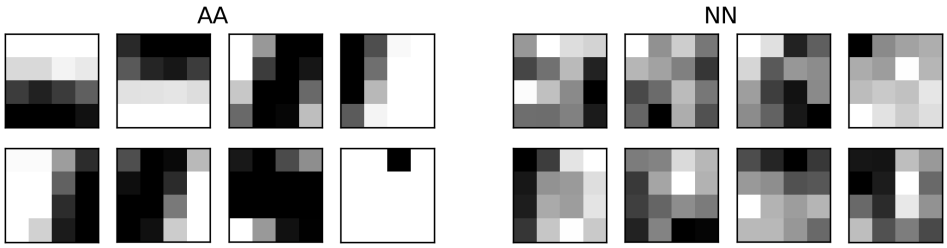

Figure 4.8: A visualization of the learned archetypes (left) and filters (right).

The learned filters (archetypes) are shown in Figure 4.8 for $k = 8$, yielding an embedding of size $m = 392$. The archetypes in the left part of the figure resemble edge and border detectors, while the filters on the right appear random. Moreover, every patch is represented as a convex combination of the shown archetypes. The classification of a patch can be explained by the corresponding weights to render the results interpretable.

4.4 Conclusion

We proposed a novel method for computing the frame of a data set. The frame is a subset of the data set containing only the extreme points, *i.e.*, the vertices of the convex hull of the data. While standard approaches like QuickHull were infeasible for scenarios with more than three dimensions, we computed the frame by leveraging the well known active-set method for non-negative least squares problems called NNLS. We provided a theoretical underpinning for our approach and conducted a series of experiments to compare the computation of the frame with two LP-based methods to show our competitiveness.

Moreover, we proposed an approximation of archetypal analysis called Frame-AA by restricting the optimization of the archetypes to the frame. We showed that if a small number of data points approximated the frame sufficiently, the resulting approximation of AA on the whole data set was appropriate. This approximation does not only work for archetypal analysis but also for all variants of it, like, for instance, Mørup and Hansen (2010), Chen et al. (2014), and Bauckhage et al. (2015).

Empirically, we showed that the error of Frame-AA is only slightly higher than standard archetypal analysis but was often much faster once the frame has been computed. We leveraged this observation and proposed an efficient model selection strategy to find the optimal number of archetypes. We proposed to compute the frame only once and then reused it for all subsequent computations for other numbers of archetypes. We observed an enormous acceleration in scenarios with low frame densities.

Low frame densities were attained when only small portions of the data were extreme. On the other hand, high frame densities diminish this speed-up and the computational time converges to that of a standard archetypal analysis. However, we argued that these scenarios are not suited for archetypal analysis in the first case.

Furthermore, we compared Frame-AA to several state-of-the-art baselines. Frame-AA either outperformed its competitors or performed on-par. On the example of an autoencoder, Frame-AA led to nicely interpretable classifications in contrast to convolutional neural networks.

Chapter 5

Frame-based Optimal Design

In the previous chapter, we exploited the geometry of the data and proposed to use the frame as a representative subset for archetypal analysis. We now consider another setting in which the data boundary, or frame, contains enough information for the learning task at hand. Specifically, we are concerned with the setting of optimal experimental design (OED) (Fedorov, 1972). In a nutshell, the idea of OED is as follows. We consider a supervised learning task with n unlabeled data points $\mathcal{X}$. Obtaining labels for all instances is assumed to be costly or time-consuming, thus prohibitive. Still, there is a budget k allowing a small portion of $k \ll n$ points to be labeled. The goal is to select the best subset of $\mathcal{X}$ of size k for labeling such that the learned model is optimal with respect to some optimality measure, which quantifies the uncertainty of the model. This selection process can be formulated as an optimization problem over n data points. Reducing the search space by providing a representative subset thus reduces the computational complexity of the selection procedure. Figure 5.1 shows an example. Choosing points from the data boundary, which is the frame, obtaining labels for them, and training the linear regression model on those points is better than choosing an arbitrary subset of data.

The key contributions of this chapter are as follows. We (i) show that restricting the optimization problem to the frame yields competitive results in terms of optimality and predictive performance but comes with a much smaller computational cost. To leverage OED for non-linear problems, we (ii) devise a novel approach to compute the frame in kernel-induced feature spaces; this allows us to sample random designs for non-linear regression models without knowing the explicit feature mapping. Our approach of computing the frame can be seen as a transposed LASSO that selects data points instead of features. We also (iii) discuss the relation to LASSO (Tibshirani, 1996) in greater detail and address the connection to active learning. Finally, we (iv) empirically verify the efficiency of our frame-based approach to OED.

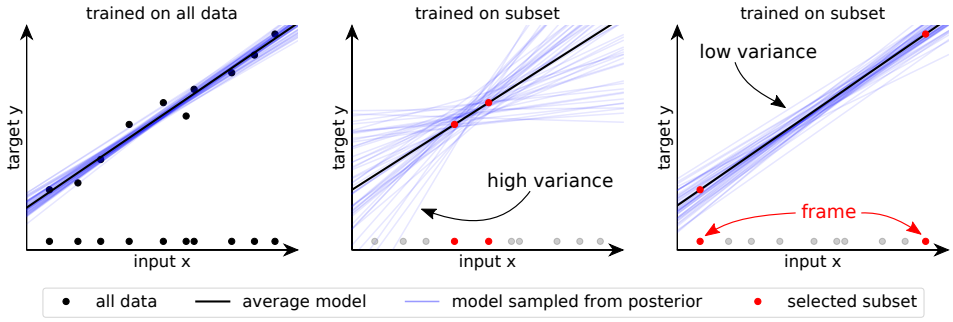

Figure 5.1: Illustration of using different subsets for training. Left: model trained on all data. Center: model trained on random subset. Right: model trained on the frame which serves as a representative subset.

5.1 Preliminaries and Optimal Experimental Design

We consider a discrete input set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_n\} \subseteq \mathbb{R}^d$ consisting of n data points in d dimensions. As introduced in Chapter 3, $\mathcal{F}$ denotes the so-called frame of $\mathcal{X}$ (*cf.* Definition 3.1), $q = |\mathcal{F}|$ is the size of the frame, and we call the portion of points in $\mathcal{X}$ belonging to the frame $\mathcal{F}$ the frame density q/n.

In the classical setting of optimal experimental design (OED), the task is a linear regression model

$$\mathbf{y} = \mathbf{X}\mathbf{w} + \boldsymbol{\varepsilon}, \tag{5.1}$$

where $\mathbf{y}$ is the vector of targets $y_i \in \mathbb{R}$, the design matrix $\mathbf{X} \in \mathbb{R}^{n \times d}$ denotes the pool of n experiments $\mathbf{x}_i \in \mathbb{R}^d$, the model parameters are given by $\mathbf{w} \in \mathbb{R}^d$, and $\boldsymbol{\varepsilon} \sim \mathcal{N}(\mathbf{0}, \sigma^2 \boldsymbol{I})$ is a vector of i.i.d. Gaussian noise. The maximum likelihood estimate of the parameters $\mathbf{w}$ has a closed-form and is given by

$$\mathbf{w}^\star = \arg\min_{\mathbf{w}} \sum_{i=1}^{n} (\mathbf{w}^\top \mathbf{x}_i - y_i)^2 = (\mathbf{X}^\top \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{y}.$$

The goal of OED is to choose a subset $\mathcal{S}$ of size k out of the n points for which the estimation of $\mathbf{w}$ is optimal in some sense. As common, we require $d \leq k \ll n$. Optimality can be measured in several ways. One idea is to increase the confidence of learning the parameters by minimizing the covariance of the parameter estimation. For the regression problem stated above, the covariance matrix is given by

$$\mathrm{Cov}_{\mathcal{S}}[\mathbf{w}] = \sigma^2 \left(\sum_{\mathbf{x} \in \mathcal{S}} \mathbf{x}\mathbf{x}^\top \right)^{-1},$$

where $\mathcal{S} \subset \mathcal{X}$ is the selected subset with $|\mathcal{S}| = k$. The selection of $\mathcal{S}$ leads to a combinatorial optimization problem as follows

$$\begin{aligned} \min_{\lambda} \quad & \Lambda\left(\sum_{i=1}^{n} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top\right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \lambda_i \leq k \;\text{ and }\; \lambda_i \in \{0,1\} \;\; \forall i. \end{aligned} \tag{5.2}$$

Here, λ_i selects data points and $\Lambda : \mathbb{S}_d^+ \to \mathbb{R}$ is an optimality criterion that assigns a real number to every feasible experiment (positive semi-definite matrices $\mathbb{S}_d^+$). The setting can be seen as maximizing the information we obtain from executing the experiment with fixed effort. The most popular choices for Λ are D-, E-, and A-optimality (Pukelsheim, 2006), given by

$$\begin{aligned} \Lambda_D(\Sigma) &= (\det(\Sigma))^{-1/d}, && \text{(D-optimality)}, \\ \Lambda_E(\Sigma) &= \|\Sigma^{-1}\|_2, && \text{(E-optimality)}, \\ \Lambda_A(\Sigma) &= d^{-1} \operatorname{tr}(\Sigma^{-1}), && \text{(A-optimality)}. \end{aligned}$$

Unfortunately, the combinatorial optimization problem in Equation (5.2) cannot be solved efficiently. A remedy is to use a continuous relaxation, which is efficiently solvable:

$$\begin{aligned} \min_{\lambda} \quad & \Lambda\left(\sum_{i=1}^{n} \lambda_i \mathbf{x}_i \mathbf{x}_i^\top\right) \\ \text{s.t.} \quad & \sum_{i=1}^{n} \lambda_i \leq k \;\text{ and }\; \lambda_i \in [0,1] \;\; \forall i. \end{aligned} \tag{5.3}$$

The following lemma characterizes the solution of the optimization problem stated above.

Lemma 5.1. *Let $\boldsymbol{\lambda}^\star$ be the optimal solution of Problem* (5.3)*. Then $\|\boldsymbol{\lambda}^\star\|_1 = k$.*

However, the support of $\boldsymbol{\lambda}^\star$ is usually much larger than k, and the solution needs to be sparsified to end up with k experiments. Approaches therefore include pipage rounding schemes (Ageev and Sviridenko, 2004), sampling (Wang et al., 2017), regret minimization (Allen-Zhu et al., 2017), and greedy removal strategies (Mariet and Sra, 2017; Wang et al., 2017).

5.2 Restricting OED to the Frame

As introduced above, a D-optimal design minimizes the determinant of the error covariance matrix. Its dual problem is known as minimum volume confidence ellipsoid (Dolia et al., 2006). Geometrically, the optimal solution is

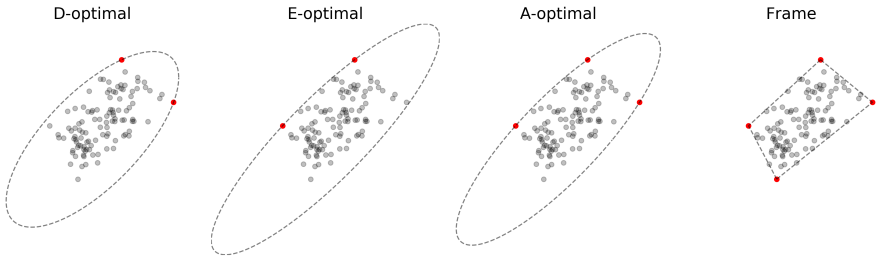

Figure 5.2: An example of D-, E-, A-optimal designs and the frame. The support of the optimal solution $\boldsymbol{\lambda}^\star$ is highlighted in red. The dashed ellipsoids denote the dual problems. Note that the support of $\boldsymbol{\lambda}^\star$ is at the boundary of the corresponding ellipsoid.

an ellipsoid that encloses the data with minimum volume. For E-optimality, the dual problem can be interpreted as minimizing the diameter of the confidence ellipsoid (Boyd and Vandenberghe, 2004). In A-optimal designs, the goal is to find the subset of points that optimizes the total variance of parameter estimation. Figure 5.2 depicts the support of the optimal solution $\boldsymbol{\lambda}^\star$ for D-, E-, and A-optimal designs as well as their confidence ellipsoids derived from their dual problems. The right-hand figure shows the frame of the same data. The confidence ellipsoids clearly touch the points at the border of the data while the interior points are enclosed. Hence, we propose to discard all interior points entirely in the optimization and restrict the optimization to the frame, that is, to the points lying on the border of the convex hull.

Non-linear regression can be done by applying a feature mapping $\phi : \mathcal{X} \to \mathcal{X}'$ to the data. The model then becomes $y_i = \mathbf{w}^\top \phi(\mathbf{x}_i)$, which is still linear in parameters. Considering the dual of the regression problem, we can employ kernels that implicitly do a feature mapping. However, the regression is still a linear model, but in feature space $\mathcal{X}'$. Knowing the frame in $\mathcal{X}'$ would allow us to sample random designs rendering a naive version of non-linear or kernelized OED possible. We already showed how to compute the frame for $\mathcal{X}$ in Section 4.1.1 and focus now on how to compute the frame in kernel-induced feature spaces $\mathcal{X}'$.

Computing the Frame in Kernel-induced Feature Spaces

Let $\phi : \mathbb{R}^d \to \mathbb{R}^D$ be a feature mapping, $\Phi \in \mathbb{R}^{D \times n}$ be the mapped design matrix, and $\mathbf{K}$ be the kernel matrix induced by a kernel $k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$. As before, the idea is to solve a linear system subject to the constraints that the solution $\mathbf{s}$ is non-negative, sums up to one, and uses only points from the frame; however, this time, we aim to solve the problem in feature space

spanned by ϕ. We obtain

$$\begin{aligned} \underset{\mathbf{s}}{\text{solve}} \quad & \Phi\mathbf{s} = \phi(\mathbf{x}_i) \\ \text{s.t.} \quad & s_j \geq 0 \; \forall j \; \wedge \; \mathbf{1}^\top \mathbf{s} = 1 \; \wedge \; s_j \neq 0 \Rightarrow \phi(\mathbf{x}_j) \in \mathcal{F} \quad \forall j. \end{aligned}$$

The constraint $\mathbf{1}^\top \mathbf{s} = 1$ can be incorporated into the system of linear equations by augmenting Φ with a row of ones and $\phi(\mathbf{x})$ with a static 1. Let $\psi(\mathbf{x}) = (\phi(\mathbf{x})^\top, 1)^\top$ and $\Psi = (\psi(\mathbf{x}_1), \ldots, \psi(\mathbf{x}_n)) \in \mathbb{R}^{(D+1)\times n}$, we obtain

$$\text{solve } \Psi\mathbf{s} = \psi(\mathbf{x}_i) \quad \text{s.t.} \quad s_j \geq 0 \; \wedge \; s_j \neq 0 \Rightarrow \phi(\mathbf{x}_j) \in \mathcal{F}.$$

The approach can be kernelized by multiplying from the left with $\Psi^\top$:

$$\text{solve } \Psi^\top\Psi\mathbf{s} = \Psi^\top\psi(\mathbf{x}_i) \quad \text{s.t.} \quad s_j \geq 0 \; \wedge \; s_j \neq 0 \Rightarrow \phi(\mathbf{x}_j) \in \mathcal{F}.$$

Since there is always a solution (*cf.* Chapter 4), we can equivalently solve the non-negative least squares problem

$$\begin{aligned} \underset{\mathbf{s}\geq 0}{\arg\min} \quad & \frac{1}{2}\|\Psi^\top\Psi\mathbf{s} - \Psi^\top\psi(\mathbf{x}_i)\|_2^2 \\ \text{s.t.} \quad & s_j \neq 0 \Rightarrow \phi(\mathbf{x}_j) \in \mathcal{F}. \end{aligned} \tag{5.4}$$

A kernel can now be applied by exploiting the relationship between Ψ, Φ, and $\mathbf{K}$ as follows

$$\Psi^\top\Psi = \Phi^\top\Phi + \mathbb{1}_{nn} = \mathbf{K} + \mathbb{1}_{nn} =: \mathbf{L} \tag{5.5}$$

$$\Psi^\top\psi(\mathbf{x}_i) = \Phi^\top\phi(\mathbf{x}_i) + \mathbb{1}_{n1} = \mathbf{K}_{\cdot i} + \mathbb{1}_{n1} = \mathbf{L}_{\cdot i}, \tag{5.6}$$

where $\mathbb{1}_{nm} \in \mathbb{R}^{n\times m}$ denotes the matrix of ones. The resulting problem becomes

$$\begin{aligned} \underset{\mathbf{s}\geq 0}{\arg\min} \quad & \frac{1}{2}\|\mathbf{L}\mathbf{s} - \mathbf{L}_{\cdot i}\|_2^2 \\ \text{s.t.} \quad & s_i \neq 0 \Rightarrow \phi(\mathbf{x}_i) \in \mathcal{F}. \end{aligned} \tag{5.7}$$

A standard non-negative least squares (NNLS) problem can be solved, for example, by the algorithm of Lawson and Hanson (1995). Bro and De Jong (1997) increase the efficiency of NNLS (*cf.* Algorithm 4.2) by caching the quantities in Equation (5.5). This renders the problem in Equation (5.7) feasible.

Theorem 5.1. *The active-set method from Bro and De Jong (1997) solves the problem in Equation* (5.7).

Proof. The algorithm of Bro and De Jong (1997) selects points that contribute to the solution $\mathbf{s}$ by maximizing the negative gradient of the objective. The

Algorithm 5.1 Kernel-Frame

Input: kernel matrix $\mathbf{K}$
Output: indices of frame points $\mathcal{E} = \{\, i \in \{1, 2, \ldots, n\} \mid \mathbf{x}_i \in \mathcal{F} \,\}$
$\mathbf{L} = \mathbf{K} + \mathbb{1}_{nn}$
$\mathcal{E} = \emptyset$
for $i = 1, 2, \ldots, n$ **do**
  $\mathbf{s}_i = \text{bro-dejong}(\mathbf{L}, \mathbf{L}[:, i])$
  $\mathcal{P}_i = \{\, j \in \{1, 2, \ldots, n\} \mid (\mathbf{s}_i)_j > 0 \,\}$
  $\mathcal{E} = \mathcal{E} \cup \mathcal{P}_i$
end for

selection is implemented by the criterion $j = \arg\max_j \; [\mathbf{L}_{\cdot i} - \mathbf{L}\mathbf{s}]_j$, where j is the index of the selected point. Thus, the selection process is maximizing a linear function and Lemma 3.1 assures that this point belongs to the frame. □

Solving problem (5.4) for the i-th data point $\mathbf{x}_i$ yields either its index i in case $\mathbf{x}_i$ is a point on the frame or, if $\mathbf{x}_i$ is an interior point, the solution is the index set $\mathcal{P}_i$ of points on the frame that recover $\mathbf{x}_i$ as a convex combination. The entire frame is recovered by solving Equation (5.4) for all points in $\mathcal{X}$ as stated in Corollary 5.1 and depicted in Algorithm 5.1.

Corollary 5.1. *Let $k(\mathbf{x}, \mathbf{z}) = \phi(\mathbf{x})^\top \phi(\mathbf{z})$ be a kernel and $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ be a data set. Then Algorithm 5.1 yields the frame of $\{\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)\}$.*

Proof. Algorithm 5.1 computes the solution $\mathbf{s}_i$ for every mapped data point $\phi(\mathbf{x}_i)$. Theorem 5.1 ensures that the positive positions of every $\mathbf{s}_i$ ($i = 1, 2, \ldots, n$) refers to points on the frame. Hence, taking the union of those positions recovers the frame indices $\mathcal{E}$. □

Note that the frame in kernel-induced feature space can be found without knowing the explicit feature map ϕ. Having access to the kernel matrix is sufficient. Besides, the `for`-loop in Algorithm 5.1 can be trivially parallelized.

5.3 Frame Densities for Common Kernels

We focus on radial basis function (RBF) and polynomial kernels to analyze the frame sizes in kernel-induced feature spaces. The former is given by $k(\mathbf{x}, \mathbf{z}) = \exp(-\gamma \|\mathbf{x} - \mathbf{z}\|_2^2)$, where $\gamma > 0$ is a scaling parameter. The induced feature mapping ϕ of the RBF kernel has an infinite dimensionality. Corollary 5.2 shows that this kernel always yields a full frame. Thus, every point belongs to the frame, and the frame density is consequently equal to one.

Corollary 5.2. *Let $\mathcal{X}$ be the data set of distinct points and k be the RBF kernel with parameter $\gamma > 0$. Then every point belongs to the frame $\mathcal{F}$ in feature space.*

Proof. Gaussian gram matrices have full rank (Schölkopf and Smola, 2002). Hence, the images $\phi(\mathbf{x}_1), \ldots, \phi(\mathbf{x}_n)$ in feature space are linearly independent. Thus, every image can only be represented by itself and, every point belongs to the frame $\mathcal{F}$. □

The polynomial kernel is given by $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^\top \mathbf{z} + c)^p$ with degree $p \in \mathbb{N}$ and constant $c \in \mathbb{R}_0^+$. A feature of the polynomial kernel is an explicit representation of the implicit feature mapping ϕ. *E.g.*, for the homogeneous polynomial kernel with $c = 0$, we have

$$\phi_{\mathbf{m}}(\mathbf{x}) = \sqrt{\frac{p!}{\prod_{j=1}^d m_j!}} \prod_{j=1}^d x_j^{m_j}$$

for all multi-indices $\mathbf{m} = (m_1, \ldots, m_d) \in \mathbb{N}^d$ satisfying $\sum_{j=1}^d m_j = p$. That is, new features consist of monomials of the input features x_j, while the multi-indices $\mathbf{m}$ denote their respective degrees. The condition $\sum_{j=1}^d m_j = p$ assures that all possible combinations are uniquely accounted for and leads to a feature space dimension of size

$$D = \binom{p+d-1}{p} = \frac{(p+d-1)!}{p!(d-1)!}.$$

For the explicit mapping corresponding to the heterogeneous kernel (where $c \neq 0$) that realizes a feature space with dimensionality

$$D = \binom{p+d}{p},$$

as well as for more details, we refer to Smola et al. (1998) and Schölkopf and Smola (2002). For the polynomial kernel we obtain a full frame if the dimension D of the feature space exceeds the number of data points n.

Corollary 5.3. *Let $\mathcal{X}$ be the normalized and distinct data set of size n in d dimensions and k be the polynomial kernel with degree p and offset $c = 0$. If $n \leq \frac{(p+d-1)!}{p!(d-1)!}$, then every point belongs to the frame $\mathcal{F}$ in feature space.*

Proof. The polynomial feature map yields linearly independent feature vectors of size $\frac{(p+d-1)!}{p!(d-1)!}$ for a data set with unique observations. Hence, if the number of data points is lower than the dimensionality of the mapping, all points belong to the frame $\mathcal{F}$. □

Although a formal proof regarding the influence of the degree p of the homogeneous polynomial kernel is missing, we would like to provide some intuition. We empirically apply a homogeneous polynomial kernel to a synthetic data set with $n = 2500$ points in $d = 5$ dimensions with an initial

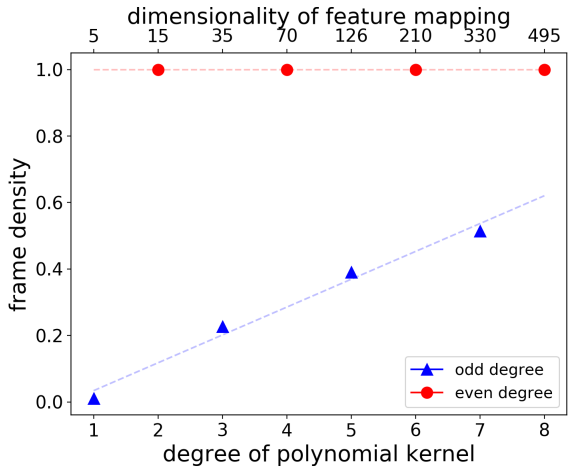

Figure 5.3: Frame density for various polynomial degrees on synthetic data of size $n = 2500$ in $d = 5$ dimensions. The initial frame density is 1%. The data set is introduced in Section 5.5.

frame density of 1%. Figure 5.3 shows the resulting frame densities. For odd degrees, the frame density is growing with increasing values of p. This is due to the increasing dimensionality in feature space. However, for even degrees, the frame is always full. We conclude with the following conjecture.

Conjecture 5.1. *Let $\mathcal{X}$ be the normalized and distinct data set of size n in d dimensions and k be the polynomial kernel with degree p and offset $c = 0$. If p is even, then every point belongs to the frame $\mathcal{F}$ in feature space.*

5.4 Computing the Frame and LASSO

LASSO (Tibshirani, 1996) solves regression tasks by combining a squared loss with an ℓ_1-regularizer on the parameters. Thus, LASSO simultaneously performs a regression and variable selection such that the influence of redundant variables is set to zero and a sparse parameter vector is obtained. The corresponding optimization problem for a regression scenario as in Equation (5.1) is given by

$$\min_{\mathbf{w}} \ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_1,$$

where $\lambda \geq 0$ is a trade-off parameter. A special case is obtained by restricting the parameters to be positive, yielding a *non-negative LASSO*:

$$\min_{\mathbf{w} \geq 0} \ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \|\mathbf{w}\|_1 \quad \Longleftrightarrow \quad \min_{\mathbf{w} \geq 0} \ \|\mathbf{X}\mathbf{w} - \mathbf{y}\|_2^2 + \lambda \mathbf{1}^\top \mathbf{w}.$$

Computing the frame can be seen as a transposed version of the LASSO problem in which not variables but data points are selected. The following proposition shows that the problem in Equation (5.7) is equivalent to a non-negative LASSO, if one ignores the constraint that elicits only frame points to contribute to the solution.

Proposition 5.1. *Problem* (5.7) *solved with the active-set method from Bro and De Jong (1997) is equivalent to a non-negative LASSO with trade-off parameter* $\lambda = n$.

Proof. By using the identities $\mathbf{L} = \mathbf{K} + \mathbb{1}_{nn}$ and $\mathbf{L}_{\cdot i} = \mathbf{K}_{\cdot i} + \mathbf{1} = \mathbf{k} + \mathbf{1}$, we rewrite the objective of the optimization problem in Equation (5.7) as follows:

$$\begin{aligned}\|\mathbf{L}\mathbf{s} - \mathbf{L}_{\cdot i}\|_2^2 &= \|(\mathbf{K} + \mathbb{1})\mathbf{s} - (\mathbf{k} + \mathbf{1})\|_2^2 = \|\mathbf{K}\mathbf{s} - \mathbf{k}\|_2^2 + \|\mathbb{1}\mathbf{s} - \mathbf{1}\|_2^2 \\ &= \|\mathbf{K}\mathbf{s} - \mathbf{k}\|_2^2 + n\|\mathbf{1}^\top\mathbf{s} - 1\|_2^2 = \|\mathbf{K}\mathbf{s} - \mathbf{k}\|_2^2 + n\mathbf{1}^\top\mathbf{s} - n \\ &\equiv \|\mathbf{K}\mathbf{s} - \mathbf{k}\|_2^2 + n\mathbf{1}^\top\mathbf{s} = \|\mathbf{K}\mathbf{s} - \mathbf{k}\|_2^2 + n\|\mathbf{s}\|_1.\end{aligned}$$

Hence, the objective is an ℓ_1-regularized least squares problem. In combination with the non-negativity constraint, we obtain a non-negative LASSO. □

5.5 Experiments

In this section, we empirically investigate frame-based optimal experimental design. Throughout this section, we compare the performance of the following different approaches. *Uniform-data* samples the subset S uniformly at random without replacement from all data points $\mathcal{X}$. A second approach *uniform-frame* uses the same strategy but samples points from the frame $\mathcal{F}$ instead of $\mathcal{X}$. If the size of $|S|$ exceeds the size of the frame, *uniform-frame* always draws the full frame and randomly selects the remaining points from the interior points $\mathcal{X} \setminus \mathcal{F}$. According to their contribution to the objective of D-optimal design, the *greedy* baseline chooses the points in S one after another. The baselines *{D,E,A}-optimal* use the continuous relaxations of the {D,E,A}-optimal design criteria, respectively. After solving the optimization problem, we sample the subset S according to a strategy outlined by Wang et al. (2017). Analogously, *{D,E,A}-optimal-frame* restricts the computation of the previous three baselines to the frame. Finally, the *Fedorov* baseline selects S according to the Fedorov Exchange algorithm (Fedorov, 1972) and optimizes D-optimality. We initialize this baseline using random samples from $\mathcal{X}$, random samples from $\mathcal{F}$, and with the output of *greedy*.

The continuous relaxations are optimized using sequential quadratic programming (Nocedal and Wright, 2006), and the number of iterations is limited to 250. We report on average performances over 100 repetitions; error

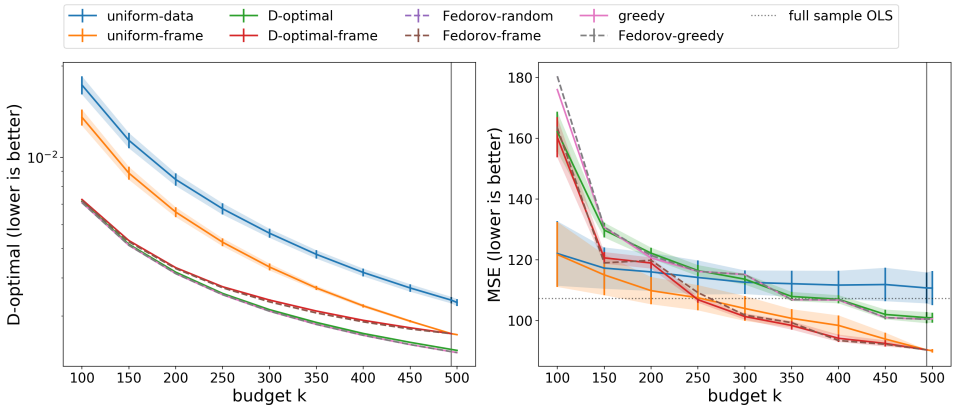

Figure 5.4: Results for D-optimal designs on Concrete data.

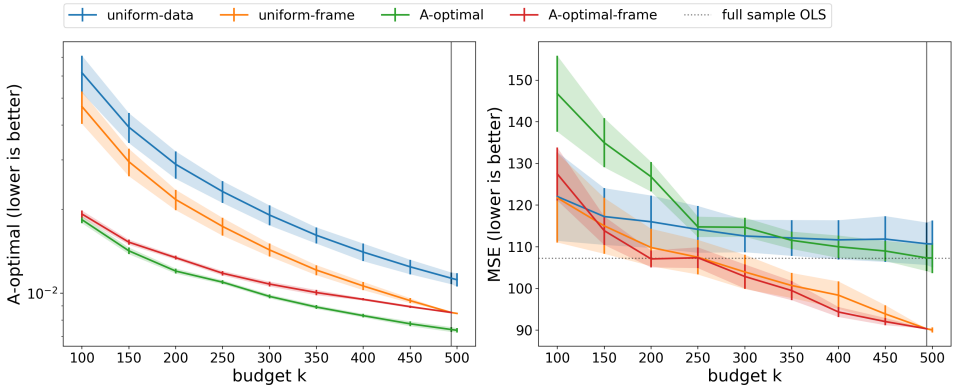

Figure 5.5: Results for A-optimal designs on Concrete data.

bars indicate one standard deviation, and a vertical line, when included, denotes the frame size. The greedy algorithm is executed once, and we conduct only ten repetitions for every Fedorov Exchange initialization due to its extensive runtime.

5.5.1 OED Restricted on the Frame

The first experiment studies the performance of optimal designs of the proposed approaches on the real-world data set Concrete (Yeh, 1998). Concrete consists of a design pool of $n = 1030$ instances with $d = 8$ dimensions and has a frame density of 48%. The task is to predict the compressive strength of different types of concrete.

We measure the performance in terms of the D-optimality criterion as well as the mean squared error (MSE), given by $\text{MSE} = \frac{1}{n}\|\mathbf{y} - \mathbf{X}\mathbf{w}\|_2^2$. For the latter, we train an ordinary least squares regression on the selected points and evaluate on the remaining $n - k$ points.

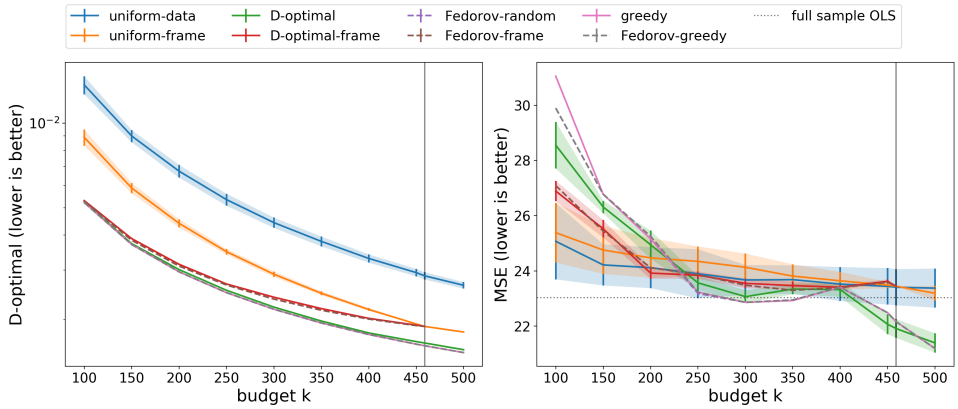

Figure 5.6: Results for D-optimal designs on Airfoil data.

Figure 5.4 (left) shows the results with respect to the D-optimality criterion. Sampling uniformly from the frame (*uniform-frame*) performs consistently better than sampling from all data (*uniform-data*). Thus, exploiting the frame allows to sample better designs without solving any optimization problem other than computing the frame. The situation changes once the designs are optimized in addition. Frame-based approaches (**-optimal-frame*) are close to their competitors computed on all data (**-optimal*) but no longer better. Interior points thus do contribute, if only marginally, to the optimization.

However, Figure 5.4 (right) shows that the slight improvement in the objective function does not carry over for the predictive performance. By contrast, the frame-based approaches (**-optimal-frame*) consistently outperform the other approaches and lead to significantly lower MSEs. For comparison, the MSE trained and evaluated on all data points is shown as a dashed horizontal line. Training only on a few points of the frame already leads to more accurate models than using all available data.

We obtain similar results when evaluating against the A- and E-optimality criteria. Due to their similar performance, we only report on the results for A-optimal designs in Figure 5.5. Once again, the frame-based optimization is only sightly worse in terms of the optimization objective (left) but clearly outperforms the traditional approaches in predictive performance (right).

We additionally experiment on the Airfoil data (Brooks et al., 1989). The task is to predict the self-noise of airfoil blades of different designs, and the data comes with $n = 1503$ experiments describing tests in a wind tunnel with $d = 5$ attributes, and the data has a frame density of 31%.

The results for D-optimal designs are shown in Figure 5.6. Once again, the frame-based approaches perform slightly worse or on par in terms of the optimality criterion. However, the predictive performance measured in MSE is no longer superior. The errors are similar to those using uniform samples

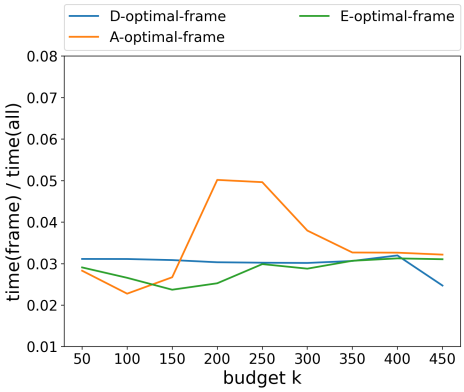

Figure 5.7: Timing results on Airfoil data.

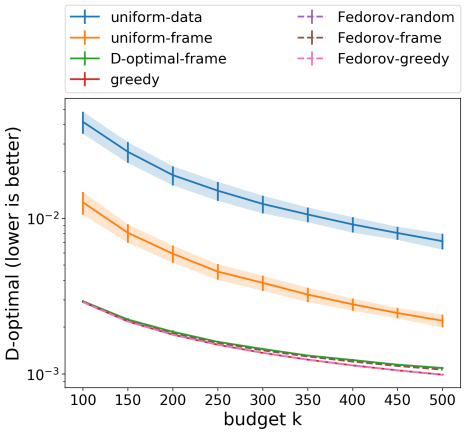

Figure 5.8: Results on California Housing data.

of the data. Thus, the dataset shows that even though the optimality criterion is well approximated, an error reduction is not guaranteed. However, this does not limit our approach as D-optimal design does not guarantee a reduction either.

5.5.2 The Efficiency of the Frame Restriction

We now report on the efficiency of our approach on Airfoil data. Figure 5.7 illustrates the relative time of the frame-based approaches compared to their traditional analogs computed on all data. The y-axis thus shows `time(frame)/time(all)`. We can report a drastically faster computation taking only 2-5% of the time of the traditional variants. We credit this finding to Airfoil's frame density of 31%. That is, restricting the data to the frame already discards 69% of the data, and the resulting optimization problems become much smaller.

Naturally, the smaller the frame size, the faster the computation as we leave out more and more interior points. We thus experiment on the California Housing data (Pace and Barry, 1997), where the task is to estimate the median housing prices for different census blocks. This data comes with $n = 20,640$ instances in $d = 8$ dimensions but possesses a frame density of only 8%.

Figure 5.8 depicts the result with respect to the D-optimal criterion. The figure again shows that naively sampling from the frame (*uniform-frame*) is significantly better than drawing random samples from all data (*uniform-data*). All other tested algorithms perform even better and realize almost identical curves. The D-optimal baseline could not be computed in a reasonable amount of time due to the size of the data. Only restricting the computations to the frame rendered the computation feasible.

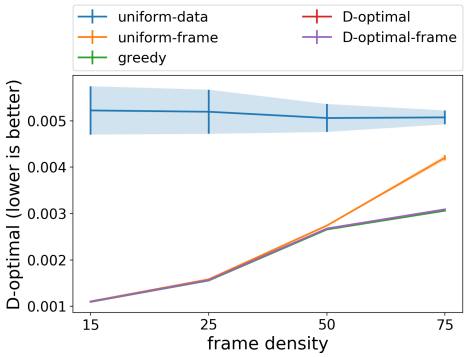

Figure 5.9: Effect of the frame size on synthetic data.

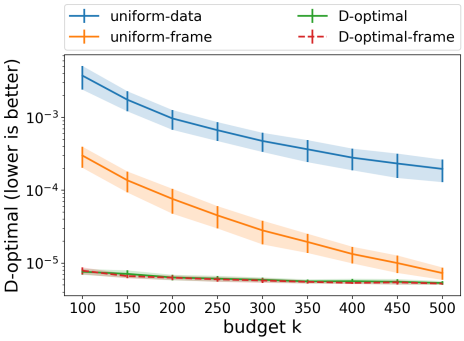

Figure 5.10: Results on synthetic data using a polynomial kernel of degree 3.

5.5.3 The Impact of the Frame Density

We already mentioned that the frame density q/n influences the efficiency of frame-based approaches. A frame density of z implies that the $(1-z)$-th part of the data are interior points and can thus be ignored in subsequent computations.

To show this influence empirically, we control the frame density on synthetic data from Lopez (2005) (*cf.* Section 4.2). The data we use consists of $n = 2,500$ instances in $d = 5$ dimensions and comes in five different sets realizing frame densities of 1%, 15%, 25%, 50% and 75% respectively. Figure 5.9 shows the resulting D-optimality criteria for the different frame densities. Up to a frame density of 50%, randomly sampling from the frame (*uniform-frame*) performs on par with all other approaches, thus showing the efficiency of our proposal. For higher frame densities the performance of *uniform-frame* diverges towards *uniform-data*. Nevertheless, restricting D-OED to the frame stays on par with its peers. This experiment suggests that the smaller the frame density, the better the competitiveness of frame-based OED.

5.5.4 Sampling in Kernel-induced Feature Spaces

In our last set of experiments, we consider sampling random designs on synthetic data for non-linear regression problems. We use synthetic data as described above with a frame density of 1%. We employ a homogeneous ($c = 0$) polynomial kernel with a degree of $p = 3$ that allows for obtaining the explicit feature mapping ϕ, which is needed for all approaches except *uniform-**.

Figure 5.10 illustrates the results. Approaches optimizing the D-optimal design criterion (*D-**) perform equally well, irrespectively of whether they sample from the frame or not. This result confirms the competitiveness of

restricting OED to the frame. However, both methods rely on the explicit feature map.

Strategies that are purely based on sampling (*uniform-**) do not need an explicit mapping. Sampling at random from all data (*uniform-data*) trivially does not rely on anything but a list of indices. Finally, sampling from the frame (*uniform-frame*) uses the proposed kernel frame algorithm (Algorithm 5.1) to sample in feature space. The figure shows that our approach samples much better designs from the frame, which is only 23% in feature space. The larger the sample size, the less relevant becomes an explicit mapping.

5.6 Related Work

Optimal experimental design is a well-studied problem in statistics (Fedorov, 1972; Pukelsheim, 2006). Recent work focuses on efficiency and performance and aims to devise approximation guarantees for relaxations of the combinatorial problems. For example, Wang et al. (2017) consider A-optimal designs and propose sampling strategies (for the settings with and without replacement) with statistical efficiency bounds as well as a greedy removal approach. Allen-Zhu et al. (2017) propose a regret-minimization strategy for the setting without replacement which works for most optimality criteria. Mariet and Sra (2017) use elementary symmetric polynomials (ESP) for OED and introduce ESP-design, an interpolation between A- and D-optimal design that includes both as special cases. They provide approximation guarantees for sampling and greedy removal strategies.

OED has close ties to many other problems. D-optimality, for example, is related to volume sampling (Avron and Boutsidis, 2013; Derezinski and Warmuth, 2018; Li et al., 2017) and determinantal point processes (DPPs) (Kulesza et al., 2012); both are used in many applications to sample informative and diverse subsets.

Moreover, the problem setting we consider is related to active learning (Sugiyama and Nakajima, 2009; Chaudhuri et al., 2015). Common active learning strategies sequentially select data points based on some uncertainty criterion or heuristic. For instance, data points are selected based on the confidence of the model to an assigned label or according to the maximal model update in the worst case. Usually, active learning iteratively selects instances and then re-trains to include the newly gained label into the model. In contrast to such iterative active learning scenarios with feedback, OED corresponds to selecting a single optimal batch prior to labeling and learning.

Convex hull algorithms can straightforwardly compute the frame. However, as mentioned in Chapter 4, many are motivated and limited to two- or three-dimensional settings. Quickhull (Barber et al., 1996) works in higher

dimensionalities but quickly becomes infeasible. If the enumeration of vertices is dropped, convex hull algorithms can be turned into methods that directly (and only) compute the frame. Common approaches for examples include linear programming to test whether a point is part of the frame or not (Dulá and Helgason, 1996; Ottmann et al., 2001; Dulá and López, 2012). Recent methods use quadratic programming to efficiently compute the frame (Mair et al., 2017) (*cf.* Chapter 4).

5.7 Conclusion

We proposed to leverage the geometry of the data to compute optimal designs efficiently. Our contribution was motivated by the observation that traditional OED variants optimize enclosing ellipsoids that are supported by extreme data points. Hence, we proposed to restrict the computations to the frame, which is the smallest subset of the data that yields the same convex hull as all data. We devised an optimization problem to compute the frame in kernel-induced feature spaces and provided a theoretical foundation for the eligibility of different kernel functions. Our contribution can be viewed as a transposed version of LASSO that selects data points instead of features.

Empirically, we showed that restricting optimal design to the frame yields competitive designs with respect to D-, E-, and A-optimality criteria on several real-world data sets. Our frame-based approaches ignore interior data points, and we observed computational speed-ups of up to a factor of twenty. Our contribution rendered OED problems feasible on data at large scales for moderate frame densities.

Chapter 6

Coresets for Archetypal Analysis

In Chapter 4, we considered the frame as a representative subset for archetypal analysis. Although the frame as a representative subset reduced the computational complexity and performed competitively, it has some drawbacks. First, the frame computation is expected to be prohibitive for high-dimensional data sets as it scales polynomially in the dimensionality d. Second, the size of the representative subset is fixed as it is an inherent property of a data set. Third, there is no theoretical guarantee that the frame actually yields a competitive performance. Despite its good performance on the evaluated data sets, data sets that induce an arbitrary error can be constructed.

Thus, in this chapter, we provide a remedy to those issues. A theoretically sound alternative to the frame is offered by coresets. They compactly represent large data sets by weighted subsets on which models perform provably competitive compared to operations on all data. Coresets have successfully been leveraged to very different methods, including k-means (Lucic et al., 2016; Bachem et al., 2018a), support vector machines (Tsang et al., 2005), logistic regression (Munteanu et al., 2018), and Bayesian inference (Huggins et al., 2016; Campbell and Broderick, 2018). The idea is as follows. A small subset of the data is selected (in linear time) according to a strategy such that the subset approximates the original data very well. A learning algorithm will then provably perform similarly on the original data and the subset, but training on the subset is much more efficient. In this chapter, we present coresets for archetypal analysis (AA). An example of the proposed idea is depicted in Figure 6.1.

The key contributions of this chapter are as follows. We (i) show that the objective function of k-means upper bounds the objective of archetypal analysis and show that every coreset for k-means is also a coreset for archetypal analysis, we (ii) propose a simple and efficient sampling strategy

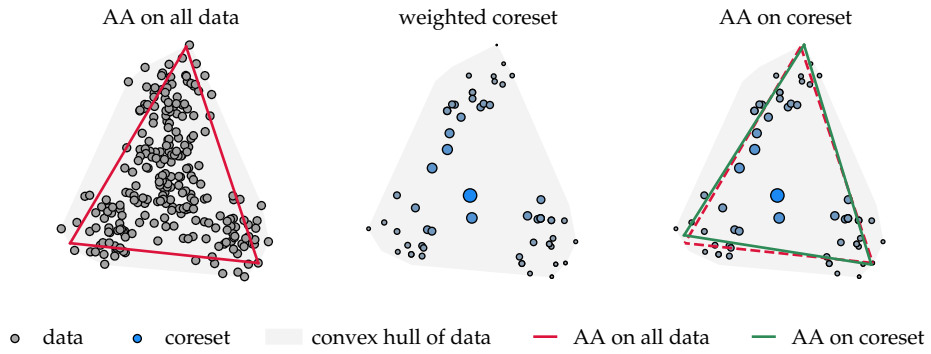

Figure 6.1: Illustration of archetypal analysis on a coreset. Left: archetypal analysis on a data set; center: coreset (weighted representative subset) with weights, where the weights are denoted by size; and right: archetypal analysis computed on the coreset.

to compute a coreset with only two passes over the data, so that (iii) a weak ε-absolute-coreset is guaranteed to be obtained after sampling sufficiently many points where (iv) the error bound does not depend on the query itself. Finally, we (v) provide empirical results on various data sets to support the theoretical derivation.

6.1 Preliminaries and Coresets

Let $\mathcal{X}$ be a data set of n points in d dimensions. Consider a learning problem with an objective function of the form $\phi_{\mathcal{X}}(Q) = \sum_{\mathbf{x}\in\mathcal{X}} d(\mathbf{x}, Q)^2$. The goal is to learn the so-called query $Q \subset \mathbb{R}^d$, with $|Q| = k$, and $d(\mathbf{x}, Q)^2$ is the minimal squared distance from a data point $\mathbf{x}$ to the query Q. For example, in k-means clustering (Lloyd, 1982), Q refers to the set of cluster centers and $d(\mathbf{x}, Q)^2 = \min_{\mathbf{q}\in Q} \|\mathbf{x} - \mathbf{q}\|_2^2$. The objective function is then given by

$$\phi_{\mathcal{X}}(Q) = \sum_{\mathbf{x}\in\mathcal{X}} d(\mathbf{x}, Q)^2 = \sum_{\mathbf{x}\in\mathcal{X}} \min_{\mathbf{q}\in Q} \|\mathbf{x} - \mathbf{q}\|_2^2.$$

A *coreset* is a possibly weighted subset $\mathcal{C}$ of the full data set $\mathcal{X}$ with cardinality $m \ll n$, which performs provably competitive with respect to the performance on $\mathcal{X}$. Using non-negative weights $w_i \geq 0$ on the data points, the objective becomes $\phi_{\mathcal{X}}(Q) = \sum_{\mathbf{x}\in\mathcal{X}} w_i \cdot d(\mathbf{x}, Q)^2$. The standard definition of a coreset is as follows.

Definition 6.1. *Let $\varepsilon > 0$ and $k \in \mathbb{N}$. A (weighted) set $\mathcal{C}$ is a (ε, k)-coreset of the data $\mathcal{X}$ if for any $Q \subset \mathbb{R}^d$ of cardinality at most k*

$$|\phi_{\mathcal{X}}(Q) - \phi_{\mathcal{C}}(Q)| \leq \varepsilon \phi_{\mathcal{X}}(Q). \tag{6.1}$$

Algorithm 6.1 Lightweight coreset construction for k-means

Input: Set of data points $\mathcal{X}$, coreset size m
Output: Coreset $\mathcal{C}$, weights $\{w_i\}_{i=1}^m$
$\mu \leftarrow$ mean of $\mathcal{X}$
for $\mathbf{x} \in \mathcal{X}$ **do**
  $q(\mathbf{x}) = \frac{1}{2}\frac{1}{|\mathcal{X}|} + \frac{1}{2}\frac{d(\mathbf{x},\mu)^2}{\sum_{\mathbf{x}'} d(\mathbf{x}',\mu)^2}$
end for
$\mathcal{C} \leftarrow$ sample m points from $\mathcal{X}$ where each point has weight $\frac{1}{m \cdot q(\mathbf{x})}$ and is sampled with probability $q(\mathbf{x})$

Note that the condition of a coreset in Equation (6.1) is equivalent to $(1-\varepsilon)\phi_{\mathcal{X}}(Q) \leq \phi_{\mathcal{C}}(Q) \leq (1+\varepsilon)\phi_{\mathcal{X}}(Q)$. Hence, the performance of the query learned on the coreset is bounded from below and above by a $(1 \pm \varepsilon)$ multiplicative of the query evaluated on the full data set. Definition 6.1 defines a *strong* coreset since the bound holds uniformly for all queries Q. If the condition in Equation (6.1) holds only for the optimal query, $\mathcal{C}$ is called a *weak* coreset.

Computing a coreset for k-means may require k sequential passes over the data (Lucic et al., 2016). Bachem et al. (2018a) introduce the notion of *lightweight-coresets*, which allow for an additional additive term on the right-hand side of the bound in Equation (6.1) and show that the solution can be computed in only two passes over the data. The definition of lightweight-coresets is as follows.

Definition 6.2. *(Bachem et al., 2018a) Let $\varepsilon > 0$, $k \in \mathbb{N}$ and $\mathcal{X} \subset \mathbb{R}^d$ be a set of points with mean $\mu \in \mathbb{R}^d$. The weighted set $\mathcal{C}$ is a (ε, k)-lightweight-coreset of the data $\mathcal{X}$ if for any $Q \subset \mathbb{R}^d$ of cardinality at most k*

$$|\phi_{\mathcal{X}}(Q) - \phi_{\mathcal{C}}(Q)| \leq \frac{\varepsilon}{2}\phi_{\mathcal{X}}(Q) \underbrace{+\frac{\varepsilon}{2}\phi_{\mathcal{X}}(\{\mu\})}_{\text{additive term}}. \tag{6.2}$$

The lightweight-coreset for k-means is constructed via importance sampling, in order to guide the sampling procedure towards more influential points. The sampling distribution q is a mixture of a uniform distribution and the normalized squared distances to the mean, *i.e.*,

$$q(\mathbf{x}) = \frac{1}{2}\frac{1}{n} + \frac{1}{2}\frac{d(\mathbf{x},\mu)^2}{\sum_{i=1}^n d(\mathbf{x}_i,\mu)^2}. \tag{6.3}$$

The underlying idea is that points that lie far away from the mean μ have a larger impact on the objective function and should thus be sampled with higher probability. The procedure of Bachem et al. (2018a) is shown in Algorithm 6.1. After sampling m data points according to q, each point

is weighted by $(m \cdot q(\mathbf{x}))^{-1}$ such that the sampling procedure yields an unbiased estimator of the quantization error:

$$\begin{aligned}\mathbb{E}_{\mathcal{C}}\left[\phi_{\mathcal{C}}(Q)\right] &= \mathbb{E}_{\mathcal{C}}\left[\sum_{\mathbf{x}\in\mathcal{C}} \frac{1}{m \cdot q(\mathbf{x})} d(\mathbf{x}, Q)^2\right] \\ &= \mathbb{E}\left[\frac{1}{q(\mathbf{x})} d(\mathbf{x}, Q)^2\right] = \sum_{\mathbf{x}\in\mathcal{X}} q(\mathbf{x}) \frac{d(\mathbf{x}, Q)^2}{q(\mathbf{x})} = \phi_{\mathcal{X}}(Q).\end{aligned}$$

The following result ensures that a (ε, k)-lightweight-coreset is obtained after sampling sufficiently many points.

Theorem 6.1 (Bachem et al. (2018a))**.** *Let $\varepsilon > 0, \delta > 0$ and $k \in \mathbb{N}$. Let $\mathcal{X}$ be a set of points in $\mathbb{R}^d$ and let $\mathcal{C}$ be the output of Algorithm 6.1 with a sample size m of at least*

$$m \geq c \frac{dk \log k + \log \frac{1}{\delta}}{\varepsilon^2},$$

where c is an absolute constant. Then, with probability of at least $1 - \delta$, $\mathcal{C}$ is a (ε, k)-lightweight-coreset of $\mathcal{X}$.

Bachem et al. (2018a) argue that dropping $\frac{\varepsilon}{2}\phi_{\mathcal{X}}(Q)$ from Equation (6.2) is not possible for the problem of k-means. Assume, for example, that the cluster centers (query Q) are placed arbitrarily far away from other data. Equation (6.2) would show an arbitrary large difference on the left-hand side, but the error on the right-hand side would be bounded by $\frac{\varepsilon}{2}\phi_{\mathcal{X}}(\{\mu(\mathcal{X})\})$. Hence, $\mathcal{C}$ cannot be a coreset because it does not hold uniformly for all queries Q.

While this observation applies to k-means, the situation is very different for archetypal analysis. We assume that the mean μ of $\mathcal{X}$ is actually *contained in the convex hull* of the query, *i.e.*, $\mu \in \operatorname{conv}(Q)$. Hence, placing some points of the query far away from the data induces a larger convex hull and thus a *lower* projection error. In the remainder, we also argue that queries of practical interest always lie on the border of the convex hulls of either $\mathcal{X}$ or $\mathcal{C}$ and that the mean μ is always included.

6.2 Coreset Construction

In the case of archetypal analysis, the query Q consists of the archetypes $\mathbf{z}_1, \ldots, \mathbf{z}_k$. The squared distance of a point $\mathbf{x}$ to the query Q is given by the length of the projection of the point to the convex set $\operatorname{conv}(Q)$, *i.e.*, $d(\mathbf{x}, Q)^2 = \min_{\mathbf{q}\in\operatorname{conv}(Q)} \|\mathbf{x} - \mathbf{q}\|_2^2$. Hence, the objective function can be

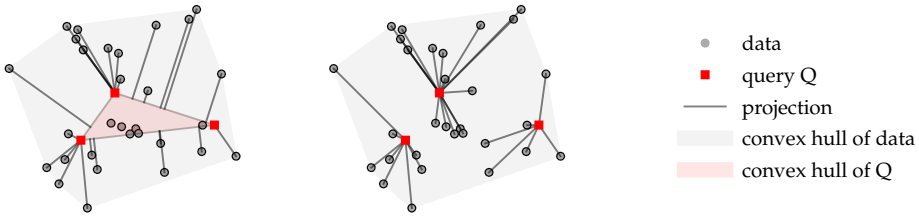

Figure 6.2: Illustration of Lemma 6.1. The projection of a point $\mathbf{x}$ to $\text{conv}(Q)$ (left) is smaller than or equal to the distance of $\mathbf{x}$ to the closest point $\mathbf{q} \in Q$ (right). Points within $\text{conv}(Q)$ (red shaded area) have no projection.

rewritten in the following way:

$$\phi_{\mathcal{X}}(Q) = \sum_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}, Q)^2 = \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{q} \in \text{conv}(Q)} \|\mathbf{x} - \mathbf{q}\|_2^2.$$

In the remainder, $\phi_{\mathcal{X}}(Q)$ refers to the above objective of archetypal analysis.

Before introducing and analyzing the coreset construction for archetypal analysis, we show that for a point $\mathbf{x}$ the quantization error of k-means upper bounds the projection of $\mathbf{x}$ to the query (*i.e.*, the archetypes $\mathbf{z}_1, \dots, \mathbf{z}_k$) in archetypal analysis. Consequently, every coreset that bounds the error of k-means must also bound the error of archetypal analysis and is thus also a coreset for archetypal analysis.

Lemma 6.1. *Let $\mathbf{x} \in \mathbb{R}^d$ be a data point, $d(\cdot, \cdot)$ be a distance metric and $Q \subset \mathbb{R}^d$ be any set of $k \in \mathbb{N}$ points, then it holds that*

$$\min_{\mathbf{q} \in \text{conv}(Q)} d(\mathbf{x}, \mathbf{q}) \leq \min_{\mathbf{q} \in Q} d(\mathbf{x}, \mathbf{q}).$$

Proof. First, note that $Q \subset \text{conv}(Q)$. Assume that $\mathbf{q}' \in \text{conv}(Q)$ minimizes $d(\mathbf{x}, \mathbf{q})$, then $\mathbf{q}'$ is either in $\text{conv}(Q) \setminus Q$, resulting in a smaller distance than any other $\mathbf{q}'' \in Q$, or $\mathbf{q}'$ is in Q, yielding the same distance as $\min_{\mathbf{q} \in Q} d(\mathbf{x}, \mathbf{q})$. Hence, the distance of $\mathbf{x}$ to the convex set $\text{conv}(Q)$ is smaller or equal to the distance to any $\mathbf{q} \in Q$. □

A direct consequence of Lemma 6.1, which is depicted in Figure 6.2, is that for any choice of Q, the objective function of archetypal analysis is upper bounded by the objective of k-means, *i.e.*,

$$\sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{q} \in \text{conv}(Q)} \|\mathbf{x} - \mathbf{q}\|_2^2 \leq \sum_{\mathbf{x} \in \mathcal{X}} \min_{\mathbf{q} \in Q} \|\mathbf{x} - \mathbf{q}\|_2^2.$$

Here, Q in archetypal analysis refers to the archetypes $\mathbf{z}_1, \dots, \mathbf{z}_k$ and in k-means, Q refers to the set of centroids. Since a coreset bounds the error of a method on the entire set, and due to Lemma 6.1, any coreset for k-means is also a coreset for archetypal analysis.

Algorithm 6.2 Coreset construction for archetypal analysis

Input: Set of data points $\mathcal{X}$, coreset size m
Output: Coreset $\mathcal{C}$, weights $\{w_i\}_{i=1}^m$
$\mu \leftarrow$ mean of $\mathcal{X}$
for $\mathbf{x} \in \mathcal{X}$ **do**
  $q(\mathbf{x}) = \frac{d(\mathbf{x},\mu)^2}{\sum_{\mathbf{x}'} d(\mathbf{x}',\mu)^2}$
end for
$\mathcal{C} \leftarrow$ sample m points from $\mathcal{X}$ where each point has weight $\frac{1}{m \cdot q(\mathbf{x})}$ and is sampled with probability $q(\mathbf{x})$

Proposition 6.1. *Every coreset for k-means is also a coreset for archetypal analysis.*

The proposition implies that the sampling strategy outlined in Algorithm 6.1 already yields a lightweight-coreset for our problem. However, we show in Section 6.2.1 that the term $\frac{\varepsilon}{2}\phi_{\mathcal{X}}(Q)$ can be dropped to obtain a weak ε-absolute-coreset for archetypal analysis, whose bound does not depend on the query Q.

Definition 6.3. *Let $\varepsilon > 0$ and $k \in \mathbb{N}$. A (weighted) set $\mathcal{C}$ is an ε-absolute-coreset of the data $\mathcal{X}$ if for any $Q \subset \mathbb{R}^d$ of cardinality at most k*

$$|\phi_{\mathcal{X}}(Q) - \phi_{\mathcal{C}}(Q)| \leq \varepsilon. \tag{6.4}$$

The set $\mathcal{C}$ is called a weak ε-absolute-coreset if the bound holds only for specific queries Q.

Due to Theorem 3.1, archetypes are guaranteed to lie on the boundary of the convex hull of the data.[1] Thus, we are interested in points lying far from the mean μ of $\mathcal{X}$. Such points increase the convex hull of the archetypes and result in smaller projections and hence in a lower value of the objective function. We thus discard the uniform term in Equation (6.3) and propose the following sampling distribution:

$$q(\mathbf{x}) = \frac{d(\mathbf{x}, \mu)^2}{\sum_{i=1}^n d(\mathbf{x}_i, \mu)^2}.$$

6.2.1 Analysis

We now provide a bound on the sample size m to show that Algorithm 6.2 computes a provably competitive coreset.

[1] Given that there is more than only a single archetype.

Theorem 6.2. *Let $\varepsilon > 0, \delta > 0$ and $k \in \mathbb{N}$. Let $\mathcal{X}$ be a set of points in $\mathbb{R}^d$ with mean $\mu \in \mathbb{R}^d$ and let $\mathcal{C}$ be the output of Algorithm 6.2 with a sample size m of at least*

$$m \geq c \frac{dk \log k + \log \frac{1}{\delta}}{\varepsilon^2},$$

where c is an absolute constant. Then, with probability of at least $1 - \delta$, the set $\mathcal{C}$ fulfills

$$|\phi_{\mathcal{X}}(Q) - \phi_{\mathcal{C}}(Q)| \leq \varepsilon \phi_{\mathcal{X}}(\{\mu\}) \tag{6.5}$$

for any query $Q \subset \mathbb{R}^d$ of cardinality at most k satisfying $\mu \in \operatorname{conv}(Q)$.

Before we prove Theorem 6.2, we introduce the concept of the pseudo-dimension (Haussler, 1992; Li et al., 2001) and based on this a result from Li et al. (2001).

Definition 6.4 (Haussler (1992); Li et al. (2001))**.** *Fix a countably infinite domain $\mathcal{X}$. The pseudo-dimension of a set $\mathcal{H}$ of functions from $\mathcal{X}$ to $[0, 1]$, denoted by $\operatorname{Pdim}(\mathcal{H})$, is the largest d' such there is a sequence $x_1, \ldots, x_{d'}$ of domain elements from $\mathcal{X}$ and a sequence $r_1, \ldots, r_{d'}$ of reals such that for each $b_1, \ldots, b_{d'} \in \{above, below\}$, there is an $f \in \mathcal{H}$ such that for all $i = 1, \ldots, d'$, we have $f(x_i) \geq r_i \iff b_i = above$.*

Theorem 6.3 (Li et al. (2001))**.** *Let $\alpha > 0$, $\nu > 0$ and $\delta > 0$. Fix a countably infinite domain $\mathcal{X}$ and let P be any probability distribution over $\mathcal{X}$. Let $\mathcal{H}$ be a set of functions from $\mathcal{X}$ to $[0, 1]$ with $\operatorname{Pdim}(\mathcal{H}) = d'$. Denote by $\mathcal{C}$ a sample of m points from $\mathcal{X}$ independently drawn according to P with*

$$m \geq \frac{c}{\alpha^2 \nu} \left(d' \log \frac{1}{\nu} + \log \frac{1}{\delta} \right),$$

where c is an absolute constant. Then, it holds with probability of at least $1 - \delta$ that

$$\mathrm{d}_\nu \left(\mathbb{E}_P[f], \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} f(x) \right) \leq \alpha \quad \forall f \in \mathcal{H},$$

where $\mathrm{d}_\nu(a, b) = \frac{|a-b|}{a+b+\nu}$. Over all choices of $\mathcal{H}$ with $\operatorname{Pdim}(\mathcal{H}) = d$, this bound on m is tight.

Furthermore, the proof of Theorem 6.2 relies on bounds of the projection of a point $\mathbf{x}$ to the convex hull of the query Q as shown in the following lemma.

Lemma 6.2. *Let $\mathcal{X}$ be a set of points in $\mathbb{R}^d$ with mean μ. For all $\mathbf{x} \in \mathcal{X}$ and $Q \subset \mathbb{R}^d$ satisfying $\mu \in \operatorname{conv}(Q)$, it holds that*

$$d(\mathbf{x}, Q)^2 \leq 2d(\mathbf{x}, \mu)^2.$$

Proof. By the triangle inequality and since

$$(|a| + |b|)^2 \leq 2a^2 + 2b^2$$

we have for any $\mathbf{x}$ and Q that

$$d(\mathbf{x}, Q)^2 \leq (d(\mathbf{x}, \mu) + d(\mu, Q))^2 \leq 2d(\mathbf{x}, \mu)^2 + 2d(\mu, Q)^2.$$

Since $\mu \in \operatorname{conv}(Q)$, the distance of μ to the query Q is zero, *i.e.*, $d(\mu, Q)^2 = 0$, yielding our claim:

$$d(\mathbf{x}, Q)^2 \leq 2d(\mathbf{x}, \mu)^2.$$

□

The proof of Theorem 6.2 can now be shown as follows.

Proof of Theorem 6.2. Let μ be the mean of $\mathcal{X}$. Consider the function

$$g_Q(\mathbf{x}) = \frac{d(\mathbf{x}, Q)^2}{2d(\mathbf{x}, \mu)^2}.$$

Due to the non-negativity of distances as well as Lemma 6.2, we know that $g_Q(\mathbf{x}) \in [0, 1]$ for any $\mathbf{x} \in \mathcal{X}$ and $Q \subset \mathbb{R}^d$ satisfying $\mu \in \operatorname{conv}(Q)$. Then, it holds that

$$\begin{aligned}
\phi_{\mathcal{X}}(Q) &= \sum_{\mathbf{x} \in \mathcal{X}} d(\mathbf{x}, Q)^2 = \sum_{\mathbf{x} \in \mathcal{X}} \frac{2d(\mathbf{x}, \mu)^2 \phi_{\mathcal{X}}(\{\mu\})}{2d(\mathbf{x}, \mu)^2 \phi_{\mathcal{X}}(\{\mu\})} d(\mathbf{x}, Q)^2 \\
&= 2\phi_{\mathcal{X}}(\{\mu\}) \sum_{\mathbf{x} \in \mathcal{X}} \frac{d(\mathbf{x}, \mu)^2}{\phi_{\mathcal{X}}(\{\mu\})} \frac{d(\mathbf{x}, Q)^2}{2d(\mathbf{x}, \mu)^2} \\
&= 2\phi_{\mathcal{X}}(\{\mu\}) \sum_{\mathbf{x} \in \mathcal{X}} q(\mathbf{x}) g_Q(\mathbf{x}) = 2\phi_{\mathcal{X}}(\{\mu\}) \mathbb{E}_q \left[g_Q(\mathbf{x}) \right]. \qquad (6.6)
\end{aligned}$$

Following the discussion in Bachem et al. (2017), and since every coreset for k-means is also a coreset for archetypal analysis (Proposition 6.1), we use the result $\operatorname{Pdim}(\mathcal{G}) \in \mathcal{O}(dk \log k)$. Hence, we can choose $d' = \frac{1}{\log 2} dk \log k$. Let $\alpha = \frac{\varepsilon}{6}$, $\nu = \frac{1}{2}$, c' be an absolute constant and $c = 72c'$. By using

$$m \geq \frac{c'}{\alpha^2 \nu} \left(d' \log \frac{1}{\nu} + \log \frac{1}{\delta} \right) = \frac{72c'}{\varepsilon^2} \left(d' \log 2 + \log \frac{1}{\delta} \right) = c \frac{dk \log k + \log \frac{1}{\delta}}{\varepsilon^2},$$

Theorem 6.3 implies that with probability of at least $1 - \delta$

$$\mathrm{d}_\nu \left(\mathbb{E}_q[g_Q(\mathbf{x})], \frac{1}{|\mathcal{C}|} \sum_{x \in \mathcal{C}} g_Q(\mathbf{x}) \right) \leq \frac{\varepsilon}{6}$$

uniformly for all sets Q of cardinality at most k including μ in their convex hull. Since both arguments of d_ν are in $[0, 1]$, the denominator of d_ν is bounded by 3. Hence, we have

$$\left| \mathbb{E}_q[g_Q(\mathbf{x})] - \frac{1}{|\mathcal{C}|} \sum_{\mathbf{x} \in \mathcal{C}} g_Q(\mathbf{x}) \right| \leq \frac{\varepsilon}{2}.$$

We now multiply both sides by $2\phi_{\mathcal{X}}(\{\mu\})$ yielding

$$\left| 2\phi_{\mathcal{X}}(\{\mu\}) \mathbb{E}_q[g_Q(\mathbf{x})] - \frac{2\phi_{\mathcal{X}}(\{\mu\})}{|\mathcal{C}|} \sum_{\mathbf{x} \in \mathcal{C}} g_Q(\mathbf{x}) \right| \leq \varepsilon \phi_{\mathcal{X}}(\{\mu\}).$$

The first part is equal to $\phi_{\mathcal{X}}(Q)$ due to Equation (6.6) and the second part can be rewritten as follows:

$$\begin{aligned}
\frac{2\phi_{\mathcal{X}}(\{\mu\})}{|\mathcal{C}|} \sum_{\mathbf{x} \in \mathcal{C}} g_Q(\mathbf{x}) &= \frac{2\phi_{\mathcal{X}}(\{\mu\})}{|\mathcal{C}|} \sum_{\mathbf{x} \in \mathcal{C}} \frac{d(\mathbf{x}, Q)^2}{2d(\mathbf{x}, \mu)^2} = \sum_{\mathbf{x} \in \mathcal{C}} \frac{\phi_{\mathcal{X}}(\{\mu\})}{|\mathcal{C}| d(\mathbf{x}, \mu)^2} d(\mathbf{x}, Q)^2 \\
&= \sum_{\mathbf{x} \in \mathcal{C}} \frac{1}{|\mathcal{C}| \frac{d(\mathbf{x}, \mu)^2}{\phi_{\mathcal{X}}(\{\mu\})}} d(\mathbf{x}, Q)^2 \\
&= \sum_{\mathbf{x} \in \mathcal{C}} \frac{1}{|\mathcal{C}| q(\mathbf{x})} d(\mathbf{x}, Q)^2 = \phi_{\mathcal{C}}(Q).
\end{aligned}$$

Finally, we obtain our claim:

$$|\phi_{\mathcal{X}}(Q) - \phi_{\mathcal{C}}(Q)| \leq \varepsilon \phi_{\mathcal{X}}(\{\mu\}).$$

□

Note that the bound on the right-hand side of Equation (6.5) is independent of the query Q and corresponds to the scaled variance of the data. For a normalized data set, Algorithm 6.2 yields an ε-absolute-coreset as the following corollary shows.

Corollary 6.1. *Let $\varepsilon > 0, \delta > 0, k \in \mathbb{N}$, and $\mathcal{X}$ be a set of points in $\mathbb{R}^d$ with mean $\mu \in \mathbb{R}^d$. Denote by $\bar{\mathcal{X}}$ the standardized set of points with $\bar{\mathbf{x}}_i = (\mathbf{x}_i - \mu)/\phi_{\mathcal{X}}(\{\mu\})$. Let $\mathcal{C}$ be the output of Algorithm 6.2 on $\bar{\mathcal{X}}$ with a sample size m of at least*

$$m \geq c \frac{dk \log k + \log \frac{1}{\delta}}{\varepsilon^2},$$

where c is an absolute constant. Then, with probability of at least $1-\delta$, $\mathcal{C}$ is an ε-absolute-coreset of $\bar{\mathcal{X}}$, i.e., it holds that

$$|\phi_{\mathcal{X}}(Q) - \phi_{\mathcal{C}}(Q)| \leq \varepsilon$$

for any query $Q \subset \mathbb{R}^d$ of cardinality at most k satisfying $\mu \in \operatorname{conv}(Q)$.

Corollary 6.1 can be interpreted in the following way: as we decrease ε, the performance gap of archetypal analysis on the full (standardized) data set and archetypal analysis on the coreset closes for a query Q satisfying $\mu \in \operatorname{conv}(Q)$. One might ask whether this restriction on the choice of Q is a drawback. The assumption within the various definitions of coresets that the bound has to hold for any choice of Q is very strong. For the problem of k-means, this makes sense as the centers could be anywhere in the space. However, Theorem 3.1 shows that the archetypes $\mathbf{z}_1, \ldots, \mathbf{z}_k$ lie on the boundary of the data, *i.e.*, $\{\mathbf{z}_1, \ldots, \mathbf{z}_k\} \subset \partial\mathcal{C}$ for $k > 1$. Hence, any meaningful query Q will be on the boundary of the coreset $\partial\mathcal{C}$ as well. Such a query will likely contain the mean μ of $\mathcal{X}$, because $\mathcal{C} \subset \mathcal{X}$ is sampled around μ.

As the following theorem shows, the optimal solution $Q^\star_{\mathcal{C}}$ computed on the coreset $\mathcal{C}$ is indeed provably competitive to the solution $Q^\star_{\mathcal{X}}$ obtained on the full data set.

Theorem 6.4. *Let $\varepsilon > 0$ and $\mathcal{X}$ be a set of points in $\mathbb{R}^d$ with mean $\mu \in \mathbb{R}^d$. Denote by $Q^\star_{\mathcal{X}}$ the optimal solution on $\mathcal{X}$ and by $Q^\star_{\mathcal{C}}$ the optimal solution on $\mathcal{C}$. Then it holds that*

$$\phi_{\mathcal{X}}(Q^\star_{\mathcal{C}}) \leq \phi_{\mathcal{X}}(Q^\star_{\mathcal{X}}) + 2\varepsilon\phi_{\mathcal{X}}(\{\mu\}).$$

Proof. Since $Q^\star_{\mathcal{C}}$ is the optimal solution on $\mathcal{C}$ we know that

$$\phi_{\mathcal{C}}(Q^\star_{\mathcal{C}}) \leq \phi_{\mathcal{C}}(Q^\star_{\mathcal{X}}) \tag{6.7}$$

and by the property in Equation (6.5) we have that

$$\phi_{\mathcal{X}}(Q) - \varepsilon\phi_{\mathcal{X}}(\{\mu\}) \leq \phi_{\mathcal{C}}(Q) \leq \phi_{\mathcal{X}}(Q) + \varepsilon\phi_{\mathcal{X}}(\{\mu\}).$$

Inserting $Q^\star_{\mathcal{C}}$ and $Q^\star_{\mathcal{X}}$ yields

$$\phi_{\mathcal{X}}(Q^\star_{\mathcal{C}}) - \varepsilon\phi_{\mathcal{X}}(\{\mu\}) \leq \phi_{\mathcal{C}}(Q^\star_{\mathcal{C}}) \leq \phi_{\mathcal{X}}(Q^\star_{\mathcal{C}}) + \varepsilon\phi_{\mathcal{X}}(\{\mu\}), \tag{6.8}$$
$$\phi_{\mathcal{X}}(Q^\star_{\mathcal{X}}) - \varepsilon\phi_{\mathcal{X}}(\{\mu\}) \leq \phi_{\mathcal{C}}(Q^\star_{\mathcal{X}}) \leq \phi_{\mathcal{X}}(Q^\star_{\mathcal{X}}) + \varepsilon\phi_{\mathcal{X}}(\{\mu\}). \tag{6.9}$$

It follows that

$$\phi_{\mathcal{X}}(Q^\star_{\mathcal{C}}) - \varepsilon\phi_{\mathcal{X}}(\{\mu\}) \overset{(6.8)}{\leq} \phi_{\mathcal{C}}(Q^\star_{\mathcal{C}}) \overset{(6.7)}{\leq} \phi_{\mathcal{C}}(Q^\star_{\mathcal{X}}) \overset{(6.9)}{\leq} \phi_{\mathcal{X}}(Q^\star_{\mathcal{X}}) + \varepsilon\phi_{\mathcal{X}}(\{\mu\}).$$

Adding $\varepsilon\phi_{\mathcal{X}}(\{\mu\})$ to both sides yields the claim. □

6.2.2 Complexity Analysis

Algorithm 6.2 needs one full pass over the data set $\mathcal{X}$ of size n in order to determine the mean μ. Then, another pass is needed to compute the distance of each point $\mathbf{x}_i$ to the mean μ, which is needed for the sampling distribution $q(\cdot)$. Hence, the complexity of Algorithm 6.2 scales in $\mathcal{O}(nd)$. In addition, since $q(\cdot)$ is a discrete distribution on n objects, the space complexity is also in $\mathcal{O}(nd)$. The same arguments apply to the lightweight-coreset construction of Bachem et al. (2018a) as outlined in Algorithm 6.1.

6.3 Weighted Archetypal Analysis

Algorithm 6.2 does not only produce a subset $\mathcal{C}$ of $\mathcal{X}$ but also corresponding weights $w_i > 0$ $(i = 1, \dots, m)$ of the sampled data points. These weights need to be properly incorporated into the learning procedure.

Eugster and Leisch (2011) propose a weighted archetypal analysis by moving the weights into the Frobenius norm in Equation (3.2) as follows:

$$\begin{aligned}
\sum_{i=1}^{n} w_i \|\mathbf{x}_i - \mathbf{Z}^T \mathbf{a}_i\|_2^2 &= \sum_{i=1}^{n} w_i \sum_{j=1}^{d} ((\mathbf{x}_i)_j - (\mathbf{Z}^T \mathbf{a}_i)_j)^2 \\
&= \sum_{i=1}^{n} \sum_{j=1}^{d} ((\sqrt{w_i}\mathbf{x}_i)_j - (\sqrt{w_i}\mathbf{X}^T \mathbf{B}^T \mathbf{a}_i)_j)^2 \\
&= \sum_{i=1}^{n} \sum_{j=1}^{d} ((\tilde{\mathbf{x}}_i)_j - (\tilde{\mathbf{X}}^T \mathbf{B}^T \mathbf{a}_i)_j)^2,
\end{aligned}$$

where $\tilde{\mathbf{x}}_i = \sqrt{w_i}\mathbf{x}_i$ $(i = 1, \dots, n)$ denotes the transformed data point and $\tilde{\mathbf{X}}$ the corresponding design matrix. Once the data has been transformed, a standard archetypal analysis can be computed. Before using the factorization, the weight matrix $\mathbf{A}$ has to be re-computed on the original data using the archetypes. Unfortunately, this approach has a major drawback: since the archetypes $\mathbf{z}_j$ live on the boundary of $\text{conv}(\mathcal{X})$, scaling $\mathcal{X}$ will change the support of the archetypes. This is, however, counterintuitive in our setting. While the weights may influence the placement of the archetypes, their support should not be affected.

We thus propose an alternative way to include weights into vanilla archetypal analysis as proposed by Cutler and Breiman (1994) and outlined in Algorithm 3.1. Without loss of generality, assume that the weights are natural numbers. Hence, having a weight w_i on data point $\mathbf{x}_i$ is equivalent to having the data point w_i times within the data set. This causes w_i times the projection $\|\mathbf{x}_i - \mathbf{Z}^T \mathbf{a}_i\|_2^2$. In addition, we have w_i times the weight vector $\mathbf{a}_i$. Thus, the weight w_i affects the intermediate update of the archetypes, which is $\mathbf{Z} = (\mathbf{A}^\top \mathbf{A})^{-1} \mathbf{A}^\top \mathbf{X}$ as outlined in Algorithm 3.1. Instead of adding a

point $\mathbf{x}_i$ multiple (w_i) times, we can incorporate the weights via the modified update rule

$$\mathbf{Z} = (\tilde{\mathbf{A}}^\top \tilde{\mathbf{A}})^{-1} \tilde{\mathbf{A}}^\top \tilde{\mathbf{X}},$$

where $\tilde{\mathbf{A}} = \mathbf{WA}$, $\tilde{\mathbf{X}} = \mathbf{WX}$ and $\mathbf{W}$ is the diagonal matrix of rooted weights, *i.e.*, $W_{ii} = \sqrt{w_i}$. Note that by this change, the location but not the support of the archetypes $\mathbf{z}_1, \ldots, \mathbf{z}_k$ is altered with respect to the weights. The generalization to real-valued weights $w_i > 0$ follows directly.

6.4 Experiments

We now evaluate the coreset construction for archetypal analysis (**abs-cs**) and compare it to the performance of archetypal analysis on the full data set, a uniform sample (**uniform**), the lightweight-coresets for k-means (**lw-cs**, Bachem et al. (2018a)), another state-of-the-art coreset construction for k-means (**lucic-cs**, Lucic et al. (2016)) as well as an approximate solution that learns archetypes on the frame (**frame**, Mair et al. (2017), *cf.* Chapter 4).

6.4.1 Setup

We initialize the archetypes $\mathbf{z}_1, \ldots, \mathbf{z}_k$ using the `FurthestSum` procedure (Mørup and Hansen, 2010, 2012). The termination criterion is reached when the relative error between iterations is less than 10^{-3}. We measure the error in terms of the residual sum of squares (RSS) as provided in Equation (3.2) and compute the relative error $\eta = (\mathrm{RSS}_{\text{full}} - \mathrm{RSS}_{\text{coreset}}) / \mathrm{RSS}_{\text{full}}$ with respect to the performance on the full data set. We report on averages over 50 independent repetitions; error bars show standard error. The code is written in Python using NumPy (Harris et al., 2020), and all experiments run on an Intel Xeon machine with 28 $\times$ 2.60GHz and 256GB memory.[2]

6.4.2 Data

We evaluate the algorithms on several data sets. **Ijcnn1** refers to data from the IJCNN 2001 neural network competition and has $n = 49,990$ instances in $d = 22$ dimensions.[3] We adopt the preprocessing from Chang and Lin (2001). **Pose** is a subset of the Human3.6M data set (Catalin Ionescu, 2011; Ionescu et al., 2014) and deals with 3D human pose estimation and is part of the ECCV 2018 PoseTrack Challenge.[4] It consists of $n = 35,832$ poses each of which is represented as 3D coordinates of 16 joints resulting in a 48-dimensional problem. **Song** is a subset of the Million Song Dataset

[2] `https://github.com/smair/archetypalanalysis-coreset`

[3] `https://www.csie.ntu.edu.tw/~cjlin/libsvmtools/datasets/`

[4] `http://vision.imar.ro/human3.6m/challenge_open.php`

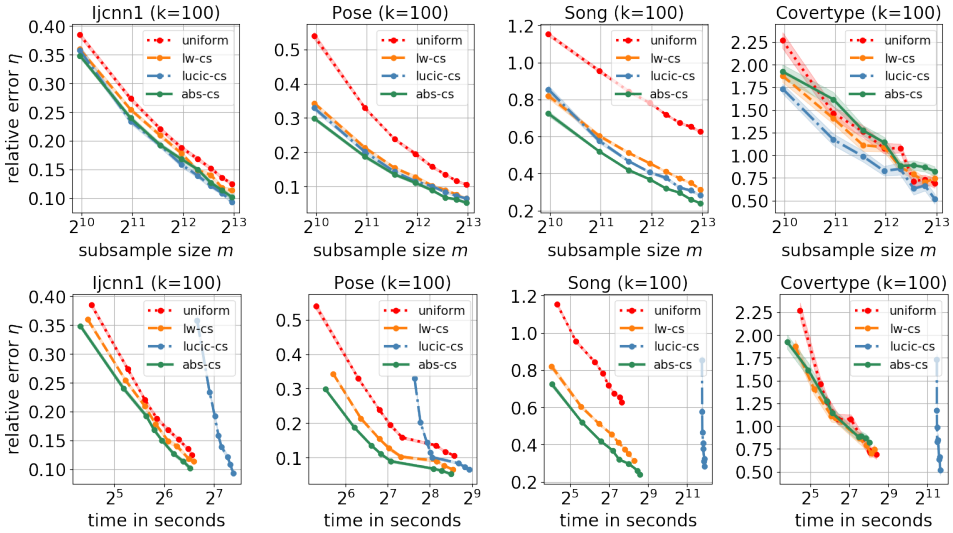

Figure 6.3: Results for $k = 100$ archetypes. Relative error η on the full data set (top) as well as computation time in seconds (bottom). Depicted are averages and standard errors of 50 independent runs.

(Bertin-Mahieux et al., 2011) which has $n = 515,345$ data points in $d = 90$ dimensions. In this data set, the task is to predict the year of a song. **Covertype** (Blackard and Dean, 1999) contains $n = 581,012$ examples in $d = 54$ dimensions. The task is to predict the forest cover type from cartographic variables.

6.4.3 Results

Figure 6.3 shows the results for $k = 100$ archetypes. In the top row, the relative error η of each approach is evaluated on the full data set and illustrated for subsample sizes ranging from $m = 1,000$ to $m = 8,000$, depicted on the x-axis. Unsurprisingly, the relative error decreases with an increasing subsample size for all approaches. The uniform sampling strategy performs almost always worse than its peers. The coreset construction of Lucic et al. (2016) (*lucic-cs*) performs in a few cases on par with our proposed approach (*abs-cs*); see, for example, Ijcnn1. In most other cases, the proposed coreset construction yields the best results and outperforms its competitors, especially on the Song data.

The bottom row in Figure 6.3 also depicts the relative error η, however, with respect to the average runtime of a single run. Theoretically, the lightweight-coreset (*lw-cs*) and the proposed coreset construction realize complexities in $\mathcal{O}(nd)$. In practice, however, the proposed approach yields consistently lower relative errors in a shorter time. We credit this finding to a better selection of coreset points resulting in a faster convergence of

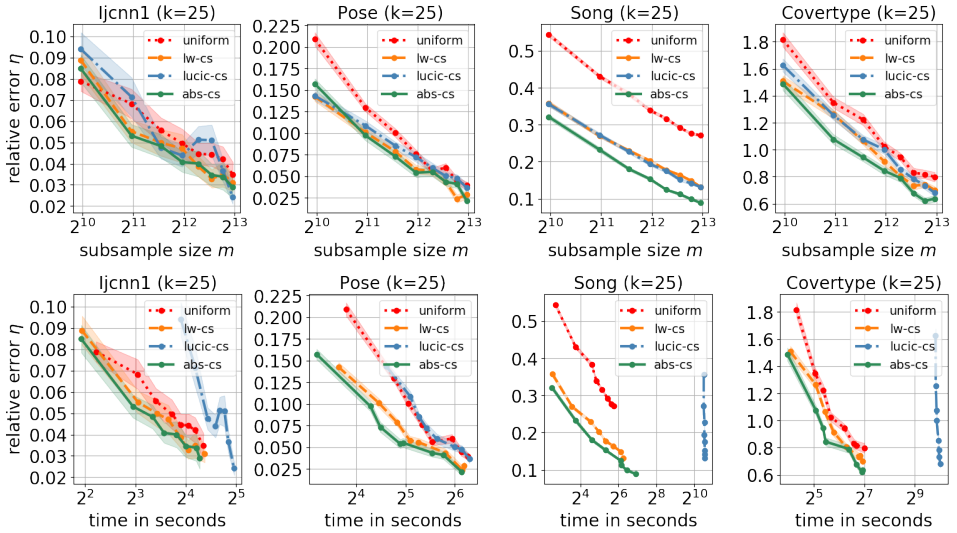

Figure 6.4: Results for $k = 25$ archetypes. Relative error η on the full data set (top) as well as computation time in seconds (bottom). Depicted are averages and standard errors of 50 independent runs.

archetypal analysis. The method of Lucic et al. (2016) (*lucic-cs*) is consistently the slowest as it requires k passes over the data for constructing the coreset.

Figure 6.4 shows the same evaluation scenario as Figure 6.3 but for only $k = 25$ archetypes. Once again, our proposed coreset construction either outperforms its peers or performs on par with the lightweight-coreset construction of Bachem et al. (2018a) while being more efficient. Table 6.1 summarizes the achieved speed-ups. On Covertype, for example, the computation of 25 archetypes with *abs-cs* and $m = 1,000$ is 601 times faster than computing the archetypes on the full data set. Although the error is 148.9% higher than the error using archetypes learned on the full data set, all other competitors are consistently outperformed. Increasing the size of the coreset to $m = 5,000$ yields a much lower relative error of 79.1% while still being 111 times faster to compute. Similar results with less speed-up but also much less relative errors are obtained for the other data sets.

The remaining competitor *frame* (*cf.* Chapter 4) precomputes all data points lying on the boundary of the convex hull of the data set (the frame). We were not able to compute the frame within a reasonable amount of time for Covertype and Song.[5] On Pose, every data point lies on the boundary; hence the performance is identical to the performance on all data. For Ijcnn1, the number of points on the frame is about $0.57n$, and the relative error η is about 0.03 for $k = 100$ archetypes. While this error is much lower, the subset

[5]Computations take about 2,000 and 4,000 hours for Covertype and Song, respectively.

Table 6.1: Relative error η in percent and speed-up compared to the full data set for $k = 25$ archetypes. Depicted are averages and standard errors of 50 independent runs for coreset sizes of $m = 1000$ and $m = 5000$.

		$m = 1000$		$m = 5000$	
Data	Method	Relative Error	Speedup	Relative Error	Speedup
Covertype	uniform	181.7%± 5.2%	468×	94.6%± 2.9%	126×
	lw-cs	150.8%± 4.1%	553×	80.6%± 2.5%	119×
	lucic-cs	162.7%± 4.8%	10×	85.4%± 2.6%	9×
	abs-cs	148.9%± 4.8%	601×	79.1%± 2.9%	111×
Song	uniform	54.4%± 0.6%	430×	31.7%± 0.4%	78×
	lw-cs	35.8%± 0.7%	480×	17.7%± 0.4%	68×
	lucic-cs	35.6%± 0.6%	2×	17.5%± 0.5%	2×
	abs-cs	32.1%± 0.6%	486×	12.4%± 0.3%	39×
Pose	uniform	20.9%± 0.8%	9×	5.6%± 0.4%	3×
	lw-cs	14.2%± 0.5%	10×	5.6%± 0.5%	3×
	lucic-cs	14.4%± 0.5%	5×	6.0%± 0.6%	3×
	abs-cs	15.7%± 0.6%	14×	5.5%± 0.5%	4×
Ijccn1	uniform	7.9%± 0.5%	17×	4.5%± 0.5%	5×
	lw-cs	8.9%± 0.7%	21×	3.9%± 0.5%	5×
	lucic-cs	9.4%± 0.8%	5×	5.1%± 0.6%	3×
	abs-cs	8.5%± 0.6%	21×	4.0%± 0.6%	6×

size is also much larger. In addition, the size of the representative subset m is not chosen but implicitly given as a property of the data set.

6.5 Conclusion

In this chapter, we introduced the first coresets for archetypal analysis. The derivation was grounded on the observation that the quantization error of k-means serves as an upper bound on the projection error of archetypal analysis; hence, every coreset for k-means is also a coreset for archetypal analysis. We devised an algorithm based on importance sampling that computes a coreset in linear time with only two passes over the data. A theoretical analysis showed that the proposed coreset performed competitively for a sufficiently large sample size. The theoretical results are supported by empiricism. The proposed algorithm outperformed its competitors on various data sets in terms of relative error and computation time. For some setups, we observed improved runtimes. In summary, our contribution rendered archetypal analysis feasible for state-of-the-art-sized data sets.

Chapter 7

Probabilistic Movement Models and Zones of Control

In the previous chapters, we considered the problem of selecting representative subsets to obtain efficient representations of data. As a result, we were able to learn more efficiently and rendered large-scale problems feasible. This chapter is the first of the second part of this thesis, in which we change the representation of data in terms of its dimensions. The goal is to derive efficient representations that ease the computation of specific operations or downstream tasks.

In this chapter, we are concerned with density estimation on trajectories in a spatio-temporal setting. Specifically, we are interested in probabilistic movement models. They allow us to study player coordination in team sports (Dick and Brefeld, 2019) but also generalize to refugee migration patterns (Hübl et al., 2017), collective animal movements (McDermott et al., 2017), and understanding dynamical systems with moving particles (Padberg-Gehle and Schneide, 2017). The task of a movement model is to predict all possible movements of a player (or refugee, animal, particle, etc.) in a given situation within a certain amount of time. In the remainder, we focus on sports analytics, and more specifically, on soccer.

Traditional movement models ground on the assumption that players are able to move in all directions equally fast and ignore velocities (Taki et al., 1996; Taki and Hasegawa, 2000; Fonseca et al., 2012), leading to implausible Voronoi-like tessellations (Voronoi, 1908) of the pitch. More sophisticated models incorporate some basic laws of physics but suffer from simplifying assumptions (Taki and Hasegawa, 2000; Fujimura and Sugihara, 2005; Gudmundsson and Wolle, 2014). All existing approaches treat every player the same by assuming that a single movement model serves all players equally well, hence, ignoring individual differences between players.

Consequences arise for applications that build upon player movement, such as the computation of *zones of control* (sometimes also called dominant

regions). The zone that is controlled by a player is characterized by her being the person on the pitch to attain any position within this region first (Taki and Hasegawa, 2000). The underlying idea is that if the ball falls inside a player's zone of control, she will likely be able to bring the ball under control after receiving it, and the more space a team controls, the more dominant they are.

The key contributions of this chapter are as follows. We (i) propose to estimate individual movement models from positional data. Our probabilistic approach leverages positions, directions, and velocities of a player at observed timestamps and returns a distribution of all reachable positions in a given time. Furthermore, we (ii) show how to turn the probabilistic movement models into zones of control. Compared to traditional one-serves-all methods, our approach leads to realistic movement models, which in turn lead to realistic zones of control. Finally, we (iii) qualitatively compare the obtained zones of control and discuss their benefits.

7.1 Related Work

Trajectory analyses are often carried out for wearable devices like smartphones, accelerometers, or gyroscopes (Zheng, 2015; Mazimpaka and Timpf, 2016). Often, the trajectories serve only as proxies for a higher level research question such as the identification of road defects (see, *e.g.*, Byrne et al. (2013); Mohan et al. (2008)), discrimination of drivers by insurance companies (Paefgen et al., 2011), or activity recognition (Avci et al., 2010; Lasek and Gagolewski, 2015).

Similarly, trajectory data in sports is used to identify movement patterns. At an individual level, Zhao et al. (2016) use Gaussian processes (GPs) to model velocity of athletes in ski races. Laube et al. (2005) propose to analyze relative motions and different temporal patterns across many subjects. As an exemplary application, the authors analyze positional data to retrieve patterns from coordinated team motions. The problem of pattern identification in groups of moving objects is also studied by Gottfried (2008, 2011). The author proposes qualitative descriptions of motion patterns using a set of atomic motions as building blocks to analyze and describe more complex behaviors; Sprado and Gottfried (2009) apply this idea to RoboCup and soccer games. Knauf et al. (2016) propose spatio-temporal convolution kernels as a similarity measure over time and space and identify game initiations and offensive patterns using a clustering approach. Similarly, Janetzko et al. (2014) group attacking patterns of strikers. Generally, frequent patterns in multi-trajectory data can also be found using episode mining algorithms (Haase and Brefeld, 2014).

Zhang et al. (2016) visualize time interval data to analyze player and team performance. They include a variety of features ranging from player velocities and ball possession as the team dominance metrics. Other methods include, for example, estimating the probabilities of a shot being made (Link et al., 2016; Harmon et al., 2016). Generally, the application of neural networks to player trajectories, represented either as sequences or images, render the need for engineering hand-crafted features unnecessary and may thus be beneficial in situations where sufficient statistics are unknown or difficult to obtain, as for analyzing player positioning. For instance, Zheng et al. (2016) and Le et al. (2017) propose to model player trajectories with recurrent neural networks for player positioning in basketball and soccer. Similarly, convolutional neural networks are used by Harmon et al. (2016) to estimate the probability of scoring opportunities. Memmert et al. (2016) and Gudmundsson and Horton (2017) provide a general overview of positional data applications in team sports. Other interesting applications include pass quality evaluation (Brooks et al., 2016) or injury prediction (Rossi et al., 2017).

Taki and Hasegawa (2000) propose a movement model based on a player's current speed, direction, and an acceleration profile along different directions. The authors discuss the dependency of acceleration on velocity and direction and also emphasize that the acceleration decreases with increasing speed. Unfortunately, the authors ignore physical details and focus on a very basic and unrealistic version of their model, in which a player is able to move in all directions with the same acceleration; hence, accepting the consequence of unbounded velocities. Fujimura and Sugihara (2005) extend this approach by adding a resistive force to prevent velocities from growing infinitely. Thus, the two approaches drastically simplify physical laws to model player movements. Note that both also constitute one-serves-all approaches as the model is not personalized to account for individual differences between players. Gudmundsson and Wolle (2014) sketch how such an individual movement model could be estimated from data. They suggest approximating a player's *reachable region* at time t by constructing a convex polygon for all historic points she reached within this time given her actual position. However, they leave it a play of thoughts and do not present technical or algorithmic details of their approach.

Once a movement model is established, it serves as a foundation for various applications in the analysis of matches. Perhaps the most important one being the computation of *zones of control*, or, alternatively, *dominant regions*. This concept has been introduced by Taki and Hasegawa (2000) as the part of the pitch that can be attained by a player before all others. Consequently, zones of control are necessary to compute and evaluate pass quality and success (Taki and Hasegawa, 2000; Nakanishi et al., 2009; Gudmundsson and Wolle, 2014; Horton et al., 2015), pressing (Taki and Hasegawa, 2000), as well as the analysis of team behavior and interaction (Fonseca et al., 2012),

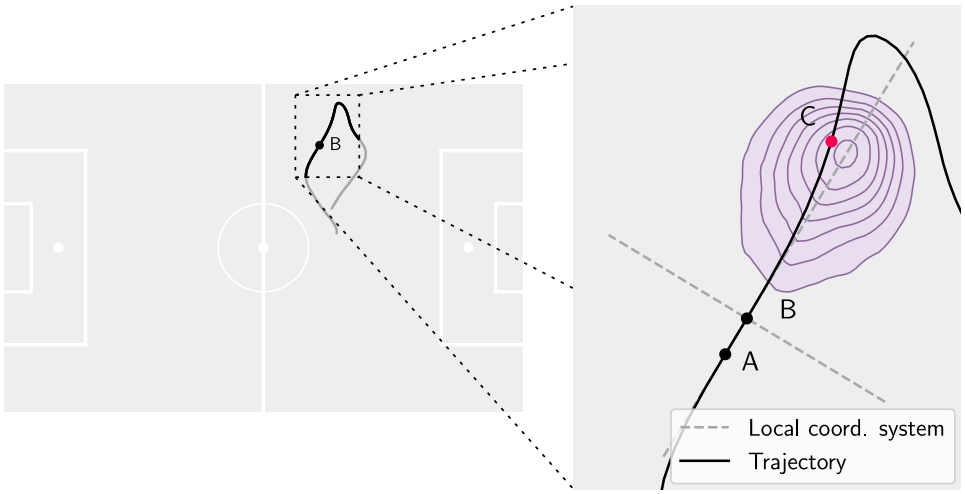

Figure 7.1: Illustration of a movement model (purple contour lines). The solid black line depicts a trajectory. The last, current, and next position of an player are marked as A, B, and C, respectively. The movement model shown in purple models the distribution of the next position C (red dot) using the information of the past and present (A and B). The dashed lines denote a local coordinate system centered at the player and aligned in the current direction of movement.

or organization and positioning in both offense and defense (Ueda et al., 2014).

7.2 Individual Movement Models

7.2.1 Preliminaries

To not clutter notation unnecessarily, we start by focusing on a single player. Let $(\mathbf{x}_t)_{t\in\mathbb{R}_{\geq 0}}$ be the trajectory of a player describing her position $\mathbf{x}_t$ at time t. In the remainder, we deal with two-dimensional movements, *i.e.*, $\mathbf{x}_t = (x_1^{(t)}, x_2^{(t)})^\top \in \mathbb{R}^2$, but the following definitions also hold for higher-dimensional movements. We further assume that the position is within an area $\mathcal{A} \subset \mathbb{R}^2$. Let $\mathbf{v}_t \in \mathbb{R}^2$ be her velocity vector at time t with its magnitude (speed) $v_t = \|\mathbf{v}_t\|_2$, where $\|\cdot\|_2$ denotes the ℓ_2-norm.[1] The time index t is typically discrete as samples are generated with equidistant timestamps $t_1, t_2, \ldots, t_n$, where $t_{i+1} - t_i = \tau > 0$ is fixed. The trajectories and the associated velocities form the data set $\mathcal{D} = \{(\mathbf{x}_{t_i}, v_{t_i})\}_{i=1}^n$. Our goal is to model the distribution of the position $\mathbf{x}_{t+t_\Delta}$, which is $t_\Delta > 0$ seconds in the

[1] Note that the velocity can be estimated from positional data in case it is not provided directly.

future, given the current position $\mathbf{x}_t$ and velocity v_t:

$$p\left(\mathbf{x} \mid \mathbf{x}_t, v_t, t_\Delta\right).$$

An example of a probabilistic movement model is depicted in Figure 7.1.

7.2.2 Existing Approaches

Before we introduce the estimation of probabilistic movement models from positional data, we briefly review existing approaches. The simplest model assumes that all players can move in all directions equally fast at a constant speed. Thus, there is no acceleration or direction of movement, and the resulting zones of control are equal to Voronoi tessellations (Voronoi, 1908) of the pitch using the players as center points. We call this model *Voronoi*.

Taki and Hasegawa (2000) improve on this by incorporating the notion of velocity and acceleration. Their model is based on the assumption that every player is able to accelerate in each direction equally fast with the magnitude of $a_{\max} > 0$. Thus, at time $t = 0$ the player begins to move with acceleration $a_{\max}$ in a direction given by the angle $\theta \in [-\pi, \pi)$. Assuming that a player is moving with speed v in the direction of the x_1-axis, in time t her position is given by

$$\mathbf{x} = (x_1, x_2), \quad \text{with} \quad \begin{cases} x_1 = \frac{1}{2} a_{\max} \cdot \cos(\theta) \cdot t^2 + vt \\ x_2 = \frac{1}{2} a_{\max} \cdot \sin(\theta) \cdot t^2. \end{cases} \tag{7.1}$$

In other words, the set of points that can be reached in time t forms a circle centered at $\mathbf{c} \in \mathbb{R}^2$ with radius $r > 0$, where

$$\mathbf{c} = (vt, 0) \quad \text{and} \quad r = \frac{1}{2} a_{\max} t^2,$$

thus, allowing for unbounded velocities. We refer to this model as *Taki & Hasegawa*.

Fujimura and Sugihara (2005) introduce a resistive force proportional to the current speed to render the movement model more realistic. The resistive force prevents the speed from growing infinitely and even clips it at maximal value $v_{\max} > 0$. Thus, at time $t = 0$, a player accelerates in direction $\theta \in [-\pi, \pi)$ with the underlying assumption that she can exert the maximum speed in any direction. The position $\mathbf{x}$ of the player at time t is given by

$$\mathbf{x} = (x_1, x_2) \quad \text{with} \quad \begin{cases} x_1 = v_{\max} \cdot \cos(\theta) \cdot \left(t - \frac{1-\exp(-\alpha t)}{\alpha}\right) + v \cdot \frac{1-\exp(-\alpha t)}{\alpha} \\ x_2 = v_{\max} \cdot \sin(\theta) \cdot \left(t - \frac{1-\exp(-\alpha t)}{\alpha}\right), \end{cases} \tag{7.2}$$

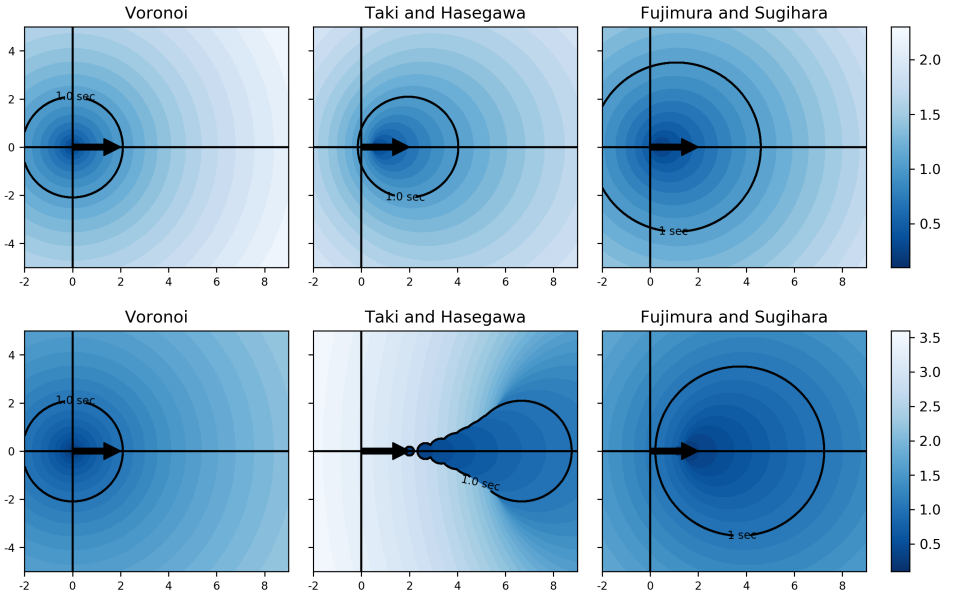

Figure 7.2: A comparison of movement models. The top row shows models for a player running with 7 km/h in direction of the x_1-axis. The bottom row shows models for a player running with 24 km/h. Voronoi is computed like Taki & Hasegawa but with a velocity of zero.

where v is the initial velocity in the direction of the x_1-axis, and the parameter $\alpha > 0$ is responsible for the resistive force. Hence, the set of points within reach of the player in time t forms a circle with center and radius given by

$$\mathbf{c} = \left(v \cdot \frac{1 - \exp(-\alpha t)}{\alpha}, 0 \right) \quad \text{and} \quad r = v_{\max} \cdot \left(t - \frac{1 - \exp(-\alpha t)}{\alpha} \right).$$

The model is referred to as *Fujimura & Sugihara*.

Figure 7.2 visualizes the existing movement models obtained by Voronoi, Taki & Hasegawa, and Fujimura & Sugihara-based approaches (from left to right). While all models realize similar circular-shaped movements for slowly moving players, differences become significant with increasing velocities. While the Voronoi-based approach yields perfect circles for any velocity, the approach by Fujimura & Sugihara leads to a conical structure assembled by nested circles. Finally, Taki & Hasegawa-based movement models become drop-shaped and oblique conical. Simply by being intrinsically circular for arbitrary velocities, it becomes evident that the existing models serve only as crude approximations of reality. Intuitively, one would expect an elliptically shaped movement model, and we will show in the next section that the data-driven models take on elliptical shapes.

Algorithm 7.1 Computation of movement samples

Input: Data set $\mathcal{D} = \{(\mathbf{x}^{(t_i)}, v^{(t_i)})\}_{i=1}^{n}$
Output: Set $\mathcal{S}_{t_\Delta}$ of attained positions in time t_Δ including initial velocities
for $t_A < t_B < t_C$ s.t. $t_A = t_B - t_\delta$ and $t_C = t_B + t_\Delta$ **do**
  $\mathbf{x}_C^{\text{local}} = \mathbf{g}(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)$
  $\mathcal{S}_{t_\Delta} = \mathcal{S}_{t_\Delta} \cup \{(\mathbf{x}_C^{\text{local}}, v^{(t_B)})\}$
end for

7.2.3 Probabilistic Movement Models

We now describe how to compute probabilistic movement models from positional data. Given a trajectory (or a set of many trajectories), we first extract triplets as shown in Figure 7.1. Let $(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C)$ be such a triplet of positions at timestamps t_A, t_B, and t_C such that $t_A < t_B < t_C$. Henceforth, we use $\mathbf{x}_B$ as a shorthand for $\mathbf{x}_{t_B}$. We denote the time difference of t_A and t_B as $t_\delta = t_B - t_A$ and the difference of t_B and t_C as $t_\Delta = t_C - t_B$. Here, $\mathbf{x}_B$ describes the current position, $\mathbf{x}_A$ is used to estimate the current direction in which the player is moving and $\mathbf{x}_C$ denotes the position in the future. Hence, the next position $\mathbf{x}_C$ describes the ability to move within a given time horizon t_Δ. Collecting multiple next positions $\mathbf{x}_C$, represented in the local coordinate system as $\mathbf{x}_C^{\text{local}}$, allows for creating a movement model. Such a model is then able to quantify the likelihood of a possible next position relative to its current position.

To obtain the next position $\mathbf{x}_C^{\text{local}}$, the triplet is transformed into the local coordinate system. The transformation realizing this first subtracts the current position $\mathbf{x}_B$ from all points of the triplet. This way, the triplet is centered at the current position. Then, the triplet is rotated such that the last position, *i.e.*, $\mathbf{x}_A$, is aligned with the x_1-axis of the local coordinate system. Let $\mathbf{g}(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = \mathbf{x}_C^{\text{local}}$ be this transformation. Mathematically, this transformation can be expressed using polar coordinates as

$$\mathbf{x}_C^{\text{local}} = \mathbf{g}(\mathbf{x}_A, \mathbf{x}_B, \mathbf{x}_C) = \begin{pmatrix} r \cdot \cos(\theta) \\ r \cdot \sin(\theta) \end{pmatrix}, \tag{7.3}$$

where r is a distance given by $r = \|\overrightarrow{\mathbf{x}_B\mathbf{x}_C}\|_2$ and θ is a signed angle given by $\theta = \measuredangle(\overrightarrow{\mathbf{x}_A\mathbf{x}_B}, \overrightarrow{\mathbf{x}_B\mathbf{x}_C})$. After processing multiple triplets along a trajectory for fixed time differences t_δ and t_Δ, we store the corresponding next positions, that are represented in the local coordinate system, in a set $\mathcal{S}_{t_\Delta}$. This approach is summarized in Algorithm 7.1.

Having obtained the set $\mathcal{S}_{t_\Delta}$, it is possible to define a probability distribution over possible player whereabouts given her position and initial velocity. This can be done with a two-dimensional kernel density estimate (KDE). Due to practical considerations, we suggest discretizing the speed range

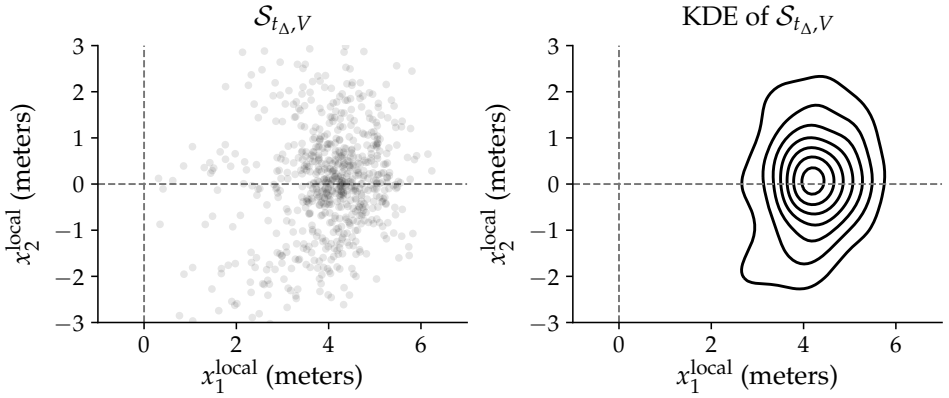

Figure 7.3: An example of the set $\mathcal{S}_{t_\Delta,V}$ with a time horizon of $t_\Delta = 1s$ and a speed range of 14-20 km/h and the corresponding kernel density estimate.

and include it in the model at a lower level of granularity. To this end, we define a subset of points

$$\mathcal{S}_{t_\Delta,V} = \{(\mathbf{x}, v) \mid v \in V\} \subseteq \mathcal{S}_{t_\Delta}$$

for a range of velocity values in the interval $V = [v_{\min}, v_{\max}]$ and compute the individual movement model using a KDE-based on samples from this set. We obtain several KDEs depending on different velocity ranges denoted as $p^{\text{KDE}}_{t_\Delta,V}$. To evaluate the likelihood of attaining a given position $\mathbf{x} \in \mathbb{R}^2$ in time t_Δ, we use

$$p_{t_\Delta}\left(\mathbf{x} \mid \mathbf{x}_{t-t_\delta}, \mathbf{x}_t, v_t\right) = p^{\text{KDE}}_{t_\Delta,V}\left(\mathbf{g}\left(\mathbf{x}_{t-t_\delta}, \mathbf{x}_t, \mathbf{x}\right)\right) \tag{7.4}$$

for $v_t \in V$. We introduce an extra conditioning on the previous player's position $\mathbf{x}_{t-t_\delta}$ utilized to estimate the direction (angle θ) in which the player moves. Figure 7.3 (left) presents a set of samples collected, and Figure 7.3 (right) a corresponding movement model based on a KDE.

The model relies on a particular discretization of the speed range $\mathcal{V}$ denoted by $\tilde{\mathcal{V}}$. Analogously, different models are obtained for different values of the time horizon parameter t_Δ. Those values are summarized in the set $\tilde{\mathcal{T}}$.

In some cases, the triplets of points used to estimate the model can contain outliers. They may stem from an interruption during a match (*e.g.*, due to a foul or corner kick) or errors in the data collecting process. Hence, triplets containing outliers should be discarded. Finally, given that a player's ability to move should be symmetric with respect to the direction she is facing. This can be achieved by augmenting the set with $(\mathbf{x}', v)$ using $\mathbf{x}' = (x_1, -x_2)$ for each sample $(\mathbf{x}, v) \in \mathcal{S}_{t_\Delta,V}$.

7.2.4 Complexity Analysis

We now consider the complexities for training, prediction, and memory consumption of the proposed approach for a single player. Let $n_{t_\Delta,V} \in \mathbb{N}$ denote the number of samples within the set $\mathcal{S}_{t_\Delta,V}$ of transformed locations generated by Algorithm 7.1 when conditioning on a specific time delta t_Δ and speed v, *i.e.*, $n_{t_\Delta,V} = |\mathcal{S}_{t_\Delta,V}|$. The complexity of training a KDE is equivalent to the complexity of obtaining the training data, which is the cardinality of the set $\mathcal{S}_{t_\Delta,V}$ and thus equal to $\mathcal{O}(n_{t_\Delta,V})$. Since there is a separate KDE for every time horizon t_Δ and speed interval V, the complexity of training all KDEs for a single player is $\mathcal{O}(\sum_{V\in\tilde{\mathcal{V}}}\sum_{t_\Delta\in\tilde{\mathcal{T}}} n_{t_\Delta,V})$ and thus linear in the player's trajectory data. The complexity of predicting, *i.e.*, obtaining the probability for a given position, using a KDE is $\mathcal{O}(n_{t_\Delta,V})$. Clearly, it holds that the larger the training set, the better the model. However, increasing the size of samples $n_{t_\Delta,V}$ makes it prohibitive to use the individual movement models based on KDEs in real-time scenarios. Considering the memory demand of the KDE-based approach, it becomes obvious that all samples are needed as the KDE is a non-parametric method. Hence, $\mathcal{O}(\sum_{V\in\tilde{\mathcal{V}}}\sum_{t_\Delta\in\tilde{\mathcal{T}}} n_{t_\Delta,V})$ points need to be stored.

7.2.5 Empirical Results

There are two typical ways of collecting positional data in sports. The first way is to attach sensors to players and ball to monitor their positions (Grün et al., 2011; Mutschler et al., 2013). The second way is to use computer vision algorithms for retrieving players' and ball's trajectories in consecutive frames (Barris and Button, 2008; D'Orazio and Leo, 2010).

Data

The positional soccer data from the German Bundesliga we use in the experiments stem from the latter and is recorded at 25 Hz.[2] This usually yields over $25 \cdot 60 \cdot 90 = 135{,}000$ samples (due to possible extra time by the end of each half) for a single match. The dimensions of a soccer field are 105.0 by 68.0 meters, and the coordinates of positions in the trajectory data are given relative to the origin of the field, which is set to $(0, 0)$. Hence, player coordinates (x_1, x_2) are within $\mathcal{A} = [-52.5, +52.5] \times [-34.0, +34.0] \subset \mathbb{R}^2$.

Baselines

As baselines, we use the Voronoi, Taki & Hasegawa, and Fujimura & Sugihara models, which we introduced in Section 7.2. Except for the Voronoi-based approach, all other approaches involve user-defined parameters

[2]We thank Deutsche Fußball Liga and Sportcast GmbH for providing the positional data.

Table 7.1: Distribution of speed classes for three different players.

Speed	Range (km/h)	Goalkeeper	Defender	Midfielder
Stand	< 1	27.19%	11.25%	12.01%
Walk	$1 - 7$	66.68%	53.22%	50.90%
Jog	$7 - 14$	5.44%	27.57%	28.11%
Run	$14 - 20$	0.57%	5.97%	6.36%
Sprint	> 20	0.12%	2.00%	2.62%

that need to be specified. For Taki & Hasegawa, the acceleration parameter $a_{\max}$ can be derived from the corresponding speed samples v_t using $a_t = \frac{1}{h}(v_{t+h} - v_t)$. Here, these are computed for a time horizon of $h = 1s$ using data from a single match. Based on this, we set $a_{\max} = 4.2\ m/s^2$, which is equal to the 0.999-quantile of the derived values. The quantile instead of the maximum acceleration observed is used to ignore outliers. The model by Fujimura & Sugihara includes two parameters, α and $v_{\max}$. We use $\alpha = 1.3$, which is the value proposed by Fujimura and Sugihara (2005), and $v_{\max} = 8.0\ m/s$. The latter corresponds to the 0.999-quantile of the observed speed values (analogously as in the case of $a_{\max}$ parameter in the previous model).

Setup

To compute the individual movement models presented in Section 7.2.3, we use $t_\delta = 0.2s$ and $t_\Delta = 1s$ in Algorithm 7.1. We use five different speed intervals shown in Table 7.1. Note that such a discretization is a common way to bin velocities to account for sparseness in real data, as the number of samples per speed interval may vary significantly (Lago-Peñas et al., 2009; Coutts et al., 2010; Gudmundsson and Wolle, 2014). Table 7.1 also presents speed distributions for three different players: a goalkeeper, a defender, and an attacking midfielder. On average, field players walk and jog and save their energy for only a few sprints.

Results

Movement estimates for these three player roles are presented in Figure 7.4 using a Gaussian KDE with bandwidth equal to 1.0 for simplicity. Note that there are small but distinctive differences between players' ability to move.[3] For example, the goalkeeper has a significantly lower probability of reaching distant positions compared to the field players. However, the reason lies not in her ability to move but in the lack of corresponding observations:

[3] Differences in Table 7.1 between the defender and the midfielder are significant according to a χ^2-test.

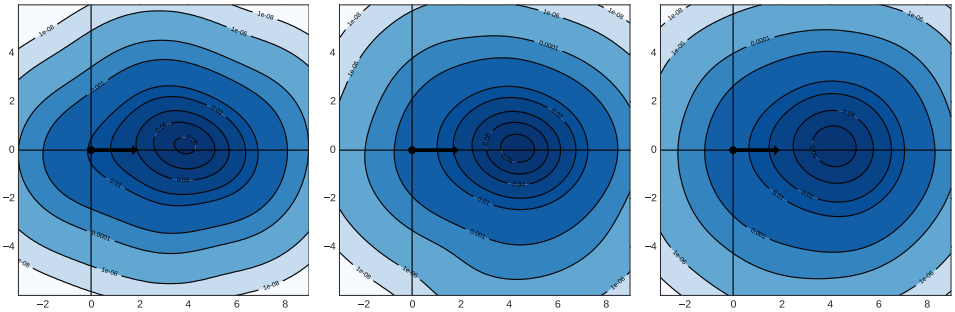

Figure 7.4: Individual movement models for three different players with initial speed in the range $14 - 20\ km/h$ in the direction of the x_1-axis: goalkeeper (left), defender (center), and midfielder (right).

goalkeepers hardly push forward and usually cover a smaller radius than field players. The figures clearly show that the midfielder covers a wider area and is, on average, moving faster than her peers. The few data samples collected for the goalkeeper could be balanced with an average model; see discussion in Section 7.4.

7.3 Zones of Control

7.3.1 Motivation

Movement models can be used to compute zones of control (or dominant regions) of individual players and teams as a whole (Taki and Hasegawa, 2000; Gudmundsson and Wolle, 2014; Horton et al., 2015). Below we formally define dominant regions for the models presented in the previous section. To do so, it is beneficial to recall the definition of the traditional movement models that are inspired by physical laws. The definition of probabilistic models is analogous and discussed later.

Let the function Γ yield the time $s \in \mathbb{R}_{\geq 0}$ needed to reach position $\mathbf{x} \in \mathbb{R}^2$ for a player j at position $\mathbf{x}_t^j$ moving with velocity v_t^j in a given direction, *i.e.*, $\Gamma\left(\mathbf{x} \mid \mathbf{x}_t^j, v_t^j\right) = s$. This function is specific to a given physical model governing player movements. In other words, for a given player, function Γ yields the minimal time s that satisfies Equations (7.1) and (7.2) for the Taki & Hasegawa and the Fujimura & Sugihara models, respectively. In Taki and Hasegawa (2000), the concept of a player's zone of control is defined as follows.

Definition 7.1. *The* zone of control *of player j is defined as the subset A^j of the playing area $\mathcal{A}$, where player j can arrive before any other player $j' \neq j$.*

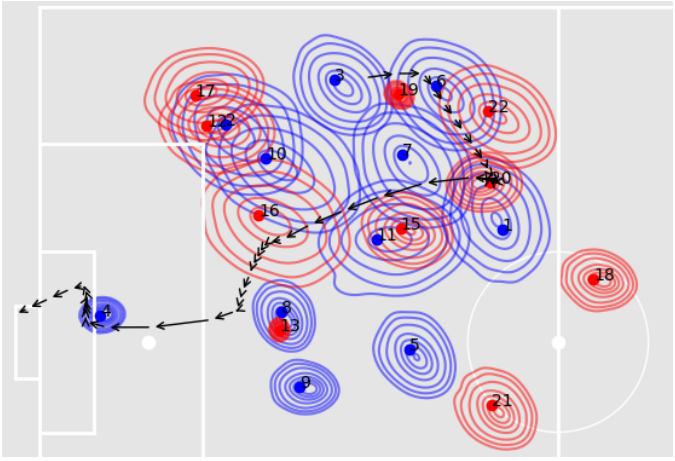

Figure 7.5: Players with their movement models on a pitch. The ball trajectory is depicted as black arrows.

Formally, this is to say that $A^j \subseteq \mathcal{A}$ is defined such that $\forall \mathbf{x} \in A^j$:

$$j = \arg\min_j \Gamma\left(\mathbf{x} \mid \mathbf{x}_t^j, v_t^j\right).$$

It should be noted that interdependencies between players may be complex enough to produce a player's zone of control that is not a single connected region (Taki and Hasegawa, 2000).

The zone of control of a team is defined analogously. Note that a different perspective is taken in our setup by considering probabilistic movement models for a given time horizon. That is, the zones of control are derived based on density functions of possible players' whereabouts. Therefore, we obtain probability distributions of individual players over the playing area. This is depicted in Figure 7.5. The computation of those regions using probabilistic movement models is presented in detail below.

7.3.2 Problem Formulation

Let $p_{t_\Delta}^j\left(\mathbf{x} \mid \mathbf{x}_{t-t_\delta}^j, \mathbf{x}_t^j, v_t^j\right)$ be the movement model of the j-th player as introduced in Equation (7.4). It quantifies the likelihood of player j to reach position $\mathbf{x}$ given her current $\mathbf{x}_t^j$ and last position $\mathbf{x}_{t-t_\delta}^j$, velocity v_t^j, and time horizon t_Δ. The position $\mathbf{x}$ is controlled by the player that has the highest likelihood, *i.e.*,

$$j = \arg\max_{j \in \{1,2,\ldots,22\}} \; p_{t_\Delta}^j\left(\mathbf{x} \mid \mathbf{x}_{t-t_\delta}^j, \mathbf{x}_t^j, v_t^j\right),$$

Algorithm 7.2 Exact computation of the zones of control

Input: Movement models $p^j_{t_\Delta}$ for players $j = 1, 2, \ldots, 22$
Output: Sets $A^1, A^2, \ldots, A^{22}$
for $j = 1, 2, \ldots, 22$ **do**
    $A^j = \{\mathbf{x} \in \mathcal{A} \mid \Xi_{t_\Delta}(\mathbf{x}) = j\}$
end for

Algorithm 7.3 Finite approximation of the zones of control

Input: Movement models $p^j_{t_\Delta}$ for players $j = 1, 2, \ldots, 22$
Output: Set $\mathcal{B}$
$\mathcal{B} = \emptyset$
for $\mathbf{x} \in \mathcal{G}$ **do**
    $\mathcal{B} = \mathcal{B} \cup \{(\mathbf{x}, \Xi_{t_\Delta}(\mathbf{x}))\}$
end for

assuming that there are 22 players. Hence, we can define a function

$$\Xi_{t_\Delta} : \mathcal{A} \to \{1, 2, \ldots, 22\}, \quad \mathbf{x} \mapsto \arg\max_{j \in \{1,2,\ldots,22\}} p^j_{t_\Delta}\left(\mathbf{x} \mid \mathbf{x}^j_{t-t_\delta}, \mathbf{x}^j_t, v^j_t\right)$$

that determines the index of the dominating player. Thus, the zone of control of a player j is given as the set of all points $A^j = \{\mathbf{x} \in \mathcal{A} \mid \Xi_{t_\Delta}(\mathbf{x}) = j\}$ that are controlled by her. It should be noted that ties may occur if the likelihood of two or more players is equal. If ties are broken, then the set $\{A^1, A^2, \ldots, A^{22}\}$ is a partition of $\mathcal{A}$. The procedure is summarized in Algorithm 7.2.

7.3.3 Approximating Zones of Control

Unfortunately, running Algorithm 7.2 is not practicable. This is because the set $\mathcal{A} \subset \mathbb{R}^2$ is not iterable since it is uncountable. A typical workaround is to use a finite approximation of the playing area (Nakanishi et al., 2009; Lucey et al., 2012; Narizuka et al., 2014; Franks et al., 2015). Let $\mathcal{G} \subset \mathcal{A}$ be a finite grid over $\mathcal{A}$ containing $n_{x_1} \cdot n_{x_2}$ equally spaced points in $\mathcal{A}$ with (axis-aligned) distances Δ to each other. The player domination is then computed using $\mathcal{G}$ rather than $\mathcal{A}$, which yields a finite approximation of the zones of control with precision Δ. The smaller Δ is, the better the approximation. The procedure is presented in Algorithm 7.3. For visualization purposes, the set $\mathcal{B} = \{(\mathbf{x}, \Xi_{t_\Delta}(\mathbf{x})) \mid \mathbf{x} \in \mathcal{G}\}$ can then be used to compute the zones of control by assigning each position $\mathbf{x} \in \mathcal{A}$ the same label as its closest neighbor from the grid $\mathcal{G}$.

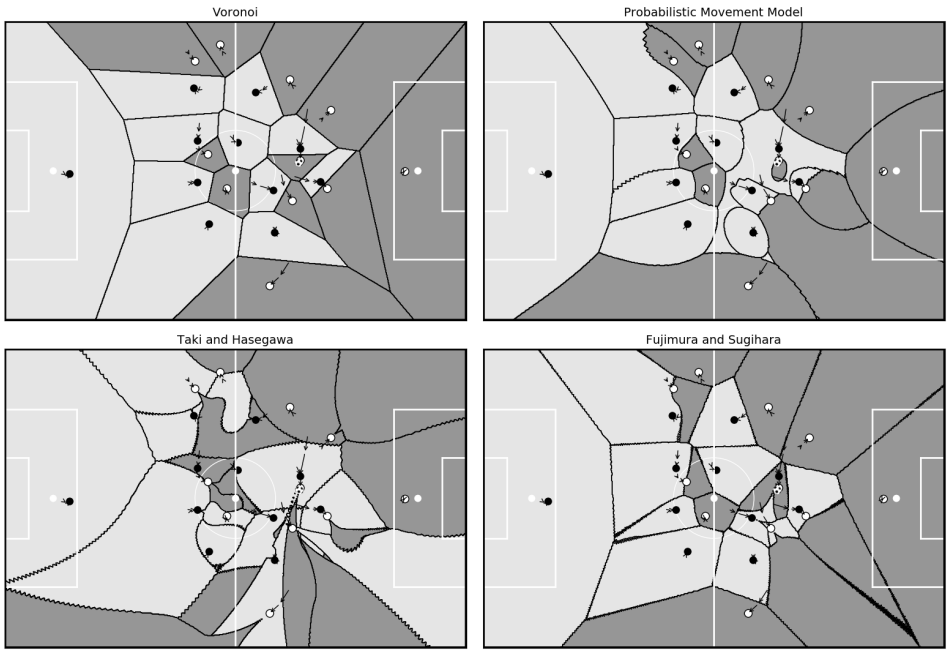

Figure 7.6: Zones of control for different movement models. Black plays from left to right. The two arrows attached to player positions indicate their whereabouts one and two seconds ago, respectively.

7.3.4 Empirical Results

We now compare zones of control obtained by a Voronoi tessellation, the movement models by Taki & Hasegawa and Fujimura & Sugihara, respectively, and the proposed data-driven movement model for the same situation. Figure 7.6 shows the resulting regions where arrows indicate directions and velocities of movements.

The top left shows a Voronoi tessellation and implements the assumption that every player is able to run in any direction equally fast, hence ignoring actually observed movements. In other words: the closest player always wins, and borders of controlled zones are half cuts between players. The assumption leads to implausible zones of control, as we showcase on the example of the white team playing right to left. The white player on the right-wing, for example, has a large zone, although she is running towards the center of the pitch. Most of the controlled area of that player lies in her back, and she would need to turn before being able to head in that direction. The Voronoi model clearly overestimates the right-wing of the white team. By contrast, their left-wing is underestimated. Although the left-winger pushes forward and although her direct opponents only move slowly and head towards the center of the pitch, her zone is small. In contrast to Voronoi tessellations, the proposed approach in the upper right part of the figure

clearly eliminates the depicted limitations. For the white team, the zone of the right-winger is realistically small, and the zone of the left-winger realistically large.

Computing controlled zones using the movement model by Taki & Hasegawa leads to the bottom left figure. Borders between zones are often curly as a direct consequence of the nested circles that originate from the assumption that players may accelerate in any direction unbounded (see Figure 7.2). The zone of the white left-winger evolves drop-like from the actual player position. The underlying movement model also assigns a big part of the right half of the pitch to the black team, although white players are closely positioned, and some of them even move in this direction. Figure 7.6 exhibits the limitations of the approach by Taki & Hasegawa.

The movement model by Fujimura & Sugihara corrects some of the limitations of the model by Taki & Hasegawa, and, correspondingly, the bottom right part of Figure 7.6 appears more realistic. For instance, similar to the proposed approach, the zone of the white left-winger seems more appropriate than the Voronoi-based zone. Nevertheless, this model has other problems, as can be seen on the right-wing of the white team. The zone of the winger has shrunk to almost zero, although her opponent is still far away, and both are moving slowly.

To sum up, out of the four movement models, only the proposed approach leads to realistic controlled zones that are in line with player movements and distances. Either of the competitors suffers from oversimplified assumptions in the movement models and yield unrealistic zones of control. Analyses that build upon one of the three competitors are likely to be crude as they rely on rough approximations of reality. We include more examples of the methods in Appendix A.

7.4 Discussion

The previous sections show theoretically and empirically that existing movement models suffer from implausible assumptions. Particularly in the previous section, we observe the clear influence of such oversimplifications in the resulting zones of control for Voronoi tessellations and underlying movement models by Taki & Hasegawa and Fujimura & Sugihara.

The idea of this chapter is to avoid cumbersome definitions of complex physics (and possibly oversimplifications) by simply observing player movements. We propose a purely data-driven movement model that intelligently combines all player movements into a probabilistic model. Depending on the application at hand, either the full distribution, some quantile thereof, or the convex hull of observed positions can be processed to compute reachable positions in a predetermined time. Further exploiting the probabilistic nature of the model may provide confidence to possible movements. Empiri-

cally, the resulting zones of control are intuitive and can be straightforwardly interpreted with player movements and, hence, constitute a realistic picture of a situation.

As a remark, we note that the zones of control for the three baseline approaches are identical when no player is moving. This can be seen by setting $v = 0$ in Equations (7.1) and (7.2) for Taki & Hasegawa and Fujimura & Sugihara, respectively, which then reduces the resulting zones of control to a Voronoi tessellation. The time needed to reach an arbitrary position is now a strictly increasing function of the distance to that position. As Figure 7.2 shows, the greater the players' velocities, the greater the differences of the resulting zones.

However, also note that using positional data for estimating the movements of players also comes with limitations. The angle estimation from trajectory data used in Equation (7.3), for instance, is based on the assumption that players always move forward. In other words, the model assumes that the direction a player is facing is in line with her movement. This is not always the case, as particularly goalkeepers often move backward. Thus, the model would overestimate or underestimate the time needed for turning around depending on the actual change of direction. A possible remedy could be a better approximation of the angle θ rather than as used in Equation (7.3) or devising the angle from an auxiliary data source. Using positional data alone is, however, not sufficient to solve this matter.

The goalkeeper serves as an example of another problem of the proposed approach as she is hardly running at full speed. Thus, just by observing her movements on the pitch, one will hardly be able to assess her full potential. The same problem occurs with players that are substituted for the first time as the proposed approach does not apply off-the-shelf to unseen players. The problem is also known as the *cold-start problem*, and similar instances occur in recommendation scenarios (see, *e.g.*, Son (2016)). To overcome this problem, a two-component mixture model can be used. The first component utilizes the actual (and continuously updated) movement model $p^j_{n_j}$ of the new player j, which is learned on n_k points. The second component is an *average* model p^{avg} over all players (with a similar role) and their data points. The idea is to blend the personalized component with the average component until the former is accurate enough to be used alone. Hence, the model is given by a convex combination of movement models

$$p^j = \lambda \cdot p^j_{n_k} + (1 - \lambda) \cdot p^{\text{avg}}, \qquad \text{with} \qquad \lambda = \min\left(\frac{n_k}{n}, 1\right).$$

Suppose $n_k = 0$, then only the average model will be used. Once n_k exceeds the number of data points n, $\lambda = 1$ and the average model is weighted by zero and hence automatically deactivated as desired. The required number of observations depends both on the domain and a player's speed. In the

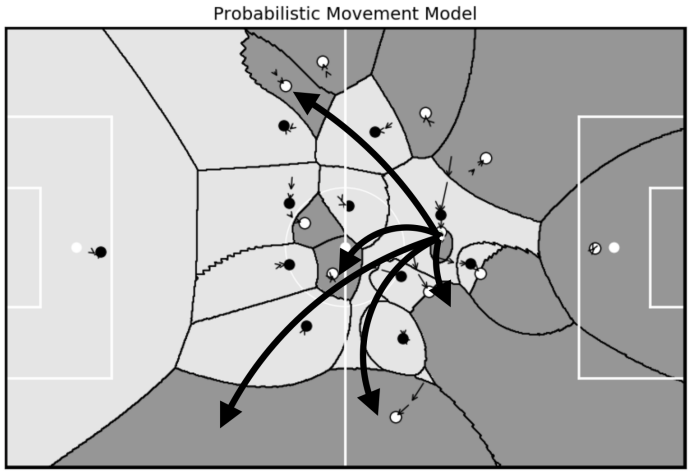

Figure 7.7: A mock-up showing possible passes.

case of soccer and for a given speed range, several thousands of samples appear sufficient to produce satisfactory results. For a field player, those samples can, for instance, be collected in a single match. However, in the case of a goalkeeper, it is recommended to always maintain an additional average movement model due to the small number of samples for higher values of initial velocities as she mostly stands or walks during a match.

There are many possible use cases where realistic movement models may give an edge toward existing techniques. For instance, player performance indices that ground on the ability to move (Taki et al., 1996; Ueda et al., 2014) may be revised accordingly. Similarly, player ratings that measure to what extend their controlled zone contributes to the overall area controlled by their team (Link et al., 2016; Harmon et al., 2016) may be revisited. Figure 7.7 shows a potential application that deals with estimating probabilities of passing and pass completion given the context of the ball possessing player to test the hypothesis that players try more difficult passes when they have enough space. While the space is directly given by their zone of control, pass interception and pass completion probabilities could be conditioned on the available area to shed light on which player to attack in what situations and also where to position the own defenders to intercept and defend the receiving player possibly.

Along these lines, there is also the prediction of pass outcomes (Nakanishi et al., 2009). The idea is to split the ball's trajectory into small units that are processed one after another. For every unit, the probability that a player reaches the ball's position during the lifespan of the unit is computed. If an opposing player fulfills this criterion, she intercepts the ball, and the computation terminates. If no player intercepts the ball, the pass is completed after processing the final unit.

7.5 Conclusion

We proposed a novel data-driven method for estimating individual movement models using positional data. The model is generated by conditioning a player's whereabouts after a given time on her initial position and velocity. We obtained tables of reachable (x_1, x_2) coordinates for every velocity and time interval and proposed to turn these tables into a probabilistic movement model using kernel density estimate. Movement models were computed for every player individually, and the computation could be distributed on many machines to compute movement models for many players and process many games at once. Empirically, we showed the limitations of existing movement models and exemplified the contribution on the example of zones of control. Computing these zones using existing approaches led to crude approximation due to oversimplified assumptions in the respective models. By contrast, the proposed movement models led to realistic and intuitive zones of control.

Chapter 8

Contextual Movement Models

In the previous chapter, we proposed non-parametric probabilistic movement models to quantify likelihoods within trajectories of movements using kernel density estimates (KDEs). Although this solution is purely data-driven and renders assumptions on underlying physics obsolete, we still need to distinguish initial conditions (*e.g.*, bins of velocities and time horizon intervals). We addressed this by maintaining many, possibly differently parameterized, models. This turns the advantage of kernel density estimation into a drawback: being non-parametric by design, predictive performance does increase proportionally with data, but every new data point also increases the computation time for the prediction. The same holds true for memory requirements. This is rather impractical.

In this chapter, we provide a remedy to the issues of the KDE. We turn conditional normalizing flows into novel movement models that consist of only a single contextualized probabilistic model. This contextual model has consistent prediction accuracies over all contexts, and its computation time is independent of the amount of data, allowing for real-time applications. Normalizing flows (NFs), as introduced in Chapter 3, provide a state-of-the-art framework for learning densities using an invertible deep neural network.

The key contributions of this chapter are as follows. We (i) extend the approach of conditional normalizing flows (Lu and Huang, 2020) in the remainder to incorporate context into flow-based movement models. Hence, our flow-based movement model is actually only *a single model* which can be conditioned on several kinds of contexts, particularly more complex ones than just bins of velocities and time horizon intervals. Furthermore, we (ii) show how to incorporate additional contextual information such as the relative positions of the other players. Empirically, we (iii) show that the proposed contextual models predict the players' trajectories more accurately as measured in terms of log-likelihood. Finally, we (iv) demonstrate that our contribution is efficient and allows for (near) real-time predictions independently of the amount of data.

8.1 Related Work

In Section 7.1 we already discussed related work regarding movement models. Hence, we now focus on normalizing flows, which recently emerged as an attractive approach to learning densities. A significant advantage over other methods for learning densities using neural networks is that they allow the direct maximization of data log-likelihood. Alternatives such as generative adversarial networks (GANs) (Goodfellow et al., 2014) and variational autoencoders (VAEs) (Kingma and Welling, 2014; Rezende et al., 2014) use instead surrogate learning objectives, such as the GAN adversarial loss and the VAE evidence lower bound (ELBO), for this task. While this does not hamper their use as generative models, they are inadequate for estimating the likelihood of a given data point.

In the last few years, several different flow-based models were proposed. A particular family of models, derived from NICE (Dinh et al., 2015), is appealing due to their computationally efficient nature while being simple to implement. Improvements such as RealNVP (Dinh et al., 2017) and Glow (Kingma and Dhariwal, 2018) were proposed over the years, achieving state-of-the-art performance to model complex, high-dimensional distributions while keeping the attractive properties of NICE. Similarly, autoregressive flow models such as NAF (Huang et al., 2018) and B-NAF (De Cao et al., 2019) have been proposed. Particularly interesting are conditional normalizing flows (Winkler et al., 2019; Lu and Huang, 2020), an improvement over Glow, allowing them to model distributions conditioned on continuous variables. For an in-depth analysis of these and their relation to other flow-based models, we refer the reader to Papamakarios et al. (2021).

8.2 Learning Flow-based Movement Models

8.2.1 Preliminaries

We consider the same notational setup as in Section 7.2.3. Hence, $(\mathbf{x}_t)_{t\in\mathbb{R}_{\geq 0}}$ represents a trajectory of positions $\mathbf{x}_t \in \mathbb{R}^d$ at time t. The positional data might come with additional information, the so-called context $\mathbf{c}_t \in \mathbb{R}^{d_c}$ at time t. An example for context is the velocity vector $\mathbf{v}_t \in \mathbb{R}^d$, or its magnitude, known as speed $v_t = \|\mathbf{v}_t\|_2$, respectively.

To make the movement model location-invariant, we consider a local coordinate system centered at the player's current position and along the last movement direction, just as in Chapter 7. The transformation $\mathbf{g}$, as introduced in Equation (7.3), allows us to transform a point of interest into a local coordinate system given the current and past positions, which are $\mathbf{x}_t$ and $\mathbf{x}_{t-t_\delta}$, respectively.

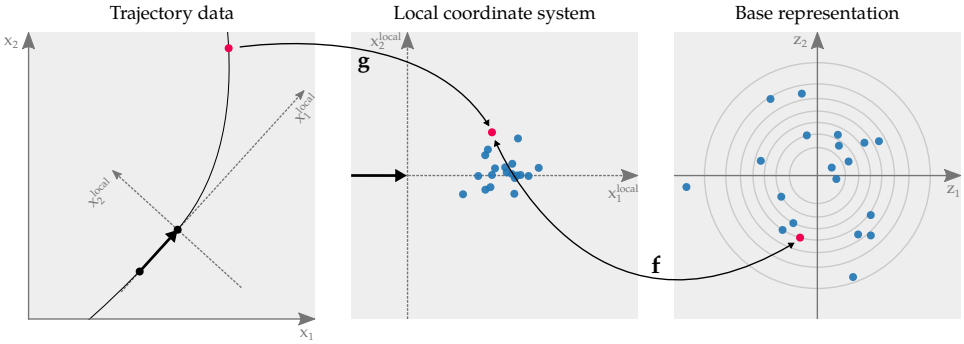

Figure 8.1: A summary of our approach. Left: triplets are extracted from a trajectory using some current and previous position (black) as well as a future position (red) reached within the time horizon t_Δ. Center: the future position is remapped using $\mathbf{g}$ into a local coordinate system. The coordinates are centered at the current position and aligned with the previous position. Right: points reached in time t_Δ are transformed using a conditional normalizing flow $\mathbf{f}$. The flow $\mathbf{f}$ is optimized to ensure that points under this transformation are distributed according to a Gaussian base distribution.

8.2.2 Movement Models without Context

Instead of constructing a probabilistic movement model based on a kernel density estimate as in Chapter 7, we now use a (conditional) normalizing flow. Hence, a position $\mathbf{x}$ is represented in a local coordinate system as $\mathbf{x}^{\text{local}}$ using the transformation $\mathbf{g}$ before the normalizing flow $\mathbf{f}$ further transforms it to the base representation $\mathbf{z}$ that matches a base distribution. An example is depicted in Figure 8.1. Hence, the likelihood of a position $\mathbf{x}$, that is, t_Δ in the future, can be evaluated via

$$p_{t_\Delta}\left(\mathbf{x} \mid \mathbf{x}_{t-t_\delta}, \mathbf{x}_t, v_t\right) = p^{\text{Flow}}_{t_\Delta, V}\left(\mathbf{g}\left(\mathbf{x}_{t-t_\delta}, \mathbf{x}_t, \mathbf{x}\right)\right),$$

where p^{Flow} uses the transformation $\mathbf{f}$ as outlined in Equation (3.3). In this scenario, we still have a separate model for each time horizon t_Δ and speed v_t. As an architecture for our flow model, we consider Glow (Kingma and Dhariwal, 2018). Glow is built from three main transformations: activation normalization (actnorm), 1×1 invertible convolution, and affine coupling. Those transformations, introduced in Section 3.3, are employed in a multi-scale architecture, reshaping the image tensors to have fewer pixels with more channels, referred to as *squeezing*. The channels are then split, and further operations are only performed on half of them. This squeezing and splitting scheme is performed several times for scalability. Squeezing, however, is not suitable for vectorial data, including d-dimensional positions. Furthermore, the dimensionality of our setting curtails the computational performance gains from splitting, allowing us to use the entire vector in

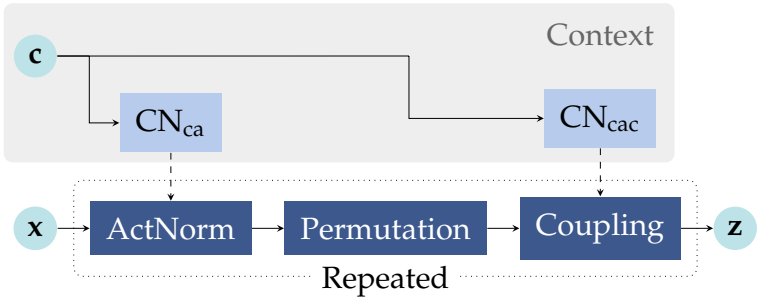

Figure 8.2: Architecture of the (contextual) flow-based movement model: the composition of the actnorm, permutation, and coupling transformations is repeated multiple times to build the flow. The contextual part is shaded in gray. The input to the flow is denoted as $\mathbf{x}$ and the output of the transformation is $\mathbf{z}$. The base representation $\mathbf{z}$ follows a predefined (base) distribution. In the conditional variant, $\mathbf{c}$ denotes the contextual information, which can be used to characterize the movement being modeled, such as speed. The networks CN_{ca} (conditional actnorm) and CN_{cac} (conditional affine coupling) augment the respective transformations with contextual information.

every transformation of the flow. An overview of the non-contextualized architecture we use is depicted in Figure 8.2 (non-grayed area).

A major benefit of this approach is the ability to compute predictions independently of the data used for training. It only requires a number of computations proportional to the data dimensionality d and the number of transformations L employed. In other words, computing the likelihood of a data point takes $\mathcal{O}(dL)$ time. Additionally, only the model parameters have to be stored in memory, resulting in a significantly smaller memory footprint than the KDE-based movement model proposed in Chapter 7 when dealing with large data sets.

8.2.3 Contextual Movement Models

Our goal is now to unify the separate models and to propose s single contextual probabilistic movement model

$$p\left(\mathbf{x} \mid \mathbf{x}_{t-t_\delta}, \mathbf{x}_t, \mathbf{c}_t\right) = p^{\text{CFlow}}\left(\mathbf{g}\left(\mathbf{x}_{t-t_\delta}, \mathbf{x}_t, \mathbf{x}\right) | \mathbf{c}_t\right).$$

In this setting, the contextual information $\mathbf{c}_t$ at time t is, *e.g.*, the speed v_t and time horizon t_Δ. Therefore, the transformations employed in the normalizing flow should be able to take into account this additional information. A conditional flow-based model that appropriately addresses those requirements is conditional Glow (c-Glow) (Lu and Huang, 2020). As the name suggests, c-Glow is based on Glow (Kingma and Dhariwal, 2018). Nevertheless, the goal is to derive a contextualized model that can be conditioned on arbitrary contexts. The most significant change is the

addition of a *conditioning network*, denoted by $\mathrm{CN}(\cdot)$, which replaces the parameters of the actnorm and affine coupling layers with surrogates that are predicted from the context $\mathbf{c} \in \mathbb{R}^{d_c}$. Hence, a conditional probability density can be devised whose log-variant is given by

$$\log p(\mathbf{x}|\mathbf{c}) = \log p_z(\mathbf{z}) + \sum_{j=1}^{L} \log \left| \det J_{\mathbf{f}_j^{(\theta(\mathbf{c}))}}(\mathbf{z}_{j-1}) \right|.$$

Here, $\log p(\mathbf{x}|\mathbf{c})$ is the conditional counterpart to $\log p_x(\mathbf{x})$ as introduced in Equation (3.4), and $\log p_z(\mathbf{z})$ still refers to the base distribution, *i.e.*, a standard Gaussian. The parameters $\theta(\mathbf{c})$ of the transformations $\mathbf{f}_j^{(\theta(\mathbf{c}))}$ are now dependent on the context $\mathbf{c}$. Note that this does not change the computation of the log-determinant of the Jacobian in both cases. The new transformations with the addition of the conditioning network $\mathrm{CN}(\cdot)$ are detailed below.

Conditional actnorm. The scaling and offset vectors $\mathbf{a}$ and $\mathbf{b}$ are now computed by $\mathrm{CN}_{\mathrm{ca}} : \mathbb{R}^{d_c} \to \mathbb{R}^{2d}$ as

$$(\mathbf{a}, \mathbf{b}) = \mathrm{CN}_{\mathrm{ca}}(\mathbf{c})$$

and the transformation is then carried out as in Equation (3.5). The original initialization procedure for $\mathbf{a}$ and $\mathbf{b}$ (*cf.* Section 3.3) is no longer needed.

Conditional affine coupling. This transformation already computes its scaling $\mathbf{a}$ and offset $\mathbf{b}$ parameters from $\mathrm{NN}_{\mathrm{cac}} : \mathbb{R}^{d'+d} \to \mathbb{R}^{d}$, similar to the previously used $\mathrm{NN}_{\mathrm{ac}}(\cdot)$ in Equation (3.6). The change introduced by conditioning on the context $\mathbf{c}$ is an additional input to $\mathrm{NN}_{\mathrm{cac}}(\cdot)$, computed by $\mathrm{CN}_{\mathrm{cac}} : \mathbb{R}^{d_c} \to \mathbb{R}^{d}$, as

$$(\log \mathbf{a}, \mathbf{b}) = \mathrm{NN}_{\mathrm{cac}}(\mathbf{z}_2, \mathrm{CN}_{\mathrm{cac}}(\mathbf{c})),$$

followed by the same operations as described for the affine coupling in Section 3.3.

Figure 8.2 provides an overview of the complete normalizing flow model with and without the conditioning on the context $\mathbf{c}$. As shown in Figure 8.1, learning contextual flow-based probabilistic movement models proceeds as follows. Given positional data, triplets (A, B, C) are extracted from the trajectories as depicted in Figure 7.1. The future position C in a time horizon t_Δ of interest is then represented in a local coordinate system for spatial invariance. The positions which can be reached in time t_Δ are then transformed, using our normalizing flow model, into a representational space that follows a standard Gaussian.

The additional context taken into account by the model, while crucial to its flexibility in dealing with a variety of movements observed in the data, incurs little additional computational cost. As in the non-contextual case, the time to compute the likelihood of a data point is $\mathcal{O}(dL)$, and the amount of memory required is proportional to the number of parameters of the model.

8.3 Experiments

We now evaluate our flow-based probabilistic movement models. First, we conduct an in-depth analysis of our proposed approach and compare its performance with several baselines. Second, we investigate whether using the positions of the remaining players as additional contextual information improves our movement model. For all experiments, we use data from professional soccer.

8.3.1 Data

The tracking data from soccer consists of coordinates of each player and the ball, recorded with camera-based systems at 25Hz for five professional games from the German Bundesliga.[1] Each game is encoded as a sequence of triplets $(x_1^{(t)}, x_2^{(t)}, v_t)$ describing the x_1 and x_2 coordinates on the pitch in meters and the current speed v_t in km/h at time t for every player. Thus, every game consists of about $25 \cdot 60 \cdot 90 = 135,000$ positions per player. The x_1 and x_2 positions are relative to the origin. Hence, the coordinates are within $[-52.5, 52.5] \times [-34.0, 34.0] \subset \mathbb{R}^2$, since the dimensions of a standard soccer field are 105×68 meters. We use the first four games for training and the last game for testing.

8.3.2 Baselines

We compare our model against the following baselines. The first baseline, denoted as **KDE**, follows the approach introduced in Chapter 7 and deploys kernel density estimates for the movement models. This approach is considered the current state of the art. We use a Gaussian kernel and select the bandwidth using Scott's rule (Scott, 2015). A second, straightforward baseline simply estimates a two-dimensional Gaussian (mean and covariance) on the point cloud $\mathcal{S}$ as introduced in Section 7.2.3; see, for instance, Figure 7.3. We refer to this baseline as **Gaussian**. Third, we test a two-dimensional histogram, again on the point cloud mentioned above. The histogram uses an equally spaced grid over $[-10, 30] \times [-20, 20]$ with 1600 cells of size 1×1

[1] We thank Deutsche Fußball Liga and Sportcast GmbH for providing the positional data.

meter. Note that this range covers a larger space than depicted in Figure 7.3 and is suitable for all velocity ranges. We refer to this baseline as **Histogram**.

Depending on the experiment, we use different configurations. We experiment with time horizons $t_\Delta \in \{1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0\}$, where every value is in seconds. The baselines use the same speed discretization as in Table 7.1 which is $\{[0, 1), [1, 7), [7, 14), [14, 20), [20, 40]\}$. Those values are in kilometers per hour. Consequently, we train per baseline $7 \cdot 5 = 35$ different configurations, one per time horizon and speed range. As for the t_δ, denoting the time difference for estimating the direction in which a player is moving, we again follow Brefeld et al. (2019) and use $t_\delta = 0.2$ seconds. We show the influence of t_δ in Appendix B.1.

The code for all baselines is written in Python using NumPy (Harris et al., 2020) and SciPy (Virtanen et al., 2020). All experiments run on a machine with an Intel Xeon CPU, 256GB of RAM, and an NVIDIA V100 GPU.

8.3.3 Evaluation Metrics

When evaluating and comparing movement models, it is important to quantify how well a model explains the observed movements. In other words, a movement model should be capable of showing where the agent or object of interest will be in the near future. Consider a model that predicts a large area of future positions. Although this model explains all future positions by its sheer broadness, it is not concise. Hence, a good movement model needs to be concise and accurate and, to fulfill both, the model has to find an optimal trade-off between area and accuracy to estimate future positions in the smallest area possible.

A natural measure for this trade-off is the (log-)likelihood. Since densities are normalized by definition, larger areas possess lower point-wise likelihoods while compact densities capture only trivial movements and do not generalize well. As a consequence, a model achieving a good trade-off will also achieve higher likelihoods.

We additionally aim to measure the complexity of the different approaches. The complexity of predicting a likelihood either scales in the size of training data (KDE, see Section 7.2.4) or in the complexity of the neural networks (flow-based models, see Section 8.2). We thus measure the average evaluation time and report its quantity in seconds. Note that the prediction speed remains constant for both the Histogram and the Gaussian baseline.

A similar argument holds for memory footprints of the different approaches. The predictive performance of KDE is not only expected to deteriorate for data at large scales; it is also expected to require an excessive amount of memory. Hence, we also analyze the memory requirements of all evaluated methods and provide the number of variables (*i.e.*, floats or double precision) that need to be stored.

Table 8.1: Neural network architectures are detailed as a sequence of fully connected layers. The constants before and after arrows indicate the dimensionalities of input and outputs for that layer, respectively, while SELU denotes the activation function. The networks are part of the affine coupling $\mathrm{NN}_{\mathrm{ac}}(\mathbf{z}_2)$ that operates on the second part of the intermediate representation $\mathbf{z}_2 \in \mathbb{R}^{d/2}$, the conditional actnorm $\mathrm{CN}_{\mathrm{ca}}(\mathbf{c})$ which uses the context $\mathbf{c} \in \mathbb{R}^{d_c}$, and the conditional affine coupling $\mathrm{NN}_{\mathrm{cac}}(\mathbf{z}_2, \tilde{\mathbf{c}})$, where $\tilde{\mathbf{c}} = \mathrm{CN}_{\mathrm{cac}}(\mathbf{c}) \in \mathbb{R}^d$.

Network	Architecture and Activation Functions
$\mathrm{NN}_{\mathrm{ac}}$	$(d/2) \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d$
$\mathrm{CN}_{\mathrm{ca}}$	$d_c \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} 2d$
$\mathrm{NN}_{\mathrm{cac}}$	$(d/2 + d) \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d$
$\mathrm{CN}_{\mathrm{cac}}$	$d_c \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d$

8.3.4 Experimental Setup

We experiment with several configurations of the proposed models for a thorough comparison of the unconditional and the conditional flow-based movement models. The unconditional **Flow** uses the same time horizons and speed discretizations as the baselines above. Hence, we consider as many different models as the baselines for obtaining a fair comparison. With Flow we intend to show that movement models based on normalizing flows achieve the same or better predictive performance as the state of the art. We introduce contextual information, given by $\mathbf{c}_t = (v_t, t_\Delta)$, directly as a part of a single model in **CFlow**. Here, we use the current speed v_t and the time horizon t_Δ as context, just as for the baselines. Note that the baselines are implicitly conditioned on the time horizon as well due to maintaining different models for different intervals. Hence, the purpose of CFlow is to have a unified model instead of many different ones. We further evaluate a model called **CFlow-extended**, which leverages the other players' positions as additional contextual information. This model uses as context $\mathbf{c}_t = (v_t, t_\Delta, \bar{x}_t^{\text{rel}})$, where $\bar{x}_t^{\text{rel}} \in \mathbb{R}^{\frac{1}{2}d_{\text{hidden}}}$ is a permutation invariant representation of the relative positions of the remaining players. We provide details of how we compute this vector in Appendix B.2.

All models follow the architecture shown in Figure 8.2, repeating the three depicted transformations (actnorm, permutation, and coupling) $L = 8$ times in sequence. The same applies to their contextual alternatives. For unconditional and conditional flow-based models, the architectures of the neural networks are outlined in Table 8.1. All networks use self-scaling exponential linear unit (SELU) (Klambauer et al., 2017) activations, d denotes the dimensionality of the input $\mathbf{x}$, and d_c is the dimensionality of the context

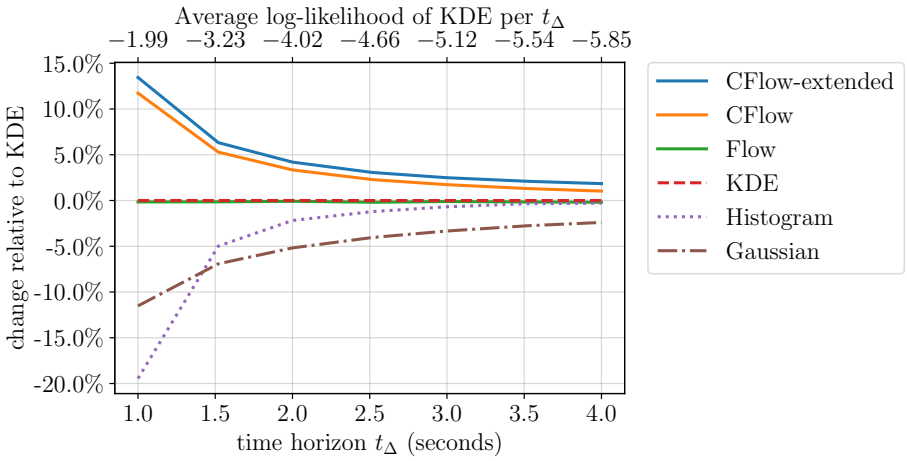

Figure 8.3: Average log-likelihood values (22 players) per model relative to the KDE baseline for various time horizons t_Δ. The specific values of the KDE are stated in the upper x-axis.

which is for CFlow and CFlow-extended, $d_c = 2$ and $d_c = 2 + \frac{1}{2}d_{\text{hidden}}$, respectively. The size of the hidden dimensionality is set to $d_{\text{hidden}} = 16$. Just as the authors of Glow (Kingma and Dhariwal, 2018), we initialize the weights and biases of the last layer of each neural network in Table 8.1 to zeros for stability. This implies that the actnorm and affine coupling layers are initialized to identity transformations before training.

The code of the flow-based models is written in Python using JAX (Bradbury et al., 2018). We employ the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3}. For stability, we clip gradients (Pascanu et al., 2013) with norms larger than 20. The batch size is set to $1,024$ for all models except the conditional models that condition on time horizon. A larger batch size of $1,024 \cdot N_{t_\Delta}$ is used to accommodate for different time horizons, where N_{t_Δ} denotes the number of different time horizons. All models are trained for 100 epochs with no early stopping.

8.3.5 Results

Predictive Performance. We first evaluate the predictive performance of the baseline movement models and compare them to the unconditional Flow. To have a fair comparison, we deal with 35 different configurations per model. We proceed as follows. We randomly sample a trajectory over three minutes for every player from the test game, leaving us with 22 such trajectories. Then, for several time horizons ranging from 1s to 4s, we compare the average log-likelihood per trajectory and model as well as the corresponding computation time for the prediction.

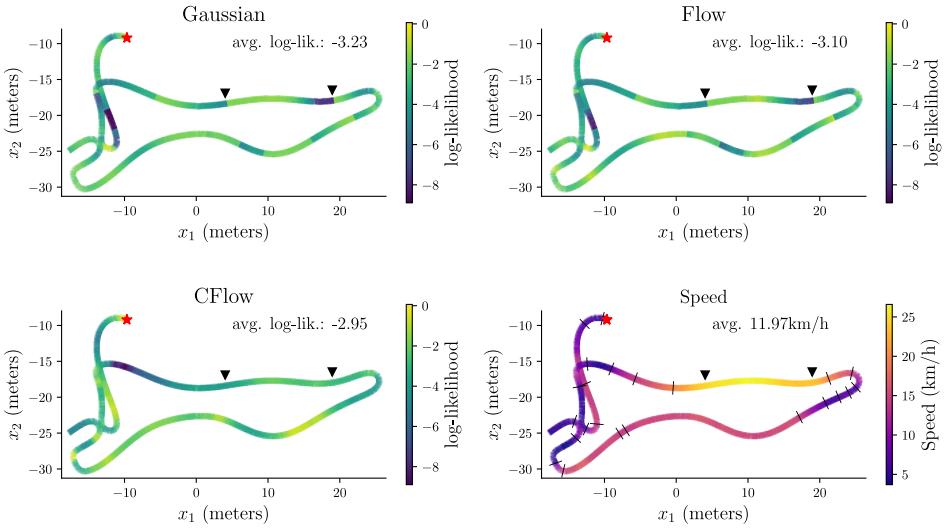

Figure 8.4: Analysis of a sample trajectory. The bottom-right plot shows the speed along the trajectory. Other plots depict log-likelihood values along the trajectory for the Gaussian, Flow, and CFlow models for a time horizon of $t_\Delta = 1s$. Darker colors in these other plots indicate the model estimates that position as less likely. We also provide average log-likelihood values for each model in this trajectory. The red star denotes the start of the trajectory. Black lines indicate model transitions for the Gaussian and Flow models, which rely on speed binning. Black triangle markers indicate an example where models behave distinctively. At those points in the trajectory, the player starts accelerating and then slows down. CFlow maintains a consistent quality of predictions. In contrast, the other two models do not.

Figure 8.3 shows the relative improvement of each model compared to the KDE. Unsurprisingly, the average log-likelihoods of the KDE decrease for larger time horizons t_Δ, as the models become progressively more uncertain with increasing t_Δ. It can be seen that the Flow model (green) performs on par with the KDE (dashed red) while the Gaussian approximation (brown) is almost consistently the worst. In addition, the performance gap of the Histogram (dotted purple) closes for larger time horizons; however, it underperforms compared to the KDE.

The relative improvement of CFlow (orange) and CFlow-extended (blue), each of which are single contextual models, shows that they clearly dominate all baselines by far. We credit this observation to conditioning. While the other models use a predefined velocity discretization, both CFlow and CFlow-extended adapt specifically to any velocity without the need for binning intervals with identical model output. If we also include the other players' relative positions into the context and condition on it, the performance increases further.

Table 8.2: Evaluation time in seconds per model averaged over 22 players including standard errors.

Model	$t_\Delta = 1.0s$
KDE	$774.3s \pm 23.0$
Histogram	$< 0.1s \pm 0.0$
Gaussian	$< 0.1s \pm 0.0$
Flow	$0.5s \pm 0.0$
CFlow	$0.1s \pm 0.0$
CFlow-extended	$0.7s \pm 0.0$

The flexibility afforded by conditioning can also be seen in Figure 8.4, which visualizes a trajectory of a soccer player drawn from the data. The bottom-right part of the figure depicts the current velocities of the player. In the remaining parts, the trajectory is colored according to the log-likelihood values using the Gaussian, Flow, and CFlow models for a time horizon of $t_\Delta = 1s$.

We also evaluate the average log-likelihood of this trajectory showing that CFlow has the highest log-likelihood, meaning it predicts the movement best. Once again, the Gaussian model yields the worst log-likelihood. We credit this due to its implicit symmetry. While being symmetric for moving left or right might be acceptable in this use case, the symmetry in the orthogonal direction is clearly not present, as seen in Figure 7.3. Visual inspection shows the effect of binning the velocities on the performance of the Gaussian and Flow. The player accelerates and decelerates at the positions marked by triangle markers while staying within a velocity bin. Hence, the model is unable to adapt to those speed changes.

By contrast, the conditional CFlow takes those speed changes into account and adapts much better to the trajectory at hand. This is indicated by a smooth transition of log-likelihood values and a higher average log-likelihood of the trajectory. In summary, CFlow and CFlow-extended can smoothly adjust to contextual changes such as speed and thus provide better predictions at each point in time.

Time and Memory Requirements. Table 8.2 compares the predictive performance of the competitors in terms of computation time. The numbers are again averaged over 22 individual trajectories and are shown with their standard errors. The baselines Gaussian and Histogram take almost no time to compute the predicted movements. KDE is consistently the slowest model by orders of magnitude. The result clearly shows the impracticability of KDE for (near) real-time applications and/or large data sets. Our proposed

Table 8.3: Memory footprint per model. The memory demand is stated in terms of floating point numbers which need to be saved.

		Memory Demand	
Model	No. of Models	per Model	Total
KDE	35	variable	175,177,098
Histogram	35	1,600	56,000
Gaussian	35	6	210
Flow	35	2,360	82,600
CFlow	1	11,704	11,704
CFlow-extended	1	15,056	15,056

flow-based models are only marginally slower than the straightforward competitors and, as expected, can be computed very efficiently.

The KDE baseline is not only the slowest approach when comparing evaluation time; it also has the highest memory demand, as Table 8.3 shows. This is due to the non-parametric nature of the approach. The more training data is observed and integrated, the better the model. Trivially, integrating more data into a KDE also means storing exactly these additional data. In comparison, the Histogram baseline has a fixed grid and velocity binning and, hence, a fixed memory footprint. The Gaussian exploits strong assumptions (*e.g.*, unimodality, symmetry) and needs the least amount of memory but is quite limited in expressiveness. Our proposed flow-based models have moderate memory requirements, which neither grow with training data as the KDE nor with higher precision or finer-grained bins as the Histogram.

8.4 Discussion

Binning velocities is a common trick in histogram-based movement models since continuous movements can be treated as discrete events that are instances of one or another bin. A major issue that is usually not appropriately addressed is how to find the optimal size of bins for a problem at hand. Often, optimal sizes are not equidistant but grow with the values of the variable of interest, such as speed. The bins also induce hard thresholds that may lead to treating similar values very differently if they end up in two neighboring bins. The presented contextual models do not have these limitations as they work directly on the continuous signal.

In the case of movement models, binning leads to maintaining several models and renders the space and time complexity of the models highly inefficient. By contrast, the contextual models do not rely on binning and simply condition on all variables of interest such as current velocity, time horizon,

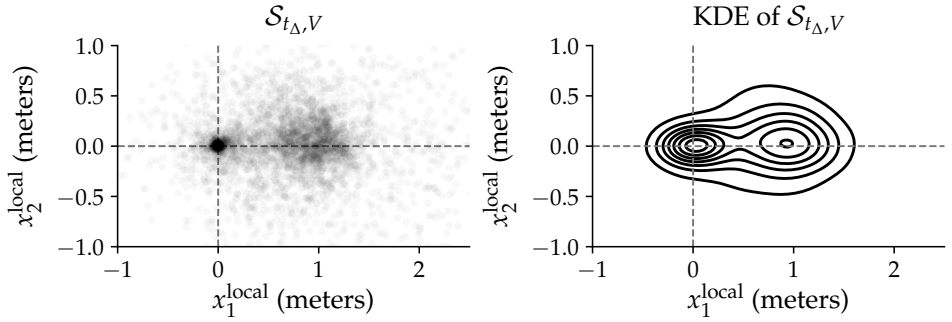

Figure 8.5: An example of the set $\mathcal{S}_{t_\Delta,V}$ (left) and corresponding kernel density estimate (right) for a goalkeeper.

and even the position of the other players. Thus, we end up with only a single model which is efficient and allows for (near) real-time processing. Our experiments show that both contextual models are moreover achieving superior performance in terms of log-likelihood and clearly outperform all other approaches.

As another (straightforward) baseline, we incorporated a Gaussian in the experiments. Naturally, a Gaussian is trivial to estimate and turns out efficient in time and space. However, these advantages imply assumptions on the densities being unimodal and symmetric. While some scenarios may support these claims, others may suffer from this oversimplification, as shown in Figure 8.5. It depicts the movements of a goalkeeper, and the point cloud has clearly two modes. Estimating the density of the points with a Gaussian introduces unnecessary errors by assigning a great deal of probability mass to unlikely events. Again, the proposed contextual models do not make any assumption on the nature of the densities and adapt to any multi-modal distribution of points.

A practical application for movement models is to compute so-called zones of control (*cf.* Section 7.3) that describe the areas which are controlled by a player. The underlying idea is that the payer who controls the zone is expected to arrive first at every position within this area (Taki and Hasegawa, 2000; Gudmundsson and Wolle, 2014; Horton et al., 2015; Spearman et al., 2017; Brefeld et al., 2019). Figure 8.6 shows an example. Using any of the CFlow models to compute the zones of control removes the necessity of binning velocities and time horizons; computation time and memory requirements are very low, even when training on massive amounts of data, compared to KDE-based approach in Chapter 7. Thus, our models allow for (near) real-time processing of live data and may be used in live analysis and broadcasts to visualize key situations.

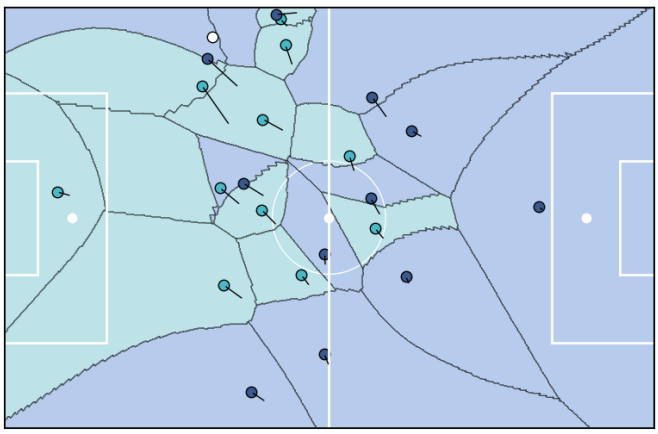

Figure 8.6: Soccer pitch showing players and the controlled spaces per player derived from the individual movement models.

Finally, while movement models, by definition, compute the area of possible (near-)future positions, they do not allow for predicting positions. Our contextual movement models, however, may overcome this issue by explicitly conditioning the models also on positions of other players (CFlow-extended) and the ball, ball possession indicators, etc. In principle, any CFlow should adapt to the current positioning of players and ball on the pitch, hence, shifting the probability mass towards areas that better represent motions of professional athletes given the current situation. Although these assumptions need to be confirmed in additional (empirical) studies and are clearly out of scope for the present thesis, the sheer possibility indicates the potential impact and importance of this line of work.

8.5 Conclusion

In this chapter, we studied the problem of learning movement models in a purely data-driven fashion, which we evaluated on data from professional soccer matches. We circumvented the limitations of previous approaches and cast the problem as a conditional density estimation task. We exploited characteristics of the low-dimensional problem formulation to devise conditional normalizing flows for modeling movements. In contrast to the state of the art, our contextual model consists of only a single model. Having a conditional model proved important for the integration of contextual information, such as velocities of movements. In principle, any relevant information can be used as context. Moreover, they outperformed all competitors by far when it came to predictive performance and turned out very efficient. Their computation times were comparable to trivial baselines and orders of magnitude faster than the state of the art.

Chapter 9

Principled Interpolation in Normalizing Flows

Learning high-dimensional densities is a common task in unsupervised learning. Normalizing flows (NFs) provide a state-of-the-art framework for transforming complex distributions into simple ones: a chain of parametrized bijective functions converts data into another representation that follows a given base distribution. The log-likelihood of the data can then be expressed as the log-likelihood of the base distribution and the log-determinants of the Jacobians of the transformations.

In the previous chapter, we leveraged normalizing flows as density estimators since our primary goal was to evaluate log-likelihoods. Now, we shift our focus to generating new samples from a learned distribution. In flow-based generative models, data are generated by drawing new samples from a base distribution, which is usually a standard Gaussian. Those samples are then mapped to real data using the aforementioned chain. A prevalent operation is to linearly interpolate samples and consider the interpolation path in data space. Such interpolations are, for example, frequently used to evaluate the quality of the learned model and to demonstrate that the model generalizes beyond what was seen in the training data (Radford et al., 2016).

In this chapter, we highlight the problems of vanilla linear interpolations while using Gaussian base distribution and propose several approaches to circumvent these problems. In summary, the key contributions of this chapter are as follows: we (i) highlight interpolation issues in normalizing flows using a Gaussian base distribution, (ii) propose to address those issues by enforcing data to lie on specific manifolds and use appropriate base distributions on them, and (iii) conduct quantitative and qualitative empirical evaluations on several real-world image data sets. Finally, (iv) our approach yields better interpolations as measured by Fréchet inception distance (FID) scores, which have been shown to correlate highly with human judgment of visual quality.

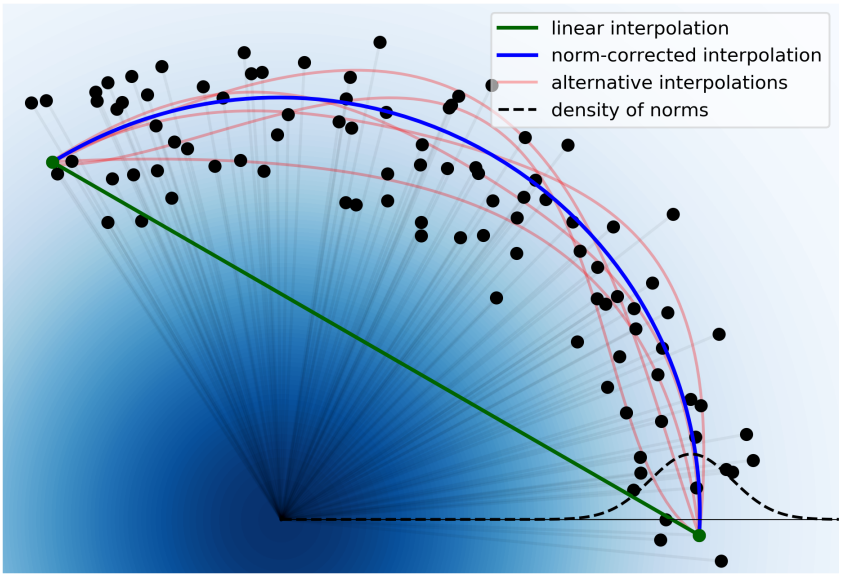

Figure 9.1: Illustration of different interpolation paths of points from a high-dimensional Gaussian. The figure also shows that, in high dimensions, points are not concentrated at the origin.

9.1 The Interpolation Problem and a Simple Heuristic

When using a standard Gaussian as a base distribution (*cf.* $p_z(\cdot)$ in Section 3.3), the consequences for interpolation, however, are not immediately apparent. Figure 9.1 shows a *linear interpolation (lerp)* of high-dimensional samples from a Gaussian. The squared Euclidean norms of the samples follow a χ^2_d-distribution as indicated by the dashed black line. Data points have an expected squared Euclidean norm of length d, where d is the dimensionality. This implies that there is almost no point around the origin. As seen in the figure, the norms of a linear interpolation path (green line) of two samples drop significantly and lie in a low-density area w.r.t. the distribution of the norms (dashed black line) (White, 2016).

Instead of a linear interpolation (green line), an interpolation that preserves the norm distribution of interpolants is clearly preferable (blue and red lines): the blue and red interpolation paths stay in the data manifold and do not enter low-density areas. The observation suggests that interpolated samples with norms in a specific range should generally result in better interpolations. This can be achieved, *i.e.*, by shrinking the variance of the density or norms (dashed lines), which yields a subspace or manifold that has a fix norm. The blue path is obtained by a norm correction of the linear

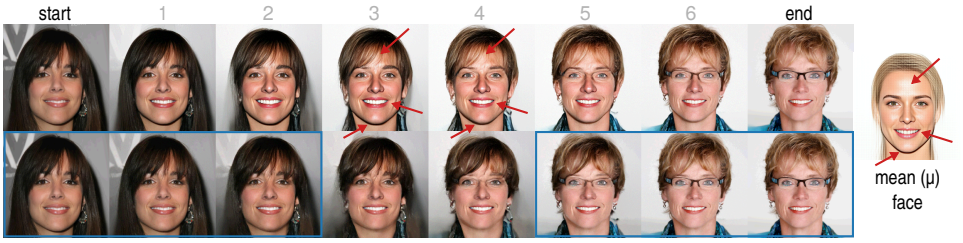

Figure 9.2: Two interpolation paths of samples from CelebA. *Top*: a linear interpolation path. The central images resemble features of the mean face as annotated in red. *Bottom*: an alternative interpolation path using a norm-correction. Note that the first and last three images are almost identical as annotated in blue. *Right*: decoded expectation of base distribution, *i.e.*, the mean face.

interpolation via also interpolating the norms. Mathematically, that is

$$\gamma(\lambda) = \underbrace{((1-\lambda)\mathbf{z}_a + \lambda\mathbf{z}_b)}_{\text{linear interpolation}} \cdot \underbrace{\frac{(1-\lambda)\|\mathbf{z}_a\|_2 + \lambda\|\mathbf{z}_b\|_2}{\|(1-\lambda)\mathbf{z}_a + \lambda\mathbf{z}_b\|_2}}_{\text{norm correction}}, \tag{9.1}$$

for endpoints $\mathbf{z}_a, \mathbf{z}_b$ and $\lambda \in [0, 1]$. We refer to this approach as *norm-corrected linear interpolation (nclerp)*. However, the depicted red lines also stay within the manifold; hence it remains unclear how a unique interpolation path can be obtained.

Figure 9.2 depicts two interpolation paths for faces taken from CelebA (Karras et al., 2018) created using Glow (Kingma and Dhariwal, 2018), a state-of-the-art flow-based model that uses a standard Gaussian as base distribution. The leftmost and rightmost faces of the paths are real data, while the other ones are computed interpolants. The face on the right depicts the so-called *mean face*, which is given by the mean of the Gaussian base distribution and is trivially computed by decoding the origin of the space. The top row shows a linear interpolation similar to the green line in Figure 9.1. The interpolation path is close to the origin, and the interpolants consequentially resemble features of the mean face, such as the nose, mouth, chin, and forehead shine, which neither of the women have. We highlight those features in red in Figure 9.2.

By contrast, the bottom row of Figure 9.2 shows the norm-corrected interpolation sequence (as the blue line in Figure 9.1): the background transition is smooth and not affected by the white of the mean face, and also subtleties like the shadow of the chin in the left face smoothly disappears in the transition. The norm correction clearly leads to a better transition from one image to the other. However, the simple heuristic in Equation (9.1) causes another problem: the path after norm correction is no longer equally spaced when λ values are equally spaced in $[0, 1]$. Hence, control over the interpolation is lost. Implications of this can be seen in blue in the bottom

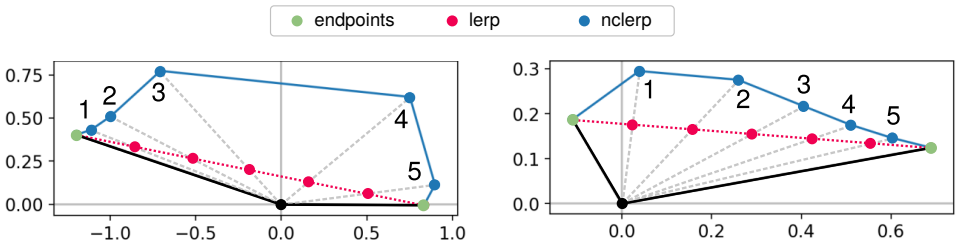

Figure 9.3: Two examples showing the issues caused by a norm-corrected linear interpolation (*nclerp*).

row of Figure 9.2, where the first and last three faces are almost identical. We provide additional examples in Appendix C.

In Figure 9.3, we illustrate two examples, comparing a linear interpolation (lerp) and a norm-corrected linear interpolation (nclerp) between points from a high-dimensional Gaussian (green points). For equally-spaced λ values in $[0, 1]$, a linear interpolation yields an equally-spaced interpolation path (red line). Evidently, the norm-corrected interpolation (blue line) keeps the norms of interpolants within the range observed in data. However, the interpolants are no longer evenly spaced along the interpolation path. Hence, control over the interpolation mixing is lost. This problem is more pronounced on the left example, where points 1 through 3 are closer to the start point, while points 4 and 5 are closer to the endpoint. Consequently, evaluations such as FID scores (Heusel et al., 2017) will be affected. Such scores are computed by comparing two sets of samples, in this case, the real data and interpolated data. As those points will be clearly more similar to the endpoints, which are samples from the data set itself, the scores are in favor of the norm-corrected interpolation.

9.2 Base Distributions on p-Norm Spheres

Motivated by earlier observations illustrated in Figures 9.1 and 9.2, we intend to reduce ambiguity by shrinking the variance of the norms of data. We achieve this by considering base distributions on restricted subspaces. More specifically, we focus on unit p-norm spheres defined by

$$\mathbb{S}_p^d = \left\{ \mathbf{z} \in \mathbb{R}^{d+1} \;\middle|\; \|\mathbf{z}\|_p^p = \sum_{j=1}^{d+1} |z_j|^p = 1 \right\}. \tag{9.2}$$

We distinguish two choices of p and discuss the challenges and desirable properties that ensue from their use. We consider $p \in \{1, 2\}$ as those allow us to use well-known distributions, namely the Dirichlet distribution for $p = 1$ and the von Mises-Fisher distribution for $p = 2$.

9.2.1 The Case p=1

For $p = 1$, the Dirichlet distribution defined on the standard simplex Δ^d is a natural candidate. Its probability density function is given by

$$p(\mathbf{s}) = \frac{1}{Z(\alpha)} \prod_{k=1}^{d+1} s_k^{\alpha_k - 1} \quad \text{with} \quad Z(\alpha) = \frac{\prod_{k=1}^{d+1} \Gamma(\alpha_k)}{\Gamma(\sum_{k=1}^{d+1} \alpha_k)}$$

for $\mathbf{s} \in \Delta^d$, where Γ is the gamma function and $\alpha_k > 0$ are the parameters. In order to make use of it, we also need to impose a non-negativity constraint in addition to Equation (9.2).

Let $\mathbf{z} \in \mathbb{R}^d$ be an unconstrained variable. The function $\phi : \mathbb{R}^d \to (0,1)^d$ transforms $\mathbf{z}$ into a representation $\mathbf{s}$ on the standard simplex by first transforming each dimension z_k into intermediate values v_k with

$$v_k = \sigma\left(z_k - \log\left(d + 1 - k\right)\right)$$

which are used to write $\mathbf{s} \in \Delta^d$ as

$$s_k = \left(1 - \sum_{l=1}^{k-1} s_l\right) \cdot v_k,$$

where $\sigma(\cdot)$ denotes the sigmoid function. We note a few details of this transformation. First, a property of ϕ is that $0 < \sum_{k=1}^{d} s_k < 1$. Therefore, a point in Δ^d can be obtained with an implicit additional coordinate $s_{d+1} = 1 - \sum_{k=1}^{d} s_k$. Second, the difference in dimensionality does not pose a problem for computing its Jacobian as ϕ establishes a bijection within $\mathbb{R}^d$ while the mapping to Δ^d is given implicitly. Third, ϕ maps $\mathbf{z} = \mathbf{0}$ to the center of the simplex $\mathbf{s} = (d+1)^{-1}\mathbf{1}$. Fourth, since $\mathbf{s}$ consists of solely positive numbers which sum up to one, numerical problems may arise for high-dimensional settings. We elaborate on this issue in Section 9.3.

The Jacobian J_ϕ has a lower triangular structure and solely consists of non-negative entries. Hence, the log-determinant of this transformation can be efficiently computed in $\mathcal{O}(d)$ time via

$$\log|\det J_\phi| = \sum_{k=1}^{d} \log\left(v_k(1 - v_k)\right) + \log\left(1 - \sum_{l=1}^{k-1} s_l\right).$$

The inverse transformation $\phi^{-1} : \mathbb{R}^d \to \mathbb{R}^d$ is given by

$$z_k = \sigma^{-1}\left(\frac{s_k}{1 - \sum_{l=1}^{k-1} s_l}\right) + \log(d + 1 - k).$$

The interpolation of two points $\mathbf{a}, \mathbf{b} \in \Delta^d$ within the unit simplex is straightforward. A linear interpolation $(1-\lambda)\mathbf{a} + \lambda\mathbf{b}$ using $\lambda \in [0,1]$ is guaranteed to stay within the simplex by definition.

9.2.2 The Case p=2

For $p = 2$, data points lie on the surface of a d-dimensional hypersphere. The von Mises-Fisher (vMF) distribution, defined on $\mathbb{S}_2^d$, is frequently used in directional statistics. It is parameterized by a *mean direction* $\boldsymbol{\mu} \in \mathbb{S}_2^d$ and a *concentration* $\kappa \geq 0$, with a probability density function given by

$$p(\mathbf{s}) = C_{d+1}(\kappa) \exp(\kappa \boldsymbol{\mu}^\top \mathbf{s}),$$
$$\text{with} \quad C_\nu(\kappa) = \frac{\kappa^{\nu/2-1}}{(2\pi)^{\nu/2} I_{\nu/2-1}(\kappa)},$$

where I_w denotes the modified Bessel function of the first kind at order w.

Again, let $\mathbf{z} \in \mathbb{R}^d$ be an unconstrained variable. We employ a stereographic projection, for both its invertibility and its Jacobian, whose log-determinant can be efficiently computed. The transformation $\psi : \mathbb{R}^d \to \mathbb{S}_2^d$ maps a point $\mathbf{z} \in \mathbb{R}^d$ to a point $\mathbf{s} \in \mathbb{S}_2^d \subset \mathbb{R}^{d+1}$ on the hypersphere via

$$\psi(\mathbf{z}) = \mathbf{s} = \begin{bmatrix} \mathbf{z}\rho_\mathbf{z} \\ 1 - \rho_\mathbf{z} \end{bmatrix}, \quad \text{with} \quad \rho_\mathbf{z} = \frac{2}{1 + \|\mathbf{z}\|_2^2}.$$

The transformation ψ, which has no additional parameters, ensures that its image is on the unit hypersphere, allowing the use of a vMF distribution to model $p(\mathbf{s})$. Two points in $\mathbb{S}_2^d$ are of special interest, namely the *south pole* and the *north pole*, where the last coordinate of $\mathbf{s}$ is either -1 or 1, respectively. By construction, the transformation is symmetric around zero and sends $\mathbf{z} = \mathbf{0}$ to the *south pole*, which we choose as the mean direction $\boldsymbol{\mu}$. Furthermore, it is bijective up to an open neighborhood around the *north pole*, as $\rho_\mathbf{z} \to 0$ whenever $\|\mathbf{z}\|_2^2 \to \infty$. For this reason, we avoid choosing a uniform distribution on the hypersphere, which is obtained for $\kappa = 0$. Figure 9.4 shows an example.

Contrary to the previous case, the log-determinant of J_ψ alone is not enough to accommodate the density change when transforming from $\mathbb{R}^d$ to $\mathbb{S}_2^d$ (Gemici et al., 2016). The correct density ratio change is scaled by $\sqrt{\det J_\psi^\top J_\psi}$ instead, whose logarithm can be computed in $\mathcal{O}(d)$ time as

$$\log \sqrt{\det J_\psi^\top(\mathbf{z}) J_\psi(\mathbf{z})} = d \log \frac{2}{1 + \|\mathbf{z}\|_2^2} = d \log \rho_\mathbf{z},$$

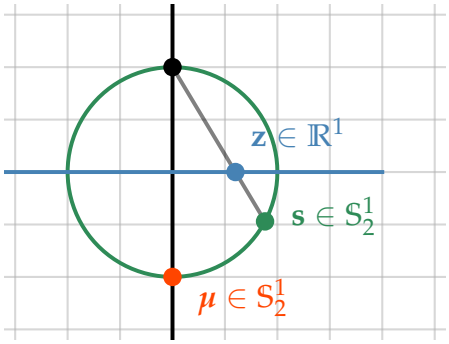

Figure 9.4: A stereographic projection mapping $\mathbf{z} \in \mathbb{R}^1$ to $\mathbf{s} \in \mathbb{S}_2^1 \subset \mathbb{R}^2$ using the north pole depicted as a black dot. For the vMF distribution we use $\boldsymbol{\mu} \in \mathbb{S}_2^1$ as the mean direction.

with $\rho_{\mathbf{z}}$ given as stated above. The inverse transformation $\psi^{-1} : \mathbb{S}_2^d \subset \mathbb{R}^{d+1} \to \mathbb{R}^d$ is

$$\psi^{-1}(\mathbf{s}) = \mathbf{z} = \frac{[\mathbf{s}]_{1:d}}{1 - [\mathbf{s}]_{d+1}},$$

where $[\mathbf{s}]_{1:d}$ denotes the first d coordinates of $\mathbf{s}$ and $[\mathbf{s}]_{d+1}$ is the $(d+1)$-th coordinate of $\mathbf{s}$.

To interpolate points on the hypersphere, a *spherical linear interpolation (slerp)* (Shoemake, 1985) can be utilized. It is defined as follows. Let $\mathbf{s}_a$ and $\mathbf{s}_b$ be two unit vectors and $\omega = \cos^{-1}(\mathbf{s}_a^\top \mathbf{s}_b)$ be the angle between them. The interpolation path is then given by

$$\gamma(\lambda) = \frac{\sin((1-\lambda)\omega)}{\sin(\omega)}\mathbf{s}_a + \frac{\sin(\lambda\omega)}{\sin(\omega)}\mathbf{s}_b, \quad \text{for} \quad \lambda \in [0,1].$$

9.2.3 Sampling with Temperature

Prior research (Kingma and Dhariwal, 2018; Chen et al., 2019) indicates that sampling with a low temperature yields better samples for a Gaussian base distribution. Sampling with temperature refers to sampling from a base distribution $p_T(\mathbf{z}) \propto p(\mathbf{z})^{-T^2}$. In the case of a standard Gaussian base distribution, we obtain

$$p(z)^{\frac{1}{T^2}} \propto \exp\left(-\frac{\frac{1}{T^2}}{2}z^2\right) = \exp\left(-\frac{1}{2}\frac{z^2}{T^2}\right),$$

which is equivalent to sample from a Gaussian with zero mean and standard deviation of $\sigma' = T$. Since the squared Euclidean norm of a standard

Gaussian sample follows a χ^2_d distribution, *i.e.*, $\|\mathbf{z}\|_2^2 \sim \chi^2_d$, a sample has an expected length of $\mathbb{E}\left[\|\mathbf{z}\|_2^2\right] = d$. A sample with a different standard deviation can be obtained by first drawing a sample from a standard Gaussian and then scaling it by T^2. Therefore, by sampling with temperature $T < 1$, we essentially sample closer to the mean, since

$$\mathbb{E}\left[\|T \cdot \mathbf{z}\|_2^2\right] = \mathbb{E}\left[T^2\|\mathbf{z}\|_2^2\right] = T^2\mathbb{E}\left[\|\mathbf{z}\|_2^2\right] = T^2 d.$$

An analogous change can be performed in a vMF distribution. Using temperature, we have

$$p(\mathbf{z})^{-T^2} = p(\mathbf{z})^{\frac{1}{T^2}} \propto \exp\left(\frac{\kappa}{T^2}\boldsymbol{\mu}^\top \mathbf{z}\right),$$

which corresponds to having a vMF distribution with $\kappa' = \frac{\kappa}{T^2}$. Consequently, $T \to \infty$ yields a uniform distribution on the hypersphere and $T \to 0$ results in sampling the mean direction $\boldsymbol{\mu}$.

9.3 Experiments

We now evaluate the restriction of a normalizing flow to a unit p-norm sphere and compare them to a Gaussian base distribution. As we focus on a principled way of interpolating in flow-based generative models, we employ a fixed architecture per data set instead of aiming to achieve state-of-the-art density estimation. We use Glow (Kingma and Dhariwal, 2018) as the flow architecture for the experiments in the remainder of this section. However, our approach is not limited to Glow, and the transformations and changes in the base distribution can also be used in other architectures. We also do not compare against other architectures as our contribution is a change of the base distribution, which allows for obtaining better interpolations.

9.3.1 Performance Metrics and Setup

Performance is measured in terms of: *bits per dimension (BPD)*, calculated using $\log_2 p(\mathbf{x})$ divided by d; *Fréchet inception distance (FID) scores*, which have been shown to correlate highly with human judgment of visual quality (Heusel et al., 2017); and *kernel inception distance (KID) scores* (Bińkowski et al., 2018). KID is similar to FID as it is based on Inception scores (Salimans et al., 2016). While the FID first fits a Gaussian distribution on the scores of a reference set and a set of interest and then compares the two distributions, the KID score is non-parametric, *i.e.*, it does not assume any distribution and compares the Inception scores based on maximum mean discrepancy (MMD). We follow Bińkowski et al. (2018) and employ a polynomial kernel with degree three for our evaluations.

We measure bits per dimension on the test set and on interpolated samples. FID and KID scores are evaluated on generated and interpolated samples and then compared to a reference set, which is the training data. When generating data, we draw as many samples from the base distribution as we have for training. For interpolation, we focus on interpolation within classes and adopt regular linear interpolation for Gaussian-distributed samples, while using a spherical linear interpolation on the sphere for vMF-distributed samples. In this operation, we sample $n/5$ pairs of images from the training set and generate five equally spaced interpolated data instances per pair, resulting in n new images. From those interpolation paths, we only use the generated points and not the points which are part of training data. Hence, we are only considering previously unseen data.

We also compare against the *norm-corrected linear interpolation (nclerp)* as defined in Equation (9.1). Note that a linearly spaced interpolation path is no longer linearly spaced after norm correction. The resulting interpolation paths are composed of images located closer to the endpoints and thus bias the evaluation. We include the results nevertheless for completeness and refer to Section 9.1 for a detailed discussion.

The reported metrics are averages over three independent repetitions and include standard errors. The code is written in Python using PyTorch (Paszke et al., 2019), and all experiments run on an Intel Xeon CPU with 256GB of RAM using an NVIDIA V100 GPU.

9.3.2 Data

We utilize **MNIST** (LeCun et al., 2010), **Kuzushiji-MNIST** (Clanuwat et al., 2018), and **Fashion-MNIST** (Xiao et al., 2017), which contain gray-scale images of handwritten digits, Hiragana symbols, and images of Zalando articles, respectively. All MNIST data sets consist of 60,000 training and 10,000 test images of size 28×28. In addition, we evaluate on **CIFAR10** (Krizhevsky and Hinton, 2009), which contains natural images from ten classes. The data set has 50,000 training and 10,000 test images of size 32×32.

9.3.3 Architecture

Following Kingma and Dhariwal (2018), we employ the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 10^{-3}, clip gradients at 50, and use linear learning rate warm-up for the first ten epochs. Models were trained on MNIST data and CIFAR10 using mini-batches of size 256 and 128, respectively. All models are trained for 100 epochs without early stopping. We keep all architectures as close as possible to Glow, with the following deviations. For MNIST data, we use random channel permutations instead of invertible 1×1 convolutions. The number of filters in the convolutions of the affine coupling layers is 128. In Glow terms, we employ $L = 2$ levels

Table 9.1: Results for generative modeling averaged over three independent runs including standard errors.

		Test	Sample	
	Base dist.	BPD	FID	KID
MNIST	Gaussian	1.59 ± 0.06	**34.53** ± 0.83	**0.033** ± 0.001
	vMF $\kappa = 1d$	**1.46** ± 0.07	40.07 ± 2.46	0.037 ± 0.001
	vMF $\kappa = 1.5d$	1.54 ± 0.09	40.39 ± 1.40	0.036 ± 0.001
	vMF $\kappa = 2d$	1.82 ± 0.08	39.82 ± 0.26	0.038 ± 0.001
	Dirichlet $\alpha = 2$	1.76 ± 0.12	40.08 ± 0.72	0.039 ± 0.001
K-MNIST	Gaussian	2.58 ± 0.11	35.34 ± 0.76	0.041 ± 0.001
	vMF $\kappa = 1d$	2.63 ± 0.06	36.63 ± 0.37	0.041 ± 0.001
	vMF $\kappa = 1.5d$	**2.48** ± 0.06	**35.00** ± 0.61	**0.040** ± 0.001
	vMF $\kappa = 2d$	2.51 ± 0.04	36.45 ± 0.42	0.041 ± 0.001
	Dirichlet $\alpha = 2$	2.50 ± 0.05	35.54 ± 0.39	**0.040** ± 0.001
F-MNIST	Gaussian	3.24 ± 0.04	66.64 ± 1.29	0.064 ± 0.003
	vMF $\kappa = 1d$	**3.16** ± 0.03	**60.45** ± 3.34	**0.055** ± 0.005
	vMF $\kappa = 1.5d$	3.30 ± 0.07	61.89 ± 1.29	0.056 ± 0.002
	vMF $\kappa = 2d$	3.22 ± 0.06	60.60 ± 3.47	**0.055** ± 0.004
CIFAR10	Gaussian	3.52 ± 0.01	71.34 ± 0.45	**0.066** ± 0.001
	vMF $\kappa = 1d$	3.43 ± 0.00	71.07 ± 0.78	0.069 ± 0.001
	vMF $\kappa = 1.5d$	**3.42** ± 0.00	**70.58** ± 0.40	0.068 ± 0.001
	vMF $\kappa = 2d$	**3.42** ± 0.01	71.00 ± 0.28	0.068 ± 0.001

of $K = 16$ steps each. For CIFAR10, our models have $L = 3$ levels of $K = 24$ steps each, while the affine coupling layers have convolutions with 512 filters. The architecture is kept the same across base distributions, except for the additional parameterless transformations to the restricted subspaces introduced in Section 9.2.

When comparing base distributions, we consider the following hyperparameters. For the vMF distribution, we use concentration values for which the partition function is finite. For consistency, the values we use are the same multiples of the data dimensionality d for each data set. The concentration values for the Dirichlet distribution are set to $\alpha = 2$, which refers to $2 \cdot \mathbf{1}^{d+1} \in \mathbb{R}^{d+1}$.

9.3.4 Quantitative Results

We first evaluate the generative modeling aspects of all competitors. Table 9.1 summarizes the results in terms of bits per dimension on test data, and FID and KID scores on generated samples for all data sets. Experiments

Table 9.2: Results for interpolation averaged over three independent runs including standard errors. Interpolations are in-class only and use five intermediate points; *lerp* refers to a linear interpolation; *nclerp* refers to the norm-corrected linear interpolation (Section 9.1) and *slerp* refers to the spherical interpolation.

	Base dist.	Type	BPD	FID	KID
MNIST	Gaussian	lerp	1.33 ± 0.05	5.10 ± 0.14	0.003 ± 0.000
	Gaussian	nclerp	1.44 ± 0.06	5.12 ± 0.30	0.003 ± 0.000
	vMF $\kappa = 1d$	slerp	**1.31**± 0.09	**3.84**± 0.36	**0.002** ± 0.000
	vMF $\kappa = 1.5d$	slerp	1.40 ± 0.10	4.22 ± 0.12	**0.002** ± 0.000
	vMF $\kappa = 2d$	slerp	1.63 ± 0.10	4.45 ± 0.06	**0.002** ± 0.000
	Dirichlet $\alpha = 2$	lerp	1.61 ± 0.10	5.81 ± 0.36	0.004 ± 0.001
K-MNIST	Gaussian	lerp	1.91 ± 0.17	19.71 ± 1.59	0.021 ± 0.002
	Gaussian	nclerp	2.15 ± 0.15	17.60 ± 1.48	0.020 ± 0.002
	vMF $\kappa = 1d$	slerp	2.08 ± 0.15	17.93 ± 3.72	0.020 ± 0.004
	vMF $\kappa = 1.5d$	slerp	**1.80**± 0.07	22.72 ± 2.65	0.025 ± 0.003
	vMF $\kappa = 2d$	slerp	2.03 ± 0.14	**14.54**± 2.51	**0.016**± 0.003
	Dirichlet $\alpha = 2$	lerp	1.81 ± 0.04	24.09 ± 2.35	0.026 ± 0.003
F-MNIST	Gaussian	lerp	2.84 ± 0.10	13.06 ± 0.62	0.007 ± 0.001
	Gaussian	nclerp	2.93 ± 0.03	*7.80*± 0.13	0.004 ± 0.000
	vMF $\kappa = 1d$	slerp	**2.66**± 0.03	**12.16**± 0.13	**0.006**± 0.000
	vMF $\kappa = 1.5d$	slerp	2.84 ± 0.07	12.19 ± 1.07	**0.006**± 0.001
	vMF $\kappa = 2d$	slerp	2.70 ± 0.05	15.11 ± 0.85	0.008 ± 0.001
CIFAR10	Gaussian	lerp	2.64 ± 0.06	58.63 ± 1.26	0.053 ± 0.001
	Gaussian	nclerp	3.32 ± 0.01	*14.29*± 0.16	0.010 ± 0.000
	vMF $\kappa = 1d$	slerp	2.78 ± 0.05	**51.08**± 0.37	0.010 ± 0.000
	vMF $\kappa = 1.5d$	slerp	2.66 ± 0.05	55.23 ± 5.14	0.047 ± 0.005
	vMF $\kappa = 2d$	slerp	**2.58**± 0.08	52.65 ± 3.34	0.044 ± 0.004

with the Dirichlet base distribution were not successful on all data sets. The restrictions imposed to enable the use of the distribution demand a high numerical precision since every image on the simplex is represented as a non-negative vector that sums up to one. Consequently, we only report results on MNIST and Kuzushiji-MNIST. Using the vMF as a base distribution clearly outperforms the Gaussian in terms of bits per dimension on test data. As seen in the FID and KID scores, we perform competitive compared to the Gaussian for generating new data. Hence, the generative aspects of the proposed approach are either better or on par with the default choice of a Gaussian. Note that lower BPD on test data and lower FID/KID scores on generated data might be obtained with more sophisticated models.

We now evaluate the quality of interpolation paths generated via various approaches. Table 9.2 shows the results in terms of bits per dimension, FID,

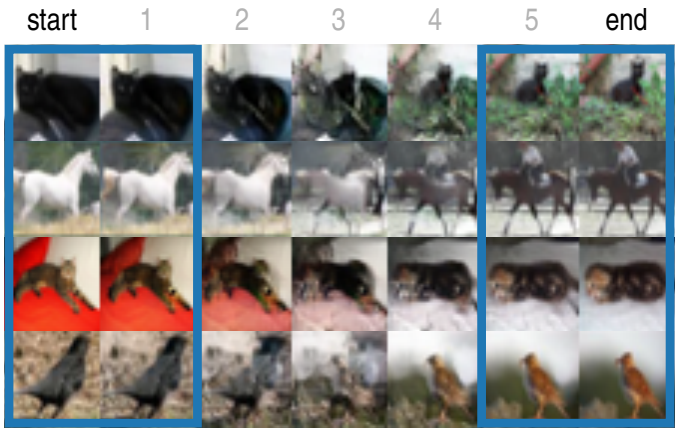

Figure 9.5: Four interpolation paths of the norm-corrected linear interpolation (*nclerp*) depicting the problem of almost repeated endpoints (highlighted in blue) and thus a biased evaluation on CIFAR10.

and KID scores for all data sets. The experiments confirm our hypothesis that an interpolation on a fixed-norm space yields better results as measured in BPD, FID, and KID scores. The norm-corrected interpolation yields better FID and KID scores for Fashion-MNIST and CIFAR10. However, this heuristic produces interpolation paths that are biased towards the endpoints and hence are naturally closer to observed data, thus yield better FID and KID scores. This is depicted in Figure 9.5. More results for general interpolations within classes and across classes are provided in Appendix C.

9.3.5 Qualitative Results

Figure 9.6 displays interpolation paths with five interpolants of four pairs of data from CIFAR10, created using the same architecture trained on different base distributions. We pick the best performing model on BPD on test data from the multiple training runs for each base distribution. We visually compare a linear interpolation using a Gaussian base distribution against a spherical linear interpolation using a vMF base distribution with different concentration values. Naturally, the images in the center show the difference and the effects resulting from the choice of base distribution and, hence, the interpolation procedure.

Overall, the linear interpolation with a Gaussian tends to show mainly darker objects on a brighter background (almost black and white images) in the middle of the interpolation path. This is not the case for the spherical interpolations using a vMF base distribution. Specifically, in the second example showing dogs, the checkerboard background of the left endpoint smoothly fades out for the vMF ($\kappa = 2d$) model while the Gaussian shows an almost white background. A similar effect happens in the last pair of

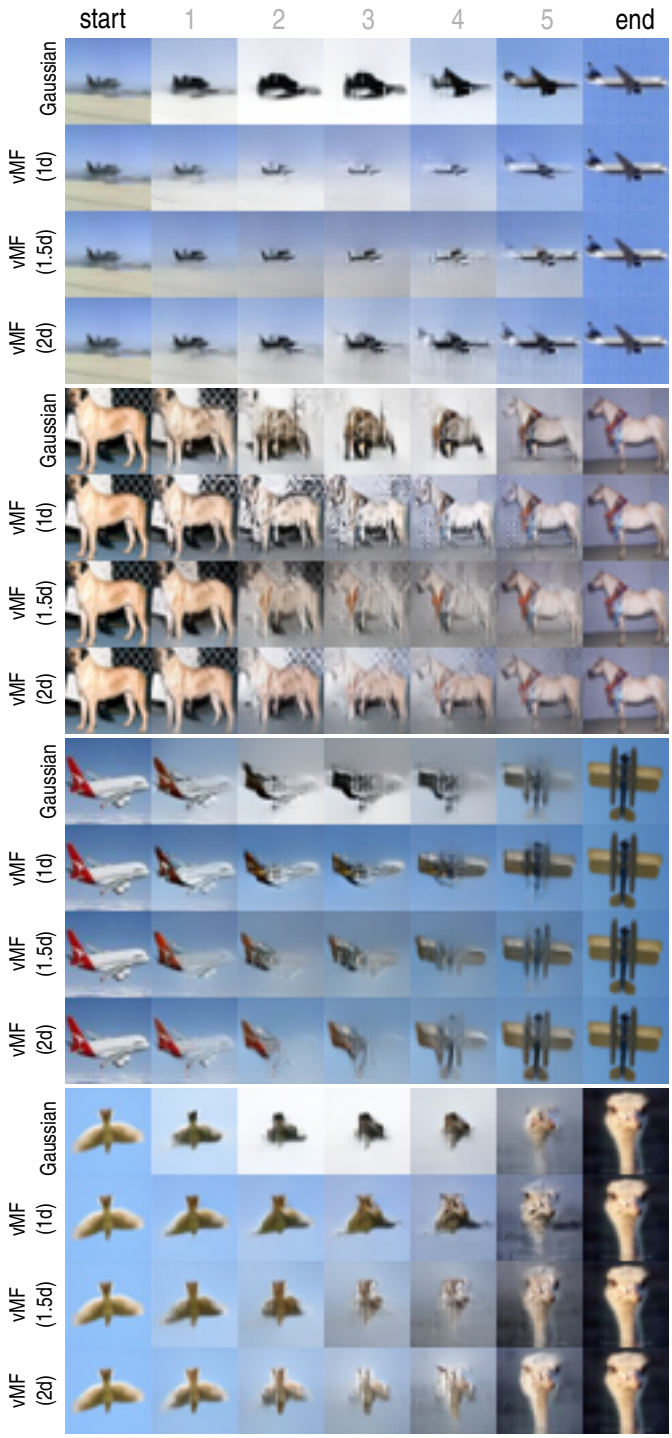

Figure 9.6: Interpolation paths of four pairs of data from CIFAR10 using different models.

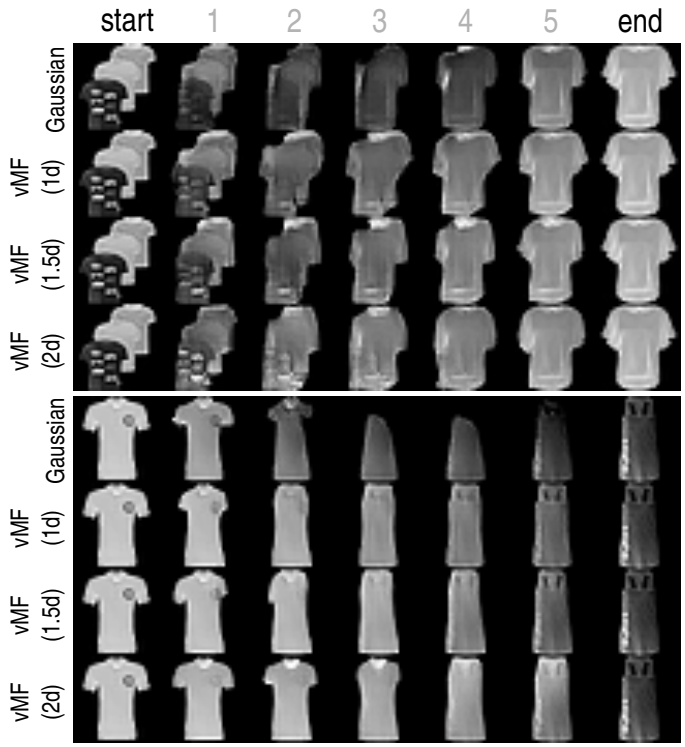

Figure 9.7: Interpolation paths of two pairs of data from Fashion-MNIST using different models.

images, highlighting the weaknesses of a linear interpolation once again. By contrast, the vMF models generate images where those effects are either less prominent or non-existent, suggesting a path that strictly follows the data manifold. We provide more interpolation paths on CIFAR10 in Appendix C.

Figure 9.7 depicts interpolation paths with five interpolants on two pairs of data from Fashion-MNIST. In both cases, the Gaussian model produces suboptimal images with visible changes in color, which is not consistent with the endpoints. Furthermore, there is visible deformation of the clothing items.

9.4 Related Work

Interpolations are commonplace in generative modeling, being particularly useful for evaluating them. White (2016) proposes to use a spherical linear interpolation to circumvent the problems depicted in Figure 9.1 in generative adversarial networks (GANs) and variational autoencoders (VAEs). However, as the Gaussian is kept as a base distribution, the difference in norms causes problems similar to the norm corrected approach. The problem of interpolation is also investigated by Agustsson et al. (2019) for

GANs. Specifically, they show that the quality of the generated images in the interpolation path improves when attempting to match the distribution of norms between interpolants and the GAN prior, such as a Gaussian or uniform distribution. The problem with the distribution mismatch while interpolating is also studied in Kilcher et al. (2018).

Brehmer and Cranmer (2020) propose to simultaneously learn a manifold and corresponding normalizing flow on it. By contrast, in this chapter, we employ a prescribed manifold, *i.e.*, a p-norm sphere, on which the interpolation can be done in a principled way. Davidson et al. (2018) and Xu and Durrett (2018) investigate the vMF distribution as a prior of VAEs. Their motivation is to encourage the model to learn better latent representations on data with hyperspherical structure. While results show improvements over a Gaussian prior, properties of our interest, such as interpolation, are not addressed. Similarly, Sinkhorn autoencoders (Patrini et al., 2020) employ principles that allow them to work with different latent spaces and priors, such as the simplex with the Dirichlet prior and a hyperspherical space with the vMF prior.

Employing normalizing flows on non-Euclidean spaces, such as the hypersphere, was first proposed by Gemici et al. (2016). They introduce a mapping for doing normalizing flows on hyperspherical data. The main difference from our setting is that the data is already on a sphere and is moved to $\mathbb{R}^d$, an unrestricted space, performing the entire flow in there instead, before moving back to the sphere. This avoids defining a flow on the sphere, which is studied in Rezende et al. (2020) for tori and spheres. Furthermore, Bose et al. (2020) define normalizing flows on hyperbolic spaces, which is beneficial for graph-structured data.

Arvanitidis et al. (2018) provide a geometric analysis of autoencoders, showing that they learn latent spaces which can be characterized by a Riemannian metric. With this, interpolations follow a geodesic path under this metric, leading to higher quality interpolations. Compared to our contribution, these approaches do not change the standard priors but propose alternative ways to interpolate samples. In contrast, we propose an orthogonal approach by changing the base distribution and imposing constraints on the representation in our training procedure. Consequently, standard interpolation procedures, such as the spherical linear interpolation, can be used in a principled way.

9.5 Conclusion

In this chapter, we highlighted the limitations of linear interpolation in flow-based generative models using a Gaussian base distribution. As a remedy, we proposed to focus on base representations with a fixed norm where the interpolation naturally overcomes those limitations and intro-

duced normalizing flows onto unit p-norm spheres. Specifically, we showed for the cases $p \in \{1, 2\}$ that we can operate on the unit simplex and unit hypersphere, respectively. We introduced a computationally efficient way of using a Dirichlet distribution as a base distribution for the case of $p = 1$ and leveraged a vMF distribution using a stereographic projection onto a hypersphere for the case $p = 2$. Although the former suffered from numerical instabilities in a few experiments, our experimental results showed superior performance in terms of bits per dimension on test data and FID and KID scores on interpolation paths that resulted in natural transitions from one image to another. This was also confirmed by visually comparing interpolation paths on CIFAR10 and Fashion-MNIST.

Chapter 10

Conclusions

This thesis dealt with unsupervised representation learning. Specifically, we considered two orthogonal views to representation learning. First, we changed the representation of a data set in terms of its sample size and selected representative subsets for various learning tasks. The goal was that machine learning models trained on those small subsets learn approximately the same model as on all data and perform competitively at a much smaller cost. Second, we transformed the representation of data in terms of the dimensions to facilitate specific tasks such as density estimation and interpolation. We derived multiple novel algorithms and concepts that advance the current state of the art for both scenarios. Theoretical and empirical evaluations showed that our contributions outperform existing baselines on several metrics and data sets.

The Sample Size View

We first introduced the concept of the frame and investigated its use for two learning tasks, namely archetypal analysis (AA) and linear regression, within the setting of optimal experimental design (OED). The frame consists of the data points on the boundary of the data set and can be seen as extreme data points. Computing this subset in high dimensions with existing methods is challenging. Hence, we proposed a new and efficient way based on the well-known non-negative least squares (NNLS) (Lawson and Hanson, 1995) optimization algorithm. We then showed that the frame serves as a representative subset for archetypal analysis. This matrix factorization can be seen as an approximation of the boundary of data with a pre-defined number of vertices, making the frame a natural candidate. Another learning task in which the boundary of data is highly informative is linear regression, specifically in the setting of optimal experimental design. Considering the dual formulations of often-used optimality criteria showed, extreme points of the boundary will most likely contribute to the solution. Hence, we leveraged our proposed frame concept and showed empirically that computing

OED on the frame yields competitive results in much less time. Besides, we showed how to compute the frame in kernel-induced feature spaces by having access only to the kernel matrix. This allows for using the frame concept in primal formulations and dual versions of learning problems. We also showed theoretically that some kernel functions will always yield a full frame. In other words, all points belong to the boundary, which means that all of them are informative in highly non-linear settings. In summary, the idea of using the boundary of data as a representative subset turned out to work well in practice: competitive results were obtained in a much shorter time, allowing for learning at much larger scales.

Although the frame worked empirically well for archetypal analysis, it had some shortcomings. First, the size of the representative subset is an inherent property of every data set and hence cannot be freely chosen. Second, the representative subset comes with no theoretical guarantees on the performance. As a remedy, we were the first to propose coresets for archetypal analysis. Coresets are (weighted) subsets of data that are representative and come with theoretical guarantees in terms of a bound on the loss function on the data and the computed coreset. The size of the coreset is directly linked to the probability that the bound holds and how tight that bound is. Furthermore, the computation of the coreset for AA is highly efficient as it scales linearly in the number of data points. We provided a rigorous theoretical analysis of the proposed coreset and backed our evaluation with empirical results on large data sets. Our contribution rendered archetypal analysis feasible for large data sets, comes with theoretical guarantees, and is straightforward to implement.

The Dimension View

The second part of this thesis dealt with a change of representation in terms of dimensions. We began by studying movement models, which are key to spatio-temporal problems and help us predict the movements of objects such as animals, particles, or players. We derived probabilistic movement models along trajectories from positional data. A change of representation into a local coordinate system allowed us to learn a location-invariant distribution of possible whereabouts. At first, we employed a kernel density estimate (KDE) as a non-parametric density estimator and maintained a separate model per context. As contextual information, we used the time horizon, *i.e.*, the time offset in the future for which we want to have the prediction, and the initial velocity. Compared to traditional movement models, our proposed approach does not rely on unrealistic assumptions on the underlying physics since we use a purely data-driven approach. We then leveraged our probabilistic movement models to derive so-called zones of control. In the example of soccer, every player has an area or zone on which she arrives first and hence controls that zone. Hence, the zones induce a par-

tition of the playing area or pitch. Empirically, we showed that our models produce better and more realistic zones of control than baseline competitors. However, using KDEs came with issues. Contextual information was sub-optimally handled, and the non-parametric nature of the KDE implied increasing computation and memory requirements with every new data point. As a remedy, we proposed the usage of normalizing flows, which are state-of-the-art density estimators based on invertible neural networks. Employing a conditional version of a normalizing flow allowed us to use a single contextual model instead of the previously used bag of KDE models. Our experimental evaluation confirmed that the prediction performance drastically increased, while the computational effort remained static for a growing data set. In addition, the contextual model allowed for conditioning on more complex information, such as the relative positions of the other players.

After employing normalizing flows as density estimators, we used them as generative models. The chain of transformations that transforms data into a representation that follows a simple base distribution allows for easy generation of new data due to its invertibility. A prevalent operation in generative modeling is to linearly interpolate samples and thus generate new data. We highlighted issues when using a standard interpolation and a Gaussian base distribution at the same time. Those issues caused either deteriorated the generative performance visually or introduced biases towards the expectation of the base representation, as we demonstrated. The problem is that linear interpolates leave the data manifold, which a high-dimensional Gaussian imposes. As a remedy, we enforced a specific manifold on which it is easy to interpolate. Specifically, we used the hypersphere and the probability simplex as manifolds and the von Mises-Fisher (vMF) and Dirichlet distributions on those manifolds. To use those manifolds, we proposed appropriate transformations that move from an unrestricted space to the manifold. We then conducted quantitative and qualitative empirical evaluations on various well-known image data sets. Our proposed approach did not only convince visually by yielding more natural interpolation paths, but it also produced better interpolations as measured by bits per dimension (BPD), Fréchet inception distance (FID) scores, and kernel inception distance (KID) scores. Hence, we were able to produce better interpolations without sacrificing generative performance.

Future Work

In this thesis, we considered the two aforementioned views on unsupervised representation learning rather independently. Besides, we put our focus on different objectives, *i.e.*, scalability and the facilitation of operations. However, some settings concern both views simultaneously. One example is privacy, which is about preserving private information for specific users. Since representative subsets are usually (weighted) subsets of data, any subset will contain original data points that might reveal private information. Thus, a natural question is how to derive representative subsets which preserve differential privacy. Moreover, privacy is not only of interest for data summaries but also for generative modeling. Another example that is relevant for both views is the setting of fairness in machine learning. Here, the goal is to reduce biases and assure fairness. Such biases might be of algorithmic nature or inherently present in the data. It can be expected that representative subsets also carry inherent biases. Thus, it should be studied whether those subsets introduce bias and how fair representative subsets can be derived. Finally, the most important research direction is the combination of both views. For example, when dealing with data sets that are large in sample size and dimensionality, we could first compute a representative subset and then reduce the dimensionality. However, the same could be done in the reversed order. Since both operations are usually approximations and thus introduce errors, it is important to study how to combine both views optimally. Last but not least, an important question that follows is whether and how the combination of both views can be done simultaneously.

Bibliography

Agarwal, P. K., Har-Peled, S., and Varadarajan, K. R. (2004). Approximating extent measures of points. *Journal of the ACM (JACM)*, 51(4):606–635.

Ageev, A. A. and Sviridenko, M. I. (2004). Pipage rounding: A new method of constructing algorithms with proven performance guarantee. *Journal of Combinatorial Optimization*, 8(3):307–328.

Agustsson, E., Sage, A., Timofte, R., and Gool, L. V. (2019). Optimal transport maps for distribution preserving operations on latent spaces of generative models. In *International Conference on Learning Representations*.

Allen-Zhu, Z., Li, Y., Singh, A., and Wang, Y. (2017). Near-optimal design of experiments via regret minimization. In *International Conference on Machine Learning*, pages 126–135.

Arora, R., Gupta, M., Kapila, A., and Fazel, M. (2011). Clustering by left-stochastic matrix factorization. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 761–768.

Arora, R., Gupta, M. R., Kapila, A., and Fazel, M. (2013). Similarity-based clustering by left-stochastic matrix factorization. *Journal of Machine Learning Research*, 14(1):1715–1746.

Arvanitidis, G., Hansen, L. K., and Hauberg, S. (2018). Latent space oddity: on the curvature of deep generative models. In *International Conference on Learning Representations*.

Avci, A., Bosch, S., Marin-Perianu, M., Marin-Perianu, R., and Havinga, P. (2010). Activity recognition using inertial sensing for healthcare, wellbeing and sports applications: A survey. In *23th International Conference on Architecture of Computing Systems 2010*, pages 1–10.

Avron, H. and Boutsidis, C. (2013). Faster subset selection for matrices and applications. *SIAM Journal on Matrix Analysis and Applications*, 34(4):1464–1499.

Bachem, O., Lucic, M., Hassani, S. H., and Krause, A. (2017). Uniform deviation bounds for k-means clustering. In *International Conference on Machine Learning*, pages 283–291.

Bachem, O., Lucic, M., and Krause, A. (2018a). Scalable k-means clustering via lightweight coresets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1119–1127. ACM.

Bachem, O., Lucic, M., and Lattanzi, S. (2018b). One-shot coresets: The case of k-clustering. In *International conference on artificial intelligence and statistics*, pages 784–792. PMLR.

Barber, C. B., Dobkin, D. P., and Huhdanpaa, H. (1996). The quickhull algorithm for convex hulls. *ACM Transactions on Mathematical Software (TOMS)*, 22(4):469–483.

Barris, S. and Button, C. (2008). A review of vision-based motion analysis in sport. *Sports Medicine*, 38(12):1025–1043.

Bauckhage, C., Kersting, H., Thurau, C., et al. (2015). Archetypal analysis as an autoencoder. In *Workshop New Challenges in Neural Computation*.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

Bertin-Mahieux, T., Ellis, D. P., Whitman, B., and Lamere, P. (2011). The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*.

Billingsley, P. (2008). *Probability and measure*. John Wiley & Sons.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. (2018). Demystifying MMD GANs. In *International Conference on Learning Representations*.

Blackard, J. A. and Dean, D. J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Computers and electronics in agriculture*, 24(3):131–151.

Bose, J., Smofsky, A., Liao, R., Panangaden, P., and Hamilton, W. (2020). Latent variable modelling with hyperbolic normalizing flows. In *International Conference on Machine Learning*, pages 1045–1055. PMLR.

Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 144–152.

Bowman, S., Vilnis, L., Vinyals, O., Dai, A., Jozefowicz, R., and Bengio, S. (2016). Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., and Wanderman-Milne, S. (2018). JAX: composable transformations of Python+NumPy programs.

Brefeld, U., Lasek, J., and Mair, S. (2019). Probabilistic movement models and zones of control. *Machine Learning*, 108(1):127–147.

Brefeld, U., Lasek, J., and Mair, S. (2020). Analyzing positional data. In Ley, C. and Dominicy, Y., editors, *Science meets Sports : when Statistics are more than Numbers*, chapter 4, pages 81–94. Cambridge Scholars Publishing.

Brehmer, J. and Cranmer, K. (2020). Flows for simultaneous manifold learning and density estimation. *Advances in Neural Information Processing Systems*, 33.

Bro, R. and De Jong, S. (1997). A fast non-negativity-constrained least squares algorithm. *Journal of chemometrics*, 11(5):393–401.

Brondsted, A. (2012). *An introduction to convex polytopes*, volume 90. Springer Science & Business Media.

Brooks, J., Kerr, M., and Guttag, J. (2016). Using machine learning to draw inferences from pass location data in soccer. *Statistical Analysis and Data Mining: The ASA Data Science Journal*, 9(5):338–349.

Brooks, T. F., Pope, D. S., and Marcolini, M. A. (1989). Airfoil self-noise and prediction.

Byrne, M., Parry, T., Isola, R., and Dawson, A. (2013). Identifying road defect information from smartphones. *Road & Transport Research*, 22(1):39–50.

Campbell, T. and Beronov, B. (2019). Sparse variational inference: Bayesian coresets from scratch. *Advances in Neural Information Processing Systems*, 32:11461–11472.

Campbell, T. and Broderick, T. (2018). Bayesian coreset construction via greedy iterative geodesic ascent. In *International Conference on Machine Learning*, pages 697–705.

Campbell, T. and Broderick, T. (2019). Automated scalable bayesian inference via hilbert coresets. *The Journal of Machine Learning Research*, 20(1):551–588.

Catalin Ionescu, Fuxin Li, C. S. (2011). Latent structured models for human pose estimation. In *International Conference on Computer Vision*.

Chang, C.-c. and Lin, C.-J. (2001). Ijcnn 2001 challenge: Generalization ability and text decoding. In *IJCNN'01. International Joint Conference on Neural Networks. Proceedings (Cat. No. 01CH37222)*, volume 2, pages 1031–1036. IEEE.

Chao, W.-L., Gong, B., Grauman, K., and Sha, F. (2015). Large-margin determinantal point processes. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, pages 191–200.

Chaudhuri, K., Kakade, S. M., Netrapalli, P., and Sanghavi, S. (2015). Convergence rates of active learning for maximum likelihood estimation. In *Advances in Neural Information Processing Systems*, pages 1090–1098.

Chen, K. (2009). On coresets for k-median and k-means clustering in metric and euclidean spaces and their applications. *SIAM Journal on Computing*, 39(3):923–947.

Chen, T. Q., Behrmann, J., Duvenaud, D. K., and Jacobsen, J.-H. (2019). Residual flows for invertible generative modeling. In *Advances in Neural Information Processing Systems*, pages 9913–9923.

Chen, Y., Mairal, J., and Harchaoui, Z. (2014). Fast and robust archetypal analysis for representation learning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1478–1485.

Clanuwat, T., Bober-Irizar, M., Kitamoto, A., Lamb, A., Yamamoto, K., and Ha, D. (2018). Deep learning for classical japanese literature.

Cortes, C. and Vapnik, V. (1995). Support-vector networks. *Machine learning*, 20(3):273–297.

Coutts, A. J., Quinn, J., Hocking, J., Castagna, C., and Rampinini, E. (2010). Match running performance in elite Australian Rules Football. *Journal of Science and Medicine in Sport*, 13(5):543–548.

Cutler, A. and Breiman, L. (1994). Archetypal analysis. *Technometrics*, 36(4):338–347.

Davidson, T. R., Falorsi, L., De Cao, N., Kipf, T., and Tomczak, J. M. (2018). Hyperspherical variational auto-encoders. In *Proceedings of the Thirty-Fourth Conference on Uncertainty in Artificial Intelligence*, pages 856–865.

De Cao, N., Titov, I., and Aziz, W. (2019). Block neural autoregressive flow. *35th Conference on Uncertainty in Artificial Intelligence (UAI19)*.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22.

Derezinski, M. and Warmuth, M. (2018). Subsampling for ridge regression via regularized volume sampling. In *International Conference on Artificial Intelligence and Statistics*, pages 716–725. PMLR.

Deshpande, A., Rademacher, L., Vempala, S., and Wang, G. (2006). Matrix approximation and projective clustering via volume sampling. *Theory of Computing*, 2(1):225–247.

Dick, U. and Brefeld, U. (2019). Learning to rate player positioning in soccer. *Big data*, 7(1):71–82.

Ding, C. H., Li, T., and Jordan, M. I. (2010). Convex and semi-nonnegative matrix factorizations. *IEEE transactions on pattern analysis and machine intelligence*, 32(1):45–55.

Dinh, L., Krueger, D., and Bengio, Y. (2015). NICE: non-linear independent components estimation. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Workshop Track Proceedings*.

Dinh, L., Shol-Dickstein, J., and Bengio, S. (2017). Density estimation using Real NVP. In *International Conference on Learning Representations*.

Dolia, A. N., De Bie, T., Harris, C. J., Shawe-Taylor, J., and Titterington, D. M. (2006). The minimum volume covering ellipsoid estimation in kernel-defined feature spaces. In *European Conference on Machine Learning*, pages 630–637. Springer.

D'Orazio, T. and Leo, M. (2010). A review of vision-based systems for soccer video analysis. *Pattern Recognition*, 43(8):2911–2926.

Drineas, P. and Mahoney, M. W. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(Dec):2153–2175.

Dulá, J. H. and Helgason, R. V. (1996). A new procedure for identifying the frame of the convex hull of a finite collection of points in multidimensional space. *European Journal of Operational Research*, 92(2):352–367.

Dulá, J. H. and López, F. J. (2012). Competing output-sensitive frame algorithms. *Computational Geometry*, 45(4):186–197.

DuMouchel, W., Volinsky, C., Johnson, T., Cortes, C., and Pregibon, D. (1999). Squashing flat files flatter. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 6–15.

Dwork, C. (2008). Differential privacy: A survey of results. In *International conference on theory and applications of models of computation*, pages 1–19. Springer.

Epifanio, I., Vinué, G., and Alemany, S. (2013). Archetypal analysis: contributions for estimating boundary cases in multivariate accommodation problem. *Computers & Industrial Engineering*, 64(3):757–765.

Eugster, M. J. and Leisch, F. (2011). Weighted and robust archetypal analysis. *Computational Statistics & Data Analysis*, 55(3):1215–1225.

Fadel, S. G., Mair, S., da Silva Torres, R., and Brefeld, U. (2021a). Contextual movement models based on normalizing flows. *AStA Advances in Statistical Analysis*, pages 1–22.

Fadel, S. G., Mair, S., da Silva Torres, R., and Brefeld, U. (2021b). Principled interpolation in normalizing flows. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 116–131. Springer.

Farahat, A., Ghodsi, A., and Kamel, M. (2011). A novel greedy algorithm for nyström approximation. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 269–277.

Fedorov, V. V. (1972). *Theory of optimal experiments*. Elsevier.

Feldman, D. (2020). Core-sets: Updated survey. In *Sampling Techniques for Supervised or Unsupervised Tasks*, pages 23–44. Springer.

Feldman, D. and Langberg, M. (2011). A unified framework for approximating and clustering data. In *Proceedings of the forty-third annual ACM symposium on Theory of computing*, pages 569–578.

Feldman, D., Schmidt, M., and Sohler, C. (2020). Turning big data into tiny data: Constant-size coresets for k-means, pca, and projective clustering. *SIAM Journal on Computing*, 49(3):601–657.

Feldman, D., Volkov, M., and Rus, D. (2016). Dimensionality reduction of massive sparse datasets using coresets. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, pages 2774–2782.

Fonseca, S., Milho, J., Travassos, B., and Araújo, D. (2012). Spatial dynamics of team sports exposed by Voronoi diagrams. *Human Movement Science*, 31(6):1652–1659.

Franks, A., Miller, A., Bornn, L., and Goldsberry, K. (2015). Characterizing the spatial structure of defensive skill in professional basketball. *Ann. Appl. Stat.*, 9(1):94–121.

Fujimura, A. and Sugihara, K. (2005). Geometric analysis and quantitative evaluation of sport teamwork. *Systems and Computers in Japan*, 36(6):49–58.

Gemici, M. C., Rezende, D., and Mohamed, S. (2016). Normalizing flows on riemannian manifolds. *arXiv preprint arXiv:1611.02304*.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. `http://www.deeplearningbook.org`.

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

Gottfried, B. (2008). Representing short-term observations of moving objects by a simple visual language. *Journal of Visual Languages & Computing*, 19(3):321–342.

Gottfried, B. (2011). Interpreting motion events of pairs of moving objects. *GeoInformatica*, 15(2):247–271.

Grün, T. v. d., Franke, N., Wolf, D., Witt, N., and Eidloth, A. (2011). *A real-time tracking system for football match and training analysis*, pages 199–212. Springer Berlin Heidelberg, Berlin, Heidelberg.

Gudmundsson, J. and Horton, M. (2017). Spatio-temporal analysis of team sports. *ACM Comput. Surv.*, 50(2):22:1–22:34.

Gudmundsson, J. and Wolle, T. (2014). Football analysis using spatio-temporal tools. *Computers, Environment and Urban Systems*, 47:16 – 27.

Haase, J. and Brefeld, U. (2014). Mining positional data streams. In *International Workshop on New Frontiers in Mining Complex Patterns*, pages 102–116. Springer.

Har-Peled, S. and Kushal, A. (2007). Smaller coresets for k-median and k-means clustering. *Discrete & Computational Geometry*, 37(1):3–19.

Har-Peled, S. and Mazumdar, S. (2004). On coresets for k-means and k-median clustering. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, pages 291–300.

Harmon, M., Lucey, P., and Klabjan, D. (2016). Predicting shot making in basketball learnt from adversarial multiagent trajectories. *ArXiv e-prints*.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., et al. (2020). Array programming with numpy. *Nature*, 585(7825):357–362.

Haussler, D. (1992). Decision theoretic generalizations of the pac model for neural net and other learning applications. *Information and Computation*, 100(1):78–150.

Heinz, G., Peterson, L. J., Johnson, R. W., and Kerk, C. J. (2003). Exploring relationships in body dimensions. *Journal of Statistics Education*, 11(2).

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

Horton, M., Gudmundsson, J., Chawla, S., and Estephan, J. (2015). Automated classification of passing in football. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 319–330. Springer.

Horton, P. and Nakai, K. (1996). A probabilistic classification system for predicting the cellular localization sites of proteins. In *Ismb*, volume 4, pages 109–115.

Huang, C.-W., Krueger, D., Lacoste, A., and Courville, A. (2018). Neural autoregressive flows. In *International Conference on Machine Learning*, pages 2078–2087.

Hübl, F., Cvetojevic, S., Hochmair, H., and Paulus, G. (2017). Analyzing refugee migration patterns using geo-tagged tweets. *ISPRS International Journal of Geo-Information*, 6(10):302.

Huggins, J., Campbell, T., and Broderick, T. (2016). Coresets for scalable bayesian logistic regression. In *Advances in Neural Information Processing Systems*, pages 4080–4088.

Ionescu, C., Papava, D., Olaru, V., and Sminchisescu, C. (2014). Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Janetzko, H., Sacha, D., Stein, M., Schreck, T., Keim, D. A., and Deussen, O. (2014). Feature-driven visual analytics of soccer data. In *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 13–22.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of GANs for improved quality, stability, and variation. In *International Conference on Learning Representations*.

Keller, S. M., Samarin, M., Torres, F. A., Wieser, M., and Roth, V. (2020). Learning extremal representations with deep archetypal analysis. *International Journal of Computer Vision*, pages 1–16.

Keller, S. M., Samarin, M., Wieser, M., and Roth, V. (2019). Deep archetypal analysis. In *German Conference on Pattern Recognition*, pages 171–185. Springer.

Kilcher, Y., Lucchi, A., and Hofmann, T. (2018). Semantic interpolation in implicit models. In *International Conference on Learning Representations*.

Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Kingma, D. P. and Dhariwal, P. (2018). Glow: Generative flow with invertible 1x1 convolutions. In *Advances in Neural Information Processing Systems*, pages 10215–10224.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. (2016). Improved variational inference with inverse autoregressive flow. In *Advances in Neural Information Processing Systems*, pages 4743–4751.

Kingma, D. P. and Welling, M. (2014). Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Klambauer, G., Unterthiner, T., Mayr, A., and Hochreiter, S. (2017). Self-Normalizing Neural Networks. In *Advances in Neural Information Processing Systems 30*, pages 971–980.

Knauf, K., Memmert, D., and Brefeld, U. (2016). Spatio-temporal convolution kernels. *Machine Learning*, 102(2):247–273.

Krizhevsky, A. and Hinton, G. (2009). Learning multiple layers of features from tiny images. Technical report.

Kulesza, A. and Taskar, B. (2010). Structured determinantal point processes. *Advances in Neural Information Processing Systems*, 23:1171–1179.

Kulesza, A. and Taskar, B. (2011). k-dpps: fixed-size determinantal point processes. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 1193–1200.

Kulesza, A., Taskar, B., et al. (2012). Determinantal point processes for machine learning. *Foundations and Trends® in Machine Learning*, 5(2–3):123–286.

Lago-Peñas, C., Rey, E., Lago-Ballesteros, J., Casais, L., and Domínguez, E. (2009). Analysis of work-rate in soccer according to playing positions. *International Journal of Performance Analysis in Sport*, 9(2):218–227.

Lasek, J. and Gagolewski, M. (2015). The winning solution to the AAIA'15 Data Mining Competition: Tagging firefighter activities at a fire scene. In *2015 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 375–380.

Laube, P., Imfeld, S., and Weibel, R. (2005). Discovering relative motion patterns in groups of moving point objects. *International Journal of Geographical Information Science*, 19(6):639–668.

Lawson, C. L. and Hanson, R. J. (1995). *Solving least squares problems*, volume 15. SIAM.

Le, H. M., Carr, P., Yue, Y., and Lucey, P. (2017). Data-driven ghosting using deep imitation learning. In *MIT Sloan Sports Analytics Conference*.

LeCun, Y. (1987). *Modeles connexionnistes de l'apprentissage (connectionist learning models)*. PhD thesis, Université P. et M. Curie (Paris 6).

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W., and Jackel, L. D. (1989). Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551.

LeCun, Y., Cortes, C., and Burges, C. (2010). Mnist handwritten digit database. *ATT Labs [Online]. Available: http://yann.lecun.com/exdb/mnist*, 2.

Lee, D. D. and Seung, H. S. (1999). Learning the parts of objects by non-negative matrix factorization. *Nature*, 401(6755):788–791.

Lee, D. D. and Seung, H. S. (2000). Algorithms for non-negative matrix factorization. *Advances in Neural Information Processing Systems*, 13:556–562.

Li, C., Jegelka, S., and Sra, S. (2017). Polynomial time algorithms for dual volume sampling. In *Advances in Neural Information Processing Systems*, pages 5045–5054.

Li, Y., Long, P. M., and Srinivasan, A. (2001). Improved bounds on the sample complexity of learning. *Journal of Computer and System Sciences*, 62(3):516–527.

Link, D., Lang, S., and Seidenschwarz, P. (2016). Real time quantification of dangerousity in football using spatiotemporal tracking data. *PLOS ONE*, 11(12):1–16.

Lloyd, S. (1982). Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137.

Lopez, F.-J. (2005). Generating random points (or vectors) controlling the percentage of them that are extreme in their convex (or positive) hull. *Journal of Mathematical Modelling and Algorithms*, 4(2):219–234.

Lu, Y. and Huang, B. (2020). Structured output learning with conditional generative flows. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Lucey, P., Bialkowski, A., Carr, P., Foote, E., and Matthews, I. (2012). Characterizing multi-agent team behavior from partial team tracings: Evidence from the English Premier League. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, AAAI'12, pages 1387–1393. AAAI Press.

Lucic, M., Bachem, O., and Krause, A. (2016). Strong coresets for hard and soft bregman clustering with applications to exponential family mixtures. In *Artificial Intelligence and Statistics*.

Macchi, O. (1975). The coincidence approach to stochastic point processes. *Advances in Applied Probability*, 7(1):83–122.

Mair, S., Boubekki, A., and Brefeld, U. (2017). Frame-based data factorizations. In *International Conference on Machine Learning*, pages 2305–2313.

Mair, S. and Brefeld, U. (2019). Coresets for archetypal analysis. In *Advances in Neural Information Processing Systems*, pages 7247–7255.

Mair, S., Rudolph, Y., Closius, V., and Brefeld, U. (2018). Frame-based optimal design. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 447–463. Springer, Cham.

Manousakas, D., Xu, Z., Mascolo, C., and Campbell, T. (2020). Bayesian pseudocoresets. *Advances in Neural Information Processing Systems*, 33.

Mariet, Z. E. and Sra, S. (2017). Elementary symmetric polynomials for optimal experimental design. In *Advances in Neural Information Processing Systems*, pages 2136–2145.

Mathieu, E., Le Lan, C., Maddison, C. J., Tomioka, R., and Teh, Y. W. (2019). Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in Neural Information Processing Systems 32*, pages 12565–12576. Curran Associates, Inc.

Mazimpaka, J. D. and Timpf, S. (2016). Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*, 2016(13):61–99.

McDermott, P. L., Wikle, C. K., and Millspaugh, J. (2017). Hierarchical nonlinear spatio-temporal agent-based models for collective animal movement. *Journal of Agricultural, Biological, and Environmental Statistics*, 6(3):294–312.

Memmert, D., Lemmink, K. A. P. M., and Sampaio, J. (2016). Current approaches to tactical performance analyses in soccer using position data. *Sports Medicine*, pages 1–10.

Miao, Y., Yu, L., and Blunsom, P. (2016). Neural variational inference for text processing. In *International conference on machine learning*, pages 1727–1736. PMLR.

Mohan, P., Padmanabhan, V. N., and Ramjee, R. (2008). Nericell: Rich monitoring of road and traffic conditions using mobile smartphones. In *Proceedings of the 6th ACM Conference on Embedded Network Sensor Systems*, SenSys '08, pages 323–336. ACM.

Mørup, M. and Hansen, L. K. (2010). Archetypal analysis for machine learning. In *Machine Learning for Signal Processing (MLSP), 2010 IEEE International Workshop on*, pages 172–177. IEEE.

Mørup, M. and Hansen, L. K. (2012). Archetypal analysis for machine learning and data mining. *Neurocomputing*, 80:54–63.

Munteanu, A., Schwiegelshohn, C., Sohler, C., and Woodruff, D. (2018). On coresets for logistic regression. In *Advances in Neural Information Processing Systems*, pages 6561–6570.

Musco, C. and Musco, C. (2017). Recursive sampling for the nystrom method. In *Advances in Neural Information Processing Systems*, pages 3833–3845.

Mutschler, C., Ziekow, H., and Jerzak, Z. (2013). The DEBS 2013 Grand Challenge. In *Proceedings of the 7th ACM International Conference on Distributed Event-based Systems*, DEBS '13, pages 289–294, New York, NY, USA. ACM.

Nakanishi, R., Maeno, J., Murakami, K., and Naruse, T. (2009). An approximate computation of the dominant region diagram for the real-time analysis of group behaviors. In *Robot Soccer World Cup*, pages 228–239. Springer.

Narizuka, T., Yamamoto, K., and Yamazaki, Y. (2014). Statistical properties of position-dependent ball-passing networks in football games. *Physica A: Statistical Mechanics and its Applications*, 412:157–168.

Nocedal, J. and Wright, S. J. (2006). *Sequential quadratic programming*. Springer.

Ottmann, T., Schuierer, S., and Soundaralakshmi, S. (2001). Enumerating extreme points in higher dimensions. *Nordic Journal of Computing*, 8(2):179–192.

Paatero, P. and Tapper, U. (1994). Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126.

Pace, R. K. and Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297.

Padberg-Gehle, K. and Schneide, C. (2017). Trajectory-based computational study of coherent behavior in flows. *PAMM*, 17(1):11–14.

Paefgen, J., Michahelles, F., and Staake, T. (2011). GPS trajectory feature extraction for driver risk profiling. In *Proceedings of the 2011 International Workshop on Trajectory Data Mining and Analysis*, TDMA '11, pages 53–56, New York, NY, USA. ACM.

Papamakarios, G., Nalisnick, E., Rezende, D. J., Mohamed, S., and Lakshminarayanan, B. (2021). Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, pages 1310–1318.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pages 8024–8035.

Patrini, G., van den Berg, R., Forre, P., Carioni, M., Bhargav, S., Welling, M., Genewein, T., and Nielsen, F. (2020). Sinkhorn autoencoders. In *Uncertainty in Artificial Intelligence*, pages 733–743. PMLR.

Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011).

Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Phillips, J. M. (2016). Coresets and sketches. *arXiv preprint arXiv:1601.00617*.

Phillips, J. M. and Tai, W. M. (2020). Near-optimal coresets of kernel density estimates. *Discrete & Computational Geometry*, 63(4):867–887.

Pukelsheim, F. (2006). *Optimal Design of Experiments*. SIAM.

Radford, A., Metz, L., and Chintala, S. (2016). Unsupervised representation learning with deep convolutional generative adversarial networks. In *International Conference on Learning Representations, ICLR 2016*.

Rahimi, A. and Recht, B. (2007). Random features for large-scale kernel machines. *Advances in Neural Information Processing Systems*, 20:1177–1184.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA.

Rezende, D. and Mohamed, S. (2015). Variational inference with normalizing flows. In *International Conference on Machine Learning*, pages 1530–1538. PMLR.

Rezende, D. J., Mohamed, S., and Wierstra, D. (2014). Stochastic backpropagation and approximate inference in deep generative models. In *International Conference on Machine Learning*, pages 1278–1286.

Rezende, D. J., Papamakarios, G., Racanière, S., Albergo, M., Kanwar, G., Shanahan, P., and Cranmer, K. (2020). Normalizing flows on tori and spheres. In *International Conference on Machine Learning*, pages 8083–8092. PMLR.

Rippel, O. and Adams, R. P. (2013). High-dimensional probability estimation with deep density models. *arXiv preprint arXiv:1302.5125*.

Rossi, A., Pappalardo, L., Cintia, P., Fernandez, J., Iaia, F. M., and Medina, D. (2017). Who is going to get hurt? Predicting injuries in professional soccer. In *Proceedings the Machine Learning and Data Mining for Sports Analytics workshop (MLSA'17), ECML/PKDD*, CGI '00, pages 227–235.

Rumelhart, D., Hinton, G., and Williams, R. (1986). Learning internal representations by error propagation. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1: foundations*, pages 318–362.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pages 2234–2242.

Santoro, A., Raposo, D., Barrett, D. G., Malinowski, M., Pascanu, R., Battaglia, P., and Lillicrap, T. (2017). A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems*, pages 4967–4976.

Schölkopf, B., Smola, A., and Müller, K.-R. (1998). Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Schölkopf, B. and Smola, A. J. (2002). *Learning with kernels: support vector machines, regularization, optimization, and beyond*. MIT press.

Schreurs, J., Fanuel, M., and Suykens, J. A. (2019). Towards deterministic diverse subset sampling. In *Artificial Intelligence and Machine Learning*, pages 137–151. Springer.

Scott, D. W. (2015). *Multivariate density estimation: theory, practice, and visualization*. John Wiley & Sons.

Shoemake, K. (1985). Animating rotation with quaternion curves. In *Proceedings of the 12th annual conference on Computer graphics and interactive techniques*, pages 245–254.

Smola, A. J., Schölkopf, B., and Müller, K.-R. (1998). The connection between regularization operators and support vector kernels. *Neural Networks*, 11(4):637–649.

Snelson, E. and Ghahramani, Z. (2005). Sparse gaussian processes using pseudo-inputs. *Advances in Neural Information Processing Systems*, 18:1257–1264.

Son, L. H. (2016). Dealing with the new user cold-start problem in recommender systems: A comparative review. *Information Systems*, 58:87–104.

Spearman, W., Pop, P., Basye, A., Hotovy, R., and Dick, G. (2017). Physics-based modeling of pass probabilities in soccer. In *Proceedings of the 11th MIT Sloan Sports Analytics Conference*, pages 1–14.

Sprado, J. and Gottfried, B. (2009). *What motion patterns tell ss about soccer teams*, pages 614–625. Springer Berlin Heidelberg.

Sugiyama, M. and Nakajima, S. (2009). Pool-based active learning in approximate linear regression. *Machine Learning*, 75(3):249–274.

Tabak, E. G. and Turner, C. V. (2013). A family of nonparametric density estimation algorithms. *Communications on Pure and Applied Mathematics*, 66(2):145–164.

Tabak, E. G. and Vanden-Eijnden, E. (2010). Density estimation by dual ascent of the log-likelihood. *Communications in Mathematical Sciences*, 8(1):217–233.

Taki, T. and Hasegawa, J. (2000). Visualization of dominant region in team games and its application to teamwork analysis. In *Proceedings of the International Conference on Computer Graphics*, CGI '00, pages 227–235, Washington, DC, USA. IEEE Computer Society.

Taki, T., Hasegawa, J., and Fukumura, T. (1996). Development of motion analysis system for quantitative evaluation of teamwork in soccer games. In *Proceedings of 3rd IEEE International Conference on Image Processing*, volume 3, pages 815–818 vol.3.

Thurau, C., Kersting, K., Wahabzada, M., and Bauckhage, C. (2011). Convex non-negative matrix factorization for massive datasets. *Knowledge and information systems*, 29(2):457–478.

Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

Titsias, M. (2009). Variational learning of inducing variables in sparse gaussian processes. In *Artificial Intelligence and Statistics*, pages 567–574.

Tsang, I. W., Kwok, J. T., and Cheung, P.-M. (2005). Core vector machines: Fast svm training on very large data sets. *Journal of Machine Learning Research*, 6(Apr):363–392.

Tukan, M., Baykal, C., Feldman, D., and Rus, D. (2020a). On coresets for support vector machines. In *International Conference on Theory and Applications of Models of Computation*, pages 287–299. Springer.

Tukan, M., Maalouf, A., and Feldman, D. (2020b). Coresets for near-convex functions. *Advances in Neural Information Processing Systems*, 33.

Ueda, F., Masaaki, H., and Hiroyuki, H. (2014). The causal relationship between dominant region and offense-defense performance – Focusing on the time of ball acquisition. *Football Science*, 11:1–17.

van Dijk, D., Burkhardt, D. B., Amodio, M., Tong, A., Wolf, G., and Krishnaswamy, S. (2019). Finding archetypal spaces using neural networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2634–2643. IEEE.

Vapnik, V. (1991). Principles of risk minimization for learning theory. In *Proceedings of the 4th International Conference on Neural Information Processing Systems*, pages 831–838.

Vinué, G. (2017). Anthropometry: An R package for analysis of anthropometric data. *Journal of Statistical Software*, 77(6):1–40.

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., et al. (2020). Scipy 1.0: fundamental algorithms for scientific computing in python. *Nature methods*, 17(3):261–272.

Voronoi, G. (1908). Nouvelles applications des paramètres continus à la théorie des formes quadratiques. premier mémoire. sur quelques propriétés des formes quadratiques positives parfaites. *Journal für die reine und angewandte Mathematik*, 133:97–178.

Wang, Y., Yu, A. W., and Singh, A. (2017). On computationally tractable selection of experiments in measurement-constrained regression models. *Journal of Machine Learning Research*, 18(143):1–41.

White, T. (2016). Sampling generative networks. *arXiv preprint arXiv:1609.04468*.

Williams, C. K. I. and Seeger, M. (2001). Using the nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., and Tresp, V., editors, *Advances in Neural Information Processing Systems 13*, pages 682–688. MIT Press.

Winkler, C., Worrall, D., Hoogeboom, E., and Welling, M. (2019). Learning likelihoods with conditional normalizing flows. *arXiv preprint arXiv:1912.00042*.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

Xu, J. and Durrett, G. (2018). Spherical latent spaces for stable variational autoencoders. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4503–4513.

Yeh, I.-C. (1998). Modeling of strength of high-performance concrete using artificial neural networks. *Cement and Concrete research*, 28(12):1797–1808.

Zhang, J. Y., Khanna, R., Kyrillidis, A., and Koyejo, O. (2020). Bayesian coresets: An optimization perspective. *arXiv preprint arXiv:2007.00715*.

Zhang, P., Beernaerts, J., Zhang, L., and de Weghe, N. V. (2016). Visual exploration of match performance based on football movement data using the Continuous Triangular Model. *Applied Geography*, 76(Supplement C):1–13.

Zhao, Y., Yin, F., Gunnarsson, F., Hultkratz, F., and Fagerlind, J. (2016). Gaussian processes for flow modeling and prediction of positioned trajectories evaluated with sports data. In *2016 19th International Conference on Information Fusion (FUSION)*, pages 1461–1468.

Zheng, S., Yue, Y., and Hobbs, J. (2016). Generating long-term trajectories using deep hierarchical networks. In *Advances in Neural Information Processing Systems 29*, pages 1543–1551.

Zheng, Y. (2015). Trajectory data mining: An overview. *ACM Trans. Intell. Syst. Technol.*, 6(3):29:1–29:41.

Zhou, T., Kuscsik, Z., Liu, J.-G., Medo, M., Wakeling, J. R., and Zhang, Y.-C. (2010). Solving the apparent diversity-accuracy dilemma of recommender systems. *Proceedings of the National Academy of Sciences*, 107(10):4511–4515.

Appendix A

Appendix of Probabilistic Movement Models and Zones of Control

To shed more light on the different movement models and their implication on the zones of control, the following Figures show four exemplary situations for Voronoi-based movement models, approaches by Taki & Hasegawa and Fujimura & Sugihara as well as the proposed probabilistic movement model.

Figure A.1 shows perhaps the most relevant situation for coaches and analysts. The probabilistic movement model on the top right clearly identifies a scoring opportunity for the white team. The white player in the center of the pitch creates a large zone of control behind the defending black players. If the ball possessing player plays the ball into this zone, the white player may have enough time to control the ball and to create a one-on-one with the goal keeper. Except for Taki & Hasegawa, the baselines fail to detect this.

In general, Figure A.2 shows that the baselines either lead to unnatural square- and rectangle-like shapes (Voronoi and Fujimura & Sugihara) or implausible drop-like areas (Taki & Hasegawa) as a consequence of implicit assumptions and constraints in the models. Our approach allows to capture movements irrespectively of the resulting shapes of the zones as there are no assumptions on the movements.

Velocities are generally an issue for the baseline approaches. Figure A.3 shows an example where we focus only on the ball possessing player and the white striker that runs towards her. The region in the Voronoi-based approach are clearly too small for the running player. By contrast, Taki & Hasegawa and Fujimura & Sugihara overestimate the impact of the approaching white player and render the ball possessing player outside of her own region of control. The approach by Taki & Hasegawa even credits a surprisingly large area to the second white player from the left. Interestingly,

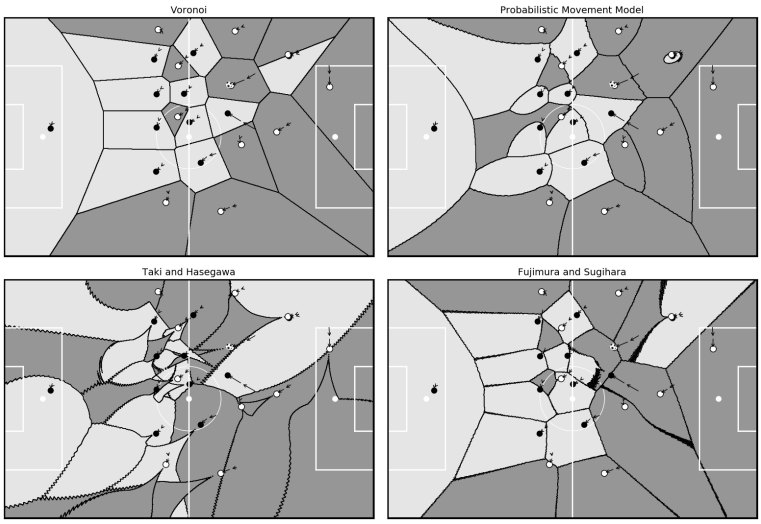

Figure A.1: Illustration of a scoring opportunity for the white team.

this player is almost standing and only gets an area this large because the zones of the other players evolve drop-like into the direction of movement. A remedy to such artefacts is to compute the controlled zones with underlying probabilistic movement models. The respective figure on the top right shows realistic areas that are easily interpreted.

Figure A.2: Implicit assumptions in baselines constrain possible shapes of zones.

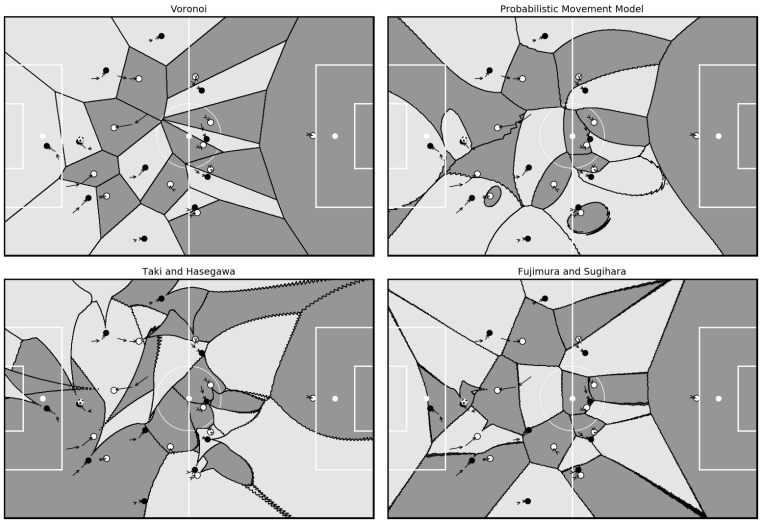

Figure A.3: High velocities not appropriately captured by baselines.

Appendix B

Appendix of Contextual Movement Models

B.1 Influence of t_δ

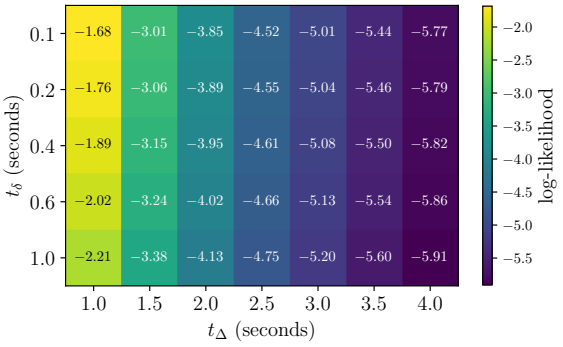

Figure B.1: Evaluation of the impact of t_δ.

Figure B.1 depicts the log-likelihood values of CFlow for the same setting as Figure 8.3, but now also varying t_δ. The figure shows that there is little influence from varying t_δ when compared to t_Δ. Although a choice of $t_\delta = 0.1s$ seems to be better, we use $t_\delta = 0.2s$ just as in Chapter 7.

B.2 Details of CFlow-extended

CFlow-extended uses contextual information given by $\mathbf{c}_t = (v_t, t_\Delta, \bar{x}_t^{\text{rel}})$, which extends the current speed of the player v_t and the time horizon t_Δ by the positions of the other players, relative to the player under consideration, *i.e.*, $\bar{x}_t^{\text{rel}}$. For soccer, there are $R = 21$ remaining players whose relative positions are summarized in the set $\mathcal{X}_t^{\text{rel}} = \{\mathbf{x}_t^{(1)}, \ldots, \mathbf{x}_t^{(R)}\}$.

Table B.1: Network architectures for CFlow-extended.

Network	Architecture and Activation Functions
$\mathbf{g}_{\text{pair}}$	$2d \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}}$
$\mathbf{g}_{\text{global}}$	$d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} d_{\text{hidden}} \xrightarrow{\text{SELU}} \frac{1}{2} d_{\text{hidden}}$

However, instead of simply stacking those positions into a vector, we leverage an aggregated representation $\bar{x}_t^{\text{rel}} \in \mathbb{R}^{\frac{1}{2}d_{\text{hidden}}}$ to circumvent the problem of ordering those players. Here, d_{hidden} is the dimensionality of the aggregated representation $\bar{x}_t^{\text{rel}}$. The latter is obtained by employing a relation network (Santoro et al., 2017) and defined as

$$\bar{x}_t^{\text{rel}} = \mathbf{g}_{\text{global}} \left(\frac{1}{R^2} \sum_{i=1}^{R} \sum_{j=1}^{R} \mathbf{g}_{\text{pair}} \left(\mathbf{x}_t^{(i)}, \mathbf{x}_t^{(j)} \right) \right),$$

where both $\mathbf{g}_{\text{pair}} : \mathbb{R}^{2d} \to \mathbb{R}^{d_{\text{hidden}}}$, $\mathbf{g}_{\text{global}} : \mathbb{R}^{d_{\text{hidden}}} \to \mathbb{R}^{\frac{1}{2}d_{\text{hidden}}}$ are small feed-forward neural networks with architectures outlined in Table B.1. The output of this aggregation step, $\bar{x}_t^{\text{rel}}$, is then used by the conditioning networks $\text{CN}_{\text{ca}}(\mathbf{c})$ and $\text{CN}_{\text{cac}}(\mathbf{c})$ whose input dimensionalities increase by $\frac{1}{2}d_{\text{hidden}}$. An example of the relation network is depicted in Figure B.2. The remainder of the flow is then carried out in the same way as CFlow.

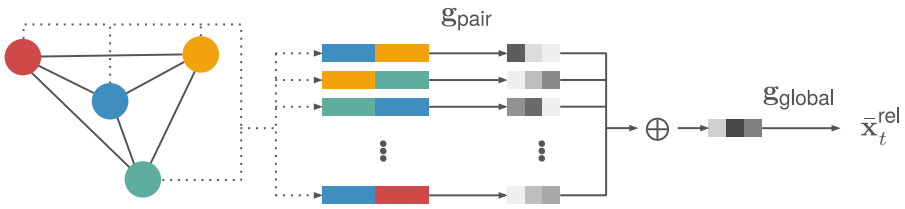

Figure B.2: An example of the relation network with only four objects $\mathbf{x}_t^{(i)}$ for simplicity. The colored nodes represent the objects $\mathbf{x}_t^{(i)}$.

Appendix C

Appendix of Principled Interpolation in Normalizing Flows

Figure C.1: Two additional interpolation paths of samples from CelebA. Left: Linear (1st and 3rd row) and norm-corrected (2nd and 4th row) interpolations paths. Right: decoded expectation of base distribution, *i.e.*, the mean face.

C.1 Additional Interpolation Paths on CelebA

Figure C.1 depicts two additional interpolation paths of images taken from CelebA. Analogously to Figure 9.2, the top row shows a linear interpolation and the bottom row an interpolation with norm correction as introduced in Equation (9.1). While this correction guarantees that the norms of interpolants stay within the observed range of data, we note that this is a

Figure C.2: Samples generated from the models for CIFAR10 (top left), Fashion-MNIST (top right), Kuzushiji-MNIST (bottom left), and MNIST (bottom right). Per data set, we show one row of samples from Gaussian, vMF ($\kappa = 1d$), vMF ($\kappa = 1.5d$), and vMF ($\kappa = 2d$), where the order is top to bottom.

rather ad-hoc way to perform the interpolation as we will point out in the remainder.

The leftmost and rightmost faces in Figure C.1 are real data while the remaining ones are interpolants. The mean face shown on the right hand side is clearly visible in the central interpolants; *e.g.*, the glasses disappear in the top row around the center and then reappear while the lips in the third row become more prominent towards the center. We credit both to the properties of the mean face. In contrast, the norm-corrected interpolation does not suffer from distortions by the mean face and has many desirable properties, but also a major limitation. The first and last interpolants are very close to the end points, which are real data. Figure 9.5 depicts the same phenomenon for CIFAR10.

C.2 Samples from the Models

Figure C.2 shows twelve samples per model. The figure essentially shows that the choice of the base distribution does not influence the quality of the generated samples. Changing the base distribution thus leads to more natural interpolations without sacrificing generative performance. Note that better looking samples can be produced with a more sophisticated architecture, but this is not the goal of this chapter.

C.3 Interpolations Across and Within Classes

Table C.1 depicts the same experiments as shown in Table 9.2. The difference, however, is that the interpolations are no longer restricted to be within classes but uniformly sampled and thus also contain interpolations across classes. Using the vMF as a base distribution clearly outperforms the

Table C.1: Results for interpolation averaged over three independent runs including standard errors. Interpolations are within and across classes and use five intermediate points; *lerp* refers to a linear interpolation; *nclerp* refers to the norm-corrected linear interpolation (Section 9.1) and *slerp* refers to the spherical interpolation.

	Base dist.	Type	BPD	FID	KID
MNIST	Gaussian	lerp	**1.34** ± 0.06	9.78 ± 0.50	0.007 ± 0.001
	Gaussian	nclerp	1.50 ± 0.12	11.66 ± 1.06	0.009 ± 0.001
	vMF $\kappa = 1d$	slerp	**1.34** ± 0.10	6.70 ± 0.46	0.005 ± 0.001
	vMF $\kappa = 1.5d$	slerp	1.41 ± 0.09	**6.65** ± 0.20	**0.004** ± 0.000
	vMF $\kappa = 2d$	slerp	1.65 ± 0.09	8.31 ± 0.20	0.006 ± 0.000
	Dirichlet $\alpha = 2$	lerp	1.61 ± 0.10	8.96 ± 0.32	0.006 ± 0.000
K-MNIST	Gaussian	lerp	2.00 ± 0.16	31.51 ± 2.53	0.034 ± 0.003
	Gaussian	nclerp	1.85 ± 0.18	29.76 ± 2.44	0.033 ± 0.003
	vMF $\kappa = 1d$	slerp	2.15 ± 0.13	29.39 ± 5.66	0.032 ± 0.006
	vMF $\kappa = 1.5d$	slerp	**1.65** ± 0.06	37.35 ± 3.49	0.041 ± 0.004
	vMF $\kappa = 2d$	slerp	2.09 ± 0.13	**23.74** ± 4.41	**0.026** ± 0.005
	Dirichlet $\alpha = 2$	lerp	1.90 ± 0.04	37.80 ± 4.30	0.042 ± 0.005
F-MNIST	Gaussian	lerp	2.84 ± 0.04	**16.93** ± 0.06	0.011 ± 0.000
	Gaussian	nclerp	2.86 ± 0.03	*13.61* ± 0.52	**0.009** ± 0.000
	vMF $\kappa = 1d$	slerp	**2.69** ± 0.02	20.10 ± 2.11	0.012 ± 0.001
	vMF $\kappa = 1.5d$	slerp	2.78 ± 0.05	17.56 ± 1.46	**0.009** ± 0.001
	vMF $\kappa = 2d$	slerp	2.72 ± 0.06	21.41 ± 0.79	0.013 ± 0.001
CIFAR10	Gaussian	lerp	2.82 ± 0.04	63.17 ± 0.99	0.059 ± 0.001
	Gaussian	nclerp	3.33 ± 0.01	*16.83* ± 0.24	*0.013* ± 0.000
	vMF $\kappa = 1d$	slerp	2.93 ± 0.04	**56.13** ± 0.38	**0.049** ± 0.000
	vMF $\kappa = 1.5d$	slerp	2.85 ± 0.04	59.89 ± 4.95	0.054 ± 0.004
	vMF $\kappa = 2d$	slerp	**2.79** ± 0.06	57.14 ± 3.45	0.051 ± 0.004

Gaussian in terms of bits per dimension on test data. The only exception is a tie on MNIST. This is caused by the same effect that caused Figure 9.1: since the norm drops when conducting a linear interpolation, the interpolants are much closer to the mean. Hence, they possess higher likelihoods in terms of the base distribution and yield better scores, because bits per dimension is a rescaled negative log-likelihood. The results in terms of FID and KID scores on general interpolations are in line with the results in Table 9.2: the vMF base distribution yields better scores on MNIST and K-MNIST. The norm-corrected linear interpolation (*nclerp*) yields lower FID and KID scores on K-MNIST and CIFAR10. We credit this to the biased evaluation as discussed in Section 9.1.

C.4 Additional Interpolations on CIFAR10

Figure C.3 shows interpolation paths on four additional pairs of images from CIFAR10. The order is the same as in Figure 9.6. The first row shows the Gaussian, the second, third, and fourth rows depict the vMF with $\kappa \in \{1d, 1.5d, 2d\}$, respectively. Within the top left part, the Gaussian prior yields the worst interpolation path since the airplane loses its wings. The vMF ($\kappa = 1d$) shows the most natural transition. While a meaningful transition of the images in the top right and bottom left parts is not obvious, the Gaussian base distribution fails to produce a smooth transition of the background and falls back to white. In contrast, all vMF models yield a smooth change of background. The interpolation of the cars depicted in the bottom right part shows an example where both the Gaussian as well as the vMF ($\kappa = 1.5d$) have issues.

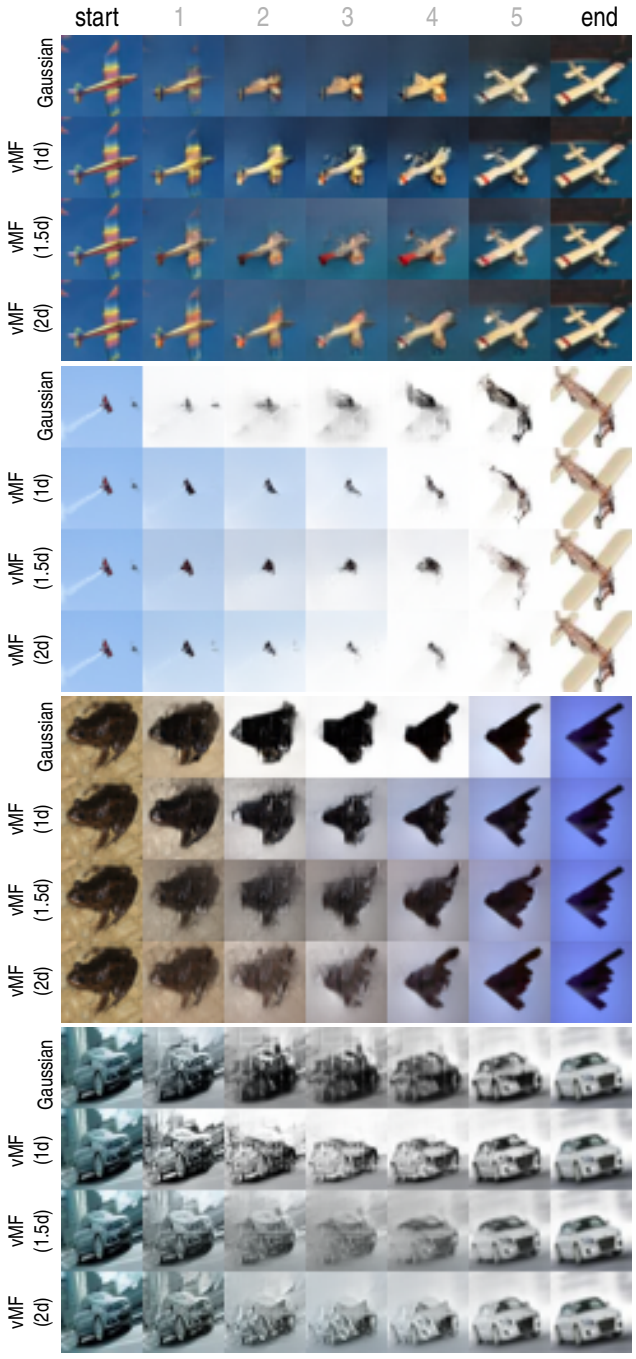

Figure C.3: Additional interpolation paths of four pairs of data from CIFAR10 using different models.

Acknowledgements

Before I close this thesis, I want to take the time to appreciate and thank everybody who helped me, in one way or another, and who made the last couple of years such a wonderful journey for me. Needless to say, it was not always easy, but rather challenging and often exhausting. Nevertheless, and beyond a doubt, it was worth all efforts and an awesome experience overall.

First and foremost, I would like to express my gratitude to my supervisor Ulf. You were extremely supportive since my master studies and you still are. I sincerely appreciate that you let me find my own direction of research, always had an open door to discuss problems and research ideas, and that you provided me with guidance when I needed it. Thank you for pushing me. I learned and enjoyed working with you a lot! Besides, I further appreciate the other members on my committee, Stephan and Søren. I am thankful for meeting you personally at conferences and to discuss my research with you.

Many thanks go of course to the entire ML3 research group for all discussions, team meetings, and activities we had: Ahcène, Daniel, Jennifer, Kai, Maria, Maryam, Samuel, Uwe, and Yannick. Most importantly, I want to thank Ahcène and Maryam with whom I shared an office with right from the beginning for quite some time. I miss all the fun times we had. Special thanks go to Samuel. I really enjoyed all our collaborations and discussions. It was a pleasure. Moreover, I also have to mention *the other PhD office*. I am grateful for all the non-scientific discussions I had with Christoph, Dennis, Felix, and Vincent. It was a fun time!

Another big thank you goes to Madlen and Tanja for their help with administrative matters and all the conversations we had.

I thank my family and friends for all their unconditional support even though you had no idea what I was working on during the past years in Lüneburg.

Last, and most importantly, I am deeply tankful for Cornelia. You were always there, no matter what, and you believed in me even when I had doubts. The amount of thank you's I owe you is uncountable.