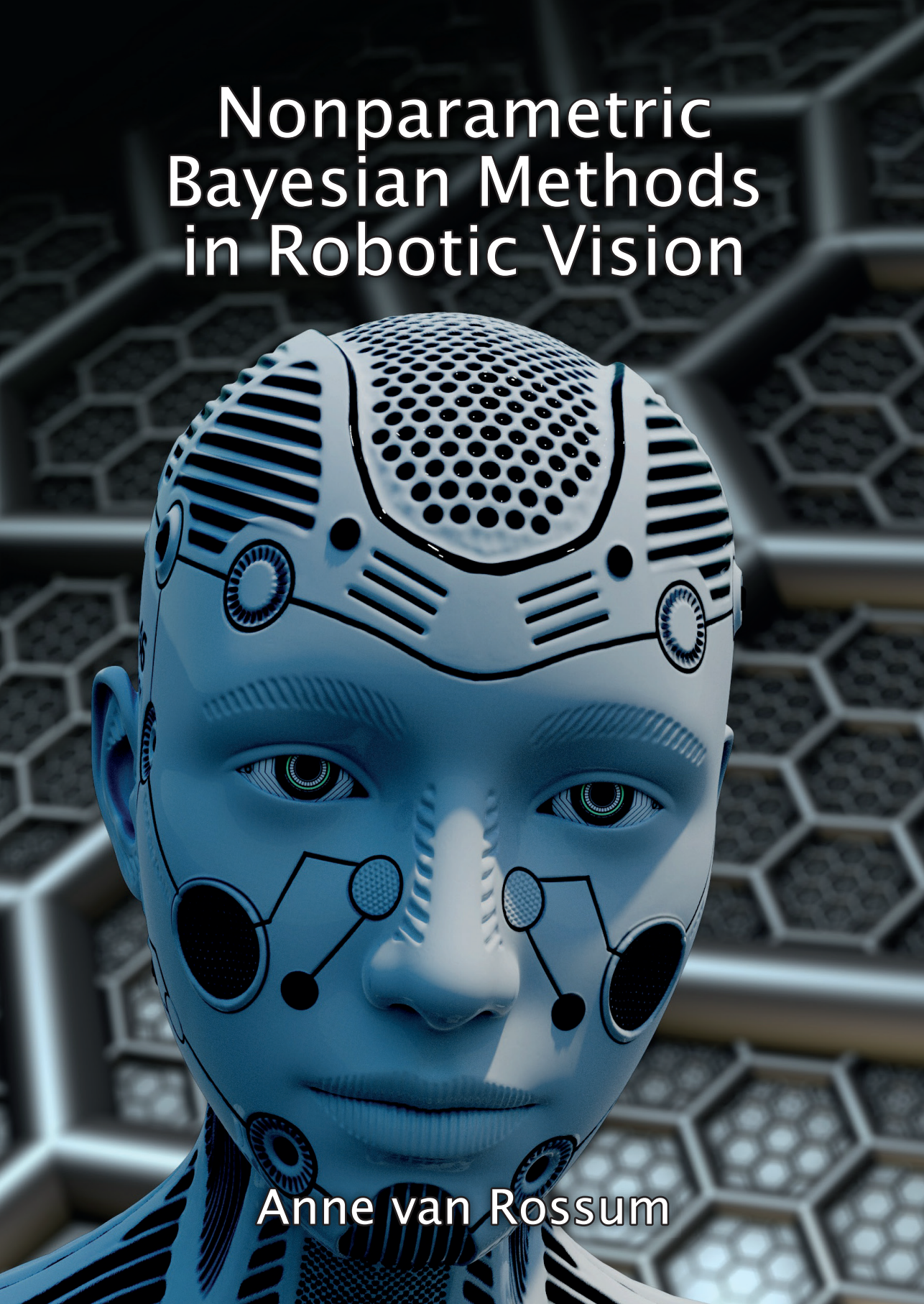
Cover Page

Universiteit Leiden

The handle http://hdl.handle.net/1887/3170176 holds various files of this Leiden University dissertation.

Author: Rossum, A.C. van
Title: Nonparametric Bayesian methods in robotic vision
Issue date: 2021-06-03

Nonparametric Bayesian Methods in Robotic Vision



Anne van Rossum

Nonparametric Bayesian Methods in Robotic Vision

Nonparametric Bayesian Methods in Robotic Vision

PROEFSCHRIFT

ter verkrijging van
de graad van Doctor aan de Universiteit Leiden,
op gezag van Rector Magnificus Prof. dr. ir. H. Bijl,
volgens besluit van het College voor Promoties
te verdedigen op donderdag 3 juni 2021
klokke 12:30 uur

door

Anne Cornelis van Rossum,

geboren te Dirksland
in 1980

Promotoren:	Prof. dr. H. J. van den Herik, Prof. dr. ir. H. X. Lin
Copromotor:	Dr. J. L. A. Dubbeldam (Technische Universiteit Delft)
Promotiecommissie:	Prof. dr. E. R. Eliel, Prof. dr. J. J. Meulman, Prof. dr. P. D. Grünwald, Prof. dr. ir. A. W. Heemink (Technische Universiteit Delft), Prof. dr. A. W. van der Vaart, Dr. F. H. van der Meulen (Technische Universiteit Delft)

SIKS Dissertation Series No. 2021-11

The research reported in the thesis has been carried out under the auspices of SIKS, the Dutch Research School for Information and Knowledge Systems.

ISBN 978-94-6423-258-5

Copyright © 2021, A. C. van Rossum

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronically, mechanically, photocopying, recording or otherwise, without prior permission of the author.

> *The study of mental objects with reproducible properties is called mathematics.*

The Mathematical Experience (Davis and Hersch, 1981)

> *The study of physical objects with reproducible properties is called science.*

The dawning of the age of stochasticity, Mathematics: frontiers and perspectives (Mumford, 2000)

Contents

Appendices

Chapter 1

Introduction

Contents The thesis addresses nonparametric Bayesian methods in robotic vision. Nonparametric Bayesian models can be simultaneously employed to perform inference over the number of entities observed and over the shape or nature of these entities. This chapter introduces nonparametric Bayesian models, the research methodology based on the Bayesian methodology, the main contribution towards robotic vision, and the general organization of the thesis.

Outline The scope of this thesis is to apply nonparametric Bayesian methods to robotic vision (Section 1.1). Bayesian nonparametric models define entities together with noise in such a way that inference can be performed in an optimal manner (Section 1.2). Particular problems in robotic vision that can benefit from Bayesian nonparametric methods are formulated and detailed (Section 1.3). The research methodology is described (Section 1.4). Our main contribution is to introduce nonparametric Bayesian models in robotic vision (Section 1.5). At the end of this chapter the organization of the thesis is given (Section 1.6).

1.1 Scope of the Thesis

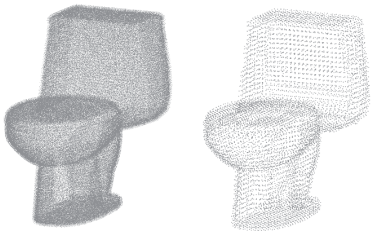
In the thesis, modern Bayesian nonparametric methods are used to answer long-standing questions within computer and robotic vision. The following three challenging questions are typical examples. Is there a Bayesian form of line detection rather than applying the traditional Hough transform? Which of the nonparametric Bayesian priors can be used to detect multiple features simultaneously? What are efficient inference methods for these priors?

The scope of the thesis is the transfer of knowledge on Bayesian nonparametrics to well-described application domains. It will not establish a new body of work around a new family

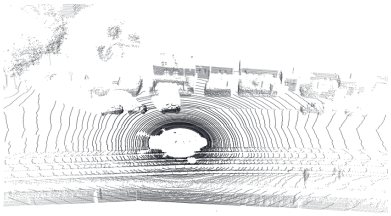
of stochastic processes. The detailed application of complex models towards robotic vision is expected to help and encourage people in entirely different application domains, such as collaborative filtering, search engine optimization, and audio processing. All these different applications do not always need dedicated algorithms, but do deserve and can exploit the same optimal general inference techniques from Bayesian nonparametrics.

1.2 Bayesian Nonparametrics

In robotic vision (computer vision and depth perception) traditionally custom-made algorithms have been developed for a given task. There are specific methods to detect corners (e.g., Förstner and Gülch, 1987; Harris and Stephens, 1988; Shi and Tomasi, 1994), to detect edges (e.g., Sobel, 1970; Canny, 1986), to detect features (e.g., Hough, 1962), and to describe features (e.g., Lowe, 1999; Dalal and Triggs, 2005; Bay et al., 2006).



(a) The PointNet40 dataset (Wu et al., 2015) has forty examples of common objects in the house of which the toilet is an example (Garcia-Garcia et al., 2016).



(b) The KITTI dataset (Geiger et al., 2012) has point clouds made by the Velodyne lidar. It is data that can be used on self-driving cars to become more aware of their surroundings. This example is from (Ioannou et al., 2012).

(c) Example of a point cloud generated by the Kinect depth sensor. This type of sensor can be used for applications indoors, e.g. autonomous cleaning robots. This particular example is a test on how a transparent sheet will show up on such a structured light sensor.

Figure 1.1: Examples of point clouds.

On the one hand, it is desirable that such sophisticated methods are generalizable to other application domains. On the other hand, it is important to take particular information about an application domain into account. The methods described in the previous paragraph are

limited to their specific task. An example of limited generalizability can be found in the Hough transform. The Hough transform can be used to detect lines, but the way inference is performed in the algorithm does limit its application to basic forms of object detection. An example of limited specificity can be found in linear regression. Linear regression assumes a linear relationship between input and output variables.

Both generalization and specificity are formalized by a Bayesian model. A Bayesian model is general, because the problem modeled by it can be solved with general inference methods. One of such general inference methods is a Markov-Chain Monte Carlo method. A Bayesian model is also specific in that it can incorporate application-specific know-how by the definition of priors. This power of Bayesian models can be seen in many disciplines, from robotic localization (Blanco et al., 2010), and dynamical systems (Dubbeldam et al., 2011), to forensic and legal arguments on evidence (Wieten et al., 2019).

Typical problems in robotic vision will be about the recognition of several objects, multiple shapes, or objects that have multiple parts. Models that represent such objects do not have knowledge about the number of such objects, shapes, or parts. To incorporate application-specific know-how on the number of objects it is possible to define a prior that assigns a probability to this quantity. The number of objects can even be potentially infinite. The Bayesian models that define a prior on the number of objects, shapes, or parts are called nonparametric Bayesian models. This means that in contrast with conventional methods such as k-means clustering (Forgy, 1965; Lloyd, 1982) the number of objects does not need to be predefined.

1.3 Problem Statement and Research Questions

Many methods in robotics - and in particular in robotic vision - have been developed in times where computational resources were limited. Then, highly optimized algorithms have been developed, leveraging pecularities of the application domain. Recent advances in Bayesian methods, both with respect to concept development, as well as computational efficient solution strategies, now open up new ways to solve old problems (Seeger, 2000; Murphy, 2012; Huszár and Duvenaud, 2012; Gal and Ghahramani, 2016; Mandt et al., 2017). However, extending only the old methods themselves would lead to ad hoc solution strategies that will miss benefits from potential optimal and more widely applicable algorithms.

This observation leads us to the formulation of our problem statement (PS).

> **PS:** *How can robotic vision problems effectively be generalized and their structure exploited in a wider Bayesian framework?*

The problem statement is rather general. In our research, we focus on robotic vision, in the form of point cloud recognition and depth perception. In particular, we look at objects, lines, line segments, and more complex shapes.

From this problem statement we derive three research questions (RQs).

RQ 1 How can we estimate the number of objects simultaneously with the fitting of these objects?

RQ 2 How can we optimize inference over both the number of objects and fitting of those objects in the robotic vision domain?

RQ 3 How can we recognize more general 3D objects?

1.4 Research Methodology

The research methodology advocated in the thesis follows the Bayesian methodology (cf. Savage, 1972; Jaynes, 2003). So, our research methodology consists of two phases. In the first phase a Bayesian model is defined. This model exists of (1) a definition of parameters and relations between these parameters, (2) a definition of the noise, and (3) the data. In the second phase, the Bayesian method dictates all remaining unknowns, from the number of parameters to the values of the parameters. To perform Bayesian inference efficiently new methods are required if the model is complex (as is the case with robotic vision).

The Bayesian methodology aims to establish the rationale for practical questions. The following two questions are clear examples.

- If we observe a single point in an image, can we expect it to be part of a line?
- If we have two lines and we live in a world with squares, what are we able to infer?

The two questions tap into our capabilities to define models that makes our prior knowledge explicit. Moreover, if we are able to quickly assign (1) points to segments, (2) segments to lines, and (3) objects to categories, we can enrich it with all corresponding group properties without the need to have them observed for this individual.

In robotic vision we take as an example the task of line detection. Both the Hough transform (Hough, 1962) and the RANSAC method (Bolles and Fischler, 1981) do detect lines, but they do not explicitly take noise into account. We can apply a Bayesian methodology to these tasks if we extend it to perform inference over a variable number of objects. This is called nonparametric Bayesian inference (Ghosal and Van der Vaart, 2017) . The Bayesian inference method is optimal in an information-theoretic sense (Zellner, 1988), Moreover, nonparametric Bayesian models are consistent in the sense that they approach the underlying true distribution (Wasserman, 1998).

Let us write this down informally in a straightforward manner. If we have formulated a problem in the Bayesian sense, there is no better way to solve it than using Bayesian inference. Given a Bayesian model, there is no need to search for another method to infer lines in a line detection task. No variant on Hough or RANSAC will outperform the Bayesian model. If someone would find a method that seems to outperform a Bayesian method it is either (1)

because the signal or noise has not been correctly modeled, or (2) because the method overfits with respect to the available data. Moreover, if approximations are used with respect to optimal Bayesian inference (either variational approximations or Markov-Chain Monte Carlo), there are theoretical guarantees on convergence (Andrieu et al., 2003). A Bayesian model is recommended also in those cases, compared to models that do not have such guarantees.

A well known problem with nonparametric Bayesian models is the curse of dimensionality. Compared to maximum likelihood methods or other non-probabilistic methods that do not take noise into account at all, the nonparametric Bayesian models require significant computational resources. Our research methodology first establishes the correct models, even if solving them seems computationally infeasible. Subsequently, our approach is to develop approximations using more efficient samplers while theoretical guarantees on convergence are preserved.

As described before, the Bayesian inference method is optimal in an information-theoretic sense (Zellner, 1988). In this thesis we take the optimality of the Bayesian method as a given. Our research methodology is to perform experiments to study the efficacy of inference methods for the proposed models. We will restrict the scope of the thesis to the (subjective) priors and noise models we propose for particular models. We will not study alternative noise models and priors.

1.5 Main Contribution

Our contribution to robotic vision can be subdivided into three parts that correspond with the three research questions.

The first part addresses the problem of inference about objects from a nonparametric Bayesian perspective. Contemporary methods in robotic vision do not allow for astute statements about their performance. In practice, this means that when using computer vision to detect cells under a microscope, someone cannot be confident about the number of detected cells. An autonomous cleaning robot in a supermarket cannot be confident about the aisle it is driving into. To be able to properly take into account models and uncertainty simultaneously, Bayesian models have found mainstream adoption. State-of-the-art Bayesian methods that reason about the number of objects alongside object models are a recent object of study (cf. Ferguson, 1973; Hjort, 1990; Lijoi and Prünster, 2010; Joho et al., 2011). The thesis applies such nonparametric Bayesian models towards the applications of robotic vision and depth perception. Models such as the infinite line model and the infinite line segment model are introduced.

The second part addresses the problem of high-dimensional data. To efficiently sample more complex geometric structures, new MCMC (Markov-Chain Monte Carlo, Section 2.2.4) methods are required. The thesis introduces such an MCMC sampler, namely a new Split-Merge sampler, and applies it to complex geometric structures.

The third part addresses more complex robotic vision problems, in the form of object recognition of point clouds in 3D. It combines nonparametric Bayesian inference with models from deep learning.

1.6 Organization of the Thesis

Chapter 1 (this chapter) introduces the problem of contemporary methods in computer vision and depth perception. Due to the fact that these methods are not optimal by construction, it is hard to articulate how they perform. The need for a Bayesian methodology is sketched briefly. The problem statement and three research questions are formulated. Moreover, the research methodology is described and the organization of the thesis is outlined.

Chapter 2 describes (1) probability theory using measure theory, (2) random measures known as random processes of which five are described as nonparametric Bayesian models, and (3) six inference methods that infer model parameters of such nonparametric Bayesian models given the data. It is followed by a discussion that indicates which parts will be most useful for chapters 3 and 4.

Chapter 3 examines a first nonparametric Bayesian model, i.e., the infinite line model. The infinite line model represents a countably infinite set of lines. Gibbs sampling is used to perform simultaneous inference over (1) the number of lines and (2) line parameter values such as slope and intercept.

Chapter 4 examines a second nonparametric Bayesian model, i.e., the infinite line segment model. The infinite line segment model represents a countably infinite set of line segments. A split-merge MCMC sampling method is used to perform simultaneous inference over (1) the number of line segments and (2) line segment parameter values such as slope, intercept, and segment size. Chapters 2 to 4 answer the first research question.

Chapter 5 investigates a new MCMC method, the Triadic Split-Merge sampler. It is tailored to clustering problems and accelerates inference of the models in Chapters 3 and 4. This chapter answers the second research question.

Chapter 6 examines more complex objects, like cubes and multiple cubes in a 3D space. It employs deep learning methods, in particular an autoencoder on point clouds, to perform inference on this type of data. This chapter answers the third research question.

Chapter 7 discusses the relevance of the developed models and inference methods. The answers to the research questions are discussed. Then the problem statement is answered and conclusions are formulated. Finally, recommendations are given and future research is envisaged.

CHAPTER 2

RELATED WORK

Contents In robotics depth sensors generate point clouds. The tasks of robotic object recognition, positioning, and navigation require models that represent such point clouds. It is unclear whether the current methods that perform inference over point clouds are appropriate for these tasks. The current models do not model uncertainty explicitly. This chapter presents models that can be used for point cloud modeling and that represent uncertainty. This (partially) answers research question RQ1. The chapter concludes with recommendations for the development of point cloud inference models. They will be implemented in a new model for line inference in Chapter 3 and line segment inference in Chapter 4.

Outline This chapter describes one particular nonparametric Bayesian model, the Dirichlet process. The process is presented in four ways: (1) as a measure, (2) with a sequential representation (using a restaurant metaphor), (3) as a prior for a mixture, and (4) exhibiting a stick-breaking distribution for cluster sizes (Section 2.1). Six inference methods are described: (1) inverse transform sampling, (2) rejection sampling, (3) approximate Bayesian computation, (4) Gibbs sampling, (5) Metropolis-Hastings sampling, and (6) Split-Merge Markov chain Monte Carlo sampling (Section 2.2). Inference about point clouds in the chapters to follow will use adaptations of the described models and inference methods for which some recommendations are given (Sections 2.3 and 2.4) .

2.1 Dirichlet Process

The Dirichlet process is presented as a measure (Section 2.1.1), is shown to have a sequential representation in the form of the Chinese restaurant process (Section 2.1.2), is used as a prior for a mixture (Section 2.1.3), and a stick-breaking representation (Section 2.1.4). We

compare the Dirichlet process, the Dirichlet process as prior for a mixture model, the Chinese restaurant process and the stick-breaking representation in Figure 2.1 using plate notation (cf. Fox et al., 2007).

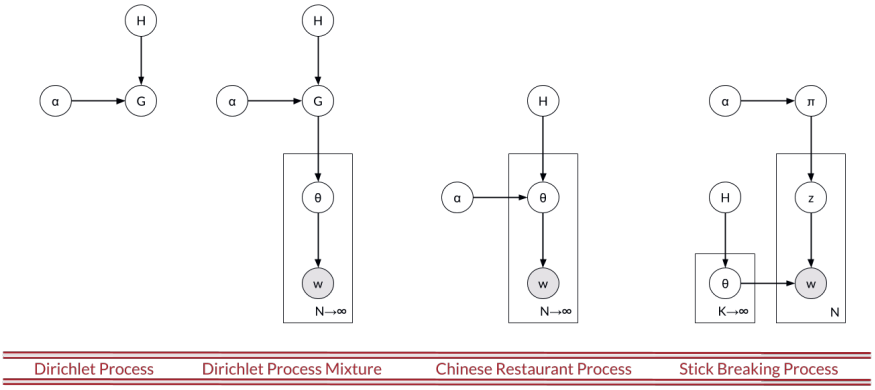


Figure 2.1: From left to right. (1) The Dirichlet process $G \sim DP(\alpha, H)$. (2) The Dirichlet mixture model with G as a prior: $\theta_i|G \sim G$. The parameters θ_i generate observations w_i through $p(w_i|\theta_i)$. (3) The Chinese restaurant process with G marginalized out. (4) The stick-breaking process with a distribution over partition sizes π and indicator variables z_i (cf. Fox et al., 2007).

2.1.1 Dirichlet Process as a Measure

The Dirichlet process (DP) is a distribution over distributions (Ferguson, 1973).

▼ **Definition 2.1** — ***Dirichlet process***

A **Dirichlet process** DP over a set S can be used to draw sample paths G:

$$G \sim DP(\alpha, H)$$

with α the dispersion parameter and H a measure on S and for which any measurable partition $\{B_0, \ldots, B_{n-1}\} \in S$ is drawn from a Dirichlet distribution:

$$(G(B_0), \ldots, G(B_{n-1})) \sim \text{Dirichlet}(\alpha H(B_0), \ldots, \alpha H(B_{n-1}))$$

The Lévy intensity of the Dirichlet process is complicated, because it is a so-called normalized process, see Regazzini et al. (2003).

2.1.2 Chinese Restaurant Process

De Finetti's theorem (Definition A.45) can be used to establish the existence of an infinitely exchangeable sequence. In the particular case of the Dirichlet process the sequence is an exchangeable *distribution over partitions* and is called the Chinese restaurant process (Aldous, 1985).

▼ **Definition 2.2 — *Chinese restaurant process***

A **Chinese restaurant process** is a sequential process that is an exchangeable distribution over partitions:

$$p(z_i = k | z_0, \ldots, z_{i-1}) = \begin{cases} \frac{n_k}{\alpha+i} & \text{if } k \leq K_+ \\ \frac{\alpha}{\alpha+i} & \text{if } k > K_+ \end{cases} \tag{2.1}$$

The conditional probability of a cluster assignment z_i (for sample θ_i) given the cluster assignments $z_0, \ldots, z_{i-1}$ is proportional to the number of samples n_k assigned to an existing cluster k, or proportional to α for a new cluster. The existing clusters are denoted by $k \leq K_+$ with K_+ the number of clusters.

Instead of sampling cluster assignments, we can generate the samples $\{\theta\}$ from the base distribution H directly:

$$\theta_{i+1} | \theta_1, \ldots, \theta_i \sim \begin{cases} \delta_{\theta_1}, & \text{with probability } 1/(\alpha+i), \\ \vdots & \vdots \\ \delta_{\theta_i}, & \text{with probability } 1/(\alpha+i), \\ H, & \text{with probability } \alpha/(\alpha+i). \end{cases} \tag{2.2}$$

The random measure G is marginalized out (see also the plate notation in Figure 2.1). This particular representation is especially convenient for Gibbs sampling about which we will learn in Section 2.2.4. This can also written equivalently as:

$$\theta_{i+1} \mid \theta_1, \ldots, \theta_i \sim \frac{1}{\alpha+i} \left(\alpha H + \sum_{j=1}^{i} \delta_{\theta_j} \right). \tag{2.3}$$

The Chinese restaurant process is visualized in Figure 2.2. In a restaurant with K^+ tables there are i customers seated, $\{\theta_0, \ldots, \theta_{i-1}\}$. The probability that a new customer, θ_i, will be seated at an existing table depends on the number of customers, n_k, already at the table. The probability that the customer is assigned a new table is proportional to the dispersion parameter α.

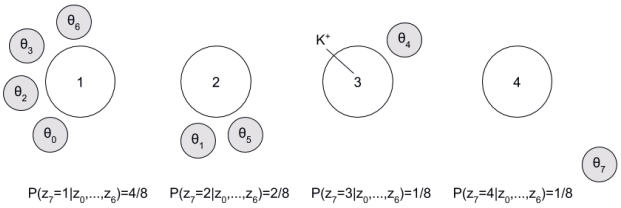


Figure 2.2: The Chinese restaurant process with i customers, $\theta_0, \dots \theta_{i-1}$, already sitting down. A new customer θ_7 arrives and gets assigned. The assignment variable is z_7. The customer gets assigned an existing table $\{1, 2, 3\}$ with a probability proportional to the number of customers n_k sitting at that table: $n_k/(\alpha + i)$. The customer gets assigned a new, empty table $\{4\}$ with probability $\alpha/(\alpha + i)$. In the visualized Chinese restaurant process the dispersion factor α is equal to 1. Thus for the first table $n_k = 4$ (four customers) and the probability the customer will be assigned this table: $P(z_7 = 1|z_0, \dots, z_6) = n_k/(\alpha + i) = 4/(1 + 7) = 4/8$.

2.1.3 Dirichlet Process Mixture

The Dirichlet process can be used as a prior for an infinite mixture model (Definition A.42). This is visualized in (Figure 2.3).

	θ0	θ1	θ2	θ3	θ4	θ5	θ6	θ7	θ8	θ9	sum
data 0	0	0	1	0	0	0	0	0	0	...	1
data 1	1	0	0	0	0	0	0	0	0	...	1
data 2	1	0	0	0	0	0	0	0	0	...	1
data 3	0	1	0	0	0	0	0	0	0	...	1
data 4	1	0	0	0	0	0	0	0	0	...	1
data 5	0	1	0	0	0	0	0	0	0	...	1
data 6	0	0	0	0	1	0	0	0	0	...	1

Figure 2.3: An infinite matrix representation of an infinite mixture model. At the horizontal axis we see the latent variables. There are potentially an infinite number of latent variables. At the vertical axis we see the data items. A data item is assigned to a single latent variable. The rows sum up to one.

The representation as a distribution over infinite matrices can be found in the literature (Ghahramani and Griffiths, 2005). We observe that a column has multiple nonzero values. A parameter is not unique, it can be sampled multiple times. The Dirichlet process lends itself as a prior for a mixture. First, the Dirichlet process exhibits this property of being *almost surely discrete* (Ferguson, 1973; Blackwell, 1973; Basu and Tiwari, 1982). If the base measure H is atomless then with probability one the ith value θ_i drawn from the Chinese restaurant process is *distinct* from the values $\theta_1, \dots, \theta_{i-1}$ if it is drawn from H. Informally,

this can be seen as defining a prior over more than one column. Second, for a finite dispersion parameter α, the probability of sampling *identical* values (from $\sum_j \delta_{\theta_j}$) is nonzero. Informally, this can be seen as a prior that allows for multiple nonzero values in a column.

2.1.4 Stick-breaking Representation of the Dirichlet process

The *stick-breaking representation* by Freedman and Diaconis (1983), also known as the residual allocation model (Sawyer and Hartl, 1985; Hoppe, 1986), represents a random process through breaking sticks.[1] A stick of unit length is broken in subsequently smaller pieces with each piece broken off put aside. Here we introduce the stick-breaking representation according to the exposition in Ishwaran and James (2001).

▼ **Definition 2.3** — ***stick-breaking***

An infinite sequence of random variables $\pi = \{\pi_0, \pi_1, \ldots\}$ has a **stick-breaking representation** with parameters α and β denoted by $\pi \sim GEM(\alpha, \beta)$ with

$$w_k \overset{iid}{\sim} Beta(1-\beta, \alpha + k\beta), \qquad k = 1, \ldots, \tag{2.4}$$

$$\pi_k = w_k \prod_{i=1}^{k-1} (1 - w_i). \tag{2.5}$$

The random variables w_k are drawn iid from a Beta distribution. The stick lengths are represented by π_k and when $k \to \infty$ the infinite dimensional prior is well defined under some mild conditions (Ishwaran and James, 2001). An illustration of the stick-breaking presentation can be found in Appendix A.8. The letters *GEM* stand for Griffiths, Engen, and McCloskey as proposed - and considered appropriate as acronym because of its beautiful properties - by Ewens (1990).

▼ **Definition 2.4** — ***stick-breaking representation of the Dirichlet process***

The **stick-breaking representation** of the Dirichlet process states that if

$$\pi_k \sim GEM(\alpha, 0), \qquad k = 1, \ldots, \infty, \tag{2.6}$$

$$\theta_k \overset{iid}{\sim} H, \qquad k = 1, \ldots, \infty, \tag{2.7}$$

$$G = \sum_{k=1}^{\infty} \pi_k \delta_{\theta_k}, \tag{2.8}$$

then $G \sim DP(\alpha, H)$.

[1] Not using the stick analogy but preceding the work by Freedman and Diaconis (1983) and using the same multiplicative representation is the study on how one pound of gold dust can be distributed among a countably infinite number of beggars by Halmos (1944).

The weights π_k are sampled from the stick-breaking process $GEM(\alpha, 0)$. The parameter values θ_k are independently sampled from the base measure H. To sample from the Dirichlet process we sample the parameters θ_k with the weights π_k.

The stick-breaking process can be used as a prior for a mixture. Sample the cluster assignments z_i according to the mixing proportions π and generate the observations from the cluster parameters:

$$z_i \sim Mult(\pi), \tag{2.9}$$
$$w_i \sim F(\theta_{z_i}) \tag{2.10}$$

Here $\theta_k = \theta_{z_i}$ for observation w_i with index i and cluster assignment k: $z_i = k$.

2.2 Inference

There will be six inference methods described, all sampling methods. Inverse transform sampling is described in Section 2.2.1. Rejection sampling in Section 2.2.2. Approximate Bayesian computation in Section 2.2.3. Gibbs sampling in Section 2.2.4. Metropolis-Hastings sampling in Section 2.2.5. Split-Merge MCMC sampling in Section 2.2.6. We report for every inference method the corresponding algorithm in pseudo code. We compare the inference methods in Section 2.2.7.

2.2.1 Inverse Transform Sampling

Let $p(x)$ be a discrete probability distribution with two possible values $x = f$ and $x = g$. The probability distribution sums up to one: $\sum_v p(x = v) = 1$. Sample from a uniform distribution $u \sim U(0, 1)$. If $u < p(x = f)$ generate f, else generate g. This procedure samples f with probability $p(x = f)$ and g with probability $p(x = g)$. It can be readily generalized to more than two values by making use of the cumulative distribution function. In Algorithm 1 we sample from $f(x)$ by making use of the inverse cumulative distribution.

Algorithm 1 Inverse transform sampling for $f(x)$

procedure INVERSE TRANSFORM SAMPLING($f(x)$) ▷ Distribution to sample from.
  $F(x) = \int_{-\infty}^{x} f(s)ds$ ▷ Create cumulative distribution function $F(x)$.
  $X = \varnothing$
  for $t = 1 \rightarrow T$ **do**
    $u \sim U(0, 1)$ ▷ Sample from uniform distribution.
    $x \sim F^{-1}(u)$ ▷ Sample x from (the inverse) $F^{-1}(x)$.
    $X = X \cup x$
  end for
  return X ▷ X will have the distribution of $f(x)$.
end procedure

The term "inverse" stems from the fact that we return x (or $f(x)$) given u. Inverse transform sampling is a common component in sampling methods. When one of the steps in an algorithm samples from a uniform distribution, it is often an inverse transform sampling step.

2.2.2 Rejection Sampling

Let $f(x)$ be a complicated function from which it is hard to take samples. Let $g(x)$ be a simple function that is easy to sample from. Then we can sample from $f(x)$ by making sure $Mg(x) \geq f(x)$. The function $Mg(x)$ is an *envelope* function. This sampling method S generates the sample set X using $f(x)$ and $g(x)$.

$$X = S(f(x), g(x), M, T) \tag{2.11}$$

The rejection sampling method (Halperin and Burrows, 1960) for $f(x)$ is described in Algorithm 2.

Algorithm 2 Rejection sampling for $f(x)$

procedure REJECTION SAMPLING($f(x), g(x), M, T$) ▷ Target and proposal distribution and scalars M and T.
  $X = \emptyset$
  for $t = 1 \rightarrow T$ **do**
    $x^t \sim g(x)$ ▷ Generate x^t from $g(x)$
    $u \sim U(0, 1)$ ▷ Inverse transform sampling
    $p_0 = f(x)/(Mg(x))$
    if $u < p_0$ **then**
      $X = X \cup x^t$ ▷ Accept
    end if
  end for
  return X ▷ S will have the distribution of $f(x)$
end procedure

We can use rejection sampling to *sample* from the *posterior* $f(\theta|x)$ given that (1) we know the *exact* likelihood function and (2) we can *sample* from the prior. Here, we know that we can sample from the posterior by sampling from $p(\theta)p(x|\theta)$. Moreover, we know that the prior $p(\theta)$ necessarily has to be larger than $p(\theta)p(x|\theta)$ for any observation, because $p(x|\theta)$ is a probability density function, hence for each x and θ it is smaller than one. Therefore we can use rejection sampling with $Mg(x) \geq f(x)$ with $M = 1$, $p(\theta) = g(x)$ and $p(x|\theta = f(x)$.

We introduce the following notation. We make explicit that we need $p(x|\theta)$ for each combination of observations and parameters, but that we only need to *sample*[2] from the prior, which we indicate by a tilde, $\sim p(\theta)$.

[2]The notation $\sim p(\theta)$ means that we can sample from $p(\theta)$ but that we do not have access to a closed-form probability density function.

$$\Theta = S(\sim p(\theta), p(x|\theta), x, T) \tag{2.12}$$

Algorithm 3 Rejection sampling for $f(\theta|x)$

procedure REJECTION SAMPLING($p(\theta), p(x|\theta), x$) ▷ Requires prior, likelihood and observations.
  $\Theta = \varnothing$
  for $t = 1 \rightarrow T$ **do**
    $\theta^t \sim p(\theta)$ ▷ Generate θ^t from prior
    $u \sim U(0, 1)$ ▷ Inverse transform sampling
    $p_0 = p(x|\theta)$
    if $u < p_0$ **then**
      $\Theta = \Theta \cup \theta^t$ ▷ Accept
    end if
  end for
  return Θ ▷ Θ will have the distribution of $f(\theta|x)$
end procedure

In Algorithm 3 the envelope distribution $p(\theta)$ and the target distribution $p(\theta)p(x|\theta)$, cancel in such way that only $p(x|\theta)$ remains.

Most examples illustrate rejection sampling by estimating the area of a circle, but let us visualize the method in the context of sampling (Figure 2.4).

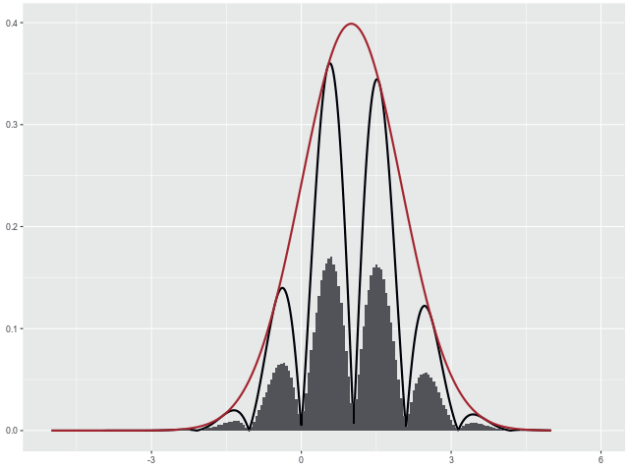


Figure 2.4: A Gaussian is placed over the complex target probability density function. Subsequently the samples that fall in between these two 'envelopes' are rejected. This results in a sampling scheme that follows exactly the more complicated probability density function. Note that if the function is scaled by a factor, the sampling scheme stays the same. Such a scaling factor is only important if we want, for example, to know the area under the graph.

2.2.3 Approximate Bayesian Computation

In approximate Bayesian computation (ABC) (Rubin, 1984) the likelihood function does not need to be calculated[3] (Sisson and Fan, 2011). In contrast, it is assumed that there is a model available that simulates observations given the (searched for) parameters. In ABC for each configuration of parameters a set of observations is generated.

$$\Theta = S(\sim p(\theta), X, \sim M(\theta), d(X^t, X), \epsilon, T) \tag{2.13}$$

Approximate Bayesian computation uses many tuning parameters. Its most salient characteristic though, is that it generates pseudo-observations through $M(\theta)$ (see Algorithm 4).

Algorithm 4 Approximate Bayesian computation

procedure APPROXIMATE BAYESIAN COMPUTATION($p(\theta), X, M, d, \epsilon$) ▷ Requires prior, observations, model, distance function, and threshold.
  $\Theta = \varnothing$
  for $t = 1 \rightarrow T$ **do**
    $\theta^t \sim p(\theta)$ ▷ Generate θ from prior
    $X^t \sim M(\theta)$ ▷ Simulate observations X^t from model M
    $\rho = d(X^t, X)$ ▷ Calculate distance between simulated and actual observations
    if $\rho \leq \epsilon$ **then**
      $\Theta = \Theta \cup \theta^t$ ▷ Accept θ^t if distance falls under threshold ϵ.
    end if
  end for
  return Θ ▷ Θ will have the distribution of $f(\theta|X)$
end procedure

The term Bayesian reflects the fact that a prior is involved. The weight of this prior can be manipulated by the threshold ϵ. If this threshold is set very low, the prior plays no role and only observations are taken into account. If ϵ is set extremely high, all θ coming from the prior will be accepted, and the actual observations are not used in the process. There are several disadvantages to approximate Bayesian computation.

- A set of simulated observations has to be compared with the actual observations. This becomes unwieldly if there are many observations.
- It is possible to use summary statistics rather than the observations themselves. If these are sufficient statistics there will be no information loss. If not, there will be information loss in practice.
- The distance function suffers from the curse of dimensionality. In the case that the dimensionality of the individual observations becomes high, or the number of parameters becomes large, it gets increasingly difficult to come up with a distance function which is efficient and accurate at the same time.

[3] ABC is also called likelihood-free computation

2.2.4 Gibbs Sampling

Gibbs sampling (Geman and Geman, 1984) is similar to the *coordinate descent* optimization algorithm (Wright, 2015). In coordinate descent a local minimum of a function is found by iteratively performing a line search along one coordinate direction at a time. Gibbs sampling generates posterior samples by sampling all conditional distributions $p(\theta_i|\theta_{-i})$ iteratively.

$$\Theta = S(\sim p(\theta_i|\theta_{-i}), \sim p(\theta), B, T). \tag{2.14}$$

Sampling iteratively will give us the joint distribution $p(\theta_i, \theta_{-i})$. We can also sample conditional on observations X to obtain $p(\theta|X)$. We will describe Gibbs sampling in that context:

$$\Theta = S(X, \sim p(\theta_i|\theta_{-i}, X), \sim p(\theta), B, T). \tag{2.15}$$

We visualize Gibbs sampling to obtain conditional distributions as in Eq. 2.15 in Figure 2.5.

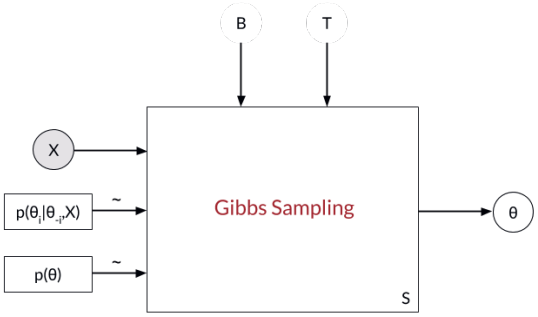


Figure 2.5: A block diagram to visualize Gibbs sampling. This representation makes the inputs and outputs of the Gibbs sampling method explicit. The inputs are observations, X, a conditional distribution to sample from $p(\theta_i, \theta_{-i}, X)$, a prior distribution to sample prior values $p(\theta)$ and a burn-in period, B, and total number of steps, T. The $\sim$ symbol at an arrow indicates sampling rather than access to a probability density function.

The Gibbs algorithm is given in Algorithm 5. Some explanation on the notation is as follows. Here θ can be understood as the parameters of a multivariate probability distribution. The individual parameters are denoted by θ_i. The set of all parameters except for parameter i is denoted by θ_{-i}.

$$\theta_i^t \sim p(\theta_i^t|\theta_{-i}^{t-1}, X). \tag{2.16}$$

The sampling of θ_i^t is here represented in a simplified manner. The actual[4] implementation samples a new parameter value using a combination of already updated, θ^t, and old values, θ^{t-1}:

[4] Also this is simplified. There is a choice to shuffle θ_i^t at each time step or to keep the same order $\theta_1^t, \ldots, \theta_k^t$.

$$\theta_i^t \sim p(\theta_i^t | \theta_1^t, \ldots, \theta_{i-1}^t, \theta_{i+1}^{t-1}, \ldots, \theta_k^{t-1}, X). \tag{2.17}$$

If we sample a parameter we write θ^t with t the iteration or sampling round. The array of parameter samples has capital letter Θ.

Algorithm 5 Gibbs sampling

procedure GIBBS SAMPLING($p(\theta_i|\theta_{-i}, X), p(\theta), X, B$) ▷ Requires conditional and prior distributions (to sample from), observations, and burn-in period.
  $\Theta = [\,]$
  $\theta^0 \sim p(\theta)$ ▷ Set parameters to some initial value.
  for $t = 1 \rightarrow T$ **do**
    for $i = 1 \rightarrow k$ **do**
      $\theta_i^t \sim p(\theta_i^t | \theta_{-i}^{t-1}, X)$ ▷ Generate θ_i^t from Eq. 2.16, equivalently, Eq. 2.17.
    end for
    $\Theta[t] = \theta^t$
  end for
  $\Theta_{B:T} = \Theta[B:T]$ ▷ Get samples from burn-in B to end of run T.
  $\Theta \sim \Theta_{B:T}$ ▷ Sample Θ from correlated $\Theta_{B:T}$.
  return Θ
end procedure

Gibbs samples are Markovian. This means that the conditional probability only takes into account values at the previous time step $t-1$. When running the Gibbs sampling algorithm long enough, it will visit all possible states eventually. The Markovian property has an undesired side effect. It makes subsequent steps correlated. Hence, when finally extracting the parameter probabilities, it is important to skip multiple steps to remove the temporal correlations. It is also important to run the algorithm for a while after its start. In that case it does not suffer from a bad choice of initial parameter values. Disregarding the first samples is called burn-in. In words, Gibbs sampling works by having the algorithm spend time in parts of the space proportionally to the probability of getting into that part of the space.

2.2.5 Metropolis-Hastings Sampling

Metropolis-Hastings (Metropolis et al., 1953; Hastings, 1970) is one of the most well-known Markov chain Monte Carlo (MCMC) algorithms. An MCMC algorithm uses a Markov chain (see Gibbs sampling, Section 2.2.4) and combines this with a stochastic (Monte Carlo) component. This sampling method can be used for high-dimensional distributions. Metropolis-Hastings calculates an acceptance factor α which takes into account if a step should be taken according to a predefined proposal distribution. In case this step is not accepted, the current sample is resampled (see Algorithm 6).

$$\Theta = S(X, \theta^0, Q(\theta^t | \theta^{t-1}), f(\theta; X)) \tag{2.18}$$

Metropolis-Hastings can be described without reference to observations just as with Gibbs sampling. However, Eq. 2.18 and Algorithm 6 is described including the observations (which seems appropriate within the Bayesian context). We observe that in contrast to Gibbs sampling, we need to be able to calculate $Q(\theta^t|\theta^{t-1})$, rather than only be able to sample from $p(\theta_i^t|\theta_{-i}^{t-1})$.

Algorithm 6 Metropolis-Hastings sampling

procedure METROPOLIS-HASTINGS SAMPLING(θ^0, X, Q, f) ▷ Requires initial parameters, observations, proposal distribution, and function proportional to desired distribution
  $\Theta = [\,]$
  for $t = 1 \rightarrow T$ **do**
    $\theta^t \sim Q(\cdot|\theta^{t-1})$ ▷ Sample from proposal distribution Q
    $\alpha = \frac{f(\theta^t;X)Q(\theta^t|\theta^{t-1})}{f(\theta^{t-1};X)Q(\theta^{t-1}|\theta^t)}$ ▷ Calculate acceptance
    $u \sim U(0,1)$ ▷ Inverse transform sampling
    if $\alpha < u$ **then**
      $\theta^t = \theta^{t-1}$ ▷ Reuse previous sample (note, different from rejection)
    end if
    $\Theta[t] = \theta^t$ ▷ Always add θ^t (accepted or repeated).
  end for
  return Θ ▷ Θ will be samples from the distribution $f(\theta;X)$
end procedure

A particular choice of a Metropolis-Hastings step is that of a proposal distribution that does not depend on the state of the chain. This is already suggested by Hastings and is called the independence sampler (Hastings, 1970).

2.2.6 Split-Merge MCMC Sampling

When we study the model as described in Section 2.1.2 we see that samples can be modelled as being clustered. The discussed sampling methods do not assume such kind of structure in the model. This means that in hierarchical models sampling either occurs through updating the to-be-estimated quantities by iterating over every single observation or over every single parameter. This has a disadvantage, an algorithm in which a new cluster is formed by sampling over individual data points can be quite slow. It would be more efficient if multiple data points can be assigned at once to a new parameter. This corresponds to methods in which we can split or merge clusters of data points.

Split-merge samplers are such methods that can update cluster assignments for multiple observations at once. These samples adjust the acceptance method in the Metropolis-Hastings algorithm. Split-Merge sampling is described in Algorithm 7.

Algorithm 7 Split-Merge MCMC sampling

procedure SPLIT-MERGE MCMC SAMPLING(θ^0, X, Q, f) ▷ Requires initial parameters, observations, proposal distribution, and function proportional to desired distribution
  $\Theta = [\,]$
  for $t = 1 \rightarrow T$ **do**
    $i \sim U(\{0, \ldots, N-1\})$ ▷ Sample observation i discretely.
    $j \sim U(\{0, \ldots, N-1\} \setminus i)$ ▷ Sample observation j discretely with $j \neq i$.
    if $c_i == c_j$ **then** ▷ Let us consider a split of this cluster ($c_i = c_j$).
      $c_{old} = c_i$
      $\theta^t_{c_{new}} \sim Q(\theta^t | \theta^{t-1})$ ▷ Sample from proposal distribution Q.
      for $k \in c_{old}$ **do**
        $c_k \sim Cat(c_{old}, c_{new})$ ▷ Assign to new cluster categorically.
      end for
    else ▷ Let us consider a merge of these clusters ($c_i \neq c_j$).
      $c_{merge} = c_i$
      for $k \in c_j$ **do**
        $c_k = c_{merge}$ ▷ Assign all observations to the first cluster.
      end for
    end if
    $\alpha = \frac{f(\theta^{t+1}, X^{t+1}) Q(\theta^{t+1} | \theta^t)}{f(\theta^t, X^t) Q(\theta^t | \theta^{t+1})}$ ▷ Calculate acceptance.
    $u \sim U(0, 1)$ ▷ Inverse transform sampling.
    if $\alpha < u$ **then**
      $\theta^t = \theta^{t-1}$ ▷ Set current sample to previous sample.
    end if
    $\Theta[t] = \theta^t$ ▷ Always add θ^t (accepted or repeated).
  end for
  return Θ ▷ Θ will be samples from the distribution $f(\theta | x)$.
end procedure

The exact acceptance probability depends on the model. For the mixture model with a Dirichlet Process as prior, its performance is further improved by adjusting the assignment process from random to observation-supported by introducing intermediate restricted Gibbs sampling steps (Jain and Neal, 2004, 2007). Similarly, there are other variants that incorporate data fit to the splitting step. Labels can for example be calculated sequentially (Dahl, 2003) or methods can be used that postulate subcluster structure within clusters to optimize inference over split and merge sets (Chang and Fisher III, 2013).

2.2.7 Comparison of the Six Inference Methods

In robotic vision the type of data we are obtaining from depth sensors are point clouds. To perform inference over objects made out of point clouds, clustering algorithms benefit from two sampling strategies. If conjugate probability densities are used, Gibbs sampling, or collapsed Gibbs sampling can be used (Section 2.2.4). If the model consists of a nonconjugate

prior and likelihood function, Metropolis-Hastings sampling can be used (Section 2.2.5). If the model becomes more complicated split-merge sampling might accelerate the inference process (Section 2.2.6).

2.3 Chapter Conclusions

In this chapter we introduced the Dirichlet process in Section 2.1. The Dirichlet process is described as a measure (Section 2.1.1), represented as a Chinese restaurant process (Section 2.1.2), described as inducing a mixture (Section 2.1.3), and constructed through stick-breaking (Section 2.1.4). The four expositions serve different purposes. The Dirichlet process as a random measure puts on firm theoretic grounds. The Chinese restaurant process representation lends itself to Gibbs-like sampling methods. The representation as a prior over infinite matrices shows how the Dirichlet process can be used as a prior for a mixture model. The stick-breaking construction shows how the cluster sizes are distributed rather than the individual samples.

In Section 2.2 six inference methods are described. Inverse transform sampling, rejection sampling, approximate Bayesian computation, Gibbs sampling, Metropolis-Hastings, and Split-Merge Markov chain Monte Carlo. These methods use different types of information. Inverse transform sampling requires the cumulative distribution function (or more specific its inverse). Rejection sampling requires an envelope function that approximates the actual distribution sufficiently well, such that not many samples will be rejected. If the likelihood function is known, and samples can be drawn from the prior, the likelihood function can be used to sample from the posterior. Approximate Bayesian computation does not require a likelihood function, but manipulates the influence of the prior by a threshold and introduces a distance function to define if simulated and actual observations are close. Gibbs sampling does not use such artificial parameters, but requires a prior and likelihood that are conjugate. It can be used for models where this is the case. Metropolis-Hastings sampling is an MCMC sampling method that can be used for nonconjugate models. Its convergence rate depends on the quality of a proposal distribution. To accelerate convergence, split-merge MCMC sampling performs inference not just over individual data points, but over sets of data points (e.g., splitting a cluster or merging two clusters of data).

2.4 A Coda

Below we provide a critical confrontation of our conclusions in relation with existing mathematical theories. First of all, we would like to stress that this research work is of a technical and applied nature as the title implies: *Nonparametric Bayesian Methods in Robotic Vision*. Below we discuss some theoretical research directions that we have neither explored in this chapter, nor in the following chapters.

Robustness is an important aspect of Bayesian methods (Ghosal and Van der Vaart, 2017). The choice of prior should not influence the posterior distribution "too much". This is difficult

to study on its own, for which reason *posterior consistency* is studied instead. It loosely means that the posterior probability is eventually concentrated in a small neighborhood of the "actual value" of the parameter. The study of asymptotic properties, such as posterior consistency, is more complex in the nonparametric case. An infinite amount of data might not overcome the prior. In this thesis we rely on the use of a Dirichlet process prior for robotic (depth) vision tasks. There will neither be an analysis on the consistency of the prior, nor on the rate of contraction as advocated for by Ghosal and Van der Vaart (2017).

One of the reasons to introduce the class of Dirichlet priors is to ensure consistency in the nonparametric Bayesian setting (Diaconis and Freedman, 1986). However, it is possible to use them in such a way that they lead to inconsistent estimates. The authors describe a Dirichlet prior with a Cauchy distribution as base measure crafted in such way that it does not converge in the asymptotic case to the underlying true distribution that consists of two point masses positioned at locations $-a$ and a with respect to the origin. This means that the posterior might converge to the wrong value or that the estimates will oscillate in the asymptotic case. We will not explore possible inconsistent behavior of the base measures that we introduce in the future chapters.

The robotic vision applications do have practical constraints which are apparent from the domain. For example, in Chapter 4 the Pareto prior will not "wash out" if chosen too large. The method will not find segments if they are much smaller than the experimenter expects. There are also degenerate cases such as single outliers or line segments aligned head-to-tail forming a larger, single line segment. Facing the existing literature we observe the following.

- Our practical application will be clustering. We assign points to geometric objects. Lack in convergence towards object parameters does not imply an asymptotic error in the assignment problem.
- In practical situations, there are degenerate cases, (1) outliers, (2) accidentally aligned line segments, and (3) other types of errors such as perceptual resolution, that would require our attention in improving our clustering results.
- Theoretically, even if we do not have the "correct model", we might still come close to it in a precise sense (see Ghosal and Van der Vaart (2017) on Kullback-Leibler projections).

In the next chapters, we will refrain from going into theory. We will only apply existing theory and implement methods to perform robotic (depth) vision in general and perform inference on point clouds in particular.

CHAPTER 3

NONPARAMETRIC BAYESIAN LINE DETECTION

Contents

In this chapter the nonparametric Bayesian models from the literature (Chapter 2) are applied to perform inference over point clouds. The point cloud under study will be a point cloud distributed over lines in a two-dimensional space. Traditionally, RANSAC and the Hough transform have been used to perform inference over such lines. We use a nonparametric Bayesian model to perform inference over a countably infinite number of lines. Given a prior with respect to the noise and the distribution of points over the lines, Bayesian inference describes the optimal procedure to perform line fitting.

Published in

A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Nonparametric Bayesian Line Detection. *International Conference on Pattern Recognition and Methods*, ICPRAM 2016, Rome, Italy, February 24-26, 2016. Best paper award in theory and methods track.
A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Fundamentals of Nonparametric Bayesian Line Detection. Springer, 2017.

Outline

The infinite line model describes a collection of lines with a Dirichlet process as prior (Section 3.2). Inference in the infinite line model is performed through Gibbs sampling (Section 3.3). As is known, Gibbs sampling over *parameters* converges slowly, however it can be accelerated through sampling over *clusters* (Section 3.4). The results by the inference method are assessed using clustering performance measures (Section 3.5). The chapter summarizes the findings (Section 3.6) and introduces extensions which will be handled in the next chapters.

3.1 Four Problems with Line Detection

In computer vision and particularly in robotics, traditionally the task of line detection has been performed through sophisticated, but ad-hoc methods. Here we mention two examples of such methods, RANSAC and the Hough transform. RANSAC (Bolles and Fischler, 1981) is a method that iteratively tests a hypothesis. A line is fitted through a subset of points. Then other points that are in consensus with this line (according to a certain loss function) are added to the subset. This procedure is repeated till a certain performance level is obtained. The Hough transform (Hough, 1962) is a deterministic approach which maps points in the image space to curves in the so-called Hough space of slopes and intercepts. A line is extracted by getting the maximum in the Hough space.

There are four main problems with these methods. First, the extension of RANSAC or Hough to the detection of multiple lines is nontrivial (Chen et al., 2001; Zhang and Kŏsecká, 2007; Gallo et al., 2011). Second, the noise level is hard-coded into model parameters and it is not possible to incorporate knowledge about the nature of the noise. Third, it is hard to extend the model to hierarchical forms, for example, to lines that form more complicated structures such as squares or volumetric forms. Fourth, there are no results known with respect to any form of optimality of the mentioned algorithms.

In this chapter we postulate a method to perform inference over the number of lines and over the fitting of points on that line using the nonparametric Bayesian methods from chapter 2. The method aims at overcoming the four main problems mentioned above.

3.2 Infinite Line Model

The application we would like to address in this chapter is that of the detection of multiple lines.

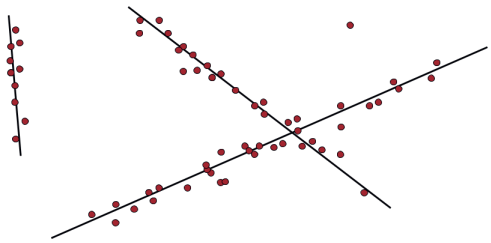


Figure 3.1: A mixture of lines. There are n points in 2D space, each point generated from a line with parameters θ_k. The number of lines k is not known beforehand.

The Dirichlet process has been previously described as prior for a mixture distribution (in Figure 2.2, see Section 2.1). It will be used in our model as a prior for the *distribution of points* over a *countably infinite set of lines*. From now on we will refer to this model as the infinite line model (ILM).

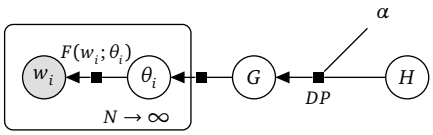


Figure 3.2: For the infinite line model we will use a Dirichlet process mixture. This visualization uses a combination of a so-called factor graph with plate notation. Compare the visualization with Eq. 3.1. The observations w_i are generated by line parameters θ_i (which do not have to be unique). The parameters θ_i are distributed according to G which is sampled from a Dirichlet process with base measure H and dispersion parameter α.

The Dirichlet process mixture nature of the infinite line model is visualized in Figure 3.2 using plate notation (see Appendix A.5). The line parameters θ_i with $i = 1, 2, \ldots$ are sampled from a distribution G. This distribution is sampled from the base distribution H with dispersion parameter α. The representation should not be seen as suggesting a form for factorization. The Dirichlet process as a prior for a mixture model we summarize as follows (compare with Figure 3.2):

$$\begin{aligned} G &\sim DP(\alpha, H), \\ \theta_i \mid G &\overset{iid}{\sim} G, \\ w_i \mid \theta_i &\overset{iid}{\sim} F(w_i; \theta_i). \end{aligned} \tag{3.1}$$

Eq. 3.1 represents a mixture model due to the fact that parameters $\{\theta_i, \theta_j\}$ can be identical for $j \neq i$. In that case y_i and y_j are considered to belong to the same cluster characterized by parameter $\theta_i = \theta_j$ (see Section 2.1.3). Here $X \sim S$ means that X has the distribution S. Independence properties, such as observation y_i given parameter θ_i being independent of other observations, are written down explicitly in Eq. 3.1. They might be silently assumed further on.

We will consider models where G is integrated out, the details of which, will follow in this chapter. We also introduce hyperparameters to the base distribution H.

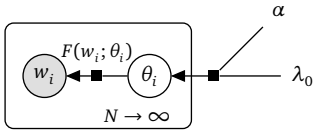


Figure 3.3: The Dirichlet process mixture with the realizations G integrated out and with a hyperparameter λ_0 for the base distribution H.

We will later on see that the base measure H will be the so-called Normal-Inverse-Gamma (NIG) distribution with hyperparameters λ_0. We will also create more detailed figures to emphasize particular aspects of the model.

As mentioned before the parameters θ_i and θ_j can be identical for $i \neq j$. We can equivalently express the Dirichlet process mixture using only k clusters and running the index k over unique clusters.

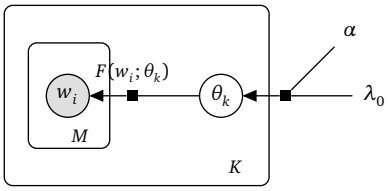


Figure 3.4: The Dirichlet process mixture over clusters with the index, k, ranging over the number of clusters. Here there are M_k observations per cluster k.

In this chapter, both θ_i and θ_k will be used. If the parameter is written as θ_i the index i runs over as many (non-unique) parameters as there are observations. If the parameter is written as θ_k the index k runs over (unique) lines.

In Section 3.2.1 it is described how θ_i is sampled from H and α. In Section 3.2.2 it is described how w_i is sampled from θ_i. In Section 3.2.3 the prior $H(\lambda_0)$ for θ_i is described. In Section 3.2.4 it is described how the hyperparameters λ_0 can be updated given the data w_i to define the posterior predictive for the line parameters, θ_i.

3.2.1 Posterior Predictive for a Line given Other Lines

The Dirichlet process (DP) is described in Section 2.1. The Dirichlet process generates a distribution $G \sim DP(\alpha, H)$ with H the so-called base distribution and α, a scalar, the dispersion parameter.

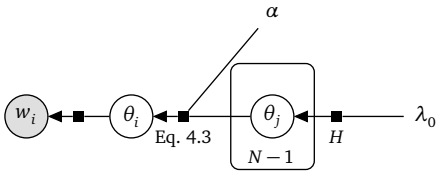


Figure 3.5: The Dirichlet process mixture highlighting the posterior predictive for the parameter θ_i given the other parameters θ_{-i}. Resampling parameters θ_{-i} is governed by Eq. 4.3. The observations w_j with $j \neq i$ with respect to θ_{-i} are not visualized.

The posterior for the Dirichlet process base distribution and dispersion parameter is a Dirichlet process with adjusted parameters:

$$G \mid \theta_1, \ldots, \theta_n \sim DP\left(\alpha + n, \frac{\alpha}{\alpha + n} H + \frac{n}{\alpha + n} \frac{\sum_{j=1}^{n} \delta_{\theta_j}}{n}\right). \tag{3.2}$$

The posterior distribution G is a weighted average between the prior base distribution H and the empirical distribution $n^{-1} \sum_{j=1}^{n} \delta_{\theta_j}$ with the weights respectively α and n (normalized). The dispersion parameter α is updated to $\alpha + n$.

The posterior predictive for a new parameter θ_n has the form (see Section 2.1):

$$\theta_n \mid \theta_1, \ldots, \theta_{n-1} \sim \frac{1}{\alpha + n - 1} \left(\alpha H + \sum_{j=1}^{n-1} \delta_{\theta_j} \right). \tag{3.3}$$

Due to the exchangeability property, any other parameter update can be written down equivalently (Neal, 2000). The prior distribution of parameters θ_i takes the form of conditional distributions:

$$\theta_i \mid \theta_{-i} \sim \frac{1}{\alpha + n - 1} \left(\alpha H + \sum_{j \neq i} \delta_{\theta_j} \right). \tag{3.4}$$

The notation θ_{-i} describes every other parameter than θ_i: the set of parameters, θ_j, with $j \neq i$. This representation with G marginalized has no independent draws anymore. The draws θ_i depend on previous draws θ_{-i}. This representation is known as the Pólya urn scheme (Blackwell and MacQueen, 1973). It lends itself well to Gibbs sampling (Eq. 2.16) as can be found in the literature (Escobar, 1994; Escobar and West, 1995).

3.2.2 Likelihood of Data given a Line

Each point in our point cloud $w_i = (x_i, y_i)$ we map into a intercept-slope representation using $X_i = [1; \quad x_i]$. A line k we model using ordinary linear regression with slope $\beta_{k,0}$, intercept $\beta_{k,1}$, and standard deviation σ_k. Thus, the line is parametrized by $\theta_k = \{\beta_{k,0}, \beta_{k,1}, \sigma_k\}$ or, equivalently, $\theta_k = \{\beta_k, \sigma_k\}$. The noise is normally distributed with the standard deviation σ_k as in ordinary least squares.

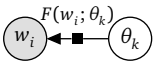


Figure 3.6: This section describes the likelihood of an observation w_i given line parameters θ_k. The likelihood is as in ordinary least squares and can be found in Eq. 3.5.

$$y_i \overset{iid}{\sim} N(X_i \beta_k, \sigma_k^2). \tag{3.5}$$

We can collect all data points $w_i = (x_i, y_i)$ on a line k, $y_i - X_i \beta_k$ for $i = 1, \ldots, n$:

$$y - X\beta_k = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_{k,0} \\ \beta_{k,1} \end{pmatrix}. \tag{3.6}$$

This allows us to write down the likelihood function as:

$$p(y \mid X, \beta_k, \sigma_k) \propto \sigma_k^{-n} \exp\left(-\frac{1}{2\sigma_k^2}(y - X\beta_k)^T(y - X\beta_k)\right). \tag{3.7}$$

Or equivalently, given the points are drawn i.i.d. from the line parameters:

$$p(y_i \mid X_i, \beta_k, \sigma_k) \propto \sigma_k^{-n} \exp\left(-\frac{1}{2\sigma_k^2}(y_i - X_i\beta_k)^T(y_i - X_i\beta_k)\right). \tag{3.8}$$

The likelihood of a data point $w_i = (x_i, y_i)$ given a line k with parameters θ_k is denoted $F(w_i; \theta_k)$ and is the ordinary linear regression model.

$$F(w_i; \theta_k) = p(y_i \mid X_i, \beta_k, \sigma_k^2). \tag{3.9}$$

We will draw the line parameters θ_k from a prior distribution. This makes this model a **Bayesian linear regression** model as can be found in the literature for single lines (Box and Tiao, 2011).

3.2.3 Conjugate Prior for a Line

We postulate the same prior as done before in the literature (Box and Tiao, 2011) for β_k and σ_k in Eq. 3.9. Those priors are defined on $p(\sigma_k^2)$ rather than $p(\sigma_k)$ which are related through a square root operation. First, we write out the joint probability as a product of the conditional probability and the marginal probability as follows:

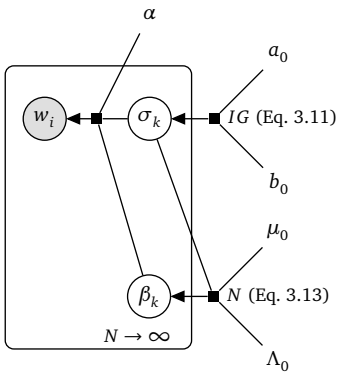


Figure 3.7: The conjugate Normal (N) and Inverse-Gamma (IG) priors for the infinite line model model. The parameter θ_k that parametrizes a line contains a slope β_k and standard deviation σ_k. The main text makes precise how σ_k or more specific, σ_k^2, will be sampled.

$$p(\beta_k, \sigma_k^2) = p(\beta_k \mid \sigma_k^2) p(\sigma_k^2). \tag{3.10}$$

The standard deviation σ_k is sampled from an Inverse-Gamma (IG) distribution:

$$\sigma_k^2 \sim IG(a_0, b_0). \tag{3.11}$$

In particular, we sample from a distribution $IG(a_0, b_0)$ with hyperparameters $a_0 = \nu_0/2$ and $b_0 = \nu_0 s_0^2/2$, see e.g. Mariotto (1989):

$$p(\sigma_k^2) \propto (\sigma_k^2)^{-(\nu_0/2+1)} \exp(-\frac{1}{2\sigma_k^2}\nu_0 s_0^2). \tag{3.12}$$

The conditional with respect to the line coefficients has a normal distribution as prior:

$$\beta_k \sim N(\mu_0, \sigma_k^2 \Lambda_0^{-1}). \tag{3.13}$$

Written out:

$$p(\beta_k \mid \sigma_k) \propto \sigma_k^{-n} \exp\left(-\frac{1}{2\sigma_k^2}(\beta_k - \mu_0)^T \Lambda_0 (\beta_k - \mu_0)\right). \tag{3.14}$$

The NIG is a distribution that combines a Normal and an Inverse Gamma distribution and can be used as a shorthand for the above exposition. Let us define $\lambda_0 = \{\Lambda_0, \mu_0, a_0, b_0\}$ and recall that $\theta_k = \{\beta_k, \sigma_k\}$, we have now a description for our base distribution H of which we can sample θ_k:

$$H = NIG(\lambda_0). \tag{3.15}$$

Summarized, the standard deviation (or more precisely, the variance, σ_k^2) is sampled from the Inverse Gamma distribution and the line line coefficients, β_k, are sampled from a Normal distribution:

$$\begin{aligned} \sigma_k^2 &\sim IG(a_0, b_0), \\ \beta_k &\sim N(\mu_0, \sigma_k^2 \Lambda_0^{-1}). \end{aligned} \tag{3.16}$$

The hyperparameters are $\lambda = \{\Lambda_0, \mu_0, a_0, b_0\}$. Given that we chose the NIG prior to be conjugate to the likelihood (Section 3.2.2), we can write down the prior predictive distribution:

$$\begin{aligned} p(w_i) &= \int F(w_i; \theta_k) p(\theta_k) d\theta_k, \\ p(y_i) &= \int p(y_i | X_i, \beta_k, \sigma_k^2) p(\beta_k, \sigma_k^2) d\beta_k d\sigma_k^2, \\ &= \int N(X_i \beta_k, \sigma_k^2) NIG(\Lambda_0, \mu_0, a_0, b_0) d\beta_k d\sigma_k^2. \end{aligned} \tag{3.17}$$

This can be written in closed form using a multivariate t-distribution (MVSt):

$$p(y_i) = MVSt_{2a_0}(X_i\mu_0, \frac{b_0}{a_0}(I + X_i\Lambda_0 X_i^T)). \tag{3.18}$$

A detailed derivation can be found in Banerjee (2008).

3.2.4 Posterior Predictive for a Line given Data

In Eq. 3.17 and Eq. 3.18 we find the probability of an observation given the prior on the hyperparameters. Similarly, we want to have an expression for the probability of an observation w_i given previous observations w_j (with $j \neq i$) and the same hyperparameters λ_0:

$$p(w_i|w_j) = \int F(w_i;\theta_k)p(\theta_k|w_j)d\theta_k \tag{3.19}$$

We will see that we arrive at an expression similar to Eq. 3.18.

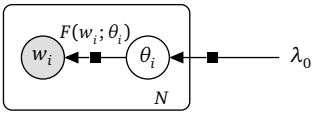


Figure 3.8: The posterior predictive can be calculated by updating the sufficient statistics $\lambda_0 \rightarrow \lambda_n$.

The NIG is a conjugate prior with respect to the normal distribution (with unknown mean and variance). Hence, we have a simplified description for updating the hyperparameters, given a set of observations. Recall, the observation $w_i = (x_i, y_i)$ can be equivalently described as (X_i, y_i) or, generalized for more observations, (X, y). The hyperparameters are updated[1] according to (cf. Denison, 2002; Walter and Augustin, 2010):

$$\begin{aligned}
\Lambda_n &= \Lambda_0 + X^T X, \\
\mu_n &= \Lambda_n^{-1}(\Lambda_0\mu_0 + X^T y), \\
a_n &= a_0 + n/2, \\
b_n &= b_0 + 1/2(y^T y + \mu_0^T\Lambda_0\mu_0 - \mu_n^T\Lambda_n\mu_n).
\end{aligned} \tag{3.20}$$

Let us define $\lambda = \{\Lambda_0, \mu_0, a_0, b_0\}$ and $\lambda^* = \{\Lambda_n, \mu_n, a_n, b_n\}$. Denote a new observation by w_i, then the update for the hyperparameters $\lambda \rightarrow \lambda^*$ can be summarized as:

$$\lambda^* = U_{up}(\lambda, w_i). \tag{3.21}$$

[1] In comparison with the notation of Denison (2002), we update Λ rather than $V = \Lambda^{-1}$ and subsequently use Λ_n at the right-hand side to simplify the notation for μ_n and b_n.

Removing observations does lead to similarly looking updates (which are at times called "downdates"[2] in the literature, see e.g. Wulsin (2013)) for the hyperparameters:

$$\begin{aligned}\Lambda_n &= \Lambda_0 - X^T X,\\ \mu_n &= \Lambda_n^{-1}(\Lambda_0\mu_0 - X^T y),\\ a_n &= a_0 - n/2,\\ b_n &= b_0 - 1/2(y^T y + \mu_n^T \Lambda_n \mu_n - \mu_0^T \Lambda_0 \mu_0).\end{aligned} \tag{3.22}$$

The downdate for the hyperparameters can then be summarized as:

$$\lambda^* = U_{down}(\lambda, w_i). \tag{3.23}$$

Hence, we can sample from $p(\theta_k \mid \lambda_0, w_i)$ by sampling from a Normal-Inverse Gamma distribution with updated hyperparameters, $NIG(\lambda_n)$. Sampling of $NIG(\lambda_n)$ is as in Eq. 3.16, but with λ_n rather than λ_0:

$$\begin{aligned}\sigma_k^2 &\sim IG(a_n, b_n),\\ \mu_k &\sim N(\mu_n, \sigma^2 \Lambda_n^{-1}).\end{aligned} \tag{3.24}$$

The posterior predictive for observation w_i given other observations w_j and the line's hyperparameters becomes:

$$\begin{aligned}p(w_i|w_j) &= \int F(w_i;\theta_k)p(\theta_k)d\theta_k,\\ p(y_i|y_j,X_j) &= \int p(y_i|X_i,\beta_k,\sigma_k^2)p(\beta_k,\sigma_k^2)d\beta_k d\sigma_k^2,\\ &= \int N(X_i\beta_k,\sigma_k^2)NIG(\Lambda_n,\mu_n,a_n,b_n)d\beta_k d\sigma_k^2.\end{aligned} \tag{3.25}$$

This can be written in closed form using a multivariate t-distribution (MVSt):

$$p(y_i|y_j,X_j) = MVSt_{2a_n}(X_i\mu_n, \frac{b_n}{a_n}(I + X_i\Lambda_n X_i^T)). \tag{3.26}$$

A detailed derivation can again be found in Banerjee (2008).

[2]The terminology might have originated from the literature on rank-one updates and downdates on the Cholesky decomposition.

3.3 Inference for the Infinite Line Model

The prior distribution of the parameters can be represented in terms of successive conditional distributions as given in Eq. 4.3 is:

$$\theta_i \mid \theta_{-i} \sim \frac{1}{\alpha + n - 1}\left(\alpha H + \sum_{j \neq i} \delta_{\theta_j}\right). \tag{3.27}$$

The conditional prior of θ_i given the parameters θ_j with $j \neq i$ and observation w_i is described by (Escobar, 1988; Escobar and West, 1995; MacEachern and Müller, 1998; Neal, 2000):

$$\theta_i \mid \theta_{-i}, w_i \sim r_i H_i + \sum_{j \neq i} F(w_i; \theta_j)\delta_{\theta_j}. \tag{3.28}$$

We use mainly the notation by Neal (2000), also compare Theorem 5.3 in Ghosal and Van der Vaart (2017). The α-weighted posterior r_i defines the probability that a new cluster will be sampled:

$$r_i = \alpha \int F(w_i; \theta) dH(\theta; \lambda_0). \tag{3.29}$$

In the case of the infinite line model the probability of an observation w_i given the hyperparameter λ_0 is given by Eqs. 3.17 and 3.18. In Eq. 3.29 we multiply the posterior with α which will govern the probability of a new cluster being created.

The distribution H_i is the posterior distribution for the parameter θ given base distribution H and a single observation w_i. We do not need to have calculate the probability for particular parameter values, we only have to sample them from this distribution. Sampling from H_i must be feasible.

$$\theta_i \sim H_i. \tag{3.30}$$

We can sample from H_i by performing a single update of the hyperparameters λ in Eq. 3.21 with observation w_i ($n = 1$, $a_n = a_0 + 1/2$, and so on) and then sampling from $NIG(\lambda_{n=1})$, see Eq. 3.24.

The probability of sampling a new parameter is given by[3]:

$$p(\theta_{new}) = \frac{r_i}{r_i + \sum_{j \neq i} F(w_i; \theta_j)}. \tag{3.31}$$

This Gibbs algorithm[4] has been described before in the context of a Dirichlet process mixture, without particular likelihoods or priors in mind (see algorithm 1 in Neal, 2000). As shown in Algorithm 8 after initialization[5], we perform a loop in which for T iterations each

[3] This can be derived from Neal (2000) by $\sum_{i \neq j} bF(y_i, \theta_i) + b\alpha \int F(y_i; \theta_j) dG_0(\theta) = 1$, which gives an expression for the normalization factor b, which can be found as denominator in Eq. 3.31.

[4] The implementation can be found at `https://code.annevanrossum.nl/dpm` in the folder inference (gibbsDPM_algo2), written such that it is compatible with octave.

[5] Initialization details can be found in Appendix B.

Algorithm 8 Gibbs sampling over parameters θ_i

procedure GIBBS ALGORITHM 1(w, λ_0, α) ▷ Accepts points w, hyperparameters λ_0, α and returns k line coordinates
  $\theta_i =$ GIBBS ALGORITHM 1 INITIALIZATION(w, λ_0, α) ▷ See Algorithm 15.
  for all $t = 1 : T$ **do**
    for all $i = 1 : N$ **do**
      $r_i = \alpha \int F(w_i; \theta) dH$ ▷ Weighted posterior predictive of w_i (Eq. 3.29)
      for all $j = 1 : N, j \neq i$ **do**
        $L_{i,j} = F(w_i; \theta_j)$ ▷ Likelihood of a line given an observation (Eq. 3.9)
      end for
      $p(\theta_{new}) = \frac{r_i}{r_i + \sum_{j \neq i} L_{i,j}}$ ▷ Probability of sampling a new parameter (Eq. 3.31)
      $u \sim U(0, 1)$
      if $p(\theta_{new}) > u$ **then** ▷ Sample with probability $p(\theta_{new})$
        $\lambda_n = U_{up}(w_i, \lambda_0)$ ▷ Update hyperparameters with w_i (Eq. 3.21)
        $\theta_i \sim NIG(\lambda_n)$ ▷ Sample θ_i from NIG (Eq. 3.24)
      else
        $i \sim Mult(N, p(\theta_{old}))$ ▷ Sample i from existing parameters, θ_{old}
        $\theta_i = \theta_{old=i}$ ▷ Pick θ_i given index i
      end if
    end for
  end for
  return summary on θ_k for k lines
end procedure

θ_i belonging to observation w_i is updated in succession. The loop consists of four steps. First, the posterior predictive for w_i given the hyperparameters $p(w_i \mid \lambda_0)$ is calculated. Second, the likelihood $F(w_i; \theta_j)$ for all θ_j given w_i (with $j \neq i$) is calculated. Third, the fraction with r_i defines the probability for θ_i to be sampled from a new or existing cluster. Fourth, depending on the probability u, (1) a new cluster is sampled, the hyperparameters are updated with information on w_i and thereafter θ is sampled from a Normal-Inverse-Gamma distribution with the updated hyperparameters, or (2) an existing cluster is sampled.

3.4 Accelerating Inference for the Infinite Line Model

In the previous section we sampled over individual parameters. It is possible to iterate only over the clusters. The derivation takes a few steps (Neal, 2000) but leads to a simple update for the component indices that only depends on the number of data items per cluster, the parameter α, and the available data.

The probability to sample from an existing cluster depends on the number of items in that cluster (the current data item excluded). This is expressed in equation 3.32.

$$p(c_i = c \text{ and } c_i = c_j \text{ and } i \neq j \mid c_{-i}, w_i, \alpha, \theta) \propto \frac{n_{c,-i}}{\alpha + n - 1} F(w_i; \theta_i). \tag{3.32}$$

The probability to sample a new cluster only depends on α and the total number of data items. This is formally described in equation 3.33.

Algorithm 9 Gibbs sampling over clusters c_k

procedure GIBBS ALGORITHM 2(w, λ_0, α) ▷ Accepts points w and hyperparameters λ_0 and α, returns k line coordinates
  $\theta_k, \lambda_c =$ GIBBS ALGORITHM 2 INITIALIZATION(w, λ_0, α) ▷ See Algorithm 16
  for all $t = 1 : T$ **do**
    for all $i = 1 : N$ **do**
      $c =$ cluster(w_i) ▷ Get cluster c currently assigned to observation w_i
      $\lambda_c = U_{down}(w_i, \lambda_c)$ ▷ Adjust cluster hyperparameters on removing w_i (Eq. 3.23)
      $m_c = m_c - 1$ ▷ Adjust cluster size m_c (and bookkeeping of K)
      for all $k = 1 : K$ **do**
        $L_k = m_k$ F$(w_i; \theta_k)$ ▷ Likelihood for cluster k given w_i (Eq. 3.34)
      end for
      $r_i = \alpha \int F(w_i; \theta) dH$ ▷ Weighted posterior predictive of w_i (Eq. 3.29)
      $p(\theta_{new}) = \frac{r_i}{r_i + \sum_k L_k}$ ▷ Calculate probability of a new parameter
      $u \sim U(0, 1)$
      if $p(\theta_{new}) > u$ **then** ▷ Sample with probability $p(\theta_{new})$
        $k = K + 1$ ▷ New cluster index (and bookkeeping of K, $K = K + 1$)
        $\lambda_k = U_{up}(w_i, \lambda_0)$ ▷ Set hyperparameter λ_k with prior pred. given w_i
        $\theta_i \sim NIG(\lambda_k)$ ▷ Sample θ_i from NIG
      else
        $k \sim Mult(K, L_k)$ ▷ Sample k from existing clusters (weighed by m_k)
        $\lambda_k = U_{up}(w_i, \lambda_k)$ ▷ Update hyperparameter k with post. pred. given w_i
      end if
      $m_k = m_k + 1$ ▷ Increment cluster size m_k
    end for
    for all $k = 1 : K$ **do**
      $\theta_k \sim NIG(\lambda_k)$ ▷ Sample θ_k from NIG with up to date λ_k
    end for
  end for
  return summary on θ_k for k lines
end procedure

$$p(c_i \in \Omega(c) \text{ and } c_i \neq c_j \text{ and } i \neq j \mid c_{-i}, \alpha) \propto \frac{\alpha}{\alpha + n - 1} \int F(w_i; \theta_i) dH(\theta). \tag{3.33}$$

Here $\Omega(c)$ denotes all admitted values for c_i. The importance of conjugacy is obvious from Eq. 3.33, it will lead to an analytic form of the integral. The inference method using Eqs. 3.32 and 3.33 is described in Section 3.2.

One benefit of iterating over clusters rather than non-unique parameters is that we can calculate the likelihood by multiplying it with the number of observations at that cluster (rather than per parameter). If we write the number of observations as $n_{c,-i} = m_k$, we can update the likelihood on a cluster level like this:

$$L_k = m_k F(w_i; \theta_k). \tag{3.34}$$

Directly sampling over the clusters is described in its general form (see algorithm 2 in Neal, 2000). Rather than updating each θ_i per observation w_i, an entire cluster θ_k is updated. In Algorithm 8 the update of a cluster would require a first observation to generate a new cluster at θ_j and then moving all observations of the old cluster θ_i to θ_j. In contrast, in

Algorithm 9 when a data item either is added or deleted from a cluster, the cluster parameters are updated for all data items in that cluster at once. For this algorithm this means that when w_i is excluded from calculating the likelihood we have to[6] "downdate" the corresponding hyperparameters (as described in Eq. 3.23). In Algorithm 9 after all observations have been iterated over and assigned the corresponding cluster k, an outer loop iterates over all clusters to obtain new parameters θ from the NIG prior.

3.5 Results

The infinite line model (see Section 3.2) is able to fit an infinite number of lines through a point cloud in two dimensions. These lines are no line segments, but infinite lines. However, to test the model a variable number of lines are generated of a length that is considerably larger compared to the spread caused by the standard deviation of points from that line.

As described before, Gibbs sampling leads to correlated samples. Our choice is to get the Maximum A Posterior estimates for the clusters by picking the median values for all the parameters involved. In Section 3.5.1 we discuss the clustering performance, in Section 3.5.2 we compare with the Hough transform, in Section 3.5.3 we provide two clustering examples, and in Section 3.5.4 we inspect visually if the model converges through trace plots.

3.5.1 Clustering Performance

The results of the clustering algorithms are measured using conventional metrics. Let us first define the contingency table (see Table 3.1).

Table 3.1: Contingency table. The overlap between clusters X and Y is characterized by the numbers n_{ij} with each number denoting the number of objects common to X and Y: $n_{ij} = |X_i \cap Y_j|$.

$X \backslash Y$	Y_1	Y_2	$\dots$	Y_s	Sums
X_1	n_{11}	n_{12}	$\dots$	n_{1s}	a_1
X_2	n_{21}	n_{22}	$\dots$	n_{2s}	a_2
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
X_r	n_{r1}	n_{r2}	$\dots$	n_{rs}	a_r
Sums	b_1	b_2	$\dots$	b_s	

(3.35)

[6] We don't strictly have to "downdate". However, this would become quickly computationally intensive. At first sight, we might consider storing a tree of hyperparameters, with a branch for each sequence of observations that we explore. Removal of an observation assigned to a particular cluster would then correspond to backtracking. We go up the the parent node in the tree and get its previously calculated λ value. However, on reaching a leaf through adding observation w_i, we might need to update the hyperparameters by removing observation w_j. This means we need to calculate updates for λ_n for all permutations of θ_i we encounter (permutations, not sequences).

There are basically four types of object pairs possible.

- A pair of points that are placed in the same class in X and in the same class in Y.
- A pair of points that are placed in a different class in X and in a different class in Y.
- A pair of points that are placed in a different class in X and in the same class in Y.
- A pair of points that are placed in the same class in X and in a different class in Y.

Here X can be considered the ground truth, Y the inferred cluster assignment. Note that there is no mention of the indices of those classes (they are exchangeable). The first and second type can be considered "agreements". The third and fourth type can be seen as "disagreements". Let us define, the total number of distinct point pairs:

$$T = \binom{n}{2} = n(n-1)/2. \tag{3.36}$$

The agreements can be counted as (Brennan and Light, 1974):

$$A = \binom{n}{2} + \sum_{i=1}^{r}\sum_{j=1}^{s} n_{ij} - \frac{1}{2}\left(\sum_{i=1}^{r} a_i^2 + \sum_{j=1}^{s} b_j^2\right). \tag{3.37}$$

We will use four performance metrics, the Rand Index, the Mirkin index, the Hubert index, and the Adjusted Rand Index,

Let us first define the Rand Index. It describes the accuracy of cluster assignments (Rand, 1971) as a the ratio of agreements with respect to the total number of possible pairs.

▼ Definition 3.1 — *Rand index*

The Rand index RI is defined by:

$$RI = \frac{A}{T}. \tag{3.38}$$

The Mirkin index (Mirkin and Cherny, 1970) is a metric for disagreement.

▼ Definition 3.2 — *Mirkin index*

The Mirkin index MI is defined by:

$$MI = \frac{T-A}{T}. \tag{3.39}$$

The Hubert index (Hubert, 1977) is a metric takes into account the difference between agreement and disagreement.

▼ **Definition 3.3 — *Hubert index***

The Hubert index HI is defined by:

$$HI = \frac{A-(T-A)}{T}. \tag{3.40}$$

However, if we randomly assign points to clusters there is a chance that we assign some of the points correctly. The adjusted Rand index (Hubert and Arabie, 1985) is a "corrected-for-chance" version of the Rand index. The correction calculates the expected index given that X and Y are chosen from a generalized hypergeometric contribution (given number of classes and objects in each):

$$E = \frac{n(n^2+1)}{2(n-1)} - \frac{n+1}{2(n-1)}\left(\sum_{i=1}^{r} a_i^2 + \sum_{j=1}^{s} b_j^2\right) + \frac{2}{2n(n-1)}\left(\sum_{i=1}^{r} a_i^2 \sum_{j=1}^{s} b_j^2\right). \tag{3.41}$$

We can then define the correction to the Rand index by making sure that $A = E$ maps to 0 and that $A = T$ maps to 1.

▼ **Definition 3.4 — *Adjusted Rand index***

The Adjusted Rand index AR is defined by:

$$AR = \frac{A-E}{T-E} \tag{3.42}$$

The clustering performance is quite different from the line estimation performance. If the points are not properly assigned, the line will not be estimated correctly. Due to the fact that line estimation has this secondary effect, line estimation performance is not taken into account. Moreover, from lines that generated only a single, or very few points, we can extract point assignments, but line coefficients are impossible to derive. In fact, any derivation would lead to introducing a threshold for the number of points per cluster. Then the performance would need to be measured by weighting the fitting versus the assignment.

The performance of Algorithm 8 can be seen in Figure 3.9 and is rather disappointing. On average the inference procedure agrees upon the ground truth for 75% of the cases considering the Rand Index. Even worse, if we adjust for chance as with the Adjusted Rand Index, the performance would then drop to only having 25% correct cases!

Algorithm 9 leads to stellar performance measures (Figure 3.10). Apparently, updating entire clusters at once with respect to their parameter values leads at times to perfect clustering, bringing the performance metrics close to their optimal values (see also Van Rossum et al., 2016b).

The lack of performance of Algorithm 8 is not only caused by slow mixing. Even when allowing it ten times the number of iterations of Algorithm 8, it does not reach the same

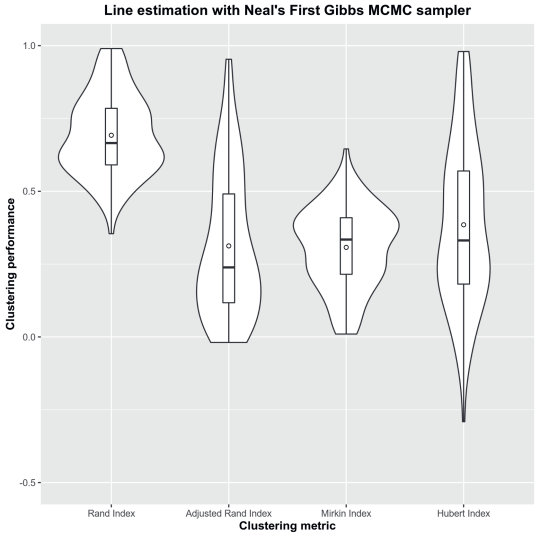


Figure 3.9: The performance of Algorithm 8 with respect to clustering is measured using the Rand Index, the Adjusted Rand Index, the Mirkin metric, and the Hubert metric. A score of 1 means perfect clustering for all metrics, except Mirkin's where 0 denotes perfect clustering.

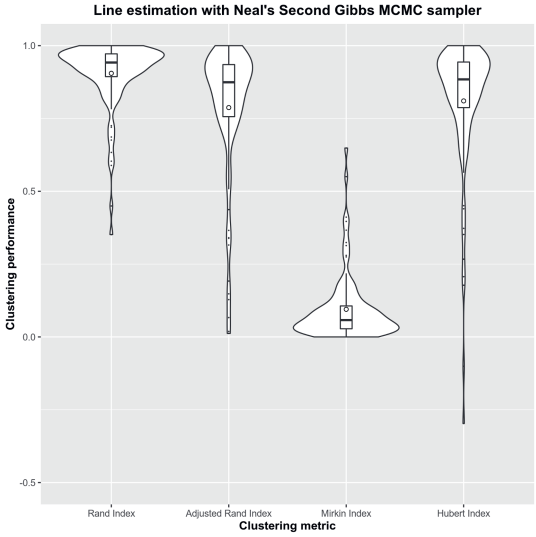


Figure 3.10: The performance of Algorithm 9 with respect to clustering is measured using the Rand Index, the Adjusted Rand Index, the Mirkin metric, and the Hubert metric. A score of 1 means perfect clustering for all metrics, except Mirkin's where 0 denotes perfect clustering.

performance levels. A line seems to form local regions of high probability, making it difficult for points to postulate slightly changed line coordinates.

3.5.2 Hough Transform

A full Bayesian method, in contrast to ad-hoc methods such as the Hough transform, means optimal inference given the model and noise definition. In practice, the model might be misspecified or the actual realization of lines might not have enough data points to benefit from the Bayesian approach. Nevertheless, it is interesting to compare with the Hough transform.

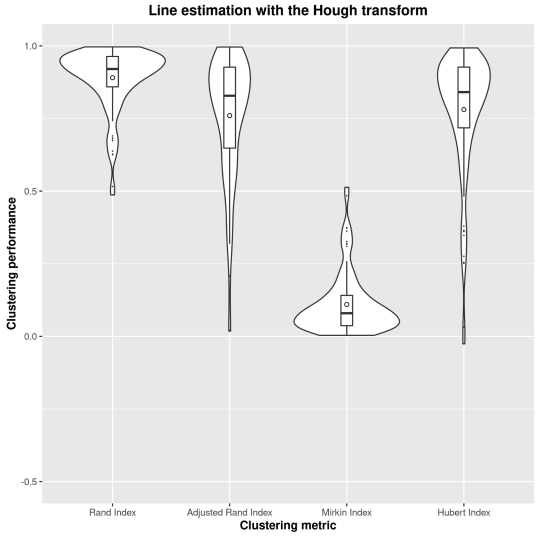


Figure 3.11: The results of the Hough transform on the same dataset. The performance of the Hough transform is slightly worse than line estimation using Neal's second MCMC sampler (Figure 3.10).

The implementation details of the Hough transform are irrelevant to the thesis[7]. We briefly summarize here the key points. The random Hough transform takes two points at random, fits a line between those points, and establishes slope and intercept of this line. A discrete object, the accumulator, exists of, in this case, 100 by 100 cells. Each cell represents an interval of slope and intercept values. For each two points, the accumulator increments a counter per cell. After running over (many or) all point pairs, those cells with large accumulated values are considered to be the detected lines. What constitutes large is determined by a threshold that is application specific.

3.5.3 Two Examples

First, we show two examples of line estimations as we would expect them (see Fig. 3.12).

In contrast to the pictures of Fig. 3.12, we show two examples with typical mistakes. These examples can guide us to understand the inference process better. The first example in seen

[7]Implementation can be found at `https://code.annevanrossum.nl/hough`

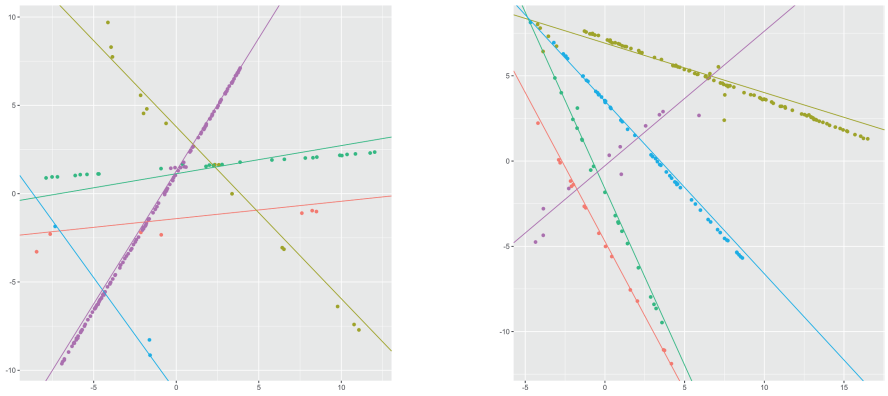
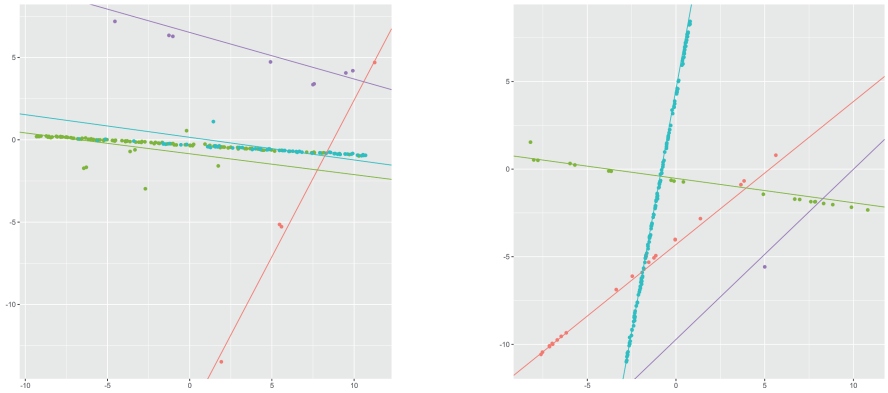


Figure 3.12: Examples of the line estimation process. Apart from slightly different angles and perhaps a few misclassifications the assignments look good.

in Figure 3.13a. It shows the assignment of two lines to a series of points that originated from a single line in the ground truth. Such an assignment can happen after a single Gibbs step in Algorithm 8 or after a long run as final assignment if the system does not convergence to underlying correct assignment.

There is a single line that is represented by two clusters. Algorithm 8 does not have merge or split steps to perform inference about sets of data points, it thus has to move each data point one by one. In passing we mention that there are split-merge algorithms that take these more sophisticated Gibbs steps into account (Jain and Neal, 2004) and we will see these in the following two chapters.



(a) This shows a mistake where a single line is fitted by two separate lines. One of the lines, the horizontal one in the center has been assigned to multiple clusters.

(b) This shows that outliers are no problem for this type of estimation. An outlier (see the purple point), even if it is a single point, can be assigned its own line.

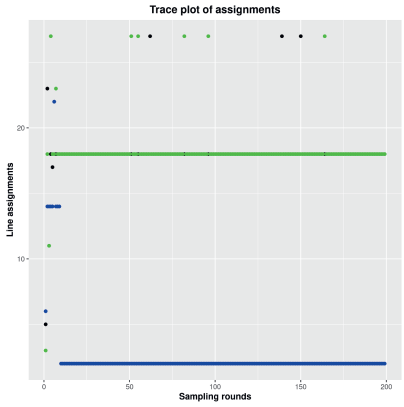
Figure 3.13: Examples of incorrect assignments in line estimation.

The second example (Figure 3.13b) shows that a single point as an outlier is not a problem for our method. A single point might throw off Bayesian linear regression, but because there are multiple lines to be estimated in our Infinite Line Mixture Model, this single point is assigned its own line.

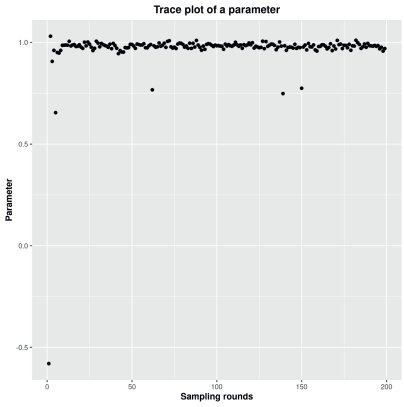
The extension to more points as outliers would, of course, require us to postulate a distribution for these outlier points as well. For instance, a uniform distribution might be used in tandem with the proposed model. However, this would lead to a non-conjugate model and hence it would require different inference methods.

3.5.4 Trace Plots

To study the convergence of parameters in an MCMC model, one of the visual aids that is in use, are so-called trace plots. A trace plot does plot values over the course of the simulation run. If we study the trace plot of individual assignments of points over lines, they are not assigned very often to other lines.



(a) This plot traces three points that are assigned to clusters (limited to around 30). Two of the points are assigned to one cluster. The other point to the cluster at the bottom. The plot only shows accepted assignments. The acceptance of a new assignment takes rarely hold.



(b) This plot traces a line parameter β_i belonging to point w_i. It exhibits exploratory behavior around a particular value (in this case 1). So now and then it shows other points (probably from w_i being assigned to a different line, compare with the plot at the right).

Figure 3.14: Two examples of trace plots. Left: a trace plot of the assignment of points to cluster (it changes not so often). Right: a trace plot of a parameter value β_i assigned to w_i.

In Figure 3.14 there are two trace plots. The first plot shows the trace plot of assignments themselves. The MCMC chain steadily assigns the same parameter to the visualized observations. At the start there is a burn-in period visible in which the assignment is more variable. After the burn-in period there are still reassignments, but they are rare. The second plot shows the trace plot of a value of one of the parameters to which on observation has been fitted.

3.6 Chapter Conclusions

The infinite line model proposed in this chapter extends the familiar Bayesian linear regression model to an infinite number of lines using a Dirichlet Process as prior. The model is a full Bayesian method to detect multiple lines. A full Bayesian method, in contrast to ad-hoc methods such as RANSAC or the Hough transform, means optimal inference (Zellner, 1988) given the model and noise definition.

Results in section 3.5 show high values for different performance metrics for clustering, such as the Rand Index, the Adjusted Rand Index, and other metrics (Van Rossum et al., 2016a,b). The Bayesian model is solved through two types of algorithms. Algorithm 8 iterates over all observations and suffers from slow mixing. The individual updates make it hard to reassign a large number of points at the same time. Algorithm 9 iterates over entire clusters. This allows updates for groups of points leading to much faster mixing. We note that even optimal inference may occasionally result in misclassifications. The dataset is generated by a random process. Hence, occasionally two lines are generated with almost the same slope and intercept. Points on these lines are impossible to assign to the proper line.

This chapter contributes to answering our first research question.

> **RQ 1** How can we estimate the number of objects simultaneously with the fitting of these objects?

We use a Bayesian method that we demonstrate on line objects. Its nonparametric nature allows for simultaneous establishing the number of lines as well as their fit.

The essential contribution of this chapter is the introduction of a fully Bayesian method to infer lines. For such a model, it holds that there are two ways in which it can to be extended for full-fledged inference in computer vision as required in robotics. First, the extension of lines in 2D to planes in 3D. This is an extension that does not change anything of the model except for the dimension of the data points. Second, somehow a prior needs to be incorporated to cut the lines (of infinite length) to line segments. It means that we need to restrict the points on the line to a uniform distribution of points over a line segment. A symmetric Pareto distribution can be used as prior for the end points of the line segment (see next Chapter). Modeled in this manner, this would subsequently allow for a hierarchical model in which the end points of the line segment are on their turn part of more complicated objects. Hence, the Infinite Line Mixture Model is an essential step towards the use of Bayesian methods (and thus properly formulated priors) for robotic computer vision.

CHAPTER 4

NONPARAMETRIC BAYESIAN SEGMENT ESTIMATION

Contents In this chapter, we introduce a Bayesian method to perform inference over line segments. In this model, infinite segment model (ISM), the prior for the location is given by a Normal distribution, the prior for the length of the segment is given by a Pareto distribution. Due to the fact that the prior and likelihood do not form a conjugate pair, a more general inference method is used (than the inference methods for the conjugate model in Chapter 3), namely Gibbs sampling with auxiliary variables.

Published in A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Nonparametric Segment Detection. *Proceedings of the Eighth European Starting AI Researcher Symposium*, STAIRS 2016, the Hague, the Netherlands, August 26-27, 2016.

Outline Our proposed model is using both a Normal-Inverse-Gamma distribution and a Normal and Pareto distribution as priors for an individual line segment (Section 4.1). Inference over the infinite segment model is done using Gibbs sampling over auxiliary variables (Section 4.2). The results for inference over line segments are compared with those for lines (Section 4.3). Finally, weak aspects of the current MCMC method are established (Section 4.4). They will form the basis for new inference methods in the next chapters.

4.1 Infinite Segment Model

The application we would like to address in this chapter is that of the detection of multiple segments rather than lines. We will label the model is the infinite segment model. The term

infinite relates to the use of a nonparametric Bayesian prior. The term does not reflect the size of the segments.

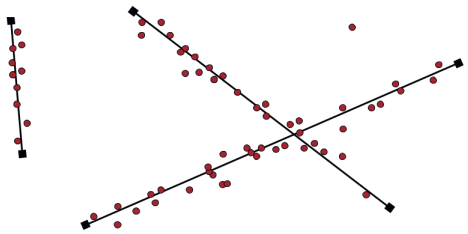


Figure 4.1: A mixture of segments. The segments have two more parameters compared to lines: the length of the segment and its center (or alternatively, the endpoints of the segment). Analogous to the line detection application, there are n points in 2D space, each point generated from a segment with parameters θ_k. The number of segments, k, is not known beforehand. Compare with Figure 3.1.

We will model the infinite segment model similar to the infinite line model, namely as a Dirichlet process mixture:

$$\begin{aligned} G &\sim DP(\alpha, H), \\ \theta_i \mid G &\overset{iid}{\sim} G, \\ w_i \mid \theta_i &\overset{iid}{\sim} F(w_i; \theta_i). \end{aligned} \tag{4.1}$$

The likelihood function F describes the mapping from parameters θ_i to observations w_i. In the previous chapter this has been a likelihood function that describes points on lines. In this chapter the likelihood function describes points on line segments.

Along the same lines as in Chapter 3 we have a base distribution H, a dispersion factor α, and hyperparameters for the base distribution λ_0.

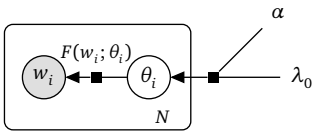


Figure 4.2: The Dirichlet process mixture with hyperparameter λ_0 for the base distribution H.

For each point w_i the segment parameters are given by θ_i. The parameters θ_i are not necessarily unique (for $i \neq j$). When we iterate over unique segments we will use the subscript k rather than i or we will mention this explicitly.

4.1.1 Posterior Predictive for a Segment given Other Segments

This follows exactly the same derivation as for the infinite line model in Section 3.2.1. The posterior predictive is given by (Neal, 2000):

$$\theta_n \mid \theta_1, \ldots, \theta_{n-1} \sim \frac{1}{\alpha + n - 1} \left(\alpha H + \sum_{j=1}^{n-1} \delta_{\theta_j} \right). \tag{4.2}$$

The prior distribution of parameters θ_i takes the form of conditional distributions:

$$\theta_i \mid \theta_{-i} \sim \frac{1}{\alpha + n - 1} \left(\alpha H + \sum_{j \neq i} \delta_{\theta_j} \right). \tag{4.3}$$

The notation θ_{-i} describes every other parameter than θ_i: the set of parameters, θ_j, with $j \neq i$.

4.1.2 Likelihood of Data given Segment Parameters

The likelihood $F(w_i, \theta_i)$ describes the mapping from parameters θ_i to observations w_i. We create a likelihood function by the combination of two probability density functions. The observation w_i has x-coordinate x_i and y-coordinate y_i. We sample x_i from a uniform distribution only giving it nonzero probability on a particular segment on the x-axis:

$$x_i \mid c, d \overset{iid}{\sim} U(c - d, c + d). \tag{4.4}$$

This defines a segment on the x-axis centered at c which extends in both directions with size d. We will use an intercept-slope representation (Chapter 3). Let us define $X_i = [1, x_i]$ with x_i distributed as in Eq. 4.4. The column vector $\beta = [\beta_0, \beta_1]$ contains two parameters: the y-intercept β_0 and the slope parameter β_1 (compare Section 3.2.2). And we assume a normally distributed random variable across $y - X\beta$, the same as in the line model (Eq. 3.5):

$$y_i \overset{iid}{\sim} N(X_i \beta_k, \sigma_k^2). \tag{4.5}$$

The combination of Eq. 4.4 and Eq. 4.5 generates points across a segment on a line.

4.1.3 Prior for a Segment

We postulate a prior that is a combination of Bayesian linear regression with restrictions on the size of the line:

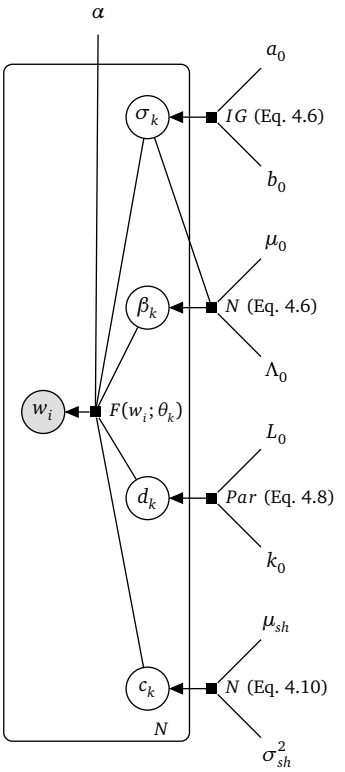


Figure 4.3: The segment parameters for segment k are $\theta_k = \{\sigma_k, \beta_k, d_k, c_k\}$. Here σ_k and β_k are sampled from the same distributions (an Inverse-Gamma, respectively, a Normal distribution) as in the infinite line model. The extend of the segment, d_k, is defined by a Pareto distribution and its center, c_k, by a Normal distribution.

The slope and intercept parameters of the segment are sampled according to a Normal-Inverse-Gamma distribution (compare Eq. 3.16 for line parameters):

$$\begin{aligned} \sigma_k^2 &\sim IG(a_0, b_0), \\ \beta_k &\sim N(\mu_0, \sigma_k^2 \Lambda_0^{-1}). \end{aligned} \tag{4.6}$$

Recall that the data on a line segment is distributed uniformly (Eq. 4.4). This is parametrized through two parameters, the center of the segment, c, and its extent, d:

$$x \mid c, d \sim U(c - d, c + d). \tag{4.7}$$

We propose as a prior for the extend of the line segment d, a Pareto distribution (Par):

$$d \mid L_0, k_0 \sim Par(L_0, k_0). \tag{4.8}$$

The Pareto distribution (Par) is given by:

$$p(d|L_0,k_0) = \begin{cases} k_0 L_0^{k_0} d^{-k_0-1} & if\ d \geq L_0, \\ 0 & \text{otherwise.} \end{cases} \tag{4.9}$$

The parameter L_0 can be seen as a prior parameter that sets a minimal size to the line segment. The parameter k_0 is the shape parameter of the Pareto distribution.

The center of the segment is sampled from a Normal distribution:

$$c \mid \mu_{sh}, \sigma_{sh}^2 \sim N(\mu_{sh}, \sigma_{sh}^2). \tag{4.10}$$

The subscript in μ_{sh} and σ_{sh}^2 stands for shifted. The center of the segment is shifted along the line.

We will collect all priors and call it a Segment prior, abbreviated to Seg.

$$\theta_k \sim Seg(\lambda_0) \tag{4.11}$$

Writing out all parameters:

$$\beta_k, \sigma_k^2, d_k, c_k \sim Seg(a_0, b_0, \mu_0, \Lambda_0, L_0, k_0, \mu_{sh}, \sigma_{sh}^2). \tag{4.12}$$

This corresponds to:

$$\begin{aligned} \beta_k, \sigma_k^2 &\sim NIG(a_0, b_0, \mu_0, \Lambda_0), \\ d_k &\sim Par(L_0, k_0), \\ c_k &\sim N(\mu_{sh}, \sigma_{sh}^2) \end{aligned} \tag{4.13}$$

4.1.4 Sampling Segment Parameters given Data

In contrast to the infinite line model there is no conjugacy between prior and likelihood in the infinite segment model. We have no closed-form updates for hyperparameters given observed data. Hence, we have to resort to sampling parameters. The proposal distribution, Q, with which we sample new parameters can be using the current state, θ_k, or it can sample from the prior λ_0, or a combination thereof:

$$\theta_k \sim Q(\theta_k, \lambda_0) \tag{4.14}$$

Observations are sampled independently from line parameters (Section 4.1.2), hence the likelihood of a set of observations is described by the product.

$$L_k = \prod_i p(\theta_k|w_i) \tag{4.15}$$

We can sample θ_{new} from $Seg(\lambda_0$ and then accept with probability L_{new}/L_k. Alternatively we can sample using an MCMC proposal distribution around θ_k:

$$\theta_k \sim N(\theta_k, \sigma^2_{prop}). \tag{4.16}$$

Alternatively, we can sample in a way that reflects our priors. For example, taking turns and sample first β_k, σ^2_k from a NIG distribution keeping d_k, c_k the same and the other way around, sample d_k, c_k from a Pareto-Normal distribution and keep β_k, σ^2_k the same.

4.2 Inference for the Infinite Segment Model

Let us introduce *Gibbs sampling with auxiliary variables* (Neal, 2000), see Algorithm 10.

Algorithm 10 Gibbs sampling with auxiliary variables

procedure GIBBS ALGORITHM WITH AUXILIARY VARIABLES(w, λ_0, α) ▷ Accepts points w and hyperparameters λ_0 and α. Requires also the number of auxiliary variables V, a proposal distribution Q. Returns k line coordinates.
  for all $t = 1 : T$ **do**
    for all $i = 1 : N$ **do**
      for all $v = 1 : V$ **do**
        $\theta_v \sim Seg(\lambda_0)$ ▷ Sample from Eq. 4.11.
        $m_v = \alpha / V$
      end for
      $c = \text{cluster}(w_i)$ ▷ Get cluster c currently assigned to observation w_i.
      $m_c = m_c - 1$ ▷ Adjust cluster size m_c (and bookkeeping of K).
      for all $k = 1 : K + m$ **do**
        $L_k = m_k F(w_i; \theta_k)$ ▷ Calculate likelihood for all θ_k.
      end for
      $k \sim Mult(K + m, L_k)$ ▷ Sample k from all clusters (weighed by m_k cq m_v).
      $\theta_i = \theta_k$ ▷ Set θ_i to sampled cluster.
      $m_k = m_k + 1$ ▷ Increment m_k (set to 1 for m_v, and adjust K).
    end for
    for all $k = 1 : K$ **do**
      $\theta_{prop} \sim Q(\theta_k, \lambda_0)$ ▷ Sample from proposal distribution (Eq. 4.14).
      $L_{prop} = \prod_i F(w_i; \theta_{prop})$ ▷ Likelihood for all i at θ_{prop}.
      $L_k = \prod_i F(w_i; \theta_k)$ ▷ Likelihood for all i at θ_k.
      $u \sim U(0, 1)$
      if $(L_{prop}/L_k) > u$ **then** ▷ Accept/reject.
        $\theta_k = \theta_{prop}$
      end if
    end for
  end for
  return summary on θ_k for k line segments.
end procedure

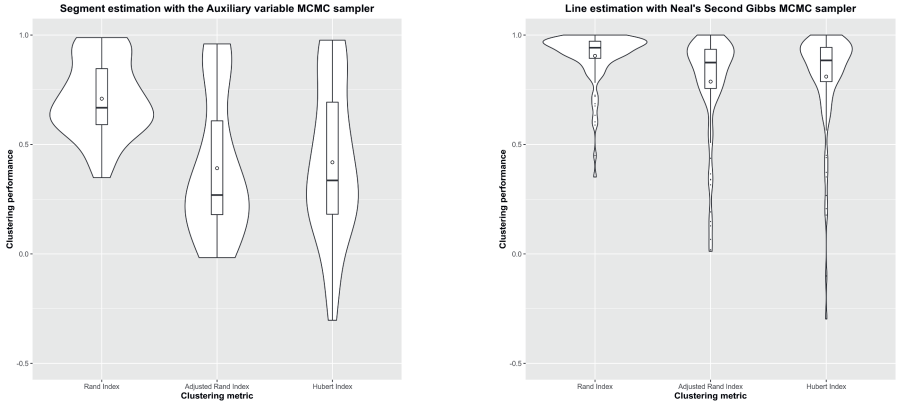
This Gibbs algorithm[1] has been described before in the context of a Dirichlet process mixture, without particular likelihoods or priors in mind (see algorithm 8 in Neal, 2000). The sampling process proposes V new values for the parameters from the hyperparameters. The V values are called auxiliary parameters. Now, to establish to which cluster a certain observation w_i needs to be assigned, the likelihood of each existing and new clusters alike are compared. The weight of an old cluster is defined through the number of data points assigned to it. The weight of a new cluster is defined through α/V. After every data item is assigned a cluster, the cluster parameters themselves are updated given the assigned data items.

4.3 Results

We show a drop in performance for segment detection compared to line detection in Section 4.3.1. Some examples of difficult to assign segments are given in Section 4.3.2. We visualize (the lack of) convergence in Section 4.3.3.

4.3.1 Clustering Performance

The results over a larger dataset can be measured with clustering metrics as visualized in Figure 4.4. The clustering performance of the segment detection algorithm, measured by the clustering index, such as the Rand Index, the Adjusted Rand Index, and the Hubert metric, show all reduced performance (see Figure 4.4) compared to line detection (without constraints on segment size).



(a) Segment detection.

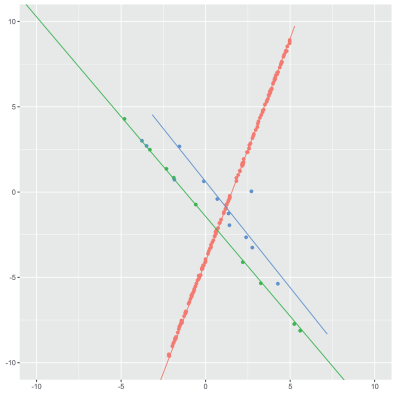
(b) Line detection.

Figure 4.4: Segment detection performs worse than line detection across all three clustering performance indicators. Perfect clustering is indicated by 1.0 for Rand Index, Adjusted Rand Index, and Hubert.

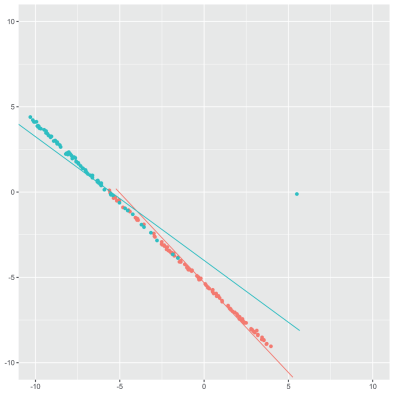
[1] The implementation can be found at `https://code.annevanrossum.nl/dpm` in the folder inference (gibbsDPM_algo8), written such that it is compatible with octave.

4.3.2 Examples

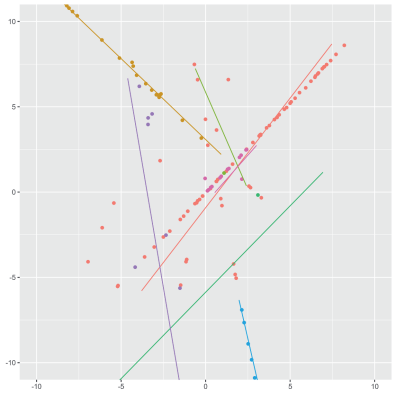
In Figure 4.5 we show four Bayesian point estimates of the sampling process. These are examples that demonstrate the type of errors that are made in the inference process. In example (a) the segments are correctly sampled. In (b) the type of error is that of recognizing multiple segments where there is only one segment to the human observer. In (c) the error is due to the fact that some segments contain very few points. In (d) the error stems from line segments being chosen orthogonal to the actual segment.



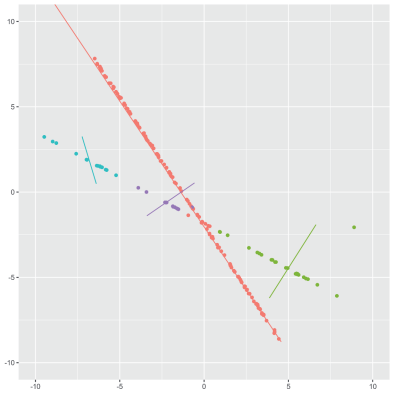
(a) There is an outlier right of the center. Also, the line segments that have fewer points, have endpoints that are recognized less "tight" (to be expected given the Pareto prior).



(b) The single line segment is incorrectly recognized as multiple segments.



(c) The segments with fewer observations are recognized poorly.

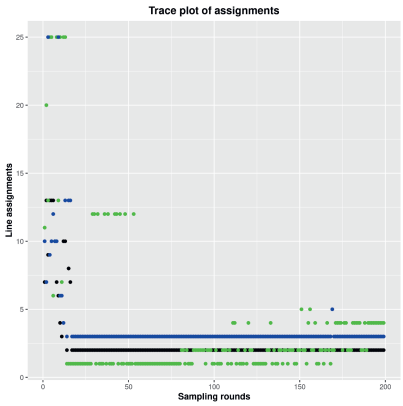


(d) Line segments are (incorrectly) chosen to be orthogonal to the lines.

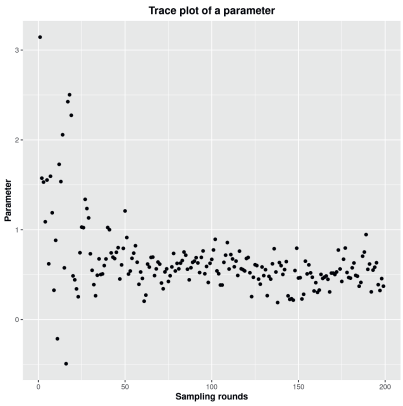
Figure 4.5: Bayesian point estimates of the sampling process with varying types of sampling errors. The descriptions indicate what type of sampling error is visualized per subfigure.

4.3.3 Trace Plots

To study the convergence of parameters in the infinite segment model, we use trace plots.



(a) This plot traces three points that are assigned to clusters (limited to around 30). Two of the points are assigned to one cluster. The other point to the cluster at the bottom. The plot only shows accepted assignments. The acceptance of a new assignment takes rarely hold.



(b) This plot traces a segment parameter belonging to point w_i. It exhibits exploratory behavior around a particular value (in this case 0.5). Compared to Figure 3.14 the variance is quite large.

Figure 4.6: Two examples of trace plots. Left: a trace plot of the assignment of points to cluster (it changes not so often). Right: a trace plot of one of the parameter values assigned to w_i.

4.4 Chapter Conclusions

From Chapter 3 we know that segment estimation is a much harder problem than line estimation. In this chapter we used an advanced method, namely Gibbs sampling with auxiliary variables to perform inference over an infinite set of line segments (Van Rossum et al., 2016c). The auxiliary variable Gibbs sampling method converges faster than the ordinary Metropolis-Hastings sampling algorithm by postulating multiple segments rather than only one.

This chapter contributes to answering our first research question.

> **RQ 1** How can we estimate the number of objects simultaneously with the fitting of these objects?

To estimate the number of objects simultaneously with the fitting of those objects, we have used a Bayesian method (as in the previous chapter). In this chapter, the prior and likelihood for the line segment model does not form a conjugate pair. Hence, different sampling methods had to be used to perform inference for the introduced Bayesian model.

However, the segment estimation problem remains a challenge for the inference method in this chapter. The target probability density has modes that each needs to be found and tend to be separated by very low probability regions. In Chapter 5 we will introduce new sampling methods that will cope with this challenge.

CHAPTER 5

TRIADIC SPLIT-MERGE SAMPLER

Contents This chapter introduces a new sampling method called the triadic split-merge sampler. The reason is that naive implementations of MCMC methods suffer from slow convergence in machine vision due to the complexity of the parameter space. Towards this blocked Gibbs and split-merge samplers have been developed that assign multiple data points to clusters at once. The triadic split-merge sampler improves on these samplers by defining split and merge steps between two and three clusters. This has two advantages. First, it reduces the asymmetry between the split and merge steps. Second, it is able to propose a new cluster that is composed out of data points from two different clusters. Both advantages speed up the convergence of the sampler on a line estimation problem.

Published in A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Triadic Split-Merge Sampler. *The 10th International Conference on Machine Vision*, ICMV 2017, Vienna, Austria, November 13-November 15, 2017.

Outline We introduce the class of split-merge samplers as part of the MCMC samplers (Section 5.1). A conventional split-merge sampler, labeled the dyadic split-merge sampler, is detailed (Section 5.2). A new split-merge sampler, the triadic split-merge sampler is introduced (Section 5.3). The results for inference over lines are compared between the conventional dyadic sampler and the new triadic sampler (Section 5.4). Finally, we provide the chapter conclusions and describe how we can further improve the inference procedure (Section 5.5). They will be the basis of the next chapter.

5.1 The Class of Split-Merge Samplers

In clustering models there is a hierarchical structure. At the lowest level there are individual data points. At a higher level the points are grouped into clusters. Split-merge samplers are

samplers that take into account such structure as already described in Section 2.2.6. Rather than moving data points one by one from an old to a new cluster, split-merge samplers can perform moves that operate on partitions of the dataset. For example, a cluster can be split into two clusters in one single move or two clusters can be merged into once cluster in another single move.

In Section 5.1.1 we describe split-merge samplers in a bit more detail than in Section 2.2.6. In Section 5.1.2 the Dirichlet Process prior is introduced in the context of split-merge samplers.

5.1.1 Split and Merge Moves

One of the first split-merge samplers has been defined for so-called point sources in nuclear imaging (Stawinski et al., 1998). This split-merge sampler proposes split and merge moves that are defined locally. For instance, two clusters that are close to each other are a candidate for a merge step into one cluster. By defining pairs of split and merge steps with the right probabilities, a properly balanced Metropolis-Hastings step an be performed (Section 2.2.6).

In the hierarchical models of lines and segments we have used a Dirichlet process as prior. Split-merge samplers have been defined with a Dirichlet prior (Dahl, 2003; Jain and Neal, 2004). These split-merge samplers operate on one or two clusters and are defined in the thesis as dyadic split-merge samplers (Section 5.2). The split step is different per sampler: (1) the simple random split procedure does not take into account the data distribution, (2) the sequentially allocated merge-split sequentially assigns data towards one of the two clusters that is the better fit.

Other examples of split-merge samplers are a sampler that uses sub-cluster splits (Chang and Fisher III, 2013) a sampler that uses data-driven jumps besides split-merge steps (Hughes et al., 2012), a sampler that uses data-driven jumps, split-merge steps, and operates on a hierarchical Dirichlet process (Bryant and Sudderth, 2012), and a sampler that generalizes split-merge steps to birth-death steps (with multiple clusters generated simultaneously) (Hughes and Sudderth, 2013). We will introduce a sampler that generalizes the dyadic sampler to moves over two and three clusters (Section 5.3).

5.1.2 Dirichlet Process Prior

Let us first reiterate the Dirichlet process. We will consider a Dirichlet process as a prior on the distribution over parameters G. The form of this model is:

$$\begin{aligned} y_i|\theta_i &\sim F(\theta_i) \\ \theta_i|G &\sim G \\ G &\sim DP(H,\alpha) \end{aligned} \tag{5.1}$$

The split and merge steps dictate the simultaneous assignment of observations y_i unto parameters θ_i. A split assigns a set of observations to a new parameter value. A merge combines multiple sets of observations with different parameter values into one set of observations with a single parameter value.

5.2 Conventional Split-Merge Sampler

The conventional split-merge sampler (see Jain and Neal, 2004) splits a single cluster into two clusters, and merges two clusters into a single cluster. Hence, this split-merge sampler operates on two clusters at each time step. Therefore we will call their algorithm a dyadic split-merge sampler in contrast with our approach (Van Rossum et al., 2017). Below we describe this dyadic split-merge sampler in pseudo-code (see Algorithm 11).

Algorithm 11 Dyadic split-merge sampler

procedure DYADIC SPLIT-MERGE SAMPLER(c) ▷ Accepts cluster assignments c of length N (besides Metropolis-Hastings acceptance factors $a(c', c)$ and a split procedure e.g. SIMPLERANDOMSPLIT) and returns a (potentially) updated cluster assignment vector c'.
  $i \sim U(\{1, \ldots, N\})$ ▷ Sample i discrete uniformly over cluster assignments.
  $j \sim U(\{1, \ldots, N\} \setminus i)$ ▷ Sample j from the discrete uniform distribution excluding i.
  $S_R = \{c_i, c_j\}$ ▷ Sampled clusters c_i, c_j.
  $S_I = \{c_x\}$ with $c_x \in S_R$ for $x \in \{1, \ldots, N\}$ ▷ All data in clusters c_i, c_j.
  $S_E = S \setminus S_R$ ▷ All data in clusters c_i, c_j excluding S_R.
  $N_S = \text{unique}(S_R)$
  if $N_S = 1$ **then** ▷ Case: i, j belong to the same cluster.
    $c_i^{(2)} = c_k$ with $c_k \notin \{c_1, \ldots, c_N\}$ ▷ Sample new cluster for $c_i^{(2)}$.
    $c_j^{(2)} = c_j^{(1)}$ ▷ Keep c_j the same.
    $c_e^{(2)} = \text{SPLITPROCEDURE}(S_E, c_i^{(2)}, c_j^{(2)})$ ▷ After $c_i^{(2)}, c_j^{(2)}$ assign S_E.
    for all $m \notin S_I$ **do**
      $c_m^{(2)} = c_m^{(1)}$ ▷ Data points in clusters other than c_i, c_j are not adjusted.
    end for
    $c' = \{c_i^{(2)}, c_j^{(2)}, c_e^{(2)}, c_m^{(2)}\}$
    $a = a_{split}(c', c)$ according to Eq. 5.3 ▷ MH acceptance for a split.
  else ▷ Case: i, j belong to different clusters $c_i \neq c_j$ ($N_S = 2$).
    for all $q \in S_I$ **do**
      $c_q^{(1)} = c_j^{(2)}$ ▷ Assign all data points in c_i and c_j to c_j.
    end for
    for all $m \notin S_I$ **do**
      $c_m^{(1)} = c_m^{(2)}$ ▷ Data points in clusters other than c_i, c_j are not adjusted.
    end for
    $c' = \{c_q^{(1)}, c_m^{(1)}\}$
    $a = a_{merge}(c', c)$ according to Eq. 5.10 ▷ MH acceptance for a merge.
  end if
  $u \sim U(0, 1)$ ▷ Sample u between 0 or 1 uniformly.
  if $a < u$ **then**
    $c' = c$ ▷ Reject c' by setting it to c
  end if
  return c', the (updated) cluster assignment vector: $c \rightarrow c'$.
end procedure

In Algorithm 11 the notation $c_i^{(2)}$ is used to signify that the cluster assignment c_i has 2 clusters under consideration. In the dyadic algorithm we could have used c_i^{merge} and c_i^{split}, however in the triadic algorithm (see Algorithm 14) with multiple split and merge operations the latter notation would become confusing.

Algorithm 12 Simple random split

1: **procedure** SIMPLERANDOMSPLIT(S, c_0, c_1) ▷ Accepts unassigned set S and cluster indices c_0, c_1, returns cluster assignment c'_m.
2: **for all** $m \in S$ **do**
3: $c'_m \sim Cat(c_0, c_1)$ with equiprobable $p(c_0) = p(c_1) = \frac{1}{2}$.
4: **end for**
5: **return** c'_m, the cluster assignment for S.
6: **end procedure**

The dyadic split-merge sampler in Algorithm 11 samples two distinct data items. If the data items belong to the same cluster a split step is attempted. If the data items belong to different clusters a merge step is attempted. The split procedure itself is the so-called simple random split (Algorithm 12) that assigns data items with the same probability to one of the parts of the split cluster without any consideration for a proper data fit.

5.2.1 Acceptance for the Split Step

The acceptance ratio contains the Metropolis ratio to step from c to c':

$$\frac{P(c')L(c'|y)}{P(c)L(c|y)}. \tag{5.2}$$

Additionally, the Hastings correction is applied because of the asymmetry of the proposal distribution in the form of $q(c|c')/q(c'|c)$:

$$a_{split}(c^{(2)}, c^{(1)}) = \min\left[1, \frac{q(c^{(1)}|c^{(2)})}{q(c^{(2)}|c^{(1)})} \frac{P(c^{(2)})}{P(c^{(1)})} \frac{L(c^{(2)}|y)}{L(c^{(1)}|y)}\right]. \tag{5.3}$$

The notation $c^{(2)}$ is used to indicate that the cluster index vector is referencing 2 unique clusters (in this case after the split step).

The prior distribution is represented by a Chinese Restaurant Process with concentration parameter α and no discount factor. Data not yet assigned is assigned (1) with probability $\alpha/(n+\alpha)$ to a new cluster and (2) with probability $n_c/(n+\alpha)$ to an existing cluster c. Here n is the total number of assigned data points, n_c is the number of data points assigned to cluster c. There are D clusters. Hence, the prior over clusters will be:

$$P(c) = \frac{\Gamma(\alpha)}{\Gamma(\alpha+n)} \alpha^D \prod_{c_l} \Gamma(n_{c_l}) = \alpha^D \frac{\prod_{c_l}(n_{c_l}-1)!}{\prod_{k=1}^{n}(\alpha+k-1)}. \tag{5.4}$$

In the prior distribution ratio before and after the split step many of the factors drop out. There is one factor α remaining and the number of data points in the split cluster is part

of the equation. There is no dependency on other clusters or the total number of data points. We can simplify the formula using the Beta function $B(a,b) = \Gamma(a)\Gamma(b)/\Gamma(a+b)$ with $\Gamma(x) = (x-1)!$ the Euler-Gamma function:

$$\frac{P(c^{(2)})}{P(c^{(1)})} = \alpha \frac{(n_{c_i^{(2)}}-1)!(n_{c_j^{(2)}}-1)!}{(n_{c_i^{(1)}}-1)!} = \alpha B(n_{c_i^{(2)}}, n_{c_j^{(2)}}). \tag{5.5}$$

The likelihood can be written as a product over all observations y_i or as a product over clusters with each cluster a product over its observations y_k:

$$L(c|y) = \prod_{c=1}^{D} \prod_{k:c_k=c} p(y_k|\phi). \tag{5.6}$$

Here we write $p(y_k|\theta_k)$ rather than assuming conjugacy between the likelihood $F(\theta_k)$ and the prior distribution $H(\theta_k)$ (see Dahl, 2005). In the case of conjugacy we can analytically calculate $\int F(\theta_k)dH(\theta_k)$ which speeds up inference, but which restricts our choice of likelihoods and priors (see Chapter 3).

With the above formula for the likelihood, we can calculate the likelihood ratio of two clusters versus a single cluster:

$$\frac{L(c^{(2)}|y)}{L(c^{(1)}|y)} = \frac{\prod_{k:c_k^{(2)}=c_i^{(2)}} p(y_k|\phi) \prod_{k:c_k^{(2)}=c_j^{(2)}} p(y_k|\phi)}{\prod_{k:c_k^{(1)}=c_i^{(1)}} p(y_k|\phi)}. \tag{5.7}$$

The split step determines the probability of a particular split. Algorithm 11 commences with picking two random points. These two points are henceforth are already assigned to distinct clusters. Only the remaining points have to be assigned (see Figure 5.1).

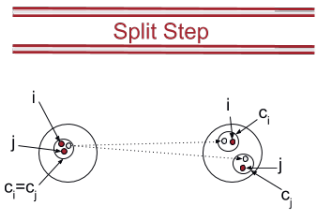


Figure 5.1: A split step. Points i and j are already assigned to separate clusters c_i and c_j. Each next point is assigned to one of the clusters with probability $\frac{1}{2}$, indicated by dotted arrows.

The remaining points are assigned with equal probability $\frac{1}{2}$ to $c_i^{(2)}$ and $c_j^{(2)}$:

$$q(c^{(2)}|c^{(1)}) = \left(\frac{1}{2}\right)^{-2+n_{c_i^{(2)}}+n_{c_j^{(2)}}} = \left(\frac{1}{2}\right)^{-2+n_{c_i^{(1)}}}. \tag{5.8}$$

The notation $n_{c_i^{(2)}}$ might seem complex, but it can be read as the number of points n in one of the clusters after the split. The split results in two clusters, indicated by the subscript (2). The cluster index is i. The probability of the reverse of the split operation is exactly 1. There is only one way in which a single cluster can be the starting state for a split operation. It must have had all points assigned to it. This means that the ratio with respect to the assignment of points over clusters in the split step becomes:

$$\frac{q(c^{(1)}|c^{(2)})}{q(c^{(2)}|c^{(1)})} = \frac{1}{\left(\frac{1}{2}\right)^{n_{c_i^{(2)}}+n_{c_j^{(2)}}-2}} = 2^{-2+n_{c_i^{(1)}}}. \tag{5.9}$$

Only basic identities are used and the fact that the number of data items does not change after a split, $n_{c_i^{(2)}} + n_{c_j^{(2)}} = n_{c_i^{(1)}}$. Note that the reverse split transition looks like a 'merging' operation. The merge step, however, is defined independently and its description can be found in the next section.

5.2.2 Acceptance for the Merge Step

Acceptance of a merge step consists of the same components as that of the split step.

$$a_{merge}(c^{(1)}, c^{(2)}) = \min\left[1, \frac{q(c^{(2)}|c^{(1)})}{q(c^{(1)}|c^{(2)})} \frac{P(c^{(1)})}{P(c^{(2)})} \frac{L(c^{(1)}|y)}{L(c^{(2)}|y)}\right]. \tag{5.10}$$

$$\frac{P(c^{(1)})}{P(c^{(2)})} = \alpha^{-1} \frac{(n_{c_i^{(1)}}-1)!}{(n_{c_i^{(2)}}-1)!(n_{c_i^{(2)}}-1)!} = \frac{1}{\alpha B(n_{c_i^{(2)}}, c_j^{(2)})}. \tag{5.11}$$

$$\frac{L(c^{(1)}|y)}{L(c^{(2)}|y)} = \frac{\prod_{k:c_k^{(1)}=c_i^{(1)}} p(y_k|\phi)}{\prod_{k:c_k^{(2)}=c_i^{(2)}} p(y_k|\phi) \prod_{k:c_k^{(2)}=c_j^{(2)}} p(y_k|\phi)}. \tag{5.12}$$

$$\frac{q(c^{(2)}|c^{(1)})}{q(c^{(1)}|c^{(2)})} = \left(\frac{1}{2}\right)^{-2+n_{c_i^{(1)}}} = 2^{2-n_{c_i^{(1)}}}. \tag{5.13}$$

The ratios of the merge step are the inverse of the ratios of the split step. That is, Eq. 5.11 is the inverse of Eq. 5.5, Eq. 5.12 is the inverse of Eq. 5.7, and Eq. 5.13 is the inverse of Eq. 5.9.

5.2.3 Sequentially-Allocated Merge-Split Sampler

A variant on the conventional split-merge sampler is the sequentially allocated merge-split (SAMS) sampler[1] (Dahl, 2003). The simple random split procedure of Algorithm 12 is replaced by a procedure that sequentially assigns observations to clusters rather than splitting the data random uniformly over the split clusters.

[1] In the naming of split-merge or merge-split samplers, the order of merge split does not bear any significance.

Algorithm 13 Sequentially Allocated Merge-Split

procedure SAMS(S, c_0, c_1) ▷ Accepts unassigned set S, cluster indices c_i, and $p(y_k|\theta_{c_i})$ with $i = 0, 1$, returns cluster assignment c'_m.
  $T =$ random_shuffle(S)
  for all $m \in T$ **do**
    $p(c_m = c_0|c_0, c_1, \theta_{c_0}, \theta_{c_1}) = \frac{N_0 p(y_k|\theta_0)}{N_0 p(y_k|\theta_0)+N_1 p(y_k|\theta_1)}$
    $p(c_m = c_1|c_0, c_1, \theta_{c_0}, \theta_{c_1}) = 1 - p(c_m = c_0|c_0, c_1, \theta_{c_0}, \theta_{c_1})$
    $c'_m \sim p(c_m|c_0, c_1, \theta_{c_0}, \theta_{c_1})$
  end for
  return c'_m, the cluster assignment for S.
end procedure

In contrast to the simple random split, observations y_k are used in the SAMS to obtain cluster assignments that correspond with the data rather than cluster assignments independent of the data.

5.3 Triadic Split-Merge Sampler

The triadic split-merge sampler that we introduce (Van Rossum et al., 2017) uses up to three clusters for a split or merge step (Fig. 5.2).

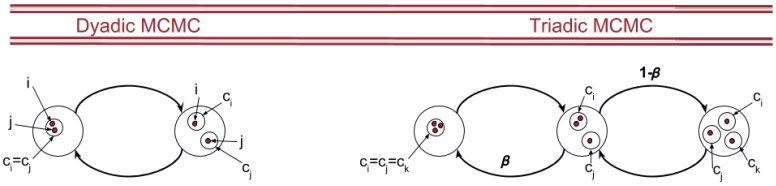


Figure 5.2: Right: dyadic MCMC picks two data items i, j random uniformly. If both are in the same cluster a split towards two clusters is attempted. If both are in distinct clusters a merge towards one cluster is attempted. Left: triadic MCMC picks three data items i, j, k random uniformly. If all three are in the same cluster a split towards two clusters is attempted. If the three items are in two clusters either a split into three (with probability $1-\beta$) or a merge into a single cluster (with probability β) is attempted. If the three data items are in three distinct clusters a merge is attempted. There are no direct transitions from a single cluster to three clusters or the other way around.

The intuition behind the triadic split-merge sampler is twofold:

- In the dyadic sampler there is a large asymmetry between split and merge steps. There is only one way in which two clusters can be merged into one single cluster, while there are many ways in which one single cluster can be split into two clusters. This asymmetry is reduced by transitioning between two and three clusters. This is a straightforward improvement in balancing split and merge steps (for alternatives, see Wang and Russell (2015)).

- In practical optimization problems it might be useful to form a third cluster out of subsets of two other clusters. The dyadic MCMC sampler requires intermediate steps in which (1) one of these clusters is split into two, (2) the other is split into two, and (3) the two new clusters are merged. This means that (a) mixing and hence convergence will be slow and (b) the intermediate steps might have very low probability and function as an unnecessary barrier between high probable states.

The algorithm is detailed in Algorithm 14. Compare with Algorithm 11. It starts with sampling three distinct points i, j, and k. These points can originate from one, two, or three distinct clusters with non-unique indices c_i, c_j, and c_k. Depending on the number of distinct clusters N_S, either a dyadic or triadic step is performed. If all points belong to the same cluster, $N_S = 1$, a split step into two clusters is attempted. If the points belong to two clusters $N_S = 2$, with probability β a merge step into one cluster is tried, with probability $1-\beta$ a triadic split into three clusters will be considered. If the points belong to three distinct clusters, $N_S = 3$, a triadic merge step into two clusters will be attempted. All steps will be accepted or rejected according to the Metropolis-Hastings probability ratios in the following section. Note that there are no steps with which a single cluster is immediately split into three, nor is there is a step that merges three clusters into one.

Sampling random uniformly for three unique items is implemented through a random shuffle algorithm, in particular the modern version of the Fisher-Yates shuffle introduced by Durstenfeld (1964) and picking the first three items.

The Metropolis-Hastings probabilities for the triadic split and merge steps are calculated in the following Sections 5.3.1 and 5.3.2.

5.3.1 Acceptance for the Split Step

In the triadic split-merge sampler there are two splitting steps. It is possible to split according to the dyadic split-merge sampler. However, given two clusters there are (split) jumps to three states as well as (merge) jumps to single states again. To account for this asymmetry another Hastings correction is applied to establish detailed balance.

$$a_{split}(c^{(2)}, c^{(1)}) = \min\left[1, \frac{r(c^{(1)}|c^{(2)})}{r(c^{(2)}|c^{(1)})} \frac{q(c^{(1)}|c^{(2)})}{q(c^{(2)}|c^{(1)})} \frac{P(c^{(2)})}{P(c^{(1)})} \frac{L(c^{(2)}|y)}{L(c^{(1)}|y)}\right]. \tag{5.14}$$

Here we have one additional term compared to the split step from one cluster to two clusters:

$$\frac{r(c^{(1)}|c^{(2)})}{r(c^{(2)}|c^{(1)})} = \frac{\beta}{1}. \tag{5.15}$$

The parameter β is free to control, as long as $0 < \beta < 1$ (to maintain ergodicity). The transition from two states to three states is another split step:

$$a_{split}(c^{(3)}, c^{(2)}) = \min\left[1, \frac{r(c^{(2)}|c^{(3)})}{r(c^{(3)}|c^{(2)})} \frac{q(c^{(2)}|c^{(3)})}{q(c^{(3)}|c^{(2)})} \frac{P(c^{(3)})}{P(c^{(2)})} \frac{L(c^{(3)}|y)}{L(c^{(2)}|y)}\right]. \tag{5.16}$$

Algorithm 14 Triadic split-merge sampler

procedure TRIADIC SPLIT-MERGE SAMPLER(c) ▷ Accepts cluster assignments c of length N (besides Metropolis-Hastings acceptance factors $a(c', c)$ and a split procedure) and returns a (potentially) updated cluster assignment vector c'.
  $i \sim U(\{1, \ldots, N\})$ ▷ Sample i discrete uniformly over cluster assignments.
  $j \sim U(\{1, \ldots, N\} \setminus i)$ ▷ Sample j from the discrete uniform distribution excluding i.
  $k \sim U(\{1, \ldots, N\} \setminus \{i, j\})$ ▷ Sample k from the discrete uniform distribution excluding $\{i, j\}$.
  $S_R = \{c_i, c_j, c_k\}$ ▷ Sampled clusters c_i, c_j, c_k.
  $S_I = \{c_x\}$ with $c_x \in S_R$ for $x \in \{1, \ldots, N\}$ ▷ All data in clusters c_i, c_j, c_k.
  $S_E = S_I \setminus S_R$ ▷ All data in clusters c_i, c_j, c_k excluding S_R.
  $N_S = \text{unique}(S_R)$
  $u \sim U(0, 1)$ ▷ Sample u between 0 or 1 uniformly.
  if $N_S = 1$ **then** ▷ Case: i, j, k belong to the same cluster.
    return $c' = $ DYADIC SPLIT-MERGE SAMPER(c)
  else if $N_S = 2$ **and** $u < \beta$ **then** ▷ Case: a cluster with one item and one with two items and $u < \beta$.
    return $c' = $ DYADIC SPLIT-MERGE SAMPER(c)
  else if $N_S = 2$ **and** $u \geq \beta$ **then** ▷ Case: a cluster with one item and one with two items and $u \geq \beta$.
    $c_i^{(3)} = c_k$ with $c_k \notin \{c_1, \ldots, c_N\}$ ▷ Sample new cluster for $c_i^{(3)}$.
    $c_j^{(3)} = c_j^{(2)}$ ▷ Keep c_j the same.
    $c_e^{(3)} = $ SPLITPROCEDURE$(S_E, c_i^{(3)}, c_j^{(3)})$ ▷ After $c_i^{(3)}, c_j^{(3)}$ assign S_E.
    for all $m \notin S_I$ **do**
      $c_m^{(3)} = c_m^{(2)}$ ▷ Data points in clusters other than c_i, c_j are not adjusted.
    end for
    $c' = \{c_i^{(3)}, c_j^{(3)}, c_e^{(3)}, c_m^{(3)}\}$
    $a = a_{split}(c', c)$ according to Eq. 5.14 ▷ MH acceptance for a split.
  else ▷ Case: i, j, k belong to thee different clusters $c_i \neq c_j \neq c_k$ ($N_S = 3$).
    $S_L = S_I \setminus \{c_i^{(3)}, c_j^{(3)}\}$ ▷ Data in clusters c_i, c_j, c_k except for i and j itself.
    $\{c_i^{(2)}, c_j^{(2)}\} = \text{SAMS}(S_L, c_i^{(3)}, c_j^{(3)})$ ▷ Assign data points in c_i, c_j, c_k to c_i, c_j.
    for all $m \notin S_L$ **do**
      $c_m^{(2)} = c_m^{(3)}$ ▷ Data points in clusters other than S_L are not adjusted.
    end for
    $c' = \{c_i^{(2)}, c_j^{(2)}, c_m^{(2)}\}$
    $a = a_{merge}(c', c)$ according to Eq. 5.21 ▷ MH acceptance for a merge.
  end if
  $u \sim U(0, 1)$ ▷ Sample u between 0 or 1 uniformly.
  if $a < u$ **then**
    $c' = c$ ▷ Reject c' by setting it to c
  end if
  return c', the (updated) cluster assignment vector: $c \to c'$.
end procedure

The fraction with r reads as follows:

$$\frac{r(c^{(2)}|c^{(3)})}{r(c^{(3)}|c^{(2)})} = \frac{1}{1-\beta}. \tag{5.17}$$

The fraction with q uses the total number of data points n_c in the clusters:

$$\frac{q(c^{(2)}|c^{(3)})}{q(c^{(3)}|c^{(2)})} = \frac{\left(\frac{1}{2}\right)^{n_c-2}}{\left(\frac{1}{3}\right)^{n_c-3}} = \left(3^{n_c-3}\right)\left(2^{2-n_c}\right) = \left(\frac{3}{2}\right)^{n_c} \frac{2^2}{3^3}. \tag{5.18}$$

To move from 2 clusters to 3 clusters the probability is a $1/3$ for each cluster index in vector c (except for the three data items already selected randomly, hence $n_c - 3$). To move back, the probability is a $1/2$ and there are only two data items randomly assigned beforehand. The fraction with P uses the number of data points in each of the clusters before and after the step:

$$\frac{P(c^{(3)})}{P(c^{(2)})} = \alpha \frac{(n_{c_i^{(3)}} - 1)!(n_{c_j^{(3)}} - 1)!(n_{c_k^{(3)}} - 1)!}{(n_{c_i^{(2)}} - 1)!(n_{c_j^{(2)}} - 1)!} = \alpha \frac{B(n_{c_i^3}, n_{c_j^3}, n_{c_k^3})}{B(n_{c_i^2}, n_{c_j^2})}. \tag{5.19}$$

Here we introduced a generalized Beta function $B(a,b,c) = \Gamma(a)\Gamma(b)\Gamma(c)/\Gamma(a+b+c)$ with $\Gamma(x) = (x-1)!$ the Gamma function. The likelihood ratio becomes:

$$\frac{L(c^{(3)}|y)}{L(c^{(2)}|y)} = \frac{\prod_{m:c_m^{(3)}=c_i^{(3)}} p(y_m|\phi) \prod_{m:c_m^{(3)}=c_j^{(3)}} p(y_m|\phi) \prod_{m:c_m^{(3)}=c_k^{(3)}} p(y_m|\phi)}{\prod_{m:c_m^{(2)}=c_i^{(2)}} p(y_m|\phi) \prod_{m:c_m^{(2)}=c_j^{(2)}} p(y_m|\phi)}. \tag{5.20}$$

5.3.2 Acceptance for the Merge Step

The merge step from two to one cluster is analogous to the split step:

$$a_{merge}(c^{(1)}, c^{(2)}) = \min\left[1, \frac{r(c^{(2)}|c^{(1)})}{r(c^{(1)}|c^{(2)})} \frac{q(c^{(2)}|c^{(1)})}{q(c^{(1)}|c^{(2)})} \frac{P(c^{(1)})}{P(c^{(2)})} \frac{L(c^{(1)}|y)}{L(c^{(2)}|y)}\right]. \tag{5.21}$$

The merge step from three clusters to two clusters is:

$$a_{merge}(c^{(2)}, c^{(3)}) = \min\left[1, \frac{r(c^{(3)}|c^{(2)})}{r(c^{(2)}|c^{(3)})} \frac{q(c^{(3)}|c^{(2)})}{q(c^{(2)}|c^{(3)})} \frac{P(c^{(2)})}{P(c^{(3)})} \frac{L(c^{(2)}|y)}{L(c^{(3)}|y)}\right]. \tag{5.22}$$

Note that all the fractions in Eq. 5.22 are the inverse of the fractions in Eq. 5.16. Inverting Eq. 5.17–5.20 will be left to the reader.

One additional issue we have to consider. When merging three clusters into two we can (1) distribute the data over all three clusters or (2) alternatively, keep the data in two clusters assigned to these clusters and only distribute the data in the third cluster over the other two clusters. The second and alternative option however would introduce unnecessary asymmetry with the merge step. In other words, Eq. 5.23 is not the inverse of Eq. 5.18. In contrast, the equation is similar to splitting one cluster across two as in Eq. 5.9:

$$\frac{q_{alt}(c^{(3)}|c^{(2)})}{q_{alt}(c^{(2)}|c^{(3)})} = 2^{-2+n_c}. \tag{5.23}$$

Hence the first option is entertained and the q-fraction is exactly the inverse of Eq. 5.18.

A second issue has to be considered, namely the inclusion or exclusion of direct operations between a single cluster and three clusters. This is because factors such as

$$\frac{P(c^{(3)})}{P(c^{(1)})} = \alpha^2 \frac{(n_{c_i^{(3)}} - 1)!(n_{c_j^{(3)}} - 1)!(n_{c_k^{(3)}} - 1)!}{(n_{c_i^{(1)}} - 1)!}, \tag{5.24}$$

become very small and although compensated by a large q fraction, remain further away from an acceptance factor of 1. Note that by the ability to split a single cluster into two and then into three, there is no ergodic argument to introduce also the immediate step.

5.4 Results

The problem we use to test our sampler is a well-known problem in computer vision, namely that of the inference of line parameters (slope and intercept) given data points. Rather than ordinary linear regression, in computer vision there is a mixture of lines that have to be estimated. Moreover, the number of lines is not known in advance. To solve this problem we use the Dirichlet process mixture (Eq. 5.1) with a normal distribution $N(0, \sigma_0^2)$ to generate the line parameters and a likelihood function that defines points to be uniformly distributed across a line of length 20 and deviating from the line according to a normal distribution $N(0, \sigma_1^2)$.

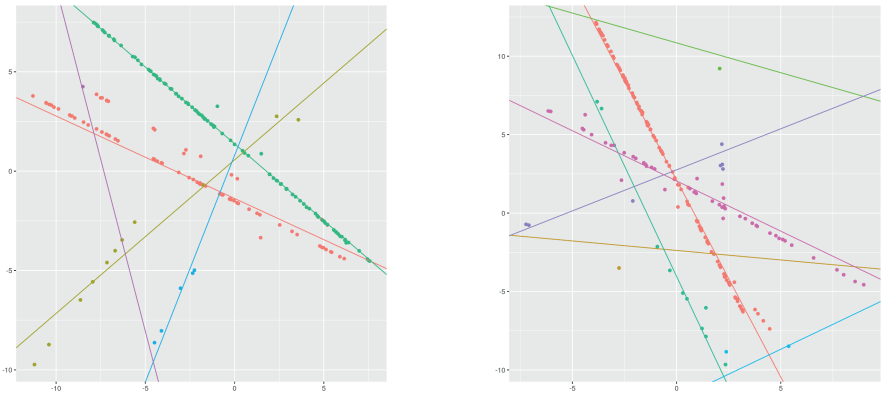


Figure 5.3: Two examples of fitting a mixture of lines to data items scattered over a two-dimensional space. The lines drawn are inferred using the triadic sampler. The lines are not the ground truth, but are meant to demonstrate the typical errors made by fitting methods. Note, for example, that there are mistakes in both the assignment of points to lines as well as the line parameters (slope and intercept). Left: In example 1 two lines with similar slope are seen as the same line. Right: In example 2 points on one vertical line are assigned to multiple lines. In example 1 and 2 slopes are not always through the points.

In Section 5.4.1 the triadic sampler's implementation is discussed. In Section 5.4.2 the triadic sampler is compared with the conventional Jain-Neal (dyadic) sampler and the auxiliary variable sampler of the previous chapter.

5.4.1 Implementation

The sampler is open-source[2] and is implemented in C++ which means that (a) it is computationally fast, (b) it can be run on embedded devices (if a cross-compiler is available and the Eigen3 library is ported). Note, that due to the fact that the simulator uses a lot of random numbers the system should use a modern compiler (g++-6 or newer) and should have enough entropy available[3]. Rather than a random scan, the implementation uses a fixed scan as advocated in the literature (MacEachern, 2007).

To speed up the sampler most calculations are done in log-space. Consider $v = u + 1$. The ratio with probabilities (Eq. 5.5 and 5.19) becomes:

$$\log \frac{P(c^{(v)})}{P(c^{(u)})} = \log(\alpha) + \sum_i \log \Gamma(n_{c_i^{(v)}}) - \sum_i \log \Gamma(n_{c_i^{(u)}}). \tag{5.25}$$

The fraction with $q(\cdot)$ (Eq. 5.9 and 5.18) becomes:

$$\log \frac{q(c^{(v-1)}|c^{(v)})}{q(c^{(v)}|c^{(v-1)})} = (v - n_c - 1)\log(v-1) - (v - n_c)\log(v). \tag{5.26}$$

The fraction with r becomes, for example, (Eq. 5.17):

$$\log \frac{r(c^{(2)}|c^{(3)})}{r(c^{(3)}|c^{(2)})} = -\log(1-\beta). \tag{5.27}$$

The log-probability to calculate the likelihood given by a multivariate Normal distribution is well-known.

5.4.2 Comparison

The Triadic sampler using SAMS is compared with the Jain-Neal Dyadic sampler using SAMS and an auxiliary variable sampler with $m = 3$ (see algorithm 8 in Neal (2000)).

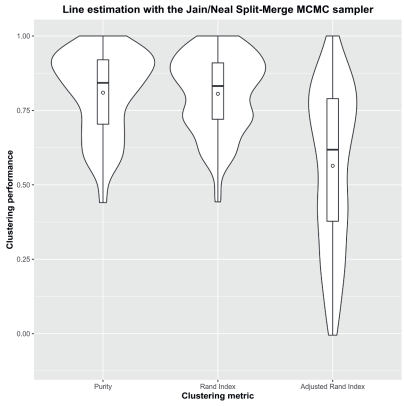
In Table 5.1 the line estimation problem is compared for the dyadic sampler, an auxiliary variables sampler, and the proposed triadic sampler. The simulation is run with $\beta = 0.1$ so that a significant number of steps are tried between two and three clusters (rather than only between one and two clusters).

In Figures 5.4a to 5.4c the different metrics are visualized in the form of violin plots.

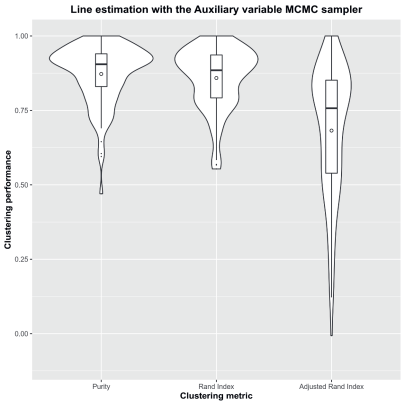
[2] Code can be found at `https://code.annevanrossum.nl/noparama`.
[3] On Linux this can be checked in /proc/sys/kernel/random/entropy_avail.

Table 5.1: The purity, rand index, and adjusted rand index establishing the quality of the clustering method. The closer the values to one, the better the method performed. The purity metric assigns high values to clusters that do not have data points from other clusters (but does not penalize the number of clusters). The rand index index computes similarity between clusters taking false negatives and false positives into account. The adjusted rand index accounts for chance. The adjusted rand index is most useful in our comparison.

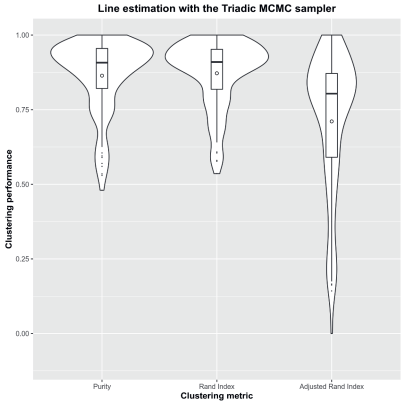
Method	Purity	Rand Index	Adjusted Rand Index
Dyadic sampler	0.80960	0.80580	0.56382
Auxiliary variables	0.87235	0.85879	0.68224
Triadic sampler	0.86405	0.87188	0.71067



(a) Line estimation with the dyadic split-merge sampler.



(b) Line estimation with the auxiliary variable sampler.



(c) Line estimation with the triadic split-merge sampler (our inference method). The values are all shifted up towards one.

Figure 5.4: The different inference methods for line estimation compared. The same results as in Table 5.1, but visualized in a violin plot. The distribution over metric values are displayed in a vertical fashion.

The improvement in clustering is especially visible with the adjusted rand index.

5.5 Chapter Conclusions

A new split-merge sampler has been introduced, implemented, and applied to the computer vision problem of line estimation. The sampler outperforms existing samplers, such as the ordinary (dyadic) split-merge sampler (Jain and Neal, 2004) and auxiliary variable sampler (Neal, 2000).

The triadic split-merge sampler has been used with likelihood functions that correspond to line fitting. It therefore estimates the number of lines and simultanously performs line fitting. Moreover, the sampler is optimized to reassign points from three lines to two lines and the other way around. This means a hypothesized third line can be composed at once from two existing lines. These triadic steps accelerate the inference process as shown in Section 5.4.

This chapter answers our second research question.

> **RQ 2** How can we optimize inference over both the number of objects and fitting of those objects in the robotic vision domain?

The triadic sampler optimizes the inference process by using spatial properties inherent to the robotic vision domain. To gain some intuition for this consider the following simplified line recognition problem. There are three lines: two vertical and one horizontal line. The horizontal line intersects both vertical lines. At some step in the inference process the vertical lines have been assigned points. The horizontal line has not been recognized yet. Now it would make sense to suggest a step where points currently assigned to both vertical lines will be combined to propose a horizontal line. The triadic sampler uses such steps. This benefits robotic vision problems where this spatial property of "object intersection" happens regularly.

Although the proposed split-merge sampler is able to mix considerably faster through a mixture model, it does not use global jumps directly based on the data. It is reasonable to suggest that MCMC methods benefit from combining the local jumps with global jumps, for example by a mixture of the local Metropolis-Hastings sampler with a Metropolized independence sampler (Jampani et al., 2015).

However, if the data is used to introduce adaptations to the sampler in this way, there are other data-driven methods that might improve performance considerably (Barbu and Zhu, 2005; Hughes et al., 2012). One of the data-driven methods that has caught considerable attention in the literature is a part of machine learning now known as deep learning methods (LeCun et al., 2015; Schmidhuber, 2015). We will introduce such methods in chapter Chapter 6.

CHAPTER 6

DEEP LEARNING OF POINT CLOUDS

Contents In the preceding chapters we have used sampling (MCMC methods) to perform inference. We extend here our point cloud datasets in 2D to much larger point cloud datasets in 3D. This requires a speed up in our inference methods.

Outline We introduce deep learning, and in particular variational autoencoders, ordinary autoencoders, sparse autoencoders, and convolutional autoencoders (Section 6.2). We show how they perform well on the MNIST dataset. We also show they do not perform well on the task of reconstructing 2D lines. We then introduce an autoencoder based on a model known in the literature as PointNet (Section 6.3). This autoencoder uses earth mover's distance (EMD) to reconstruct dense point clouds of single objects. In contrast, our dataset contains multiple objects that are sparse, such as squares and cubes. We show that the autoencoder does not learn a proper latent representation for those objects. We introduce two new metrics, the shifted earth mover's distance (SEMD) and the partitioning earth mover's distance (PEMD), to be used for datasets with objects on unknown positions or datasets with multiple objects in a single sample (Section 6.4). We test the new EMD on our dataset with multiple 3D cubes on different locations (Section 6.5). Finally, we provide the chapter conclusions and we describe some ways to improve structured autoencoders (Section 6.6).

6.1 Data-driven Methods

The previous chapter concluded with a suggestion to look into data-driven methods to accelerate inference even further than with the models described until then. One particular way in which we incorporate knowledge about the data we operate on, is by introducing nonlinear functions before we perform inference. These nonlinear functions we adapt to the data

by a learning process. A neural network can represent such a function. If a stack of neural networks is used, this becomes the field of deep learning (LeCun et al., 2015; Schmidhuber, 2015).

An autoencoder consists of two of those deep neural networks that each contains multiple layers. The encoder has layers of decreasing size and maps to the code, a layer of latent variables. The decoder has layers of increasing size and maps from the code to the same dimension as the input. A loss function can be used to quantity the difference between the input of the autoencoder and its output. The calculated loss is used to adjust the weights in the neural networks through error propagation.

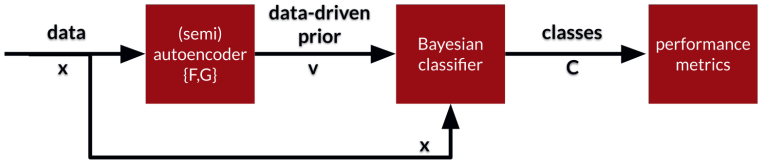


Figure 6.1: Left: the data x is used to train a (semi)autoencoder. Middle: the results of the autoencoder are used to create a data-driven prior for a Bayesian classifier. Right: the performance is measured with performance metrics like purity, rand index, as in previous chapters.

In Fig. 6.1 the autoencoder is embedded in a larger architecture The first block depicts an autoencoder that can efficiently represent 3D point cloud data. The second gets information from the autoencoder and uses this as a data-driven prior to perform classification. Typically, it is a Bayesian classifier as encountered in the previous chapters. The third block defines the performance of the autoencoder-classifier tandem.

In Section 6.2 we describe different types of autoencoders. In Section 6.3 we describe the earth mover's distance (EMD), a loss that can be used for the robotic vision domain of point clouds. We show the results of using this loss on the robotic vision task of fitting 3D cubes. We also visualize the latent representation of the autoencoder using this loss. In Section 6.4 we introduce a semi-autoencoder and introduce two generalizations of the EMD, the shifted earth mover's distance (SEMD) and the partitioning earth mover's distance (PEMD). In Section 6.5 we use the PEMD in combination with the triadic sampler of the previous chapter to perform inference over point clouds consisting of 3D cubes. We provide the chapter conclusions in Section 6.6.

6.2 Autoencoders

We introduce four autoencoders: a variational autoencoder (Section 6.2.1), an ordinary autoencoder (Section 6.2.2), a sparse autoencoder (Section 6.2.3), and a convolutional autoencoder (Section 6.2.4). We show how they perform on the MNIST dataset. We also show how they perform on a second dataset used in the previous chapter with 2D lines made out of 2D points. We will refer to the latter dataset as Lines100.

6.2.1 Variational Autoencoder

Variational autoencoders (Kingma and Welling, 2014; Rezende et al., 2014) are ordinary autoencoders with additional constraints on the latent variables. The latent variables in autoencoder parlance are called the code. In a variational autoencoder the latent variables are forced to approximately describe a standard Normal (or unit Gaussian) distribution. The autoencoder is trained using a loss function that is composed out of (1) a generative loss, a mean squared error that measures how accurately the network reconstructs its input, and (2) a latent loss, a KL-divergence that measures how closely the latent variables match a unit Gaussian. This loss is summed over all samples (and reconstructions), $L = \sum_i l_i(F,G)$.

$$l_i(F,G) = -\mathbb{E}_{h\sim q_F(h|x_i)}[\log(p_G(x_i|h)] + \mathbb{KL}(q_F(h|x_i)||p(h)) \tag{6.1}$$

To optimize the KL divergence a reparameterization trick is applied. The encoder does not generate a vector with real values, but generates a vector with means and standard deviations instead.

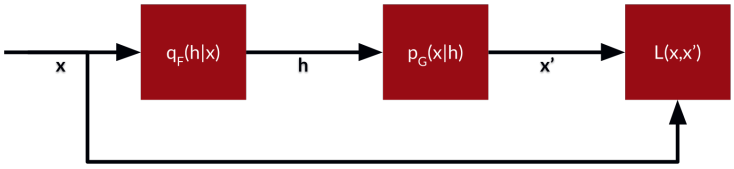


Figure 6.2: Left: $q_F(h|x)$ maps the data x to (hidden) random variables h. Middle: $p_G(x|h)$ maps the hidden random variables to reconstructed data x'. Right: $L(x,x')$ measures the similarity between x and x'.

The results are presented in the following manner. First, we visually inspect the reconstruction of the items in the dataset. Second, the test samples are encoded into the latent variable representation. The latent variables are then presented in a 2D scatterplot. Third, there is a sweep over the latent variable values to generate digits. The second and third presentations are especially useful if the encoder has only two latent variables. In that case the presentation in a 2D scatterplot does not require a dimensionality reduction step. The sweep over only two latent variables is also very easy to represent in 2D.

The MNIST digits are reconstructed by a variational autoencoder as shown in Fig. 6.3.

Figure 6.3: Reconstruction of MNIST data by a variational autoencoder. Top: the input, images with single hand-drawn digits. Bottom: the output, the images reconstructed by the autoencoder. During the training process the network weights are adjusted precisely so that the reconstruction loss between the output and the input is minimized. The variational autoencoder minimizes at the same time also the KL-divergence between the latent variables and a prior.

The scatterplot of the latent variable representation of the test set. It can be seen that similar digits are mapped to similar values in the latent space.

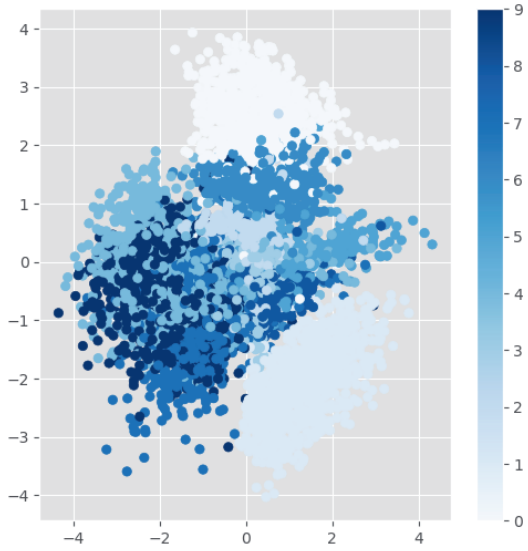


Figure 6.4: Scatterplot of latent variable representations of test samples in a variational autoencoder. There are two latent variables. The values of one latent variable are on the horizontal axis. The values of the other latent variables are on the vertical axis. The digits of the MNIST task are plotted with different color shades. For example, the digit zero is represented by values of around 0 of the "horizontal" latent variable and values around 3 of the "vertical" latent variable. Those are the whitest dots. The variables do not necessarily have (easily identifiable) semantics and are therefore not labeled.

Note that not every digit occupies the same amount of space in the latent variable layer. The amount of space emerges from the learning process.

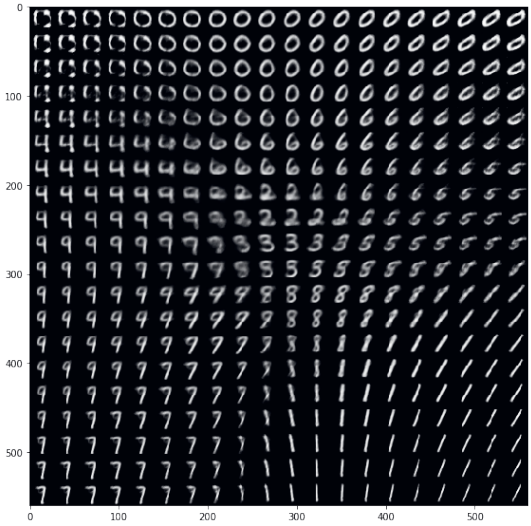


Figure 6.5: Latent variable sweep of test samples in a variational autoencoder. The two latent variables are both given values and the decoder calculates the output. This output is depicted as an image (a reconstructed "digit"). Here you see 20x20 images reconstructed starting from values at the top-left of around $(0,0)$ to the bottom-right of around $(560,560)$. It is clearly visible that if the values of the latent variables change only slightly, that the reconstruction is also changed only slightly.

6.2.2 Ordinary Autoencoder

An "ordinary" autoencoder (Rumelhart et al., 1986) has been trained with a latent variable layer of 32 nodes (rather than 2 as in the variational autoencoder above). It only minimize the reconstruction error and has no constraints[1] on the latent layers. We show the representation of the ordinary autoencoder after that of the variational autoencoder, so we can see how the representation is less structured.

$$l_i(F,G) = -\mathbb{E}_{h\sim q_F(h|x_i)}[\log p_G(x_i|h)] \tag{6.2}$$

The reconstruction is similar to that of the variational autoencoder if we just use visual inspection (see Fig. 6.6). This means that the ordinary autoencoder learns to reconstruct the digits just as the variational autoencoder.

[1]The size of the layer can be seen as a constraint, but it is not yet a regularization technique.

Figure 6.6: Reconstruction of MNIST data by an ordinary autoencoder.

The difference between the ordinary and the variational autoencoder shows when we start to study the latent representation. If we use a scatterplot for two of the latent variable nodes, there is not much structure to observe (see Fig. 6.7).

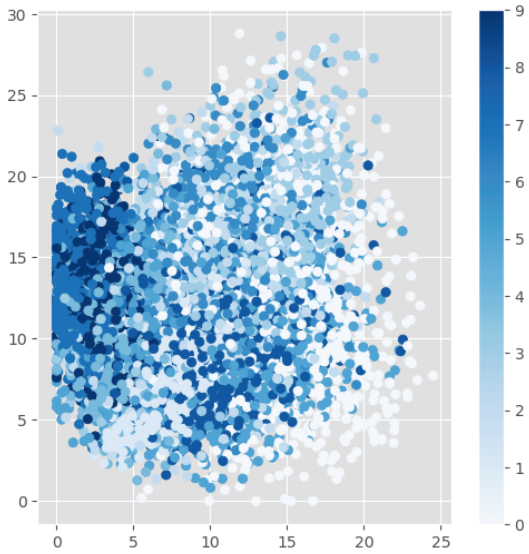


Figure 6.7: Scatterplot of latent variable representations of test samples in a ordinary autoencoder. The two axes represent only two nodes of in total 32 nodes in the latent layer. In contrast to the variational autoencoder (Fig. 6.4) there are no easily distinguishable clusters representing particular digits.

We might perform dimensionality reduction and for example use t-SNE (Van Der Maaten et al., 2009) to map to a 2D space. However, this is much more indirect than in the case that there are only two latent variables. If there is still no structure observed, it might be just an artifact of how t-SNE performs dimensionality reduction (not indicating the quality of the latent variable representation).

Let us use the ordinary autoencoder to reconstruct point clouds. In this case the reconstruction of 2D lines.

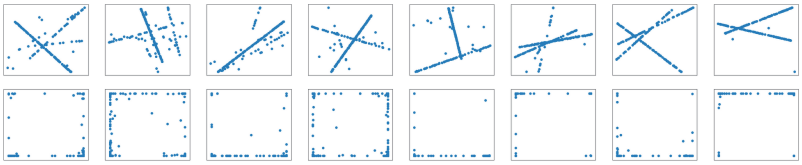


Figure 6.8: Top: the input, multiple lines per sample. Bottom: the output, the reconstructed 2D point cloud by the autoencoder. Clearly, the reconstruction fails for the ordinary autoencoder. In particular, the autoencoder seems not to be able to reconstruct the correlations between the x and y coordinates.

In Fig. 6.8 the reconstruction of the 2D lines are shown. The ordinary autoencoder is not able to reconstruct the lines.

6.2.3 Sparse Autoencoder

A sparse autoencoder (Hosseini-Asl et al., 2015) is similar to the ordinary autoencoder, but enforces sparsity through an "activity regularizer":

$$l_i(F,G) = -\mathbb{E}_{h\sim q_F(h|x_i)}[\log p_G(x_i|h)] + \mathbb{KL}\left(q_F(h|x_i)||p(h)\right) + J(h). \tag{6.3}$$

The activation in the hidden layer can be regularized by an L1 loss function, $J(h) = \lambda \sum_j |h_j|$, or by enforcing the average activation of a node to be close to zero (averaged over all m training samples). This can be achieved for example by setting $J(h) = \beta \sum_j \mathbb{KL}(\rho||\hat{\rho}_j)$ with $\hat{\rho}_j = \frac{1}{m}\sum_r^m [h_j(x_r)]$. The sharpness of the reconstructions can be tuned by the factors λ or β or both if both regularizers are used at the same time.

The reconstruction of digits (see Fig. 6.9) by a sparse autoencoder is a bit less sharp than reconstruction by an ordinary autoencoder (compare Fig. 6.6).

Figure 6.9: Reconstruction of MNIST data by a sparse autoencoder. The results are a bit less sharp than that of the ordinary autoencoder.

The scatterplot in Fig. 6.10 shows that only a few of the latent variables have non-zero values. The sparsity in the latent variables layer is achieved.

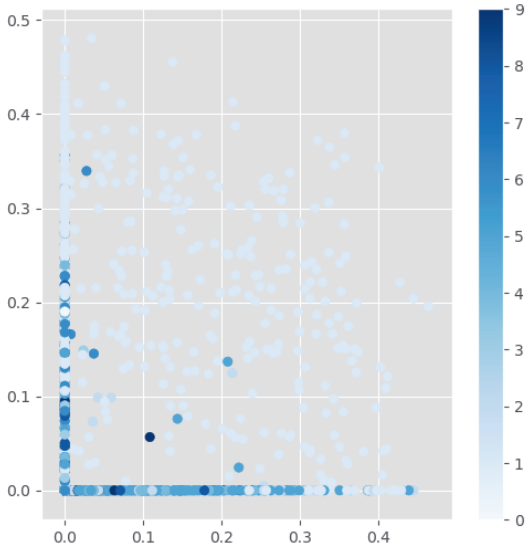


Figure 6.10: Scatterplot of latent representations in a sparse autoencoder. The two axes represent only two nodes in the latent layer. In contrast to the ordinary autoencoder (Fig. 6.7) is clear that inputs are mostly represented by either one or the other latent variable with a value that is nonnegative.

For most digits at least one of the first two nodes in the latent layer are zero (Fig. 6.10). We do not show the reconstruction of lines. The sparse autoencoder is failing in a similar manner as the ordinary autoencoder (Fig. 6.8).

6.2.4 Convolutional Autoencoder

The results for the dataset with multiple lines (Lines100) was shown for the ordinary autoencoder in Fig. 6.8. This dataset is also impossible to reconstruct for the variational and sparse encoder (not shown). The likely reason for this is that 2D data is inherently correlated. The (x, y) coordinates are dependent.

A convolutional autoencoder (Masci et al., 2011) is an autoencoder architecture that can operate on this type of data. It differs in its connectivity between the nodes in the autoencoder rather than in the loss function. The nodes are subdivided into layers. The layers are sparsely connected with each other, by convolutions. A stack of such layers has been given the name "deep neural network".

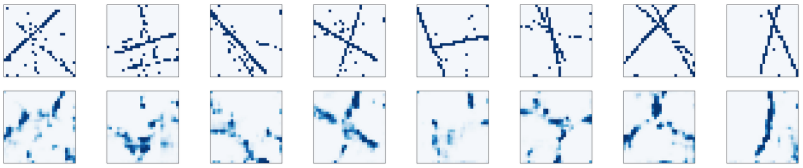


Figure 6.11: The reconstruction of lines (top row) starts to work for the convolutional autoencoder (results in bottom row). The reconstructions are blurry, but recognizable.

The convolutional autoencoder works on the 2D dataset if the lines are first mapped to a 2D grid. To use the same strategy in 3D, would mean that we have to create a voxel space. Rather than points like (x, y, z) we need to create a matrix of $LxMxN$, which becomes really large for increasing granularity. Although, the convolutional autoencoder shows that reconstructions are possible, we need something more sophisticated. In the next section we introduce autoencoders that are dedicated to point clouds.

6.3 Autoencoders on Point Clouds

Point cloud data has only recently been directly fed into deep neural networks. PointNet (Qi et al., 2017) is the first implementation of a deep network for segmentation and classification that directly operates on point clouds. In (Section 6.3.1) we describe an autoencoder from the literature which uses an architecture similar to PointNet as an encoder. The quality of this autoencoder will be assessed by visual inspection of the reconstruction quality (Section 6.3.2) and by a latent variable sweep from one representation to another (Section 6.3.3).

6.3.1 Earth Mover's Distance

The autoencoder (Achlioptas et al., 2018) architecture builds on PointNet. We refer to (Qi et al., 2017) for further details on PointNet. The autoencoder uses PointNet for the encoder. It accepts a point cloud with 2048 points (a 2048 x 3 matrix). It is constructed out of convolutional layers. The layers have kernel size 1 and an increasing number of features (representing the "neighborhood" of a point). The last layer is formed by a symmetric function, which is permutation invariant. The permutation invariance is required due to the exchangeability of points.

In a bit more detail, there are five convolutional layers, each is followed by a rectified linear unit (Nair and Hinton, 2010) and a batch-norm layer (Ioffe and Szegedy, 2015). The permutation invariant function takes a maximum (per feature) and forms a latent vector.

The decoder in this autoencoder is formed by three fully-connected layers with the first two followed by a ReLU and as output a 2048 x 3 matrix, the reconstructed point cloud.

One of the permutation-invariant objectives employed is the so-called earth mover's distance.

The earth mover's distance (EMD) is also known as the 1st Wasserstein distance. It can be considered an optimal transport problem. The concept has been first introduced in the 18th century (Monge, 1781). If there is an amount of sand and a pit where it has to go, how do we optimal transport the sand to the pit? The quantity that has to be minimized for this is called the earth mover's distance.

▼ **Definition 6.1** — ***earth mover's distance***

The earth mover's distance (EMD) between $S_1, S_2 \in \mathbb{R}^3$ of equal size $|S_1| = |S_2|$ is defined as:

$$d_{EMD}(S_1, S_2) = \min_{\phi: S_1 \to S_2} \sum_{x \in S_1} ||x - \phi(x)||_2$$

with $\phi : S_1 \to S_2$ a bijection.

In the next section we apply this autoencoder using the earth mover's distance on 3D point clouds.

6.3.2 Reconstruction of 3D point clouds

The autoencoder properly reconstructs not just 2D lines, but complete point clouds. In Figure 6.12 the autoencoder uses the earth mover's distance as loss function to reconstruct the original cubes.

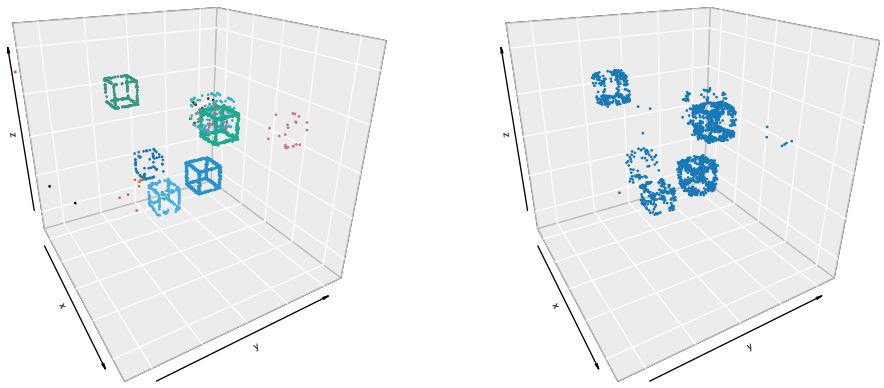
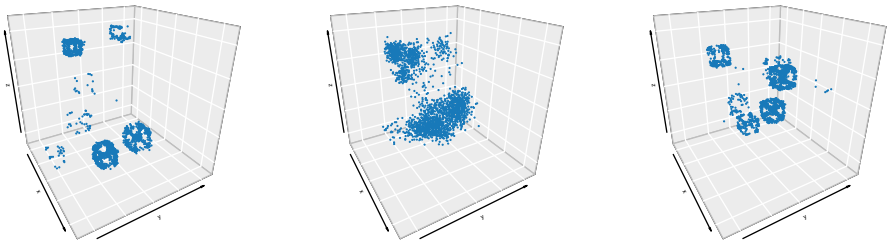


Figure 6.12: Reconstruction of a point cloud consisting of multiple cubes. The autoencoder uses the earth mover's distance. Left: the original point cloud. Right: the reconstructed point cloud. The reconstruction is not perfect. The cubes are recognizable though.

6.3.3 Quality of Latent Representation

To assess the quality of the latent representation we interpolate linearly between two latent variable representations that belong to two different cube configurations. Fig. 6.13 visualizes reconstructions of latent representations using the earth mover's distance. The results of the Chamfer distance are similar (not shown here).



(a) Reconstruction of latent representation corresponding to sample 1.

(b) Reconstruction of the latent representation linearly interpolated between latent representations of sample 1 and 2.

(c) Reconstruction of latent representation corresponding to sample 2.

Figure 6.13: Reconstruction of point clouds. This uses the earth mover's distance as a reconstruction loss. The representation in the center is generated by interpolating between two latent representations of actual samples. The reason that this is shown is to demonstrate that this latent representation is not generating geometric objects that are cubes. In contrast, it is a more general point cloud. This provides a hint that the autoencoder did not learn the concept of a cube. If it would, the intermediate representation more likely would have represented cubes at intermediate locations.

Remarkably, the objects at the interpolated steps do not resemble cubes [2]. The interpolation shows that the internal structure is not maintained. In between the two cube configurations, there is an unstructured point cloud, like Gaussian distributed blobs.

6.4 Semi-Autoencoders on Point Clouds

This section will address the limitations by the autoencoders and distance measures of the previous section. First we will explain limitation of the EMD with respect to uniformity (Section 6.4.1). We describe a first generalization of EMD called shifted earth mover's distance which addresses uniformity (Section 6.4.2). This is not able to cope with multiple objects for which we describe a second generalization called partitioning earth mover's distance (Section 6.4.3). This distance will then be used to create a semi-autoencoder (Section 6.4.4).

[2] A video can be seen at `https://bit.ly/2G2gyE2`

6.4.1 Limitations of the Earth Mover's Distance

The earth mover's distance defines a transportation map. Although the name optimal transport might suggest that there is a single optimal map, there are multiple maps possible. Depending on the implementation a particular map can be found. For example, if single grains of dirt are positioned at $x = 0, x = 1$ and need to be transported to $x = 2, x = 3$, it is possible to move one grain from $x = 0 \rightarrow 2$ and the other to $x = 1 \rightarrow 3$ or to move one grain from $x = 0 \rightarrow 3$ and the other $x = 1 \rightarrow 2$. The total distance traveled in both cases is 4. In the first case the moves are more uniform than in the second case.

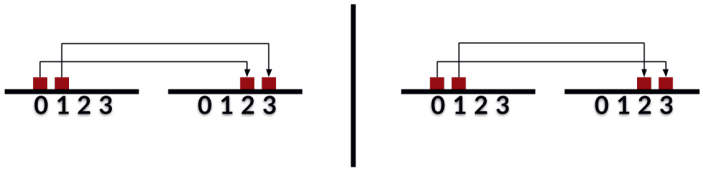
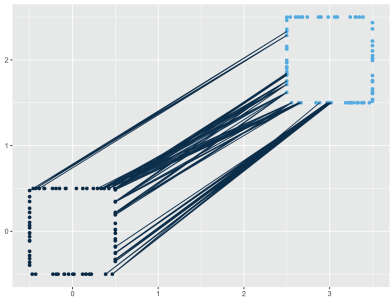


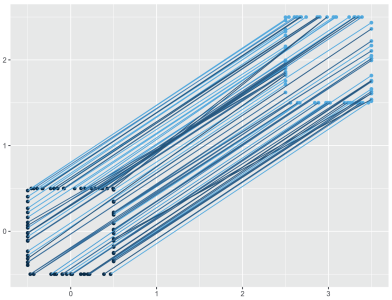
Figure 6.14: This figure shows that EMD is not unique. There are multiple maps possible and one map can be more "uniform" than another. Left: a map where each grain is moved with the same distance (of two). This is "uniform". We can achieve this with a global coordinate frame shift of two. Right: a map where one grain is moved over a larger distance (three versus one). This is not "uniform". Important to note, the total distance for both maps sums to four. The EMD can not distinguish between those cases!

The uniformity in a transportation map corresponds in the case of robotic vision with objects that are spatially translated. The transportation map that is not uniform does not correspond to a recognizable geometric operation.

In Fig. 6.15 we see at the left how we have the non-uniform transportation plan and on the right a uniform transportation plan.



(a) A non-uniform transportation plan. The square at the top-right is transported to the one at the origin. The points close to the origin are transported to the top-left of the square at the origin. Only the largest transport values are shown.



(b) A uniform transportation plan. A uniform transportation plan. The points in the square at the top-right are transported to the square at the origin in such a way that e.g. the points in a corner of the right square map to the points in the "same" corner of the other square. The map is uniform.

Figure 6.15: Earth mover's distance implementations. Both figures show "before" and "after" of the transportation map.

In Section 6.4.2 we describe an improvement on EMD which takes into account spatial translations. In Section 6.4.3 we describe a further generalization that takes into account multiple objects.

6.4.2 Shifted Earth Mover's Distance

The particular version of the earth mover's distance that is implemented is a shifted version of the EMD, the Shifted earth mover's distance. Suppose, in analogy with the original postulation of the problem by Mongei, that there is a small landslide such that the center of the source distribution becomes exactly on top of the center of the target distribution. The dirt now only has to be moved on a smaller scale.

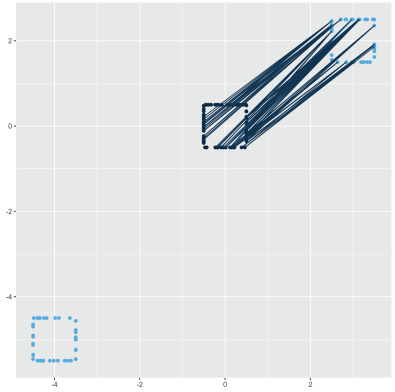
The SEMD is calculated by shifting both pointsets such that they are centered around the origin. Note, that for datasets where the objects are already centered around the origin this will not make any difference. However, our dataset is definitely not centered around the origin, so this has a large effect (see again Fig. 6.15).

6.4.3 Partitioning Earth Mover's Distance

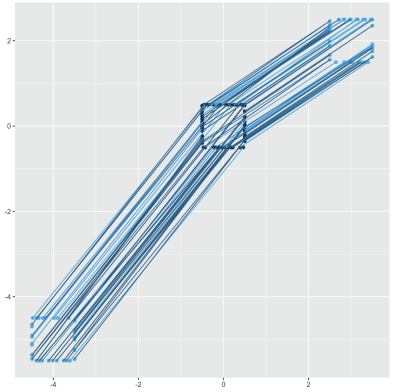
The SEMD is an improvement on EMD for single objects. However, for multiple objects the shift to the origin needs to be different for each object. PEMD partitions the space and couples each subset with its own shift operator. We visualize such a map in Fig. 6.16.

Figure 6.16: This figure demonstrates partitioning earth mover's distance (PEMD) with an example. Shown are two maps which are equivalent for EMD. Left: a possible SEMD map, the grains are moved by $\{2, 1, 1, -1, -1, -2\}$ steps (sum of absolute values is 8). Right: the PEMD map, the grains are moved by $\{2, 2, 2, -2, -2, -2\}$ steps (sum of absolute values is 12). The PEMD preserves the "structure" of the three grains in a cluster. The PEMD is larger than the SEMD.

For our robotic vision problems of identifying point clouds this can be visualized as well (see Fig. 6.17). Note that the object is a square and that there are multiple identical objects are to be recognized. This is the purpose of deploying the PEMD distance. It minimizes the distance to a "prototypical" object. In our autoencoder application, the result will be such a "prototype" object. The autoencoder will not reconstruct the complete input, it will construct a single copy of multiple identical objects that are present in the input. This is the property we use of this semi-autoencoder. We use the constructed output in Section 6.5 as input for our sampler.



(a) Shifted earth mover's distance visualization. The objects at $(-4, 5)$ and $(2, 2)$ are mapped to the object in the center. The (uniform) mapping of SEMD does not distinguish between the squares. It is not preserving the structure of the individual squares.



(b) Partitioning earth mover's distance visualization. The objects at $(-4, 5)$ and $(2, 2)$ are mapped to the object in the center. The mapping maps both squares on top of the center square. It preserves the structure of the individual squares.

Figure 6.17: Earth mover's distance implementation on three objects. Left: SEMD. Right: PEMD.

The PEMD needs to find partitions. This is actually a clustering problem on its own. Given a pointset and a repeated structure, can we divide this pointset into that structure and the number of times this structure occurs?

We solve this by the following optimization procedure. We search for local modes using a conventional algorithm, in our case mean-shift. After we have found the local modes, we use the means of the data points we found to translate the objects in both point clouds to the origin. We then perform EMD on the entire (normalized) dataset.

In Fig. 6.17 the difference can be seen in the transportation plan between a global shift (left) and shifts per partition (right). The shifts per partition shows a transportation map that preserves the local structure.

6.4.4 Semi-autoencoder

There are examples of generalizations of an autoencoder where the dimensions of the input and the output does not have to be identical (Zhang et al., 2017). However, the discrepancy between input and output can purely rise from the distance function used as well. If the distance function is truly translation invariant, the output gives rise to the object searched for. It is not a matter of reconstructing one of the input objects. A proper autoencoder would reconstruct a single object using information from all those duplicate objects in the input. This map can be seen as a generalized reconstruction objective. Alternatively, it can be seen as a general encoder-decoder network where we search for transformations of the input rather than the identity operation (Worrall et al., 2017).

The semi-autoencoder can be seen as a system that separates the object's shape (the "what") from the object's position (the "where") tailored to a situation where there are many identical objects. Separation between object shape and position through separate pathways seems ubiquitous in the (mammalian) brain (Ungerleider and Haxby, 1994; Rauschecker and Tian, 2000). Even more relevant to our architecture, there is brain-imaging evidence that the pathways for object identity and numerosity (the number of objects) are separated (Izard et al., 2008). Moreover, the ability to represent numerosity seems just as ubiquitous as the what-where pathways and can for example be found in mosquitofish (Agrillo et al., 2011) and honeybees (Bortot et al., 2019).

In this context, the semi-autoencoder fulfills the role of object identification. Given an object there is another pathway that is able to reason with these objects. In our case we will use a triadic sampler to perform class assignments. When points are assigned to multiple objects we have indirectly also represented counting those objects. In other words we have a second pipeline to establish numerosity. Note, that in contrast with subitizing autoencoders (Wever and Runia, 2018; Pecyna et al., 2019) our objective is not counting itself but clustering: the proper assignment of 3D points to multiple 3D objects.

6.5 Results

In Section 6.5.1 we describe a few implementation details for the autoencoder. In Section 6.5.2 we analyze the performance of the classifier that uses the results of the autoencoder for the task of assigning points to cubes in point clouds.

6.5.1 Implementation

The autoencoder using the partitioning earth mover's distance[3] is based on an autoencoder implementation using PointNet (Achlioptas et al., 2018). It uses Tensorflow and the metrics and their gradients have been implemented for the CPU (python3) and for the GPU (CUDA). The implementation for the GPU aims to improve the speed of the operations.

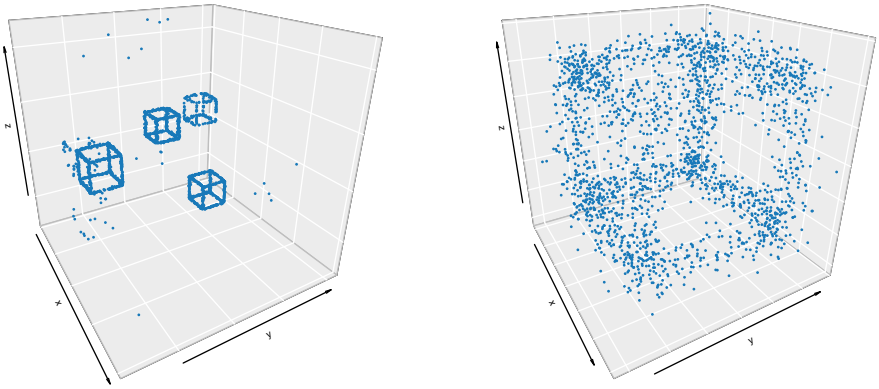


Figure 6.18: Left: the input, a sample with multiple 3D cube objects. Right: a reconstructed cube. The autoencoder uses partitioning earth mover's distance as a metric. The reconstruction creates a single object. Each cube in the original input contributes to the representation in the output. The output reflects the invariance in number of objects and their locations.

In Fig. 6.18 one of the reconstructed samples is visualized. The input, the set of cubes, is at the left. The output, the single cube at the origin, is at the right. This is a to be expected result from the reconstruction process. The loss function does not take into account the actual position of the cubes: it is on purpose invariant to these positions. This means that the generation process yielding a unity cube will have a very low loss associated to it.

6.5.2 Clustering Performance

The reconstruction of the autoencoder is used as a reference object for a triadic MCMC sampler (Chapter 5). The MCMC sampler compares the reference object with the current sample. Each point is compared with its nearest neighbor in the reference set and its match is defined through a normal distribution. The datasets are subsampled to 200 points.

More specific, it is common to compare two objects using pairwise Euclidean distances between (correspondence) points (Boutin and Kemper, 2004). We will use a Dirichlet Process with a multivariate normal distribution as base distribution, H, centered at the origin and with scalar noise ($\sigma = 1$). During the inference process the sampler will generate 3D locations, μ_i, for the hypothesized objects and we will calculate the difference between the

[3] Implementation at `https://github.com/mrquincle/latent_3d_points`.

hypothesized object with the reference object using the Euclidean kernel. In our implementation the objects do not have to have the same number of points. We sum the distances of the closest point (also with respect to the Euclidean distance). This means that the reference object can be much denser than the hypothesized object.

We can describe the Dirichlet process in the usual manner:

$$\begin{aligned} G &\sim DP(\alpha, H), \\ \mu_i \mid G &\overset{iid}{\sim} G, \\ w_i \mid \mu_i, r &\sim F(w; \mu, r). \end{aligned} \tag{6.4}$$

The likelihood function has a bit of a complicated structure:

$$F(w_i|\mu, r) = D(w_i, g(w_i, f(r, \mu))). \tag{6.5}$$

The reference point cloud r is undergoing an operation through $f(r, \mu)$, in this case a simple shift $f(r_j, \mu) = r_j - \mu$ for each j in the point cloud r. The function g is nonlinear and finds the point closest to w_i in r, which we define as r'. Then the distance is calculated between w_i and r' through D, the Euclidean distance.

The results of the triadic sampler can be found in Table 6.1.

Table 6.1: The purity, rand index, and adjusted rand index establishing the quality of the clustering method for line estimation (Chapter 5) and cube estimation (this chapter). The more complex dataset results in to be expected lower performance levels, but a significant number of assignments are correct. The results with the other types of samplers are not repeated for the cube dataset considering that they underperformed the triadic sampler already in the line estimation task.

Dataset	Purity	Rand Index	Adjusted Rand Index
Line estimation (Chapter 5)	0.86405	0.87188	0.71067
Cube estimation	0.8367	0.8359	0.6524

The spread over the samples is displayed in Figure 6.19 as a violin plot. Note that there might be still room for improvement. For some data samples an adjusted rand index of 0 corresponds to chance. Yet, it does not necessarily mean that the sampler can be improved. If there are two cubes generated at exactly the same location the "correct" cube becomes unidentifiable.

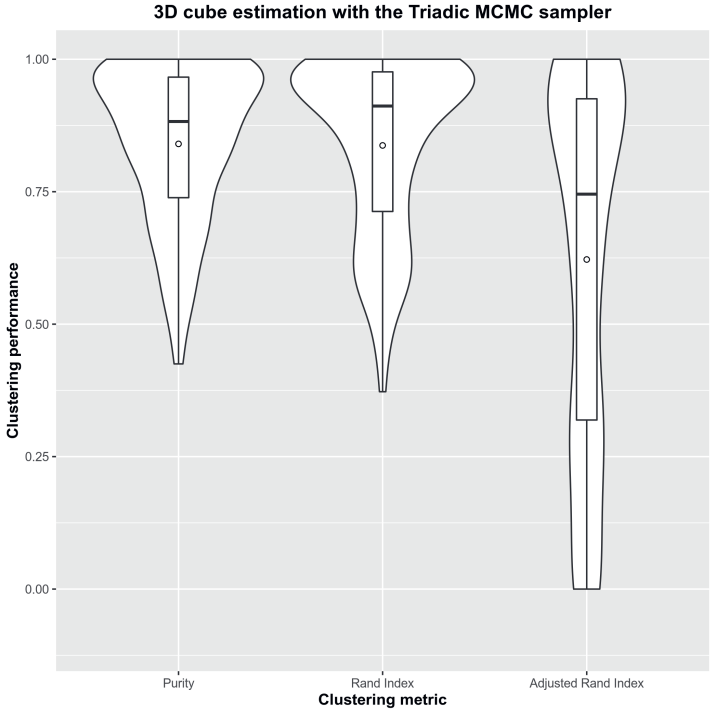


Figure 6.19: Performance of the triadic sampler on 3D cubes.

Figure 6.20 shows an assignment for a particular sample from the dataset. The cubes are properly identified as separate entities. There are also a few points that are accidentally assigned to those cubes as well.

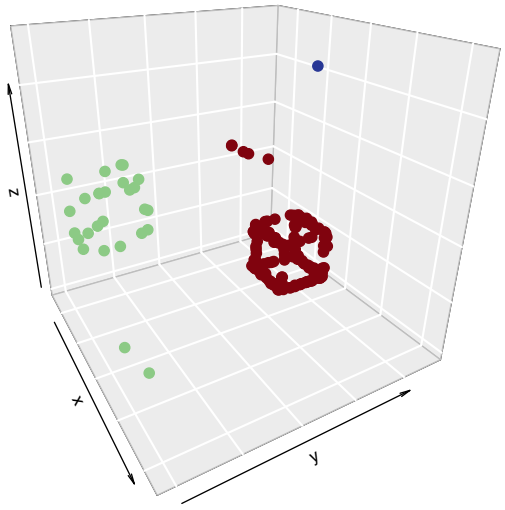


Figure 6.20: One of the samples as classified by the triadic sampler. The assignments of points to cubes is indicated by colors. It is visualized how some of the points are misclassified. In this case this happens mainly for cubes with only a few points.

6.6 Chapter Conclusions

The previous chapters went to great lengths to use Bayesian inference methods to given prior and likelihood optimally infer lines, line segments, squares, and other primitive geometrical objects. In this chapter we deployed deep learning methods to be able to reason about less primitive objects: cubes. A naive application of state-of-the-art deep learning methods for point clouds is not sufficient. Compared to dataset in the literature the objects in our dataset are shifted, and importantly, there are multiple objects in a single frame. The neural network needs to learn multiple objects at once.

The chapter introduced two new metrics for autoencoders based on earth mover's distance or Wasserstein. First, a SEMD or Shifted Wasserstein metric that can be used on dataset where objects are not centered at the origin. Second, a PEMD or Partitioning Wasserstein metric that can be used for datasets where individual frames do have multiple objects.

The results show that it is indeed possible to create a variant of an autoencoder that learns to reconstruct a particular object when there are multiple copies in its input. It is a semi-autoencoder, not an exact autoencoder. The purpose is not to exactly regenerate the input, but to represent the duplicate objects as one single object. It uses the same encoding-decoding structure as an autoencoder.

The model in this chapter is able to:

- Perform inference on multiple objects by using a partitioning earth mover's distance. Inference over multiple objects simultaneously is not part of recent optimal transport literature (Alvarez-Melis et al., 2018, 2019). The ability to perform inference over multiple objects is neither part of translation-aware work such as transforming autoencoders (Hinton et al., 2011) and capsule networks (Hinton et al., 2018).
- Perform inference on multiple objects without defining the number beforehand. This goes beyond e.g. the work on so-called barycenters (Forrow et al., 2018, 2019).

This chapter demonstrated how to perform inference on more complex volumetric objects. It showed how naive application of autoencoders - even autoencoders especially designed for point clouds - fails for inference of multiple objects.

New metrics and a new type of autoencoder is developed to take this type of structure into account. Such an autoencoder can then be used for a prior for a Bayesian model. This model can be sampled with the techniques of the previous chapters.

This chapter answers our third research question.

RQ 3 How can we recognize more general 3D objects?

First we use modern deep learning techniques to create data-driven priors. These priors can be very complex, learned - over many iterations of the data - by an autoencoders with many layers. Second, given these data-driven priors we perform inference using a nonparametric Bayesian model to detect general 3D objects. This bridges the field of deep learning with that of nonparametric Bayesian methods.

Chapter 7

Discussion and Conclusions

Autonomous systems, cars, robots, rely on depth information. Point clouds, points in a 3D space, are data streams that these systems have to encode into objects, infer object type, count, and in the end use to orient themselves in the world and act accordingly.

Bayesian methods have the advantage of - given a particular prior and likelihood - optimally deduct the posterior. In computer vision the Hough transform and other traditional methods have been used many times. Now, with the advent of more modern machines it becomes possible to implement full Bayesian methods. The Bayesian methods deployed do not just infer a particular type of line, but multiple of them. Moreover, the number of objects is not known in advance.

7.1 Research Questions

Below we summarize the answer to the three research questions, (RQs). We use RA1 to indicate the answer to RQ1. Let us start to reiterate the first research question.

RQ 1 How can we estimate the number of objects simultaneously with the fitting of these objects?

We have introduced two nonparametric Bayesian models in Chapter 3 and Chapter 4.

RA 1 We can estimate the number of objects simultaneously with fitting the objects for lines as well as segments with nonparametric Bayesian methods. For lines Gibbs sampling is sufficient. For line segments, due to nonconjugacy of prior and likelihood, Gibbs sampling with auxiliary variables has to be used.

This answers the first research question.

The second research question concerns optimization for the domain of robotic vision.

> **RQ 2** How can we optimize inference over both the number of objects and fitting of those objects in the robotic vision domain?

The point clouds in the robotic vision domain have spatial properties. One of those properties is that objects can intersect. In Chapter 5 we introduced a new MCMC method, the Triadic Split-Merge sampler.

> **RA 2** Inference method in the robotic vision domain can be improved through the use of a triadic split-merge sampler that takes into account object intersection.

This answers the second research question.

The third research question asks about generalization.

> **RQ 3** How can we recognize more general 3D objects?

Generalization to 3D, as well as considering more complex objects than lines or line segments, requires a great deal of sampling time. By introducing data-driven priors, inference can be accelerated, as shown in Chapter 6.

> **RA 3** More general 3D objects can be recognized by using data-driven priors. These priors can be generated by deep learning methods. The use of complex, data-driven priors allows us to to perform inference over more general 3D objects such as cubes, compared to simple lines or line segments.

This answers the third research question.

7.2 Problem Statement

The research questions support the answer to the general problem statement.

> **PS:** *How can robotic vision problems effectively be generalized and their structure exploited in a wider Bayesian framework?*

The answer to the problem statement is a combination of general approaches, which have been made possible to (1) advances in nonparametric Bayesian models as well as (2) advances in MCMC inference methods and (3) modern data-driven deep learning methods.

Answer: *Robotic vision problems that concern the recognition of multiple objects of which the number is unknown in advance can be modeled by nonparametric Bayesian models. The inference methods - even though there is no conjugacy - can benefit from spatial characteristics and can be further accelerated by using data-driven priors.*

7.3 Limitations

Inference methods for nonparametric Bayesian models can converge quite slowly. The triadic sampler as postulated in this thesis does accelerate inference for a particular type of robotic vision problem, but it takes too much time for more complex data objects such as 3D cubes. Standard reconstruction loss functions in deep learning methods such as autoencoders fail to take into account properties of the robotic vision domain: e.g. translation invariance and the occurrence of multiple objects. This can be mitigated by adjusted loss functions as proposed in this thesis. For robotic vision problems with multiple types of complex 3D objects, the current model is not sufficient. The autoencoder would represent multiple objects simultaneously. For a correct handling of this challenge, a more complex loss function has to be imposed on the autoencoder.

7.4 Recommendations

We recommend an in-depth study of a variety of reconstruction loss functions for autoencoders on datasets that have the spatial properties inherent to the robotic vision domain. The recommendation is to not only address spatial invariance or object copies as studied in this thesis. There are other properties, such as rotation invariance, scale invariance, and object overlap which would require our attention. There are also dynamic properties, such as temporal occlusions, temporal consistency, light conditions, and gravitational constraints, might will need to be captured by priors to perform Bayesian inference. Those priors (1) can be postulated by the domain expert as is regularly done in the Bayesian methodology or they (2) can be obtained in a data-driven manner as is done in the deep learning literature.

References

P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas. Learning representations and generative models for 3D point clouds. In *Proceedings of the 35th International Conference on Machine Learning*, pages 40–49, 2018.

C. Agrillo, L. Piffer, and A. Bisazza. Number versus continuous quantity in numerosity judgments by fish. *Cognition*, 119(2):281–287, 2011.

D. J. Aldous. Representations for partially exchangeable arrays of random variables. *Journal of Multivariate Analysis*, 11(4):581–598, 1981.

D. J. Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198. Springer, 1985.

J. Aldrich. R.A. Fisher and the making of maximum likelihood 1912-1922. *Statistical Science*, 12(3): 162–176, 1997.

D. Alvarez-Melis, T. Jaakkola, and S. Jegelka. Structured optimal transport. In *International Conference on Artificial Intelligence and Statistics*, pages 1771–1780, 2018.

D. Alvarez-Melis, S. Jegelka, and T. S. Jaakkola. Towards optimal transport with global invariances. pages 1870–1879, 2019.

C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan. An introduction to MCMC for machine learning. *Machine learning*, 50(1-2):5–43, 2003. doi: 10.1023/A:1020281327116.

S. Banach and A. Tarski. Sur la décomposition des ensembles de points en parties respectivement congruentes. *Fund. math*, 6(1):924, 1924.

S. Banerjee. Bayesian linear model: Gory details. 2008. URL `http://www.biostat.umn.edu/~ph7440`.

A. Barbu and S.-C. Zhu. Generalizing Swendsen-Wang to sampling arbitrary posterior probabilities. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1239–1253, 2005.

D. Basu and R. Tiwari. A note on the Dirichlet process. In P. R. Krishnaiah, G. Kallianpur, J. Ghosh, and C. R. Rao, editors, *Statistics and Probability: Essays in Honor of C.R. Rao*, pages 355–369. Springer, 1982.

L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *The annals of mathematical statistics*, 37(6):1554–1563, 1966.

H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

D. Blackwell. Discreteness of Ferguson selections. *The Annals of Statistics*, 1(2):356–358, 1973.

D. Blackwell and J. B. MacQueen. Ferguson distributions via Pólya urn schemes. *The annals of statistics*, pages 353–355, 1973.

J.-L. Blanco, J. González, and J.-A. Fernández-Madrigal. Optimal filtering for non-parametric observation models: applications to localization and SLAM. *The International Journal of Robotics Research*, 29(14):1726–1742, 2010.

R. C. Bolles and M. A. Fischler. A RANSAC-based approach to model fitting and its application to finding cylinders in range data. In *IJCAI*, volume 1981, pages 637–643, 1981.

M. Bortot, C. Agrillo, A. Avarguès-Weber, A. Bisazza, M. E. Miletto Petrazzini, and M. Giurfa. Honeybees use absolute rather than relative numerosity in number discrimination. *Biology Letters*, 15 (6):20190138, 2019.

A. Bosch, A. Zisserman, and X. Muoz. Scene classification using a hybrid generative/discriminative approach. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 30(4):712–727, 2008.

G. Bouchard and B. Triggs. The tradeoff between generative and discriminative classifiers. In *16th IASC International Symposium on Computational Statistics (COMPSTAT'04)*, pages 721–728, 2004.

M. Boutin and G. Kemper. On reconstructing n-point configurations from the distribution of distances or areas. *Advances in Applied Mathematics*, 32(4):709–735, 2004.

G. E. P. Box and G. C. Tiao. *Bayesian inference in statistical analysis*, volume 40. John Wiley and Sons, New York, 2011.

R. L. Brennan and R. J. Light. Measuring agreement when two observers classify people into categories not defined in advance. *British Journal of Mathematical and Statistical Psychology*, 27(2): 154–163, 1974.

M. Bryant and E. B. Sudderth. Truly nonparametric online variational inference for hierarchical Dirichlet processes. In *Advances in Neural Information Processing Systems*, pages 2699–2707, 2012.

H. Bühlmann. *Austauschbare stochastische Variabeln und ihre Grenzwertsatze*. PhD thesis, ETH Zürich, 1960.

W. L. Buntine. Operations for learning with graphical models. *JAIR*, 2:159–225, 1994.

J. Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, 8(6):679–698, 1986.

J. Chang and J. W. Fisher III. Parallel sampling of DP mixture models using sub-cluster splits. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 620–628. Curran Associates, Inc., 2013.

H. Chen, P. Meer, and D. E. Tyler. Robust regression for data with multiple structures. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I—-1069. IEEE, 2001.

D. L. Cohn. *Measure theory*. Springer, 2013.

D. B. Dahl. An improved merge-split sampler for conjugate Dirichlet process mixture models. Technical report, University of Wisconsin–Madison, Nov. 2003.

D. B. Dahl. Sequentially-allocated merge-split sampler for conjugate and nonconjugate Dirichlet process mixture models. Technical report, Texas A&M University, Nov. 2005.

N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893. IEEE, 2005.

H. Daume. Fast search for Dirichlet process mixture models. In *International Conference on Artificial Intelligence and Statistics*, pages 83–90, 2007.

D. G. Denison. *Bayesian methods for nonlinear classification and regression*, volume 386. John Wiley and Sons, New York, 2002.

P. Diaconis and D. Freedman. de Finetti's theorem for Markov chains. *The Annals of Probability*, pages 115–130, 1980.

P. Diaconis and D. Freedman. On the consistency of Bayes estimates. *The Annals of Statistics*, pages 1–26, 1986.

J. Dubbeldam, K. Green, and D. Lenstra. *The Complexity of Dynamical Systems*. Wiley Online Library, 2011.

D. B. Dunson, Y. Xue, and L. Carin. The matrix stick-breaking process. *Journal of the American Statistical Association*, 2012.

R. Durstenfeld. Algorithm 235: Random permutation. *Communications of the ACM*, 7(7):420, July 1964. ISSN 0001-0782. doi: 10.1145/364520.364540.

M. D. Escobar. *Estimating the means of several normal populations by nonparametric estimation of the distribution of the means*. PhD thesis, Yale University Unpublished dissertation, 1988.

M. D. Escobar. Estimating normal means with a Dirichlet process prior. *Journal of the American Statistical Association*, 89(425):268–277, 1994.

M. D. Escobar and M. West. Bayesian density estimation and inference using mixtures. *Journal of the american statistical association*, 90(430):577–588, 1995.

S. N. Ethier. The distribution of the frequencies of age-ordered alleles in a diffusion model. *Advances in Applied Probability*, pages 519–532, 1990.

W. J. Ewens. Population genetics theory - the past and the future. In *Mathematical and statistical developments of evolutionary theory*, pages 177–227. Springer, 1990.

S. Favaro, Y. W. Teh, et al. MCMC for normalized random measure mixture models. *Statistical Science*, 28(3):335–359, 2013.

W. Feller. *An introduction to probability theory and its applications. Vol. I.* John Wiley and Sons, New York, 1950.

T. S. Ferguson. A Bayesian analysis of some nonparametric problems. *The annals of statistics*, pages 209–230, 1973.

T. S. Ferguson. Prior distributions on spaces of probability measures. *The annals of statistics*, pages 615–629, 1974.

S. E. Fienberg et al. When did Bayesian inference become "Bayesian"? *Bayesian analysis*, 1(1):1–40, 2006.

B. de Finetti. Funzione caratteristica di un fenomeno aleatorio. *Atti Reale Accademia Nazionale dei Lincei*, VI:86–133, 1930.

B. de Finetti. La prévision: ses lois logiques, ses sources subjectives. In *Annales de l'institut Henri Poincaré*, volume 7, pages 1–68, 1937.

E. Forgy. Cluster analysis of multivariate data: Efficiency vs. interpretability of classifications. *Biometrics*, 21:768–769, 1965.

A. Forrow, J.-C. Hütter, M. Nitzan, G. Schiebinger, P. Rigollet, and J. Weed. Statistical optimal transport via geodesic hubs. *arXiv preprint arXiv:1806.07348*, 2018.

A. Forrow, J.-C. Hütter, M. Nitzan, P. Rigollet, G. Schiebinger, and J. Weed. Statistical optimal transport via factored couplings. International Conference on Artificial Intelligence and Statistics, 2019.

W. Förstner and E. Gülch. A fast operator for detection and precise location of distinct points, corners and centres of circular features. In *Proc. ISPRS intercommission conference on fast processing of photogrammetric data*, pages 281–305, 1987.

E. B. Fox, E. B. Sudderth, and A. S. Willsky. Hierarchical Dirichlet processes for tracking maneuvering targets. In *2007 10th international conference on information fusion*, pages 1–8. IEEE, 2007.

D. Freedman and P. Diaconis. On inconsistent Bayes estimates in the discrete case. *The Annals of Statistics*, pages 1109–1118, 1983.

D. H. Fremlin. *Measure theory*, volume 4. Torres Fremlin, 2000.

Y. Gal and Z. Ghahramani. Dropout as a Bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

O. Gallo, R. Manduchi, and A. Rafii. CC-RANSAC: Fitting planes in the presence of multiple surfaces in range data. *Pattern Recognition Letters*, 32(3):403–410, 2011.

A. Garcia-Garcia, F. Gomez-Donoso, J. Garcia-Rodriguez, S. Orts-Escolano, M. Cazorla, and J. Azorin-Lopez. Pointnet: A 3D convolutional neural network for real-time object class recognition. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 1578–1584. IEEE, 2016.

A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 20(5-6):721–741, 1984.

Z. Ghahramani and T. L. Griffiths. Infinite latent feature models and the Indian buffet process. In *Advances in neural information processing systems*, pages 475–482, 2005.

S. Ghosal and A. Van der Vaart. *Fundamentals of nonparametric Bayesian inference*, volume 44. Cambridge University Press, 2017.

P. R. Halmos. Random alms. *The Annals of Mathematical Statistics*, 15(2):182–189, 1944.

P. R. Halmos. *Measure theory*, volume 18. Springer, 1974.

M. Halperin and G. L. Burrows. The effect of sequential batching for acceptance—rejection sampling upon sample assurance of total product quality. *Technometrics*, 2(1):19–26, 1960.

C. Harris and M. Stephens. A combined corner and edge detector. In *Alvey vision conference*, volume 15, page 50. Citeseer, 1988.

W. K. Hastings. Monte Carlo sampling methods using Markov chains and their applications. *Biometrika*, 57(1):97–109, 1970. ISSN 00063444. doi: 10.1093/biomet/57.1.97.

G. E. Hinton, A. Krizhevsky, and S. D. Wang. Transforming auto-encoders. In *International Conference on Artificial Neural Networks*, pages 44–51. Springer, 2011.

G. E. Hinton, S. Sabour, and N. Frosst. Matrix capsules with EM routing. In *6th International Conference on Learning Representations, ICLR*, 2018.

N. L. Hjort. Nonparametric Bayes estimators based on beta processes in models for life history data. *The Annals of Statistics*, pages 1259–1294, 1990.

F. M. Hoppe. Size-biased filtering of Poisson-Dirichlet samples with an application to partition structures in genetics. *Journal of Applied Probability*, pages 1008–1012, 1986.

E. Hosseini-Asl, J. M. Zurada, and O. Nasraoui. Deep learning of part-based representation of data using sparse autoencoders with nonnegativity constraints. *IEEE transactions on neural networks and learning systems*, 27(12):2486–2498, 2015.

P. V. Hough. Method and means for recognizing complex patterns, Dec 1962. URL `https://www.google.com/patents/US3069654`. Patent US 3069654 A.

L. Hubert. Nominal scale response agreement as a generalized correlation. *British Journal of Mathematical and Statistical Psychology*, 30(1):98–103, 1977.

L. Hubert and P. Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

M. C. Hughes and E. Sudderth. Memoized online variational inference for Dirichlet process mixture models. In *Advances in Neural Information Processing Systems*, pages 1133–1141, 2013.

M. C. Hughes, E. Fox, and E. B. Sudderth. Effective split-merge Monte Carlo methods for nonparametric models of sequential data. In *Advances in neural information processing systems*, pages 1295–1303, 2012.

F. Huszár and D. Duvenaud. Optimally-weighted herding is Bayesian quadrature. In *Proceedings of the Twenty-Eighth Conference on Uncertainty in Artificial Intelligence*, pages 377–386, 2012.

Y. Ioannou, B. Taati, R. Harrap, and M. Greenspan. Difference of normals as a multi-scale operator in unorganized point clouds. In *2012 Second International Conference on 3D Imaging, Modeling, Processing, Visualization & Transmission*, pages 501–508. IEEE, 2012.

S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

H. Ishwaran and L. F. James. Gibbs sampling methods for stick-breaking priors. *Journal of the American Statistical Association*, 96(453):161–173, 2001.

V. Izard, G. Dehaene-Lambertz, and S. Dehaene. Distinct cerebral pathways for object identity and number in human infants. *PLoS biology*, 6(2), 2008.

T. S. Jaakkola, D. Haussler, et al. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

S. Jain and R. M. Neal. A split-merge Markov chain Monte Carlo procedure for the Dirichlet process mixture model. *Journal of Computational and Graphical Statistics*, 13(1):158–182, 2004. ISSN 10618600. URL `http://www.jstor.org/stable/1391150`.

S. Jain and R. M. Neal. Splitting and merging components of a nonconjugate Dirichlet process mixture model. *Bayesian Analysis*, 2(3):445–472, 2007.

V. Jampani, S. Nowozin, M. Loper, and P. V. Gehler. The informed sampler: A discriminative approach to Bayesian inference in generative computer vision models. *Computer Vision and Image Understanding*, 136:32–44, 2015.

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003.

D. Joho, G. D. Tipaldi, N. Engelhard, C. Stachniss, W. Burgard, M. Senk, F. Faber, M. Bennewitz, C. Eppner, A. Görög, et al. Unsupervised Scene Analysis and Reconstruction Using Nonparametric Bayesian Models. *Robotics and Autonomous Systems (RAS)*, 59(5):319–328, 2011.

A. Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes. *Advances in neural information processing systems*, 14:841, 2002.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proceedings of the 2nd International Conference on Learning Representations (ICLR-2014)*, volume 1050, page 1, 2014.

J. F. C. Kingman. Some further analytical results in the theory of regenerative events. *Journal of Mathematical Analysis and Applications*, 11:422–433, 1965.

J. F. C. Kingman. Completely random measures. *Pacific Journal of Mathematics*, 21(1):59–78, 1967.

J. F. C. Kingman. Random partitions in population genetics. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 361, pages 1–20. The Royal Society, 1978.

J. F. C. Kingman. *Poisson processes*, volume 3 of *Oxford Studies in Probability*. The Clarendon Press Oxford University Press, New York, 1993. ISBN 0-19-853693-3.

D. Knowles, Z. Ghahramani, and K. Palla. A reversible infinite HMM using normalised random measures. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1998–2006, 2014.

D. Koller and N. Friedman. *Probabilistic graphical models: Principles and techniques*. MIT press, 2009.

A. Kolmogorov. *Grundbegriffe der wahrscheinlichkeitsrechnung*, volume 2 of *Ergebnisse der Mathematik und ihrer Grenzgebiete*. Springer-Verlag, 1933.

K. Kurihara, M. Welling, and Y. W. Teh. Collapsed variational Dirichlet process mixture models. In *IJCAI*, volume 7, pages 2796–2801, 2007.

P.-S. Laplace. *Théorie analytique des probabilités*. V. Courcier, 1820.

H. Lebesgue. Intégrale, longueur, aire. *Annali di Matematica Pura ed Applicata (1898-1922)*, 7(1): 231–359, 1902.

Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

A. Lijoi and I. Prünster. Models beyond the Dirichlet process. *Bayesian nonparametrics*, 28:80, 2010.

S. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inform. Theory*, 28:129–137, 1982. Originally as an unpublished Bell laboratories Technical Note (1957).

D. G. Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

L. van der Maaten, E. Postma, and J. Van den Herik. Dimensionality reduction: a comparative. *J Mach Learn Res*, 10(66-71):13, 2009.

S. N. MacEachern. Comment on "Splitting and merging components of a nonconjugate Dirichlet process mixture model" by jain and neal. *Bayesian Analysis*, 2(3):483–494, 2007.

S. N. MacEachern and P. Müller. Estimating mixture of Dirichlet process models. *Journal of Computational and Graphical Statistics*, 7(2):223–238, 1998.

S. Mandt, M. D. Hoffman, and D. M. Blei. Stochastic gradient descent as approximate Bayesian inference. *The Journal of Machine Learning Research*, 18(1):4873–4907, 2017.

A. B. Mariotto. Empirical Bayes inference and the linear model. 1989.

J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *International conference on artificial neural networks*, pages 52–59. Springer, 2011.

N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculations by fast computing machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114.

B. Mirkin and L. Cherny. On a distance measure between partitions of a finite set. *Automation and remote control*, 31(5):91–98, 1970.

G. Monge. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Sciences de Paris*, 1781.

K. P. Murphy. *Machine learning: A probabilistic perspective*. MIT press, 2012.

V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814, 2010.

R. M. Neal. Markov chain sampling methods for Dirichlet process mixture models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

R. M. Neal. Defining priors for distributions using Dirichlet diffusion trees. Technical report, University of Toronto, Toronto, Ontario, Canada, 2001.

P. Orbanz and D. M. Roy. Bayesian models of graphs, arrays and other exchangeable random structures. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 37(2):437–461, 2015.

L. Pecyna, A. Di Nuovo, and A. Cangelosi. A deep neural network for finger counting and numerosity estimation. In *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*. IEEE, 2019.

J. Pitman and M. Yor. The two-parameter Poisson-Dirichlet distribution derived from a stable subordinator. *The Annals of Probability*, pages 855–900, 1997.

C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

R. Raina, Y. Shen, A. Mccallum, and A. Y. Ng. Classification with hybrid generative/discriminative models. In *Advances in neural information processing systems*, page None, 2003.

W. M. Rand. Objective criteria for the evaluation of clustering methods. *Journal of the American Statistical association*, 66(336):846–850, 1971.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*, volume 2. MIT Press, 2006.

J. P. Rauschecker and B. Tian. Mechanisms and streams for processing of "what" and "where" in auditory cortex. *Proceedings of the National Academy of Sciences*, 97(22):11800–11806, 2000.

E. Regazzini, A. Lijoi, and I. Prünster. Distributional results for means of normalized random measures with independent increments. *Annals of Statistics*, pages 560–585, 2003.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

J. S. Rosenthal. *A First Look At Rigorous Probability Theory*. World Scientific Publishing Company, 2006.

A. C. van Rossum, H. X. Lin, J. Dubbeldam, and H. J. v. d. Herik. Fundamentals of nonparametric Bayesian line detection. In *International Conference on Pattern Recognition Applications and Methods*, pages 175–193. Springer, 2016a.

A. C. van Rossum, H. X. Lin, J. Dubbeldam, and H. J. v. d. Herik. Nonparametric Bayesian line detection - towards proper priors for robotic computer vision. In *Proceedings of the 5th International Conference on Pattern Recognition Applications and Methods*, pages 119–127, Feb 2016b. ISBN 978-989-758-173-1. doi: 10.5220/0005673301190127.

A. C. van Rossum, H. X. Lin, J. Dubbeldam, and H. J. v. d. Herik. Nonparametric segment detection. In *Proceedings of the 8th European Starting AI Researcher Symposium (STAIRS)*, pages 203–208, Aug 2016c. ISBN 978-989-758-173-1. doi: 10.5220/0005673301190127.

A. C. van Rossum, H. X. Lin, J. Dubbeldam, and H. J. v. d. Herik. Triadic split-merge sampler. In *Tenth International Conference on Machine Vision (ICMV 2017)*, volume 10696, page 1069623. International Society for Optics and Photonics, Nov 2017.

D. M. Roy and Y. W. Teh. The Mondrian process. In *Advances in neural information processing systems*, pages 1377–1384, 2009.

D. B. Rubin. Bayesianly justifiable and relevant frequency calculations for the applied statistician. *The Annals of Statistics*, 12(4):1151–1172, 1984.

D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. *Parallel distributed processing*, 1:318–362, 1986.

S. Russell, P. Norvig, and A. Intelligence. Artificial Intelligence: A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

L. J. Savage. *The foundations of statistics*. Courier Corporation, 1972.

S. Sawyer and D. Hartl. A sampling theory for local selection. *Journal of Genetics*, 64(1):21–29, 1985.

R. L. Schilling. *Measures, integrals and martingales*, volume 13. Cambridge University Press, 2005.

J. Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.

M. Seeger. Relationships between Gaussian processes, support vector machines and smoothing splines. *Machine Learning*, 2000.

J. Shi and C. Tomasi. Good features to track. In *Computer Vision and Pattern Recognition, 1994. Proceedings CVPR'94., 1994 IEEE Computer Society Conference on*, pages 593–600. IEEE, 1994.

S. A. Sisson and Y. Fan. Likelihood-free MCMC. *Handbook of Monte Carlo, ed. Brooks, A., Gelman, A., Jones, GL, Meng XL*, pages 313–335, 2011.

I. Sobel. *Camera Models and Perception*. PhD thesis, Stanford University, Stanford, CA, 1970.

G. Stawinski, A. Doucet, and P. Duvaut. Reversible jump Markov chain Monte Carlo for Bayesian deconvolution of point sources. In *Bayesian Inference for Inverse Problems*, volume 3459, pages 179–191. International Society for Optics and Photonics, 1998.

M. Steel. The maximum likelihood point for a phylogenetic tree is not unique. *Systematic Biology*, 43(4):560–564, 1994.

C. Sutton and A. McCallum. An introduction to conditional random fields. *Machine Learning*, 4(4): 267–373, 2011.

T. Tao. *An introduction to measure theory*, volume 126. American Mathematical Soc., 2011.

M. K. Titsias. The infinite gamma-Poisson feature model. In *Advances in Neural Information Processing Systems*, pages 1513–1520, 2008.

L. G. Ungerleider and J. V. Haxby. "What" and "where" in the human brain. *Current opinion in neurobiology*, 4(2):157–165, 1994.

G. Walter and T. Augustin. Bayesian linear regression - different conjugate models and their (in)sensitivity to prior-data conflict. In *Statistical Modelling and Regression Structures*, pages 59–78.

Springer, 2010.

W. Wang and S. Russell. A smart-dumb/dumb-smart algorithm for efficient split-merge MCMC. In *Proceedings of the Thirty-First Conference on Uncertainty in Artificial Intelligence*, UAI15, pages 902–911, Arlington, Virginia, United States, 2015. AUAI Press. ISBN 978-0-9966431-0-8.

Y. Wang and L. Carin. Levy measure decompositions for the beta and gamma processes. *arXiv preprint arXiv:1206.4615*, 2012.

L. Wasserman. Asymptotic properties of nonparametric Bayesian procedures. In *Practical nonparametric and semiparametric Bayesian statistics*, pages 293–304. Springer, 1998.

R. Wever and T. F. Runia. Subitizing with variational autoencoders. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 0–0, 2018.

R. Wieten, F. Bex, H. Prakken, and S. Renooij. Constructing Bayesian network graphs from labeled arguments. In *European Conference on Symbolic and Quantitative Approaches with Uncertainty*, pages 99–110. Springer, 2019.

D. E. Worrall, S. J. Garbin, D. Turmukhambetov, and G. J. Brostow. Interpretable transformations with encoder-decoder networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 5726–5735, 2017.

S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920, 2015.

D. F. Wulsin. *Bayesian nonparametric modeling of epileptic events*. PhD thesis, University of Pennsylvania, 2013.

J.-H. Xue and D. M. Titterington. Comment on "On discriminative vs. generative classifiers: A comparison of logistic regression and naive Bayes". *Neural processing letters*, 28(3):169–187, 2008.

A. Zellner. Optimal information processing and Bayes's theorem. *The American Statistician*, 42(4): 278–280, 1988.

S. Zhang, L. Yao, X. Xu, S. Wang, and L. Zhu. Hybrid collaborative recommendation via semi-autoencoder. In *International Conference on Neural Information Processing*, pages 185–193. Springer, 2017.

W. Zhang and J. Kŏsecká. Nonparametric estimation of multiple structures with outliers. In *Dynamical Vision*, pages 60–74. Springer, 2007.

Appendix A

Probabilistic Concepts

Modern probability is based on measure theory (Appendix A.1). Measure theory will provide the means to formally describe random variables, random processes, and most generally, random measures. A model represented by random measures can be fitted to the data using Bayesian inference (Appendix A.2). We give three typical examples of Bayesian model compositions, among which an infinite mixture model (Appendix A.3). A number of processes are described that can be used with (for example as prior distribution) infinite mixture models (Appendix A.4). We introduce plate notation which visualizes infinite models particularly well (Appendix A.5). Then we investigate completely random measures and Lévy measures (Appendix A.6), exchangebility (Appendix A.7), and stick-breaking processes (Appendix A.8). For mathematically more thorough approaches we refer to the literature (Halmos, 1974; Rosenthal, 2006; Cohn, 2013).

A.1 Measure Theory

A random variable is a *function* that assigns values to a *set* of possible outcomes. The formal definition requires concepts such as "measurable function" and "probability space" from *measure theory* (Feller, 1950). Measure theory is used to generalize the notion of a random variable to that of a "random process".

Informally, a measure generalizes the concepts of length, area, and volume of an Euclidean object to a concept of size for sets and subsets. The definition of a measure is based on the definition of a σ-algebra. A σ-algebra ascribes a value to a sum of individual disjoint sets, even if they are infinite in number.

▼ **Definition A.1** — ***σ-algebra***

A **σ-algebra** is a *subset* $\Sigma \in 2^X$, with X a set and 2^X its powerset, with three requirements:

- Σ is non-empty: at least one $A \in X$ is in Σ;

- Σ is closed under complementation: if A in Σ, so is its complement A^c;
- Σ is closed under countable unions: if $A_1, A_2, \ldots$ in Σ, so is $A = A_1 \cup A_2 \cup \ldots$.

The members of a σ-algebra are called *measurable sets*. Let $X = \{1,2,3,4\}$ and let us define a σ-algebra $\Sigma = \{\varnothing, \{1\}, \{4\}, \{2,3\}, \{1,4\}, \{1,2,3\}, \{2,3,4\}, \{1,2,3,4\}\}$. Here $\varnothing$ denotes the empty set. The complement of A is defined with respect to X: $A \cup A^c = X$. An example of closure under complementation: let $A_1 = \{1\}$, then $A_1^c = \{2,3,4\}$ and A_1^c is indeed a member of Σ: $A_1^c \in \Sigma$. An example of closure under countable unions: let $A_1 = \{1\}$ and $A_2 = \{2,3\}$, then $A_1 \cup A_2 = \{1,2,3\}$ and $A_1 \cup A_2 \in \Sigma$.

> The notion of a σ-algebra (Fremlin, 2000) can be applied to solve the so-called Banach-Tarski paradox (Banach and Tarski, 1924). This paradox describes how a unit-ball in $\mathbf{R}^3$ can be partitioned into a finite number of disjoint infinite sets (scattering of points) and then can be reassembled into two unit-balls again. This violates the intuitive notion of preservation of volume. If the measure μ of the union of two disjoint sets is equal to the sum of the measures of the two sets, this is called *finite additivity*: $\mu(\bigcup_{i=1}^{N} A_i) = \sum_{i=1}^{N} \mu(A_i)$. In probability theory *σ-additivity* extends this to infinite disjoint sets: $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$. Measure theory solves the Banach-Tarski paradox by only assigning a measure to subsets that are measurable sets (Tao, 2011).

A *measure* assigns values to measurable sets (as stated before, measurable sets are members or subsets of Σ).

▼ Definition A.2 — *measure*

A **measure** μ is a function from Σ to $[-\infty, +\infty]$, with three requirements:

- μ is non-negative: $\mu(A) \geq 0$ for $\forall A \in \Sigma$;
- μ has a null empty set: $\mu(\varnothing) = 0$;
- μ is σ-additive: $\mu(\bigcup_{i \in I_\Sigma} A_i) = \sum_{i \in I_\Sigma} \mu(A_i)$ for A_i disjoint.

The first statement defines that a measure μ only assigns non-negative values to sets in Σ. The second statement equals the measure of the empty set $\varnothing$ to 0. The third statement defines that σ-additivity is required. For any two sets in Σ the measure of the union of the sets equals the sum of the measures of the individual sets. Here I_Σ defines an index over sets in Σ.

Informally, a measure relates the concepts of *sets* and *subsets* to notions of size. A measure can be seen as a *monotonically* increasing function. Let the set A in X be the interval $[0,1)$, an uncountable (infinite) set of real numbers. Define the σ-algebra $\{\varnothing, A\}$. The empty set has measure 0, the set A has measure 1. Let us define the σ-algebra $\{\varnothing, A_{0,0.5}, A_{0.5,1}, A\}$. The set $A_{0,0.5}$ corresponds to the interval $[0,0.5)$ and $A_{0.5,1}$ to $[0.5,1)$. Both sets are assigned measure 0.5 and their union has measure 1. This examples shows that with σ-additive unions, measures can be assigned to sets that are uncountable.

A *measurable space* (X, Σ) is defined as a pair.

▼ Definition A.3 — *measurable space*

A **measurable space** (X, Σ) is a pair with:

- X a set;
- Σ a σ-algebra over X.

A *measure space* (X, Σ, μ) is defined as a triple.

▼ Definition A.4 — *measure space*

A **measure space** (X, Σ, μ) is a triple with:

- X a set;
- Σ a σ-algebra over X;
- μ a measure from Σ to $[-\infty, \infty]$.

A finite measure μ assigns a finite real number to all A.

▼ Definition A.5 — *finite measure*

A **finite measure** μ is a measure from Σ to $[0, \infty)$:

- μ is non-negative: $\mu(A) \geq 0$ for $\forall A \in \Sigma$;
- μ has a null empty set: $\mu(\varnothing) = 0$;
- μ is σ-additive: $\mu(\bigcup_{i \in I_\Sigma} A_i) = \sum_{i \in I_\Sigma} \mu(A_i)$ for A_i disjoint;
- μ for the whole sample space, X, is finite: $\mu(X) = N$.

A σ-finite measure allows A to be a countable union of sets with finite measure.

▼ **Definition A.6 — σ-*finite measure***

A **σ-finite measure** μ is a finite measure with:

- X is a countable union of sets with finite measures.

We will now define five measures: (A.1.1) the *probability measure* (Definition A.7), (A.1.2) the *counting measure* (Definition A.9), (A.1.3) the *Borel measure* (Definition A.11), (A.1.4) the *Lebesgue measure* (Definition A.16), and (A.1.5) the *random measure* (Definition A.17). These measures are important because they are fundamental to different branches of mathematics. In probability theory a σ-algebra is interpreted as a collection of events to which probabilities are assigned. Counting measures play a fundamental role in discrete probability distributions. In integration theory a σ-algebra corresponding to the Borel and Lebesgue measures are relevant for integration in the Euclidean space $\mathscr{R}^n$. In statistics a σ-algebra formally defines the concept of sufficient statistics and generalizes random variables to random functions and measures.

A.1.1 Probability Measure

A *probability measure*, $\mathbb{P}$, is a finite measure that assigns non-negative values $\mathbb{P}$, called probabilities, to sets A, called events (see Definition A.7).

▼ **Definition A.7 — *probability measure***

A **probability measure** $\mathbb{P}$ is a measure μ with:

- $\mathbb{P}$ is non-negative: $\mathbb{P}(A) \geq 0$ for $\forall A \in \Sigma$;
- $\mathbb{P}$ has a null empty set: $\mathbb{P}(\varnothing) = 0$;
- $\mathbb{P}$ is σ-additive: $\mathbb{P}(\bigcup_{i \in I_\Sigma} A_i) = \sum_{i \in I_\Sigma} \mu(A_i)$ for A_i disjoint;
- $\mathbb{P}$ for the whole sample space, X, is unity: $\mathbb{P}(X) = 1$.

The four requirements are called the Kolmogorov axioms (Kolmogorov, 1933). The probability measure is an actual *measure*. It therefore obeys the three requirements: (1) non-negativity for any set, (2) the existence of a null empty set, and (3) σ-additivity. Here we note that a *probability* measure compared to a general measure obeys a fourth requirement, namely the restriction of the measure for the whole space X to 1. This can be seen as some kind of normalization. It influences how two probability measures have to be summed to become again a probability measure.

In Figure A.1 the probability measure is visualized as a mapping from the probability space to the unit interval $[0, 1]$.

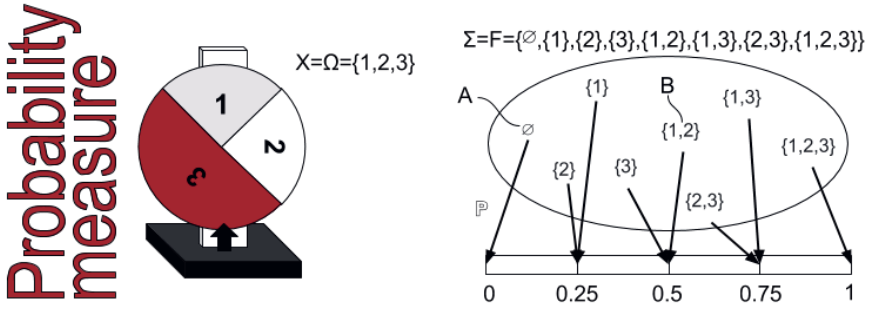


Figure A.1: A probability measure $\mathbb{P}$ mapping the probability space for 3 events to the unit interval. Left: a turning wheel representing three possible outcomes of which the third is twice as likely as the other two outcomes. Right: a probability measure $\mathbb{P}$ assigned to each outcome. The empty set, $A = \emptyset$, has probability measure 0. The set of encountering either 1 or 2, $B = \{1, 2\}$, has probability measure 0.5. Taken from Wikipedia.

A *probability space* $(X, \Sigma, \mathbb{P})$ is a measure space (X, Σ, μ) with the probability measure $\mathbb{P}$ as its measure μ.

▼ **Definition A.8** — ***probability space***

A **probability space** $(X, \Sigma, \mathbb{P})$ is a triple with:

- X a set;
- Σ a σ-algebra over X;
- $\mathbb{P}$ a probability measure from Σ to $[0, 1]$.

We will equivalently use the symbols $(X, \Sigma, \mathbb{P})$ or $(\Omega, \mathbb{F}, \mathbb{P})$ for the probability space, also called probability triple (Rosenthal, 2006). The space X is the event space Ω, the set of *elementary outcomes*. The σ-algebra over subsets of Ω is denoted by $\mathbb{F}$. The probability measure $\mathbb{P}$ assigns a value on the unit interval $[0, 1]$ to every event in $\mathbb{F}$.

A.1.2 Counting Measure

The *counting measure* forms the basis for the definition of discrete probabilities (Schilling, 2005).

▼ **Definition A.9** — ***counting measure***

A **counting measure** ν on a space X is a measure μ with:

- ν is non-negative and integer-valued for $\forall A \in \Sigma$;
- $\nu < \infty$ for $\forall A \in \Sigma$ if A bounded (of finite size);

- $\nu = \infty$ if $\exists A \in \Sigma$ with A unbounded (infinite).

A counting measure is a measure that is integer-valued. Every set A has a measure that is a positive integer or zero. The set A is unbounded if and only if its counting measure is infinite.

A.1.3 Borel Measure

The *Borel σ-algebra* defines a σ-algebra for the real line $\mathbb{R}$.

▼ Definition A.10 — *Borel σ-algebra*

A **Borel σ-algebra** $\mathbb{B}_\sigma$ on $\mathbb{R}$ is the smallest σ-algebra that contains all open subsets of $\mathbb{R}$:

- $\mathbb{B} = \Sigma(U)$ with $U = U \subseteq \mathbb{R}$: U is open.

The Borel σ-algebra contains all open subsets of $\mathbb{R}$. The property of closure under complementation of a σ-algebra means that it also contains the closed subsets of $\mathbb{R}$. If $A = (0,1)$, then $A^c = \{[-\infty, 0], [1, \infty]\}$.

A *Borel measure* assigns values to subsets of $\mathbb{B}_\sigma$.

▼ Definition A.11 — *Borel measure*

A **Borel measure** μ is a function from $\Sigma = \mathbb{B}_\sigma$ to $[-\infty, +\infty]$, with the three measure requirements:

- μ is non-negative: $\mu(A) \geq 0$ for $\forall A \in \Sigma$;
- μ has a null empty set: $\mu(\varnothing) = 0$;
- μ is σ-additive: $\mu(\bigcup_{i \in I_\Sigma} A_i) = \sum_{i \in I_\Sigma} \mu(A_i)$ for A_i disjoint.

The *Borel space* is a measureable space with a Borel σ-algebra rather than a general σ-algebra.

▼ Definition A.12 — *Borel space*

A **Borel space** $(X, \mathbb{B}_\sigma)$ is a pair with:

- X a set;
- $\mathbb{B}_\sigma$ a Borel σ-algebra over X.

A *complete measure space* is a measure space in which every subset of every null set is measurable.

▼ **Definition A.13 — *complete measure space***

A **complete measure space** (X, Σ, μ):

- $S \subseteq N \in \Sigma$ and $\mu(N) = 0 \Rightarrow S \in \Sigma$.

The Borel space is not a complete measure space. There are sets in the Borel σ-algebra that are of measure zero and that contain subsets that are undefined.

A.1.4 Lebesgue Measure

The *Lebesgue measure* defines a size to subsets of $\mathbb{R}^n$ that completes the Borel measure (Lebesgue, 1902). It makes use of the notion of an *outer measure*.

▼ **Definition A.14 — *outer measure***

An **outer measure** ϕ on a space $\mathbb{R}$ is a measure μ with:

- ϕ is non-negative and real-valued for $\forall A \in \Sigma$;
- ϕ has a null empty set: $\phi(\varnothing) = 0$;
- ϕ is σ-subadditive: $\phi(\bigcup_{i \in I_\Sigma} A_i) < \sum_{i \in I_\Sigma} \mu(A_i)$ for $\forall A_i$;
- ϕ is monotone: $A \subseteq B$ implies $\phi(A) \leq \phi(B)$;
- ϕ is translation-invariant: $\phi(A + x) = \phi(A)$ for $\forall A \in \Sigma$ and $\forall x \in \mathbb{R}$.

An outer measure relaxes σ-additivity of disjoint sets of X to σ-subadditivity for any sequence of sets. Intuitively, the outer measure of a set is an upper bound on the size of a set.

▼ **Definition A.15 — *Lebesgue outer measure***

A **Lebesgue outer measure** λ on a space $\mathbb{R}^n$ is an outer measure ϕ with:

- $\lambda(A) = \inf\left\{\sum_{k=1}^{\infty} l(I_k) : (I_k)_{k \in \mathbb{N}} \text{ is a sequence of open intervals with } A \subseteq \bigcup_{k=1}^{\infty} I_k\right\}$.

Here $A \subseteq \mathbb{R}$ is a subset of the real line. The Lebesgue outer measure λ is the infimum (greatest lower bound) of the sum of the lengths $l(I) = b - a$ of the intervals $I = [a, b]$.

The *Lebesgue measure* is defined through the Lebesgue outer measure.

▼ **Definition A.16 — *Lebesgue measure***

A **Lebesgue measure** m on a space $\mathbb{R}^n$ is a Lebesgue outer measure λ with:

- $m(B) = \lambda(B \cup A) + \lambda(B \cup A^c)$.

A.1.5 Random Measure and Random Process

A measurable function is defined between two measurable spaces.

▼ **Definition A.17** — ***measurable function***

A **measurable function** $f : X \to Y$ fulfills:

- $f^{-1}(E) \in \Sigma \qquad \text{for} \qquad \forall E \in T,$

with both (X, Σ) and (Y, T) measurable spaces.

A measurable function *preserves the structure* of the corresponding measurable spaces (captured through the σ-algebras).

A random element or (X, Σ)-valued random variable is a measurable function between two measurable spaces, with as domain a measurable space that is a probability space.

▼ **Definition A.18** — ***random element***

A **random element** or **(X, Σ)-valued random variable** X is a measurable function from probability space $(\Omega, \mathbb{F}, \mathbb{P})$ to measurable space (X, Σ).

An (X, Σ)-valued random variable is visualized in Figure A.2.

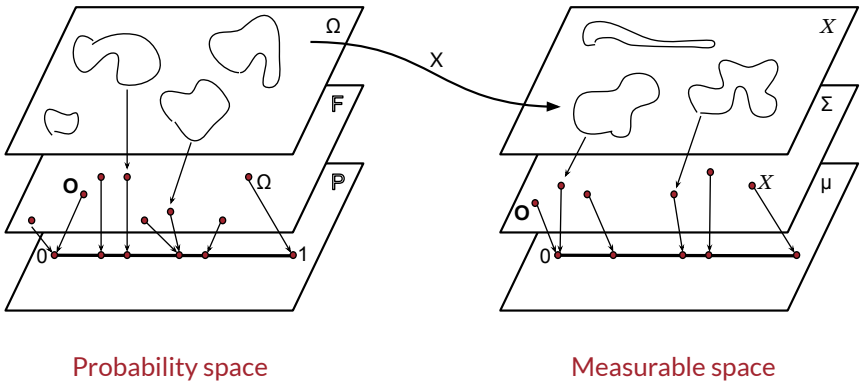


Figure A.2: An (X, Σ)-valued random variable X is a measurable function from $(\Omega, \mathbb{F}, \mathbb{P})$ (at the left) to (X, Σ) (at the right). The planes at the top depict the samples spaces Ω and X. The planes in the middle depict the σ-algebras $\mathbb{F}$ and Σ. The planes at the bottom depict measures: at the left $\mathbb{P}$, and at the right an induced measure μ. The null set is of measure 0. The set Ω is a of measure 1.

For random variables for which we do not specify the codomain explicitly, the choice for the codomain is the real line $\mathbb{R}$ and the corresponding Borel σ-algebra on the reals.

▼ **Definition A.19 — *random variable***

A **random variable** X is a measurable function from probability space $(\Omega, \mathbb{F}, \mathbb{P})$ to the real line with the Borel σ-algebra $(\mathbb{R}, \mathbb{B}_{\mathbb{R}})$.

A $(\mathbb{R}, \mathbb{B}_{\mathbb{R}})$-valued random variable is also called a real-valued random variable assuming a natural choice for the σ-algebra, or called a random variable assuming the reals.

> Random elements are a generalization of random variables. A *complex*-valued random variable or *complex random variable* is a measurable function from Ω to $\mathbb{C}$. An *elephant*-valued random variable or *random elephant* is a measurable function from Ω to a suitable space of elephants (Kingman, 1993).

This allows us to define a *measure*-valued random variable, a random measure.

▼ **Definition A.20 — *random measure***

A **random measure** is a function $\xi : \Omega \times X \to [0, +\infty]$ from probability space $(\Omega, \mathbb{F}, \mathbb{P})$ to measurable space (X, Σ) such that $\xi(\cdot, X)$ is a random variable on $(\Omega, \mathbb{F}, \mathbb{P})$ and $\xi(\omega, \cdot)$ is a measure on Σ.

We are now in the position to define a random process (the Dirichlet process in this thesis is an example of such a process). A *random process* is an ordered set of random variables. The set can be a sequence of random variables in a time series. It can be a series of steps in the spatial domain, called a random field.

▼ **Definition A.21 — *random process***

A **random process** X is a collection $\{X_t : t \in T\}$ with X_t an (S, Σ)-valued random variable on Ω and $(\Omega, \mathbb{F}, \mathbb{P})$ a probability space, (S, Σ) a measurable space, and T a totally *ordered* set.

A random process is a probability distribution with a domain that is a set of probability distributions. A random process is a distribution over distributions, a hierarchy over distribution.

Before we close this section, we will introduce two more concepts. The *distribution* of a random variable and the *probability density function* of a random variable.

We have encountered a random variable, and a probability measure $\mathbb{P}$ on the original probability space. Now, one might wonder whether there is a measure that is logically assigned to elements on the measurable space that is the codomain of this random variable. Is there a natural measure μ that can transform this measurable space into a measure space? It turns

out there is. There is a measure *induced* on this space by the random variable.[1] This measure μ is known as the *distribution* or *law* of a random variable (Rosenthal, 2006):

▼ **Definition A.22 — *distribution of a random variable***

Given a random variable X from $(\Omega, \mathbb{F}, \mathbb{P})$ to $(\mathbb{R}, \mathbb{B}_\sigma)$, the **distribution** μ of X is the induced probability measure: $\mu(B) = \mathbb{P}(X^{-1}(B))$ for all Borel sets $B \in \mathbb{B}_\sigma$.

The distribution of X is the probability measure μ induced on $(\mathbb{R}, \mathbb{B}_\sigma)$. This makes this space a measurable space $(\mathbb{R}, \mathbb{B}_\sigma, \mu)$. We will write X as being *distributed as* μ in the following shorthand notation:

$$X \sim \mu. \tag{A.1}$$

A measure ν is *absolutely continuous* with respect to a measure λ if, for every set E, $\lambda(E) = 0$ implies $\nu(E) = 0$. We also write this as $\nu \ll \lambda$. The measure ν is *dominated* by λ. The Radon-Nikodym theorem states that for two σ-finite measures one measure can be expressed as an integral of the other.

▼ **Definition A.23 — *Radon-Nikodym theorem***

The **Radon-Nikodym theorem** states that given a measurable space (X, Σ) and two σ-finite measures, ν, λ with $\nu \ll \lambda$, that there exists a Σ-measurable function $f : X \to [0, \infty)$, such that for any measurable set $A \subseteq X$,

$$\nu(A) = \int_A f \, d\lambda. \tag{A.2}$$

This allows us to define the Radon-Nikodym derivative: $f = \frac{d\nu}{d\lambda}$. The probability density function f of a random variable X is the Radon-Nikodym derivative of the induced measure (with respect to some base measure, normally the Lebesgue measure).

▼ **Definition A.24 — *probability density function***

The **probability density function** f of a random variable X is the Radon-Nikodym derivative of the induced measure μ on $(\mathbb{R}, \mathbb{B}_\sigma)$ with respect to a base measure λ,

$$f = \frac{d\mu}{d\lambda}. \tag{A.3}$$

For a discrete random variable the counting measure can be used as a base measure. For continuous random variables the Lebesgue measure is usually chosen as base measure.

[1] The measure induced on a measurable space by another measurable space by means of a measurable function is also known as a push-forward measure.

A.2 Bayesian Inference

Let x be a (S, Σ_S, μ_S)-valued random variable, y a (T, Σ_T, μ_T)-valued random variable, then we can construct z, a (C, Σ_C, μ_C)-valued random variable with the latter being a subset of the product set of x and y: $C \in S \otimes T$.

▼ **Definition A.25 — *product space***

A **product space** $(S \otimes T, \Sigma_{S\otimes T})$ has σ-algebra $\Sigma_{S\otimes T} = \sigma(F \otimes G : F \in \Sigma_S, G \in \Sigma_T)$ with (S, Σ_S, μ_S) and (T, Σ_T, μ_T) two σ-finite measure spaces.

▼ **Definition A.26 — *product measure***

A **product measure** $\mu_{S\otimes T}$ is a measure $\mu_{S\otimes T}(F \otimes G) = \mu_S(F) \otimes \mu_T(G)$ with (S, Σ_S, μ_S) and (T, Σ_T, μ_T) two σ-finite measure spaces.

The **joint probability distribution** P_C is a probability measure on the product σ-algebra Σ_C with $C \in S \otimes T$. As function of the random variables x and y the joint probability distribution is written as $_{X,Y}(x, y)$, $f(x, y)$, or $p(x, y)$.

A σ-algebra is *independent* in the following sense.

▼ **Definition A.27 — *independent σ-algebra***

Let $(\Omega, \mathbb{F}, P)$ be a probability space and $\mathbb{A}$ and $\mathbb{B}$ be a sub-σ-algebras of $\mathbb{F}$. $\mathbb{A}$ and $\mathbb{B}$ are **independent σ-algebras** if:

- $P(A \cap B) = P(A)P(B)$ $\forall A \in \mathbb{A}$ and $B \in \mathbb{B}$.

Two random variables x and y are independent if and only if the σ-algebras that they generate are independent.

▼ **Definition A.28 — *conditional probability distribution***

Let $(\Omega, \mathbb{F}, P)$ be a probability space, $\mathbb{G} \subseteq \mathbb{F}$ a sub-σ-algebra of $\mathbb{F}$, and $X : \Omega \to \mathbb{R}$ a real-valued random variable ($\mathbb{F}$-measurable with respect to the Borel σ-algebra $\mathbb{B}_\sigma$ on $\mathbb{R}$). There exists a function $\mu : \mathbb{B}_\sigma \times \Omega \to \mathbb{R}$ such that $\mu(\cdot, \omega)$ is a probability measure on $\mathbb{B}_\sigma$ for each $\omega \in \Omega$ and $\mu(H, \cdot) = P(X \in H|\mathbb{G})$ (almost surely) for every $H \in \mathbb{B}_\sigma$. For any $\omega \in \Omega$, the function $\mu(\cdot, \omega) : \mathbb{B}_\sigma \to \mathbb{R}$ is called a **conditional probability distribution** of X given $\mathbb{G}$.

Informally[2], a conditional probability is described with a sub-σ-algebra which only presents part of the structure of the full σ-algebra. As function of the random variables x and y the conditional probability distribution of y given x is written as $f_{Y|X}(y|x)$, $f(y|x)$, or $p(y|x)$.

A typical conditional probability distribution is that of the data given parameters. Another often used conditional probability distribution is that of the data given a statistic (summary) of that data. This statistic can be a so-called sufficient statistic.

▼ Definition A.29 — *sufficient statistic*

A conditional probability distribution of the data X given a *sufficient statistic* $t = T(X)$ does not depend on parameter θ:

- $P(x|t,\theta) = P(x|t)$

Random variables, or more generally, random elements x and θ define a Bayesian[3] model with observations x and parameters θ.

▼ Definition A.30 — *Bayesian model*

A **Bayesian model** $f(x,\theta)$ defines a function, a joint probability distribution, over observations x and parameters θ with both x and θ random elements.

In a **supervised learning** task both x and θ are known. In an **unsupervised learning** task x is known, but θ is unknown. The random variable θ is called a hidden or latent variable. The random variable θ can be any random element: a random vector, a random matrix, a random process.

Let the observations x be a sequence $x_0, x_1, \ldots$, then the observations x_i can be *independent and identically* distributed.

▼ Definition A.31 — *independent and identically distributed*

A collection of random variables $x = \{x_0, x_1, \ldots\}$ is **independent and identically distributed (i.i.d.)** if:

- the probability distribution $p(x_i)$ is the same for $\forall x_i \in x$
- each x_i is independent with respect to x_j with $i \neq j$.

In other words, random variables having the same distribution are said to be identically distributed.

[2] Even more informally, in "254A, Notes 0: A review of probability theory" Tao describes how conditioning can be seen as removing a *partial* amount of randomness consistent with the probabilistic way of thinking. By conditioning a random variable to be fixed, one can turn that random variable into a deterministic one, while preserving the random nature of other variables.

[3] A historic perspective on the term Bayesian can be found in (Fienberg et al., 2006).

The observations x_i can be distributed in an *exchangeable* sequence in which any order is equally likely.

▼ **Definition A.32 — *exchangeable***

A sequence of random variables $x = \{x_0, x_1, \ldots\}$ is **exchangeable** if for any finite permutation ρ of the indices $0, 1, \ldots$:

- the joint probability distribution of the permuted sequence $p(x_{\rho(0)}, x_{\rho(1)}, \ldots)$ equals that of the original sequence $p(x_0, x_1, \ldots)$.

The joint probability distribution of i.i.d. observations given parameters can be written as a product:

$$p(x_0, \ldots, x_{k-1}|\theta) = \prod_{i=0}^{k-1} p(x_i|\theta). \tag{A.4}$$

▼ **Definition A.33 — *likelihood function***

The **likelihood function** is defined as:

$$L(\theta; x) = p(x = X|\theta). \tag{A.5}$$

The likelihood indicates the probability that a particular value $x = X$ is observed when the parameter is considered to be θ.

The likelihood function allows us to find an optimal set of parameter values given the observations. We can find those parameters that maximize the likelihood, $L(\theta)$, given the observations, X, see Aldrich (1997). This maximization method is called maximum likelihood estimation (MLE).

▼ **Definition A.34 — *maximum likelihood estimation***

Maximum likelihood estimation is defined as the method optimizing:

$$\theta^* \in \underset{\theta}{\operatorname{argmax}}\, L(\theta; x). \tag{A.6}$$

The maximum likelihood method finds the maximum of $p(x|\theta)$ for all possible parameter values θ. The maximum in maximum likelihood estimation does not need to be unique (Steel, 1994). The notation makes this explicit by writing θ^* as a member (denoted by the $\in$ symbol) of the outcomes of the argmax operation (and does not use the equal sign).

In the case we have information about the parameters θ we can model this with a probability distribution.

▼ **Definition A.35 — *prior probability distribution***

A **prior probability distribution** defines a probability distribution $p(\theta)$ over parameters θ without a dependency on the observations x.

Given the definition of a prior probability distribution, we can define *maximum a posteriori* estimation.

▼ **Definition A.36 — *maximum a posteriori***

Maximum a posteriori estimation is defined as:

$$\theta^* \in \underset{\theta}{\operatorname{argmax}} \sum_{i=0}^{k-1} \log p(x_i|\theta) + \log p(\theta). \tag{A.7}$$

If we are not only interested in the parameter θ^* that maximizes $p(x|\theta)$ and $p(\theta)$, but in the complete distribution for $p(\theta)$ we need Bayes' theorem described by Laplace (1820).

▼ **Definition A.37 — *Bayesian inference***

Bayesian inference using Bayes' theorem is defined as:

$$p(\theta|x) = \frac{\overbrace{p(x|\theta)}^{\text{likelihood}}\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(x)}_{\text{normalization constant}}} = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}. \tag{A.8}$$

Bayes' theorem describes the posterior probability $p(\theta|x)$ as the likelihood times the prior probability distribution divided by a normalization constant, also called the evidence. The normalization constant is not a function of the parameters θ. If a function is known except for the normalization constant, it is indicated by the "proportional to" symbol $\propto$.

$$f(\theta|x) \propto p(x|\theta)p(\theta) \tag{A.9}$$

In Bayesian inference $p(\theta|x)$ is calculated. In contrast, in maximum likelihood and maximum a posteriori only parts of Eq. A.8 are calculated, respectively $p(x|\theta)$ and $p(x|\theta)p(\theta)$. In Section 2.2 inference methods will be described that approximate Bayesian inference. Approximation is required in the case closed-form expressions are not available. If the inference task only requires maximum a posteriori, approximation methods are also available (Daume, 2007), but this is outside of the scope of the current thesis.

It is important to note that Bayes' rule does not always apply. Recall the definition of the probability density function (Definition A.24) in Appendix A.1.5 for which we needed the notion of absolutely continuity. The posterior is not always absolutely continuous with respect to the prior. In particular for nonparametric Bayesian models this is not necessarily

the case. For example, the Dirichlet process as a prior has a posterior that is typically orthogonal to the prior. However, using appropriate care it is still the case that the posterior is well-defined and one can perform Bayesian inference without using Bayes' theorem. To read more on the exact conditions under which this is possible, we refer the reader to (Ghosal and Van der Vaart, 2017).

There are two supervised learning models, a generative model and a discriminative model. Below we provide their definitions and in Figure A.3 we give three examples for each model.

▼ **Definition A.38** — ***generative model***

A **generative** model defines the joint probability distribution $p(x, \theta)$.

▼ **Definition A.39** — ***discriminative model***

A **discriminative** model defines the conditional probability distribution $p(\theta|x)$ directly.

Figure A.3 shows three generative and three discriminative models. They are chosen for their structure. From left to right, the structure between the random variables gets enriched. The first column shows no particular structure. The second column shows a sequence structure. The third column shows a graph structure. Figure A.3 visualizes three generative models: (1) the Naive Bayes Model (Russell et al., 1995), (2) the Hidden Markov Model (Baum and Petrie, 1966), and (3) the Directional Model (Koller and Friedman, 2009). It shows also three discriminative models: (1) Logistic Regression, (2) Linear-chain Conditional Random Fields, and (3) general Conditional Random Fields.

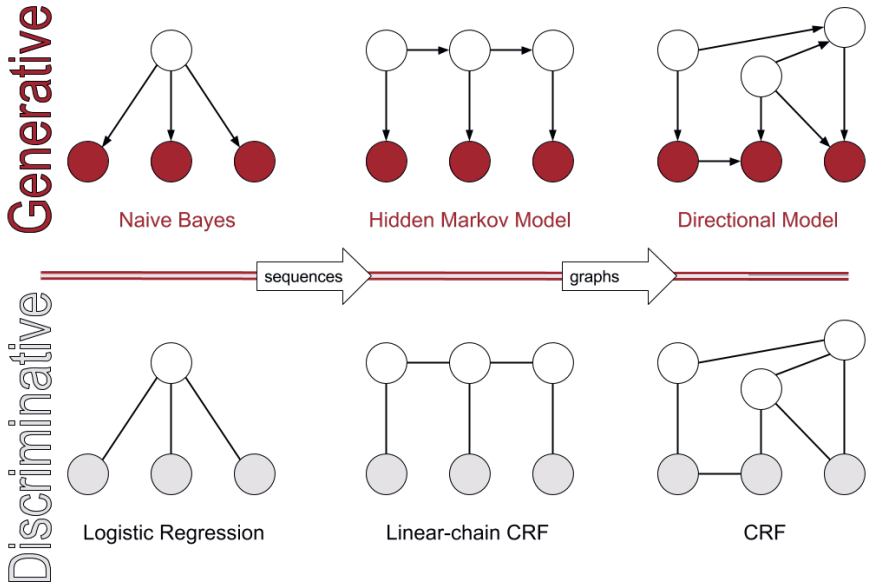


Figure A.3: Generative models: Naive Bayes Model, Hidden Markov Model, and Directional Model. Discriminative models: Logistic Regression, Linear-chain Conditional Random Fields, and general Conditional Random Fields. Figure adapted from Sutton and McCallum (2011).

There is no definitive reason to use a generative model rather than a discriminative model or vice-versa. Here we confine ourselves to two remarks. First, a discriminative model seems to have a lower asymptotic error, but a generative model seems to approaches its (higher) asymptotic error faster. This has been studied using a Naive Bayes classifier versus Logistic Regression (Jordan, 2002). This would mean that a discriminative model would be better for large datasets, while a generative model would be better for small datasets. However, Xue and Titterington (2008) doubt the existence of such precisely defined regimes depending on dataset size. According to them the behaviour seems to stem from the correctness of the conditional or the joint model specification. Second, the prior $p(\theta)$ in the generative model provides a principled way to handle missing information, while the direct modeling of decision boundaries in a discriminative model often leads to better performance in a classification task (Jaakkola et al., 1999). Apart from generative models and discriminative models, there are also hybrid models (Bouchard and Triggs, 2004; Raina et al., 2003; Bosch et al., 2008). In the thesis we will restrict ourselves to generative models.

A.3 Model Composition

A model can be composed out of a set of probability distibutions. We list three of such possible compositions. The Naive Bayes model is a *product* of probability distributions with a prior distribution (Definition A.40). The finite mixture model is a *sum* over a finite number

of probability distributions where each one is weighted (Definition A.41). The infinite mixture model is a *sum* over an infinite number of probability distributions where each one is weighted (Definition A.42).

▼ **Definition A.40 — *naive Bayes model***

The **naive Bayes model** is a product over a finite number $k \neq \infty$ of probability distributions $p(x_i|\theta)$ multiplied by the prior distribution $p(\theta)$:

$$p(\theta|x) \propto p(\theta)\prod_{i=0}^{k-1} p(x_i|\theta). \tag{A.10}$$

A finite mixture model is a sum over a finite number of probability distributions.

▼ **Definition A.41 — *finite mixture model***

A **finite mixture model** is a sum over a finite number $k \neq \infty$ of probability distributions $p(x_i)$, with each distribution weighted by a factor w_i with $\sum_i w_i = 1$.

$$p(x) = \sum_{i=0}^{k-1} w_i p(x_i). \tag{A.11}$$

The mixture model is finite in the sense that there are only $k \neq \infty$ distributions summed up. The weights of the individual distributions $p(x_i)$ are normalized (sum up to one) such that the weighted sum over the probability distributions is itself a probability distribution.

An infinite mixture model is a sum over an infinite number of probability distributions.

▼ **Definition A.42 — *infinite mixture model***

A **infinite mixture model** is a sum over an infinite number of probability distributions $p(x_i)$, with each distribution weighted by a factor w_i with $\sum_i w_i = 1$.

$$p(x) = \sum_{i=0}^{\infty} w_i p(x_i). \tag{A.12}$$

The infinite mixture model is a sum over an infinite number of probability distributions with weights that sum up to one. In this way it assigns a finite value to a countably infinite set of functions.

If the number of probability distributions is uncountable infinite, we speak about a compound distribution.

▼ **Definition A.43 — *compound probability distribution***

A **compound probability distribution** for a probability density function $p(\theta)$ (nonnegative and integrating to 1) is given by

$$p(x) = \int_{\Omega} p(\theta)p(x|\theta)d\theta. \tag{A.13}$$

Informally, $p(\theta)$ has the same function as the weight in a mixture model. From this presentation it is also clear that a compound distribution is a special case of a marginal distribution. The joint distribution $p(x,\theta) = p(\theta)p(x|\theta)$. The compound distribution is obtained through its marginal distribution: $\int p(x,\theta)d\theta$. In the thesis we will encounter infinite mixture models or compound probability distributions in Chapters 3 and 4.

A.4 General Random Elements

In section A.1 random elements were described in general. Random elements can vary from random vectors, random distributions, random clusters (partitions), to random trees. Table A.1 describes the random elements and the corresponding examples of random processes in the literature. Below we mention them with the appropriate references.

Table A.1: A list of seven mathematical structures and for each of these structures one or more random processes that can generate the structure. For example, a distribution on distributions can be generated by a Beta Process, Gamma Process, Dirichlet Process, or a Polya Tree.

Structure	Example
Distribution on functions	Gaussian Process
Distribution on distributions	Beta Process Gamma Process Dirichlet Process Polya Tree
Distribution on partition assignments	Chinese Restaurant Process Pitman-Yor Process
Distribution on partition sizes	Stick-breaking Process
Distribution on hierarchical partitions	Dirichlet Diffusion Tree Kingman's coalescence
Distribution on sparse binary matrices	Indian Buffet Process
Distribution on integer-valued matrices	Gamma-Poisson Process
Distribution on kd-trees	Mondrian Process

The Gaussian Process (Rasmussen and Williams, 2006) describes a distribution on functions. The Beta Process (Hjort, 1990), the Gamma Process (Ferguson, 1974), the Dirichlet Process and the Polya Tree (Ferguson, 1973) describe a distribution on distributions. The Chinese Restaurant Process (Aldous, 1985) and Pitman-Yor Process (Pitman and Yor, 1997) describe

a distribution on partitions (in the form of cluster assignments). The Stick-breaking Process describes a distribution on partition sizes (with no information on assignments themselves). The Dirichlet Diffusion Tree (Neal, 2001) and Kingman's coalescence (Kingman, 1965) describe a distribution on hierarchical partitions. The Indian Buffet Process (Ghahramani and Griffiths, 2005) describes a distribution over sparse binary matrices. The Gamma-Poisson Process (Titsias, 2008) describes a distribution over integer-valued matrices. The Mondrian Process (Roy and Teh, 2009) describes a distribution over kd-trees.

A.5 Plate Notation

Random processes and mixture models are visually represented by a method called *plate notation* (cf. Buntine, 1994; Koller and Friedman, 2009). Sets of variables are represented in a plate, a rectangular region (see Figure A.4).

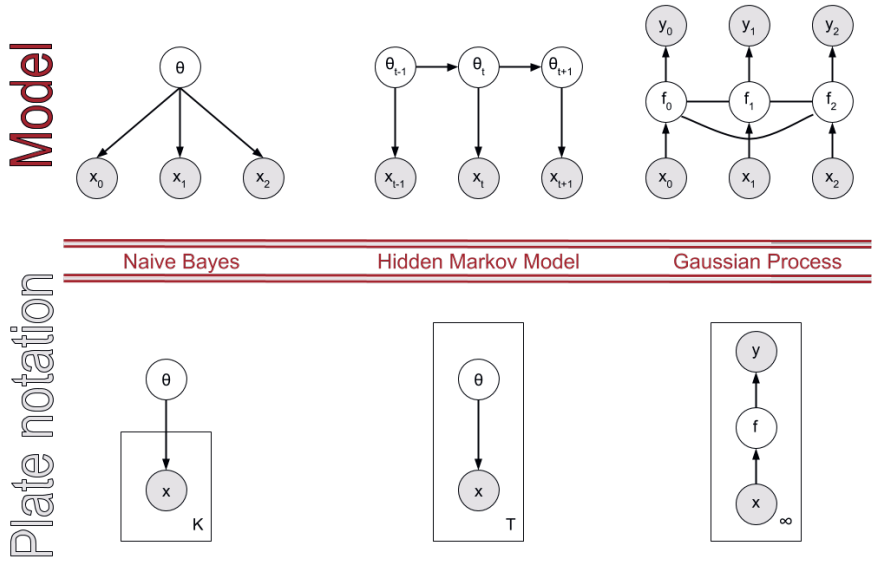


Figure A.4: Top: graphical model of a Naive Bayes, hidden Markov model, and Gaussian process. Bottom: corresponding plate notation of the Naive Bayes, hidden Markov model, and Gaussian process. Observed variables are denoted by a circle that is shaded.

Plate notation is a representation that does not preserve all dependencies between variables. For example, the dependencies between the states in the Hidden Markov Model (e.g., between θ_0 and θ_1) are not represented. The Gaussian process has a potentially infinite number of parameters. The use of plate notation for nonparametric models can be found in (Fox et al., 2007).

A.6 Completely Random Measure and Lévy Measure

Some random process are mathematically represented by a completely random measure (Kingman, 1967), which is defined as follows.

▼ **Definition A.44** — ***completely random measure***

A **completely random measure** is a random measure $\mu : \Omega \times X \to [0, +\infty]$ from probability space $(\Omega, \mathbb{F}, \mathbb{P})$ to measurable space (X, Σ) with

- for any collection of disjoint sets $A_1, \ldots, A_k \in \Sigma$ and $A_i \cap A_j = \emptyset$ for $i \neq j$ a mutual independency between $\mu(A_1), \ldots, \mu(A_k)$.

Kingman (1967) shows that a completely random measure can be decomposed into three components:

1. a deterministic function;
2. a countable set of non-negative random masses at deterministic locations;
3. a countable set of non-negative random masses at random locations.

The first component is a deterministic function. The second component has non-negative random masses, also called atoms, on deterministic locations. The third component is the one of interest. It has a set of random masses (atoms) that can be represented as a Poisson random measure on $\mathbb{R}^+ \otimes X$ with mean measure ν which is known as the Lévy intensity measure (Favaro et al., 2013).

Table A.2: Lévy measure of the Beta Process (Wang and Carin, 2012), Gamma Process (Knowles et al., 2014), the Dirichlet Process (Lijoi and Prünster, 2010) (indirectly through $F = 1 - e^{-\nu}$) .

Random Process	Lévy measure
Beta Process	$\nu(da, dw) = H(da)\alpha w^{-1}(1-w)^{\alpha-1}dw$
Gamma Process	$\nu(da, dw) = H(da)w^{-1}e^{-\alpha w}dw$
Dirichlet Process	$\nu(da, dw) = H(da)e^{-w\alpha(x,\infty)}(1-e^{-w})^{-1}dw$

For Lévy measure decompositions of other processes such as the Indian buffet process, we refer to Wang and Carin (2012).

A.7 Exchangeability

Here we recall Definition A.32 for exchangeable sequences. De Finetti's theorem states that there is parameter θ such that the data x_i is conditionally independent given this parameter for exchangeable sequences (cf. De Finetti, 1937).

▼ **Definition A.45 — *De Finetti's theorem***

A sequence $\{x_0, x_1, \ldots\}$ of (X, Σ_X)-valued random variables is an infinitely exchangeable sequence if and only if there exist a measure $\mu(d\theta)$ on θ such that

$$p(x_0, \ldots, x_{k-1}) = \int_\Omega \prod_{i=0}^{k-1} p(x_i|\theta)\mu(d\theta) \qquad \forall k \geq 1. \tag{A.14}$$

In words, de Finetti's theorem states that if we have *exchangeable* data, we have a parameter θ, a likelihood $p(x|\theta)$, and some measure μ on θ, such that the data $(x_0, \ldots, x_{k-1})$ is *conditionally independent*. Hence, although the data is not i.i.d., there are underlying, unobservable, quantities that are i.i.d. and exchangeable sequences are mixtures of these quantities. The theorem proofs that if the observations are exchangeable, they must be a random sample from some model and there must exist a prior probability distribution over the parameters of that model, hence requiring a Bayesian approach.

The theorem is not limited to exchangeable *sequences*. In contrast, there are similar theorems for other exchangeable objects (Orbanz and Roy, 2015). Five examples (see Table A.3) of exchangeable structures have a theorem describing an underlying measure that can be sampled i.i.d. are: (1) exchangeable sequences (De Finetti, 1930), (2) increments (Bühlmann, 1960), (3) partitions (Kingman, 1978), (4) arrays (Aldous, 1981), and (5) Markov chains (Diaconis and Freedman, 1980).

Table A.3: Five exchangeable structures and their theorems.

Mathematical Object	Theorem
Exchangeable Sequence	de Finetti
Exchangeable Increment	Bühlmann
Exchangeable Partition	Kingman
Exchangeable Array	Aldous-Hoover
Exchangeable Markov Chain	Diaconis-Freedman

A.8 Stick-breaking Representation

Below we introduce the *stick-breaking representation* by Freedman and Diaconis (1983), also known as the residual allocation model (Sawyer and Hartl, 1985; Hoppe, 1986).

▼ **Definition A.46 — *stick-breaking***

An infinite sequence of random variables $\phi = \{\phi_0, \phi_1, \ldots\}$ has a **stick-breaking representation** with parameters α and β denoted by $\phi \sim GEM(\alpha, \beta)$.

$$w_k \overset{iid}{\sim} Beta(1-\beta, \alpha + k\beta) \qquad k = 1, \ldots, K \tag{A.15}$$

$$\phi_k = w_k \prod_{i=1}^{k-1} (1 - w_i) \tag{A.16}$$

The stick-breaking process samples repeatedly from a $Beta(1-\beta, \alpha+k\beta)$ distribution. The result of the process is a vector of k weights ϕ_k. The abbrevation *GEM* stands for Griffiths, Engen, and McCloskey (Ewens, 1990; Ethier, 1990). There is also a variant of GEM with a single parameter α which can be obtained by setting $\beta = 0$. In that case w_k are drawn from a $Beta(1, \alpha)$ distribution. Note that although w_k are sampled i.i.d., the resulting stick sizes ϕ_k are not independent. Stick size ϕ_k depends not only on w_k, but also on the weights $w_1, \ldots w_{k-1}$ drawn previously.

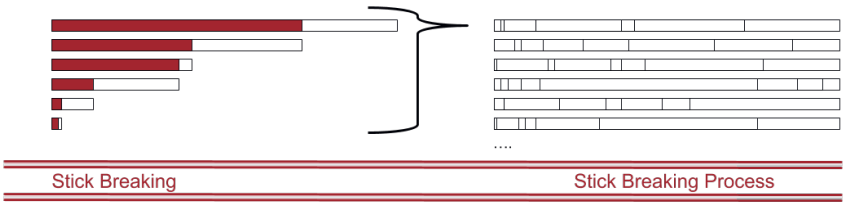


Figure A.5: The stick-breaking representation. Left: at the first row, the stick is broken at x_0, at the next rows the remaining part of the stick is broken x_i with $i > 0$. Only six iterations are shown. Right: samples of a stick-breaking process. The first row shows the stick ratios from the stick-breaking representation at the left. The next rows show other samples from the same process.

Figure A.5 visualizes the stick-breaking process. A stick of fixed length 1 gets broken at a position w_0 sampled from a Beta distribution. The remainder of the stick is broken again at position $w_1(1 - w_0)$. This process continues for an infinite number of times. A stick-breaking process generates in this manner a sequence of non-negative values that sum up to one. The stick-breaking representation can on itself give rise to more sophisticated stochastic processes (Dunson et al., 2012). Computationally it can also fulfill a useful role. Namely, it is possible to approximate a distribution over partitions by truncating a stick-breaking process. The stick-breaking procedure is then only performed a limited number of times (Kurihara et al., 2007).

In Section 2.1 the relevance of the stick-breaking process for the Dirichlet process will be shown. In that case the values generated by the stick-breaking process represent the weights of the partitions induced by the Dirichlet Process.

APPENDIX B

IMPLEMENTATION

We describe two initialization algorithms. The first algorithm initializes Gibbs sampling over parameters. The second algorithm initializes Gibbs sampling over clusters.

B.1 Initialization of Gibbs Sampling over Parameters

Algorithm 8 as shown in Section 3.3 does not describe how the parameters are initialized. The algorithm to initialize the parameters θ_i is given in Algorithm 15.

Algorithm 15 Gibbs sampling over parameters. The initialization of θ_i.

procedure GIBBS ALGORITHM 1 INITIALIZATION(w, λ_0, α) ▷ Accepts points w, hyperparameters λ_0, α and returns k initial line coordinates
  $\lambda_1 = U_{up}(w_1, \lambda_0)$ ▷ Update hyperparameter with w_1 (Eq. 3.21)
  $\theta_1 \sim NIG(\lambda_1)$ ▷ Sample θ_1 from NIG (Eq. 3.24)
  for all $i = 2 : N$ **do**
    $M = i - 1$ ▷ Let M define the number of parameters assigned up to now
    $r_i = \alpha \int F(w_i; \theta) dH$ ▷ Weighted posterior predictive of w_i (Eq. 3.29)
    for all $j = 1 : M$ **do**
      $L_{i,j} = F(w_i; \theta_j)$ ▷ Likelihood of a line given an observation (Eq. 3.9)
    end for
    $p(\theta_{new}) = \frac{r_i}{r_i + \sum_i L_{i,j}}$ ▷ Probability of sampling a new parameter (Eq. 3.31)
    $u \sim U(0, 1)$
    if $p(\theta_{new}) > u$ **then** ▷ Sample with probability $p(\theta_{new})$
      $\lambda_n = U_{up}(w_i, \lambda_0)$ ▷ Update hyperparameters with w_i (Eq. 3.21)
      $\theta_i \sim NIG(\lambda_n)$ ▷ Sample θ_i from NIG (Eq. 3.24)
    else
      $i \sim Mult(M, p(\theta_{old}))$ ▷ Sample i from existing parameters, θ_{old}
      $\theta_i = \theta_{old=i}$ ▷ Pick θ_i given index i
    end if
  end for
  return initialized θ_k for k lines
end procedure

Let us recall the posterior predictive Eq. 3.28:

$$\theta_i \mid \theta_{-i}, w_i \sim r_i H_i + \sum_{j \neq i} F(w_i; \theta_j)\delta_{\theta_j}. \tag{B.1}$$

We initialize through:

$$\begin{aligned} \theta_1 \mid w_1 &\sim H_1 \\ \theta_i \mid \theta_1, \dots \theta_{i-1}, w_i &\sim r_i H_i + \sum_{j=1}^{i-1} F(w_i; \theta_j)\delta_{\theta_j}. \end{aligned} \tag{B.2}$$

Given that j runs up to $i-1$, we do not have to specify $i \neq j$ in different lines of the algorithm (compare with Algorithm 8). The initialization algorithm is so similar from the Gibbs sampling algorithm itself, that it is recommended to write the implementation in such a way that the same function can be used.

B.2 Initialization of Gibbs Sampling over Clusters

Algorithm 9 as shown in Section 3.4 requires initialization of the hyperparameters λ_k per cluster k. In contrast to Algorithm 15 we need to initialize not just θ_k, but also the hyperparameters per cluster. This can be done by calling Eq. 3.21 successively by each observation w_i assigned to cluster k. We also require θ_k themselves to calculate $F(w_i; \theta_j)$ for $j \neq i$ in Eq. 3.9 and $p(\theta_{-i})$, or more specific, $p(\theta_{old})$.

Algorithm 16 Gibbs sampling over clusters. The initialization of θ_k and λ_k.

procedure GIBBS ALGORITHM 2 INITIALIZATION(w, λ_0, α) ▷ Accepts points w and hyperparameters λ_0 and α, returns k hyperparameters λ_k and initial parameters θ_k
  for all $k = 1 : K$ **do**
    $m_k = 0$ ▷ Set number of data points per cluster to 0
  end for
  for all $i = 1 : N$ **do**
    $k = U(\{1, \dots, K\})$ ▷ Sample k from discrete uniform distribution
    cluster$(w_i) = k$ ▷ Assign cluster index k to observation w_i
    if $m_k = 0$ **then**
      $\lambda_k = U_{up}(w_i, \lambda_0)$ ▷ Set hyperparameter λ_k with prior pred. given w_i
    else
      $\lambda_k = U_{up}(w_i, \lambda_k)$ ▷ Update hyperparameter λ_k with posterior pred. given w_i
    end if
    $m_k = m_k + 1$
  end for
  for all $k = 1 : K$ **do**
    $\theta_k \sim NIG(\lambda_k)$ ▷ Sample θ_k from NIG with up to date λ_k
  end for
  return initialized parameters θ_k and hyperparameters λ_k for k lines
end procedure

List of Figures

List of Tables

List of Abbreviations

ABC	approximate Bayesian computation
CD	Chamfer distance
CRP	Chinese restaurant process
DP	Dirichlet process
DPM	Dirichlet process mixture
EMD	earth mover's distance
GEM	Griffiths, Engen, and McCloskey
HDP	hierarchical Dirichlet process
IBP	Indian buffet process
ILM	infinite line model
ISM	infinite segment model
MAP	maximum a posteriori
MCMC	Markov chain Monte Carlo
ML	maximum likelihood
MLE	maximum likelihood estimation
NB	naive Bayes
NIG	Normal-Inverse-Gamma
PEMD	partitioning earth mover's distance
ReLU	rectified linear unit
SAMS	sequentially allocated merge-split
SEMD	shifted earth mover's distance

SUMMARY

In this dissertation non-parametric Bayesian methods are used in the application of robotic vision. Robots make use of depth sensors that represent their environment using point clouds. Non-parametric Bayesian methods can (1) determine how good an object is recognized, and (2) determine how many objects a particular scene contains. When there is a model available for the object to be recognized and the nature of perceptual error is known, a Bayesian method will act optimally.

In this dissertation Bayesian models are developed to represent geometric objects such as lines and line segments (consisting out of points). The *infinite line model* and the *infinite line segment model* use a non-parametric Bayesian model, to be precise, a Dirichlet process, to represent the number of objects. The line or the line segment is represented by a probability distribution. The lines can be represented by conjugate distributions and then Gibbs sampling can be used. The line segments are not represened by conjugate distributions and therefore a split-merge sampler is used.

A split-merge sampler fits line segments by assigning points to a hypothetical line segment. Then it proposes splits of a single line segment or merges of two line segments. A new sampler, the *triadic split-merge sampler*, introduces steps that involve three line segments. In this dissertation, the new sampler is compared to a conventional split-merge sampler. The triadic sampler can be applied to other problems as well, i.e., not only problems in robotic perception.

The models for objects can also be learned. In the disseration this is done for more complex objects, such as cubes, built up out of hundreds of points. An auto-encoder then learns to generate a representative object given the data. The auto-encoder uses a newly defined reconstruction distance, called the *partitioning earth mover's distance*. The object that is learned by the auto-encoder is used in a triadic sampler to (1) identify the point cloud objects and to (2) establish multiple occurances of those objects in the point cloud.

SAMENVATTING

In deze studie worden niet-parametrische Bayesian methoden toegepast in visuele robotperceptie. Robots maken gebruik van dieptesensoren die de omgeving representeren met behulp van puntenwolken. Niet-parametrische Bayesian methoden zijn heel goed in staat om te bepalen (1) hoe goed een object wordt herkend, en (2) hoeveel objecten een scene bevat. Als er een model voor het te herkennen object bestaat en de aard van de te verwachten perceptuele fout is bekend, dan is een Bayesian methode optimaal.

In deze studie worden Bayesian modellen ontwikkeld om geometrische objecten zoals lijnen en lijnsegmenten (opgebouwd uit punten) te representeren. Het *infinite line model* en het *infinite line segment model* gebruiken een niet-parametrisch Bayesian model, om precies te zijn, een Dirichlet process, om het aantal objecten te representeren. De lijn of het lijnsegment wordt gerepresenteerd door een *probability distribution*. Lijnen kunnen worden gerepresenteerd door *conjugate distributions* en dan kan *Gibbs sampling* worden gebruikt. De lijnsegmenten kunnen niet gerepresenteerd worden door *conjugate distributions* en daarom wordt er een *split-merge sampler* gebruikt.

Een *split-merge sampler* zoekt naar lijnsegmenten door punten toe te kennen aan een hypothetisch lijnsegment en twee lijnsegmenten samen te voegen of een enkel segment te splitsen. De *triadic split-merge sampler* stelt ook stappen voor waar drie lijnsegmenten bij zijn betrokken. In deze studie, wordt deze nieuwe sampler vergeleken met een gewone *split-merge sampler*. De *triadic sampler* kan nuttig zijn bij veel meer applicaties dan alleen dat van robotperceptie.

De modellen voor de objecten kunnen ook worden geleerd. In deze studie wordt dit gedaan voor complexere objecten zoals kubussen bestaand uit honderden punten. Om precies te zijn, er wordt een *auto-encoder* gebruikt welke aan de hand van data een representief object leert te genereren. De *auto-encoder* gebruikt een nieuw gedefinieerde reconstructie afstand, de *partitioning earth mover's distance*. Het object dat geleerd is door de *auto-encoder*, wordt gebruikt in een *triadic sampler* om in een puntenwolk objecten te identificeren en tegelijkertijd het aantal objecten te bepalen.

Acknowledgments

Wanneer ben je klaar? Deze vraag is mij zo vaak gesteld, dat ik het mezelf bijna ook begon af te vragen. Nu is voor mij de reis altijd leuker dan de aankomst, maar eens wordt het tijd om even aan te meren, al is het maar om iedereen te bedanken! Als ik iemand vergeet, dan kost me dat een biertje.

Allereerst wil ik heel graag Jaap bedanken. Je bent de meest geduldige en toegewijde promotor die ik me maar kan wensen. Van studies naar (1) "artificial gene regulatory networks", (2) "renormalization group theory" op "non-abelian sandpiles", (3) varianten op de "glueing Hough transform", tot uiteindelijk het onderzoek naar (4) "non-parametric Bayesian methods". Elke keer begon ik weer enthousiast te vertellen over de nieuwe wiskunde die ik was tegengekomen en jij begon dan te sputteren en het weer zodanig te kneden dat het gepubliceerd kon worden. Hai Xiang en Johan, dank voor alle sessies die we hebben gehad. Jullie hebben ervoor gezorgd dat de wiskundige puntjes op de i kwamen te staan.

Collega's bij Crownstone, Bart, Alex, Teresa, Arend, Peet, Hans, jullie zijn geweldig. Samen hebben we met Crownstone een geweldig bedrijf opgericht. Dank aan collega's en oud-collega's van Crownstone en DoBots, Behnaz, Merel, Farid, Marvin, Dominik, Marc, Fija, Frans, Mare, Reka, Thomas, Dewi, Aad. Oud-collega's bij Almende, Gerald, Brigit, Govert, Chloé, Judith, Alfons, Jan Peter, Andries, dankzij jullie heb ik interessant onderzoek gedaan en verdieping gevonden in programmeervraagstukken.

Alle afstudeerders en stagiaires die ik heb mogen begeleiden, enorm bedankt! Freek, Ted, Andrei, Vijeth, Remco, Marc, Jorik, Laurens, Jos, Roemer, Rashmi, Anne, Panos, Wouter, Reka, Andreas, Frerik, Jasper, Nanne, Riccardo, Victoria, Odysseas, met elke van jullie heb ik dingen geleerd, ofwel in de materie ofwel in de begeleiding. Ook Juan Pablo, Christian, Tom, Nima, Tim, Mourad, Bart, Alan, Zjhounathan, Jeandre, Edwin, Florijn, Gijs, Alex, Marc, Naveen, Aniket, Joy, Ilhan, Meralynn, Paul, Björn, Oscar, Ricardo, Wicky en Eli, dank voor jullie samenwerking!

Dank aan iedereen die ik bij workshops ben tegengekomen als participant of als organisator. Applied Bayesian Nonparametrics workshop (Como, 2014). The Statistical Physics of Inference and Control Theory (Granada, 2012). Summer school Neural Dynamics Approaches to Cognitive Robotics (Porto, 2011). Neuromorphic Engineering Workshop (Capo Caccia,

2010). Workshop on Self-Organizing Systems (Klagenfurt, 2009). Neuromorphic Engineering Workshop (Telluride, 2008). Dank vooral aan Max en Toni.

Jonathan, Michel, Achmed, Adil, Damir, Kai Fan, dank voor de gesprekken over electrotechniek en nieuwe technologie. Ernst, Bas, Jeroen, Basia, Aga, Erik, dank voor de vriendschap. Het heeft veel voor me betekent. Dank!

Lieve ouders, dank voor alles! Jullie hebben nooit druk uitgeoefend om (academisch) te presteren. Daarom houd ik, denk ik, juist zoveel van de wetenschap. Het is nooit een verplichting geweest... Eén van mijn eerste ervaringen was dat ik op D'n Diek met mijn vader de computer moest terugbrengen. Ik had als tienjarige lopen knoeien aan de *config.sys* en *autoexec.bat* bestanden om *extended memory* aan te spreken (het DOS tijdperk, 1990). Vervolgens kregen ze het bij de computerspeciaalzaak niet zo snel in orde. Mij brak het zweet uit, maar bij mijn vader niet. Hij was trots. Toen heb ik geleerd dat je nieuwsgierig mag zijn en fouten mag maken.

Broers en zus, Huibert, Marjolein, Harmen en Adriaan. Jullie zijn allemaal anders en dank voor al die perspectieven die mij ook wetenschappelijk hebben gevormd. Dank ook aan mijn oma's, het is fijn om met jullie te praten. Dank aan Adi, heel veel success met je eigen promotie binnenkort. Dank ook aan Lian, voor je interesse, je liefde, je ondersteuning en je lach!

Curriculum Vitae

Anne Cornelis van Rossum was born in Dirksland, Goeree-Overflakkee, Netherlands on the 30th of December, 1980. After attending atheneum, Anne followed a BSc of Electrical Engineering at the Delft University of Technology, also known as TU Delft, from 1999 to 2002. In the next year, Anne took a ten-month sabbatical in South-America. Thereafter, Anne completed a MSc in Media & Knowledge Engineering at the TU Delft (with courses on speech recognition, neural networks, expert systems, digital signal processing) and a minor at Leiden University in Cognitive Psychology (with courses on connectionism, memory, attention models). After these studies, in 2006, Anne started as researcher at Almende, a research institute in Rotterdam. Here Anne worked on Dutch and European projects, e.g., ProHeal, Sensei, Fireswarm, and Replicator. In the years thereafter, Anne became involved in the launching stages of new companies: Sense Observation Systems, DoBots, and Crownstone (as CEO). The role of Sense was to bring ubiquitous sensor technology into the life of people, preferably to help them to live healthier lives. At DoBots autonomous robots were brought to the market. Nowadays, Crownstone has as vision to allow everybody to own a smart home. Smart, in the sense that it is a home that knows the location of inhabitants in the house. With this knowledge the house can automatically turn on the lights when someone enters a room. In all those different roles, Anne makes sure to use artificial intelligence for a potentially brighter future.

PUBLICATIONS

The investigations performed during my Ph.D. research resulted in the following publications.

- A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Nonparametric Bayesian Line Detection. *International Conference on Pattern Recognition and Methods*, ICPRAM 2016, Rome, Italy, February 24-26, 2016. Best paper award in theory and methods track.
- A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Fundamentals of Nonparametric Bayesian Line Detection. Springer, 2017.
- A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Nonparametric Segment Detection. *Proceedings of the Eighth European Starting AI Researcher Symposium*, STAIRS 2016, the Hague, the Netherlands, August 26-27, 2016.
- A.C. van Rossum, H.X. Lin, J. Dubbeldam, and H.J. van den Herik. Triadic Split-Merge Sampler. *The 10th International Conference on Machine Vision*, ICMV 2017, Vienna, Austria, November 13-November 15, 2017.

The research done in this Ph.D. would not have been possible without preliminaries studies in the fields of robotics, that precede the publications on Bayesian methods. The following publications have the writer of this dissertation as first author.

- A.C. van Rossum and H.J. van den Herik. Circadian Robot Metamorphosis. *Tenth International Conference on Epigenetic Robotics*, EPIROB 2010, Örenäs Slott, Sweden, November 5-7, 2010.
- A.C. van Rossum and H.J. van den Herik. Designing Robot Metamorphosis. *22nd Benelux Conference on Artificial Intelligence*, BNAIC2010, Luxembourg, Luxembourg, October 25-26, 2010.
- A.C. van Rossum. Genetic Encoding of Robot Metamorphosis: How to Evolve a Glider with a Genetic Regulatory Network. *International Conference on Swarm Intelligence*, ANTS2010, Brussels, Belgium, September 8-10, 2010.

The author has contributed to the following publications.

- W. Bulten, A.C. van Rossum, W.F.G. Haselager. Human SLAM, indoor localisation of devices and users. *First International Conference on Internet-of-Things Design and Implementation*, IoTDI2016, Berlin, Germany, April 4-6, 2016.

- V. Rai, A. C. van Rossum, and N. Correll. Self-Assembly of Modular Robots from a Finite Number of Modules using Graph Grammars *International Conference on Intelligent Robots and Systems* IROS2011, San Francisco, United States of America, September 25-30, 2011.
- M. Szymanski, L. Winkler, D. Laneri, F. Schlachter, A. C. van Rossum, T. Schmickl, R. Thenius. SymbricatorRTOS: a Flexible and Dynamic Framework for Bio-inspired Robot Control Systems and Evolution. In: Evolutionary Computation, 2009.

The research performed by master students under the author's supervision has also been of large influence on the research in this dissertation. Their research can be found in the following master theses.

- O. Krystalakos. Automatic Appliance Identification. Faculty of Social Sciences, Radboud University, 2020.
- R. Bellana. Anomaly detection using autoencoders for Ambient Assisted Living. Graduate School of Natural Sciences. Utrecht University, 2018.
- N. Wielinga. Configuration of Ambient Environments by Speech. Department of Mathematics and Computer Science Software Engineering and Technology Research Group. Eindhoven University of Technology, 2018.
- J.I.J. Makkinje. Appliance Identification using Long Short-Term Memory Recurrent Neural Networks. Faculty of Science. Utrecht University, 2018.
- F.D. Andriessen, Ready for Detection. Stair-detecting in RGB-D images using OpenCV and an Adaboosting Algorithm. Space Engineering. Delft University of Technology, 2017.
- A. Sofos. Evaluation of Machine Learning Techniques for Passive Presence Detection. Erasmus School of Economics. Erasmus University Rotterdam, 2017.
- R. Hajnovicsova. Pull-E- The Assistive Shopping Trolley. Faculty of Mechanical, Maritime and Materials Engineering. Delft University of Technology, 2017.
- W. Bulten. Human SLAM. Simultaneous Localisation and Configuration (SLAC) of Indoor Wireless Sensor Networks and their Users. Faculty of Social Sciences, Radboud University, 2015.
- P. Chatzichristodoulou. Towards Lifelong Mapping in Pointclouds. Department of Knowledge Engineering. Maastricht University, 2015.
- A. Bekker. BitsLab. A Smart Product for Weight Loss Through Behavior Change. Industrial Design Engineering. Delft University of Technology, 2015.
- R.N. Kannankutty. Innovating Independent Living. Faculty of Technology, Policy and Management. Delft University of Technology, 2014.
- R.Q. Vlasveld. Temporal Segmentation using Support Vector Machines in the context of Human Activity Recognition. Department of Information and Computing Science. Utrecht University, 2014.
- J. van den Haspel. Distributed Control ofRefrigerators for the Smart Grid. Faculty of Mechanical, Maritime and Materials Engineering. Delft University of Technology, 2013.
- L. Bliek. Nonlinear System Identification and Control for Autonomous Robots. Delft Institute of Applied Mathematics. Delft University of Technology, 2013.
- J. Hulshof. ARSOn. A Robotic Search Optimization. Faculty of Mechanical, Maritime and Materials Engineering. Delft University of Technology, 2013.
- M.J. Hulscher. Joint Action Reasoning for Local Agents. Faculty of Mechanical, Maritime and Materials Engineering. Delft University of Technology, 2013.

- R. Tukker. Echo State Networks for Hierarchical Cognitive Control. Donders Institute. Radboud University Nijmegen, 2012.
- V. Rai. Self Assembly of Modular Robots with Finite Number of Modules Using Graph Grammar. Department of Electrical Computer and Energy Engineering. University of Colorado Boulder, 2011.
- T.P. Schmidt. Robotics: Environmental Awareness Through Cognitive Sensor Fusion. Department of Artificial Intelligence. University of Groningen, 2010.
- F. van Polen. Towards a Sentient Environment Usinga Neural Sensor Network. Department of Information and Computing Sciences. Utrecht University, 2008.

SIKS Dissertation Series

2011

1 Botond Cseke (RUN) *Variational Algorithms for Bayesian Inference in Latent Gaussian Models*

2 Nick Tinnemeier (UU) *Organizing Agent Organizations. Syntax and Operational Semantics of an Organization-Oriented Programming Language*

3 Jan Martijn van der Werf (TUE) *Compositional Design and Verification of Component-Based Information Systems*

4 Hado van Hasselt (UU) *Insights in Reinforcement Learning; Formal analysis and empirical evaluation of temporal-difference*

5 Base van der Raadt (VU) *Enterprise Architecture Coming of Age - Increasing the Performance of an Emerging Discipline.*

6 Yiwen Wang (TUE) *Semantically-Enhanced Recommendations in Cultural Heritage*

7 Yujia Cao (UT) *Multimodal Information Presentation for High Load Human Computer Interaction*

8 Nieske Vergunst (UU) *BDI-based Generation of Robust Task-Oriented Dialogues*

9 Tim de Jong (OU) *Contextualised Mobile Media for Learning*

10 Bart Bogaert (UvT) *Cloud Content Contention*

11 Dhaval Vyas (UT) *Designing for Awareness: An Experience-focused HCI Perspective*

12 Carmen Bratosin (TUE) *Grid Architecture for Distributed Process Mining*

13 Xiaoyu Mao (UvT) *Airport under Control. Multiagent Scheduling for Airport Ground Handling*

14 Milan Lovric (EUR) *Behavioral Finance and Agent-Based Artificial Markets*

15 Marijn Koolen (UvA) *The Meaning of Structure: the Value of Link Evidence for Information Retrieval*

16 Maarten Schadd (UM) *Selective Search in Games of Different Complexity*

17 Jiyin He (UvA) *Exploring Topic Structure: Coherence, Diversity and Relatedness*

18 Mark Ponsen (UM) *Strategic Decision-Making in complex games*

19 Ellen Rusman (OU) *The Mind's Eye on Personal Profiles*

20 Qing Gu (VU) *Guiding service-oriented software engineering - A view-based approach*

21 Linda Terlouw (TUD) *Modularization and Specification of Service-Oriented Systems*

22 Junte Zhang (UvA) *System Evaluation of Archival Description and Access*

23 Wouter Weerkamp (UvA) *Finding People and their Utterances in Social Media*

24 Herwin van Welbergen (UT) *Behavior Generation for Interpersonal Coordination with Virtual Humans On Specifying, Scheduling and Realizing Multimodal Virtual Human Behavior*

25 Syed Waqar ul Qounain Jaffry (VU) *Analysis and Validation of Models for Trust Dynamics*

26 Matthijs Aart Pontier (VU) *Virtual Agents for Human Communication - Emotion Regulation and Involvement-Distance Trade-Offs in Embodied Conversational Agents and Robots*

27 Aniel Bhulai (VU) *Dynamic website optimization through autonomous management of design patterns*

28 Rianne Kaptein (UvA) *Effective Focused Retrieval by Exploiting Query Context and Document Structure*

29 Faisal Kamiran (TUE) *Discrimination-aware Classification*

30 Egon van den Broek (UT) *Affective Signal Processing (ASP): Unraveling the mystery of emotions*

31 Ludo Waltman (EUR) *Computational and Game-Theoretic Approaches for Modeling Bounded Rationality*

32 Nees-Jan van Eck (EUR) *Methodological Advances in Bibliometric Mapping of Science*

33 Tom van der Weide (UU) *Arguing to Motivate Decisions*

34 Paolo Turrini (UU) *Strategic Reasoning in Interdependence: Logical and Game-theoretical Investigations*

35 Maaike Harbers (UU) *Explaining Agent Behavior in Virtual Training*

36 Erik van der Spek (UU) *Experiments in serious game design: a cognitive approach*

37 Adriana Burlutiu (RUN) *Machine Learning for Pairwise Data, Applications for Preference Learning and Supervised Network Inference*

38 Nyree Lemmens (UM) *Bee-inspired Distributed Optimization*

39 Joost Westra (UU) *Organizing Adaptation using Agents in Serious Games*

40 Viktor Clerc (VU) *Architectural Knowledge Management in Global Software Development*

41 Luan Ibraimi (UT) *Cryptographically Enforced Distributed Data Access Control*

42 Michal Sindlar (UU) *Explaining Behavior through Mental State Attribution*

43 Henk van der Schuur (UU) *Process Improvement through Software Operation Knowledge*

44 Boris Reuderink (UT) *Robust Brain-Computer Interfaces*

45 Herman Stehouwer (UvT) *Statistical Language Models for Alternative Sequence Selection*

46 Beibei Hu (TUD) *Towards Contextualized Information Delivery: A Rule-based Architecture for the Domain of Mobile Police Work*

47 Azizi Bin Ab Aziz (VU) *Exploring Computational Models for Intelligent Support of Persons with Depression*

48 Mark Ter Maat (UT) *Response Selection and Turn-taking for a Sensitive Artificial Listening Agent*

49 Andreea Niculescu (UT) *Conversational interfaces for task-oriented spoken dialogues: design aspects influencing interaction quality*

2012

1 Terry Kakeeto (UvT) *Relationship Marketing for SMEs in Uganda*

2 Muhammad Umair (VU) *Adaptivity, emotion, and Rationality in Human and Ambient Agent Models*

3 Adam Vanya (VU) *Supporting Architecture Evolution by Mining Software Repositories*

4 Jurriaan Souer (UU) *Development of Content Management System-based Web Applications*

5 Marijn Plomp (UU) *Maturing Interorganisational Information Systems*

6 Wolfgang Reinhardt (OU) *Awareness Support for Knowledge Workers in Research Networks*

7 Rianne van Lambalgen (VU) *When the Going Gets Tough: Exploring Agent-based Models of Human Performance under Demanding Conditions*

8 Gerben de Vries (UvA) *Kernel Methods for Vessel Trajectories*

9 Ricardo Neisse (UT) *Trust and Privacy Management Support for Context-Aware Service Platforms*

10 David Smits (TUE) *Towards a Generic Distributed Adaptive Hypermedia Environment*

11 J.C.B. Rantham Prabhakara (TUE) *Process Mining in the Large: Preprocessing, Discovery, and Diagnostics*

12 Kees van der Sluijs (TUE) *Model Driven Design and Data Integration in Semantic Web Information Systems*

13 Suleman Shahid (UvT) *Fun and Face: Exploring non-verbal expressions of emotion during playful interactions*

14 Evgeny Knutov (TUE) *Generic Adaptation Framework for Unifying Adaptive Web-based Systems*

15 Natalie van der Wal (VU) *Social Agents. Agent-Based Modelling of Integrated Internal and Social Dynamics of Cognitive and Affective Processes.*

16 Fiemke Both (VU) *Helping people by understanding them - Ambient Agents supporting task execution and depression treatment*

17 Amal Elgammal (UvT) *Towards a Comprehensive Framework for Business Process Compliance*

18 Eltjo Poort (VU) *Improving Solution Architecting Practices*

19 Helen Schonenberg (TUE) *What's Next? Operational Support for Business Process Execution*

20 Ali Bahramisharif (RUN) *Covert Visual Spatial Attention, a Robust Paradigm for Brain-Computer Interfacing*

21 Roberto Cornacchia (TUD) *Querying Sparse Matrices for Information Retrieval*

22 Thijs Vis (UvT) *Intelligence, politie en veiligheidsdienst: verenigbare grootheden?*

23 Christian Muehl (UT) *Toward Affective Brain-Computer Interfaces: Exploring the Neurophysiology of Affect during Human Media Interaction*

24 Laurens van der Werff (UT) *Evaluation of Noisy Transcripts for Spoken Document Retrieval*

25 Silja Eckartz (UT) *Managing the Business Case Development in Inter-Organizational IT Projects: A Methodology and its Application*

26 Emile de Maat (UvA) *Making Sense of Legal Text*

27 Hayrettin Gurkok (UT) *Mind the Sheep! User Experience Evaluation & Brain-Computer Interface Games*

28 Nancy Pascall (UvT) *Engendering Technology Empowering Women*

29 Almer Tigelaar (UT) *Peer-to-Peer Information Retrieval*

30 Alina Pommeranz (TUD) *Designing Human-Centered Systems for Reflective Decision Making*

31 Emily Bagarukayo (RUN) *A Learning by Construction Approach for Higher Order Cognitive Skills Improvement, Building Capacity and Infrastructure*

32 Wietske Visser (TUD) *Qualitative multi-criteria preference representation and reasoning*

33 Rory Sie (OUN) *Coalitions in Cooperation Networks (COCOON)*

34 Pavol Jancura (RUN) *Evolutionary analysis in PPI networks and applications*

35 Evert Haasdijk (VU) *Never Too Old To Learn – Online Evolution of Controllers in Swarm- and Modular Robotics*

36 Denis Ssebugwawo (RUN) *Analysis and Evaluation of Collaborative Modeling Processes*

37 Agnes Nakakawa (RUN) *A Collaboration Process for Enterprise Architecture Creation*

38 Selmar Smit (VU) *Parameter Tuning and Scientific Testing in Evolutionary Algorithms*

39 Hassan Fatemi (UT) *Risk-aware design of value and coordination networks*

40 Agus Gunawan (UvT) *Information Access for SMEs in Indonesia*

41 Sebastian Kelle (OU) *Game Design Patterns for Learning*

42 Dominique Verpoorten (OU) *Reflection Amplifiers in self-regulated Learning*

43 Withdrawn (Withdrawn) *Withdrawn*

44 Anna Tordai (VU) *On Combining Alignment Techniques*

45 Benedikt Kratz (UvT) *A Model and Language for Business-aware Transactions*

46 Simon Carter (UvA) *Exploration and Exploitation of Multilingual Data for Statistical Machine Translation*

47 Manos Tsagkias (UvA) *Mining Social Media: Tracking Content and Predicting Behavior*

48 Jorn Bakker (TUE) *Handling Abrupt Changes in Evolving Time-series Data*

49 Michael Kaisers (UM) *Learning against Learning - Evolutionary dynamics of reinforcement learning algorithms in strategic interactions*

50 Steven van Kervel (TUD) *Ontologogy driven Enterprise Information Systems Engineering*

51 Jeroen de Jong (TUD) *Heuristics in Dynamic Sceduling; a practical framework with a case study in elevator dispatching*

2013

1 Viorel Milea (EUR) *News Analytics for Financial Decision Support*

2 Erietta Liarou (CWI) *MonetDB/DataCell: Leveraging the Column-store Database Technology for Efficient and Scalable Stream Processing*

3 Szymon Klarman (VU) *Reasoning with Contexts in Description Logics*

4 Chetan Yadati (TUD) *Coordinating autonomous planning and scheduling*

5 Dulce Pumareja (UT) *Groupware Requirements Evolutions Patterns*

6 Romulo Goncalves (CWI) *The Data Cyclotron: Juggling Data and Queries for a Data Warehouse Audience*

7 Giel van Lankveld (UvT) *Quantifying Individual Player Differences*

8 Robbert-Jan Merk (VU) *Making enemies: cognitive modeling for opponent agents in fighter pilot simulators*

9 Fabio Gori (RUN) *Metagenomic Data Analysis: Computational Methods and Applications*

10 Jeewanie Jayasinghe Arachchige (UvT) *A Unified Modeling Framework for Service Design.*

11 Evangelos Pournaras (TUD) *Multi-level Reconfigurable Self-organization in Overlay Services*

12 Marian Razavian (VU) *Knowledge-driven Migration to Services*

13 Mohammad Safiri (UT) *Service Tailoring: User-centric creation of integrated IT-based homecare services to support independent living of elderly*

14 Jafar Tanha (UvA) *Ensemble Approaches to Semi-Supervised Learning Learning*

15 Daniel Hennes (UM) *Multiagent Learning - Dynamic Games and Applications*

16 Eric Kok (UU) *Exploring the practical benefits of argumentation in multi-agent deliberation*

17 Koen Kok (VU) *The PowerMatcher: Smart Coordination for the Smart Electricity Grid*

18 Jeroen Janssens (UvT) *Outlier Selection and One-Class Classification*

19 Renze Steenhuizen (TUD) *Coordinated Multi-Agent Planning and Scheduling*

20 Katja Hofmann (UvA) *Fast and Reliable Online Learning to Rank for Information Retrieval*

21 Sander Wubben (UvT) *Text-to-text generation by monolingual machine translation*

22 Tom Claassen (RUN) *Causal Discovery and Logic*

23 Patricio de Alencar Silva (UvT) *Value Activity Monitoring*

24 Haitham Bou Ammar (UM) *Automated Transfer in Reinforcement Learning*

25 Agnieszka Anna Latoszek-Berendsen (UM) *Intention-based Decision Support. A new way of representing and implementing clinical guidelines in a Decision Support System*

26 Alireza Zarghami (UT) *Architectural Support for Dynamic Homecare Service Provisioning*

27 Mohammad Huq (UT) *Inference-based Framework Managing Data Provenance*

28 Frans van der Sluis (UT) *When Complexity becomes Interesting: An Inquiry into the Information eXperience*

29 Iwan de Kok (UT) *Listening Heads*

30 Joyce Nakatumba (TUE) *Resource-Aware Business Process Management: Analysis and Support*

31 Dinh Khoa Nguyen (UvT) *Blueprint Model and Language for Engineering Cloud Applications*

32 Kamakshi Rajagopal (OUN) *Networking For Learning; The role of Networking in a Lifelong Learner's Professional Development*

33 Qi Gao (TUD) *User Modeling and Personalization in the Microblogging Sphere*

34 Kien Tjin-Kam-Jet (UT) *Distributed Deep Web Search*

35 Abdallah El Ali (UvA) *Minimal Mobile Human Computer Interaction*

36 Than Lam Hoang (TUe) *Pattern Mining in Data Streams*

37 Dirk Börner (OUN) *Ambient Learning Displays*

38 Eelco den Heijer (VU) *Autonomous Evolutionary Art*

39 Joop de Jong (TUD) *A Method for Enterprise Ontology based Design of Enterprise Information Systems*

40 Pim Nijssen (UM) *Monte-Carlo Tree Search for Multi-Player Games*

41 Jochem Liem (UvA) *Supporting the Conceptual Modelling of Dynamic Systems: A Knowledge Engineering Perspective on Qualitative Reasoning*

42 Léon Planken (TUD) *Algorithms for Simple Temporal Reasoning*

43 Marc Bron (UvA) *Exploration and Contextualization through Interaction and Concepts*

2014

1 Nicola Barile (UU) *Studies in Learning Monotone Models from Data*

2 Fiona Tuliyano (RUN) *Combining System Dynamics with a Domain Modeling Method*

3 Sergio Raul Duarte Torres (UT) *Information Retrieval for Children: Search Behavior and Solutions*

4 Hanna Jochmann-Mannak (UT) *Websites for children: search strategies and interface design - Three studies on children's search performance and evaluation*

5 Jurriaan van Reijsen (UU) *Knowledge Perspectives on Advancing Dynamic Capability*

6 Damian Tamburri (VU) *Supporting Networked Software Development*

7 Arya Adriansyah (TUE) *Aligning Observed and Modeled Behavior*

8 Samur Araujo (TUD) *Data Integration over Distributed and Heterogeneous Data Endpoints*

9 Philip Jackson (UvT) *Toward Human-Level Artificial Intelligence: Representation and Computation of Meaning in Natural Language*

10 Ivan Salvador Razo Zapata (VU) *Service Value Networks*

11 Janneke van der Zwaan (TUD) *An Empathic Virtual Buddy for Social Support*

12 Willem van Willigen (VU) *Look Ma, No Hands: Aspects of Autonomous Vehicle Control*

13 Arlette van Wissen (VU) *Agent-Based Support for Behavior Change: Models and Applications in Health and Safety Domains*

14 Yangyang Shi (TUD) *Language Models With Meta-information*

15 Natalya Mogles (VU) *Agent-Based Analysis and Support of Human Functioning in Complex Socio-Technical Systems: Applications in Safety and Healthcare*

16 Krystyna Milian (VU) *Supporting trial recruitment and design by automatically interpreting eligibility criteria*

17 Kathrin Dentler (VU) *Computing healthcare quality indicators automatically: Secondary Use of Patient Data and Semantic Interoperability*

18 Mattijs Ghijsen (UvA) *Methods and Models for the Design and Study of Dynamic Agent Organizations*

19 Vinicius Ramos (TUE) *Adaptive Hypermedia Courses: Qualitative and Quantitative Evaluation and Tool Support*

20 Mena Habib (UT) *Named Entity Extraction and Disambiguation for Informal Text: The Missing Link*

21 Kassidy Clark (TUD) *Negotiation and Monitoring in Open Environments*

22 Marieke Peeters (UU) *Personalized Educational Games - Developing agent-supported scenario-based training*

23 Eleftherios Sidirourgos (UvA/CWI) *Space Efficient Indexes for the Big Data Era*

24 Davide Ceolin (VU) *Trusting Semi-structured Web Data*

25 Martijn Lappenschaar (RUN) *New network models for the analysis of disease interaction*

26 Tim Baarslag (TUD) *What to Bid and When to Stop*

27 Rui Jorge Almeida (EUR) *Conditional Density Models Integrating Fuzzy and Probabilistic Representations of Uncertainty*

28 Anna Chmielowiec (VU) *Decentralized k-Clique Matching*

29 Jaap Kabbedijk (UU) *Variability in Multi-Tenant Enterprise Software*

30 Peter de Cock (UvT) *Anticipating Criminal Behaviour*

31 Leo van Moergestel (UU) *Agent Technology in Agile Multiparallel Manufacturing and Product Support*

32 Naser Ayat (UvA) *On Entity Resolution in Probabilistic Data*

33 Tesfa Tegegne (RUN) *Service Discovery in eHealth*

34 Christina Manteli (VU) *The Effect of Governance in Global Software Development: Analyzing Transactive Memory Systems.*

35 Joost van Ooijen (UU) *Cognitive Agents in Virtual Worlds: A Middleware Design Approach*

36 Joos Buijs (TUE) *Flexible Evolutionary Algorithms for Mining Structured Process Models*

37 Maral Dadvar (UT) *Experts and Machines United Against Cyberbullying*

38 Danny Plass-Oude Bos (UT) *Making brain-computer interfaces better: improving usability through post-processing.*

39 Jasmina Maric (UvT) *Web Communities, Immigration, and Social Capital*

40 Walter Omona (RUN) *A Framework for Knowledge Management Using ICT in Higher Education*

41 Frederic Hogenboom (EUR) *Automated Detection of Financial Events in News Text*

42 Carsten Eijckhof (CWI/TUD) *Contextual Multidimensional Relevance Models*

43 Kevin Vlaanderen (UU) *Supporting Process Improvement using Method Increments*

44 Paulien Meesters (UvT) *Intelligent Blauw. Met als ondertitel: Intelligence-gestuurde politiezorg in gebiedsgebonden eenheden.*

45 Birgit Schmitz (OUN) *Mobile Games for Learning: A Pattern-Based Approach*

46 Ke Tao (TUD) *Social Web Data Analytics: Relevance, Redundancy, Diversity*

47 Shangsong Liang (UvA) *Fusion and Diversification in Information Retrieval*

2015

1 Niels Netten (UvA) *Machine Learning for Relevance of Information in Crisis Response*

2 Faiza Bukhsh (UvT) *Smart auditing: Innovative Compliance Checking in Customs Controls*

3 Twan van Laarhoven (RUN) *Machine learning for network data*

4 Howard Spoelstra (OUN) *Collaborations in Open Learning Environments*

5 Christoph Bösch (UT) *Cryptographically Enforced Search Pattern Hiding*

6 Farideh Heidari (TUD) *Business Process Quality Computation - Computing Non-Functional Requirements to Improve Business Processes*

7 Maria-Hendrike Peetz (UvA) *Time-Aware Online Reputation Analysis*

8 Jie Jiang (TUD) *Organizational Compliance: An agent-based model for designing and evaluating organizational interactions*

9 Randy Klaassen (UT) *HCI Perspectives on Behavior Change Support Systems*

10 Henry Hermans (OUN) *OpenU: design of an integrated system to support lifelong learning*

11 Yongming Luo (TUE) *Designing algorithms for big graph datasets: A study of computing bisimulation and joins*

12 Julie M. Birkholz (VU) *Modi Operandi of Social Network Dynamics: The Effect of Context on Scientific Collaboration Networks*

13 Giuseppe Procaccianti (VU) *Energy-Efficient Software*

14 Bart van Straalen (UT) *A cognitive approach to modeling bad news conversations*

15 Klaas Andries de Graaf (VU) *Ontology-based Software Architecture Documentation*

16 Changyun Wei (UT) *Cognitive Coordination for Cooperative Multi-Robot Teamwork*

17 André van Cleeff (UT) *Physical and Digital Security Mechanisms: Properties, Combinations and Trade-offs*

18 Holger Pirk (CWI) *Waste Not, Want Not! - Managing Relational Data in Asymmetric Memories*

19 Bernardo Tabuenca (OUN) *Ubiquitous Technology for Lifelong Learners*

20 Loïs Vanhée (UU) *Using Culture and Values to Support Flexible Coordination*

21 Sibren Fetter (OUN) *Using Peer-Support to Expand and Stabilize Online Learning*

22 Luit Gazendam (VU) *Cataloguer Support in Cultural Heritage*

23 Richard Berendsen (UvA) *Finding People, Papers, and Posts: Vertical Search Algorithms and Evaluation*

24 Steven Woudenberg (UU) *Bayesian Tools for Early Disease Detection*

25 Alexander Hogenboom (EUR) *Sentiment Analysis of Text Guided by Semantics and Structure*

26 Sándor Héman (CWI) *Updating compressed colomn stores*

27 Janet Bagorogoza (TiU) *KNOWLEDGE MANAGEMENT AND HIGH PERFORMANCE; The Uganda Financial Institutions Model for HPO*

28 Hendrik Baier (UM) *Monte-Carlo Tree Search Enhancements for One-Player and Two-Player Domains*

29 Kiavash Bahreini (OU) *Real-time Multimodal Emotion Recognition in E-Learning*

30 Yakup Koç (TUD) *On the robustness of Power Grids*

31 Jerome Gard (UL) *Corporate Venture Management in SMEs*

32 Frederik Schadd (TUD) *Ontology Mapping with Auxiliary Resources*

33 Victor de Graaf (UT) *Gesocial Recommender Systems*

34 Jungxao Xu (TUD) *Affective Body Language of Humanoid Robots: Perception and Effects in Human Robot Interaction*

2016

1 Syed Saiden Abbas (RUN) *Recognition of Shapes by Humans and Machines*

2 Michiel Christiaan Meulendijk (UU) *Optimizing medication reviews through decision support: prescribing a better pill to swallow*

3 Maya Sappelli (RUN) *Knowledge Work in Context: User Centered Knowledge Worker Support*

4 Laurens Rietveld (VU) *Publishing and Consuming Linked Data*

5 Evgeny Sherkhonov (UvA) *Expanded Acyclic Queries: Containment and an Application in Explaining Missing Answers*

6 Michel Wilson (TUD) *Robust scheduling in an uncertain environment*

7 Jeroen de Man (VU) *Measuring and modeling negative emotions for virtual training*

8 Matje van de Camp (TiU) *A Link to the Past: Constructing Historical Social Networks from Unstructured Data*

9 Archana Nottamkandath (VU) *Trusting Crowdsourced Information on Cultural Artefacts*

10 George Karafotias (VUA) *Parameter Control for Evolutionary Algorithms*

11 Anne Schuth (UvA) *Search Engines that Learn from Their Users*

12 Max Knobbout (UU) *Logics for Modelling and Verifying Normative Multi-Agent Systems*

13 Nana Baah Gyan (VU) *The Web, Speech Technologies and Rural Development in West Africa - An ICT4D Approach*

14 Ravi Khadka (UU) *Revisiting Legacy Software System Modernization*

15 Steffen Michels (RUN) *Hybrid Probabilistic Logics - Theoretical Aspects, Algorithms and Experiments*

16 Guangliang Li (UvA) *Socially Intelligent Autonomous Agents that Learn from Human Reward*

17 Berend Weel (VU) *Towards Embodied Evolution of Robot Organisms*

18 Albert Meroño Peñuela (VU) *Refining Statistical Data on the Web*

19 Julia Efremova (Tu/e) *Mining Social Structures from Genealogical Data*

20 Daan Odijk (UvA) *Context & Semantics in News & Web Search*

21 Alejandro Moreno Célleri (UT) *From Traditional to Interactive Playspaces: Automatic Analysis of Player Behavior in the Interactive Tag Playground*

22 Grace Lewis (VU) *Software Architecture Strategies for Cyber-Foraging Systems*

23 Fei Cai (UvA) *Query Auto Completion in Information Retrieval*

24 Brend Wanders (UT) *Repurposing and Probabilistic Integration of Data; An Iterative and data model independent approach*

25 Julia Kiseleva (TU/e) *Using Contextual Information to Understand Searching and Browsing Behavior*

26 Dilhan Thilakarathne (VU) *In or Out of Control: Exploring Computational Models to Study the Role of Human Awareness and Control in Behavioural Choices, with Applications in Aviation and Energy Management Domains*

27 Wen Li (TUD) *Understanding Geo-spatial Information on Social Media*

28 Mingxin Zhang (TUD) *Large-scale Agent-based Social Simulation - A study on epidemic prediction and control*

29 Nicolas Höning (TUD) *Peak reduction in decentralised electricity systems -Markets and prices for flexible planning*

30 Ruud Mattheij (UvT) *The Eyes Have It*

31 Mohammad Khelghati (UT) *Deep web content monitoring*

32 Eelco Vriezekolk (UT) *Assessing Telecommunication Service Availability Risks for Crisis Organisations*

33 Peter Bloem (UvA) *Single Sample Statistics, exercises in learning from just one example*

34 Dennis Schunselaar (TUE) *Configurable Process Trees: Elicitation, Analysis, and Enactment*

35 Zhaochun Ren (UvA) *Monitoring Social Media: Summarization, Classification and Recommendation*

36 Daphne Karreman (UT) *Beyond R2D2: The design of nonverbal interaction behavior optimized for robot-specific morphologies*

37 Giovanni Sileno (UvA) *Aligning Law and Action - a conceptual and computational inquiry*

38 Andrea Minuto (UT) *MATERIALS THAT MATTER - Smart Materials meet Art & Interaction Design*

39 Merijn Bruijnes (UT) *Believable Suspect Agents; Response and Interpersonal Style Selection for an Artificial Suspect*

40 Christian Detweiler (TUD) *Accounting for Values in Design*

41 Thomas King (TUD) *Governing Governance: A Formal Framework for Analysing Institutional Design and Enactment Governance*

42 Spyros Martzoukos (UvA) *Combinatorial and Compositional Aspects of Bilingual Aligned Corpora*

43 Saskia Koldijk (RUN) *Context-Aware Support for Stress Self-Management: From Theory to Practice*

44 Thibault Sellam (UvA) *Automatic Assistants for Database Exploration*

45 Bram van de Laar (UT) *Experiencing Brain-Computer Interface Control*

46 Jorge Gallego Perez (UT) *Robots to Make you Happy*

47 Christina Weber (UL) *Real-time foresight - Preparedness for dynamic innovation networks*

48 Tanja Buttler (TUD) *Collecting Lessons Learned*

49 Gleb Polevoy (TUD) *Participation and Interaction in Projects. A Game-Theoretic Analysis*

50 Yan Wang (UvT) *The Bridge of Dreams: Towards a Method for Operational Performance Alignment in IT-enabled Service Supply Chains*

2017

1 Jan-Jaap Oerlemans (UL) *Investigating Cybercrime*

2 Sjoerd Timmer (UU) *Designing and Understanding Forensic Bayesian Networks using Argumentation*

3 Daniël Harold Telgen (UU) *Grid Manufacturing; A Cyber-Physical Approach with Autonomous Products and Reconfigurable Manufacturing Machines*

4 Mrunal Gawade (CWI) *MULTI-CORE PARALLELISM IN A COLUMN-STORE*

5 Mahdieh Shadi (UvA) *Collaboration Behavior*

6 Damir Vandic (EUR) *Intelligent Information Systems for Web Product Search*

7 Roel Bertens (UU) *Insight in Information: from Abstract to Anomaly*

8 Rob Konijn (VU) *Detecting Interesting Differences:Data Mining in Health Insurance Data using Outlier Detection and Subgroup Discovery*

9 Dong Nguyen (UT) *Text as Social and Cultural Data: A Computational Perspective on Variation in Text*

10 Robby van Delden (UT) *(Steering) Interactive Play Behavior*

11 Florian Kunneman (RUN) *Modelling patterns of time and emotion in Twitter #anticipointment*

12 Sander Leemans (TUE) *Robust Process Mining with Guarantees*

13 Gijs Huisman (UT) *Social Touch Technology - Extending the reach of social touch through haptic technology*

14 Shoshannah Tekofsky (UvT) *You Are Who You Play You Are: Modelling Player Traits from Video Game Behavior*

15 Peter Berck, Radboud University (RUN) *Memory-Based Text Correction*

16 Aleksandr Chuklin (UvA) *Understanding and Modeling Users of Modern Search Engines*

17 Daniel Dimov (UL) *Crowdsourced Online Dispute Resolution*

18 Ridho Reinanda (UvA) *Entity Associations for Search*

19 Jeroen Vuurens (TUD) *Proximity of Terms, Texts and Semantic Vectors in Information Retrieval*

20 Mohammadbashir Sedighi (TUD) *Fostering Engagement in Knowledge Sharing: The Role of Perceived Benefits, Costs and Visibility*

21 Jeroen Linssen (UT) *Meta Matters in Interactive Storytelling and Serious Gaming (A Play on Worlds)*

22 Sara Magliacane (VU) *Logics for causal inference under uncertainty*

23 David Graus (UvA) *Entities of Interest— Discovery in Digital Traces*

24 Chang Wang (TUD) *Use of Affordances for Efficient Robot Learning*

25 Veruska Zamborlini (VU) *Knowledge Representation for Clinical Guidelines, with applications to Multimorbidity Analysis and Literature Search*

26 Merel Jung (UT) *Socially intelligent robots that understand and respond to human touch*

27 Michiel Joosse (UT) *Investigating Positioning and Gaze Behaviors of Social Robots: People's Preferences, Perceptions and Behaviors*

28 John Klein (VU) *Architecture Practices for Complex Contexts*

29 Adel Alhuraibi (UvT) *From IT-BusinessStrategic Alignment to Performance: A Moderated Mediation Model of Social Innovation, and Enterprise Governance of IT*

30 Wilma Latuny (UvT) *The Power of Facial Expressions*

31 Ben Ruijl (UL) *Advances in computational methods for QFT calculations*

32 Thaer Samar (RUN) *Access to and Retrievability of Content in Web Archives*

33 Brigit van Loggem (OU) *Towards a Design Rationale for Software Documentation: A Model of Computer-Mediated Activity*

34 Maren Scheffel (OUN) *The Evaluation Framework for Learning Analytics*

35 Martine de Vos (VU) *Interpreting natural science spreadsheets*

36 Yuanhao Guo (UL) *Shape Analysis for Phenotype Characterisation from High-throughput Imaging*

37 Alejandro Montes García (TUE) *WiBAF: A Within Browser Adaptation Framework that Enables Control over Privacy*

38 Alex Kayal (TUD) *Normative Social Applications*

39 Sara Ahmadi (RUN) *Exploiting properties of the human auditory system and compressive sensing methods to increase noise robustness in ASR*

40 Altaf Hussain Abro (VUA) *Steer your Mind: Computational Exploration of Human Control in Relation to Emotions, Desires and Social Support For applications in human-aware support systems*"

41 Adnan Manzoor (VUA) *Minding a Healthy Lifestyle: An Exploration of Mental Processes and a Smart Environment to Provide Support for a Healthy Lifestyle*

42 Elena Sokolova (RUN) *Causal discovery from mixed and missing data with applications on ADHD datasets*

43 Maaike de Boer (RUN) *Semantic Mapping in Video Retrieval*

44 Garm Lucassen (UU) *Understanding User Stories - Computational Linguistics in Agile Requirements Engineering*

45 Bas Testerink (UU) *Decentralized Runtime Norm Enforcement*

46 Jan Schneider (OU) *Sensor-based Learning Support*

47 Yie Yang (TUD) *Crowd Knowledge Creation Acceleration*

48 Angel Suarez (OU) *Collaborative inquiry-based learning*

2018

1 Han van der Aa (VUA) *Comparing and Aligning Process Representations*

2 Felix Mannhardt (TUE) *Multi-perspective Process Mining*

3 Steven Bosems (UT) *Causal Models For Well-Being: Knowledge Modeling, Model-Driven Development of Context-Aware Applications, and Behavior Prediction*

4 Jordan Janeiro (TUD) *Flexible Coordination Support for Diagnosis Teams in Data-Centric Engineering Tasks*

5 Hugo Huurdeman (UvA) *Supporting the Complex Dynamics of the Information Seeking Process*

6 Dan Ionita (UT) *Model-Driven Information Security Risk Assessment of Socio-Technical Systems*

7 Jieting Luo (UU) *A formal account of opportunism in multi-agent systems*

8 Rick Smetsers (RUN) *Advances in Model Learning for Software Systems*

9 Xu Xie (TUD) *Data Assimilation in Discrete Event Simulations*

10 Julienka Mollee (VUA) *Moving forward: supporting physical activity behavior change through intelligent technology*

11 Mahdi Sargolzaei (UvA) *Enabling Framework for Service-oriented Collaborative Networks*

12 Xixi Lu (TUE) *Using behavioral context in process mining*

13 Seyed Amin Tabatabaei (VUA) *Using behavioral context in process mining: Exploring the added value of computational models for increasing the use of renewable energy in the residential sector*

14 Bart Joosten (UvT) *Detecting Social Signals with Spatiotemporal Gabor Filters*

15 Naser Davarzani (UM) *Biomarker discovery in heart failure*

16 Jaebok Kim (UT) *Automatic recognition of engagement and emotion in a group of children*

17 Jianpeng Zhang (TUE) *On Graph Sample Clustering*

18 Henriette Nakad (UL) *De Notaris en Private Rechtspraak*

19 Minh Duc Pham (VUA) *Emergent relational schemas for RDF*

20 Manxia Liu (RUN) *Time and Bayesian Networks*

21 Aad Slootmaker (OUN) *EMERGO: a generic platform for authoring and playing scenario-based serious games*

22 Eric Fernandes de Mello Araújo (VUA) *Contagious: Modeling the Spread of Behaviours, Perceptions and Emotions in Social Networks*

23 Kim Schouten (EUR) *Semantics-driven Aspect-Based Sentiment Analysis*

24 Jered Vroon (UT) *Responsive Social Positioning Behaviour for Semi-Autonomous Telepresence Robots*

25 Riste Gligorov (VUA) *Serious Games in Audio-Visual Collections*

26 Roelof de Vries (UT) *Theory-Based And Tailor-Made: Motivational Messages for Behavior Change Technology*

27 Maikel Leemans (TUE) *Hierarchical Process Mining for Scalable Software Analysis*

28 Christian Willemse (UT) *Social Touch Technologies: How they feel and how they make you feel*

29 Yu Gu (UvT) *Emotion Recognition from Mandarin Speech*

30 Wouter Beek (VU) *The "K" in "semantic web" stands for "knowledge": scaling semantics to the web*

2019

1 Rob van Eijk (UL) *Web privacy measurement in real-time bidding systems. A graph-based approach to RTB system classification*

2 Emmanuelle Beauxis- Aussalet (CWI, UU) *Statistics and Visualizations for Assessing Class Size Uncertainty*

3 Eduardo Gonzalez Lopez de Murillas (TUE) *Process Mining on Databases: Extracting Event Data from Real Life Data*

4 Ridho Rahmadi (RUN) *Finding stable causal structures from clinical data*

5 Sebastiaan van Zelst (TUE) *Process Mining with Streaming Data*

6 Chris Dijkshoorn (VU) *Nichesourcing for Improving Access to Linked Cultural Heritage Datasets*

7 Soude Fazeli (TUD) *Recommender Systems in Social Learning Platforms*

8 Frits de Nijs (TUD) *Resource-constrained Multi-agent Markov Decision Processes*

9 Fahimeh Alizadeh Moghaddam (UvA) *Self-adaptation for energy efficiency in software systems*

10 Qing Chuan Ye (EUR) *Multi-objective Optimization Methods for Allocation and Prediction*

11 Yue Zhao (TUD) *Learning Analytics Technology to Understand Learner Behavioral Engagement in MOOCs*

12 Jacqueline Heinerman (VU) *Better Together*

13 Guanliang Chen (TUD) *MOOC Analytics: Learner Modeling and Content Generation*

14 Daniel Davis (TUD) *Large-Scale Learning Analytics: Modeling Learner Behavior & Improving Learning Outcomes in Massive Open Online Courses*

15 Erwin Walraven (TUD) *Planning under Uncertainty in Constrained and Partially Observable Environments*

16 Guangming Li (TUE) *Process Mining based on Object-Centric Behavioral Constraint (OCBC) Models*

17 Ali Hurriyetoglu (RUN) *Extracting actionable information from microtexts*

18 Gerard Wagenaar (UU) *Artefacts in Agile Team Communication*

19 Vincent Koeman (TUD) *Tools for Developing Cognitive Agents*

20 Chide Groenouwe (UU) *Fostering technically augmented human collective intelligence*

21 Cong Liu (TUE) *Software Data Analytics: Architectural Model Discovery and Design Pattern Detection*

22 Martin van den Berg (VU) *Improving IT Decisions with Enterprise Architecture*

23 Qin Liu (TUD) *Intelligent Control Systems: Learning, Interpreting, Verification*

24 Anca Dumitrache (VU) *Truth in Disagreement-Crowdsourcing Labeled Data for Natural Language Processing*

25 Emiel van Miltenburg (UvT) *Pragmatic factors in (automatic) image description*

26 Prince Singh (UT) *An Integration Platform for Synchromodal Transport*

27 Alessandra Antonaci (OUN) *The Gamification Design Process applied to (Massive) Open Online Courses*

28 Esther Kuindersma (UL) *Cleared for take-off: Game-based learning to prepare airline pilots for critical situations*

29 Daniel Formolo (VU) *Using virtual agents for simulation and training of social skills in safety-critical circumstances*

30 Vahid Yazdanpanah (UT) *Multiagent Industrial Symbiosis Systems*

31 Milan Jelisavcic (VUA) *Alive and Kicking: Baby Steps in Robotics*

32 Chiara Sironi (UM) *Monte-Carlo Tree Search for Artificial General Intelligence in Games*

33 Anil Yaman (TUE) *Evolution of Biologically Inspired Learning in Artificial Neural Networks*

34 Negar Ahmadi (TUE) *EEG Microstate and Functional Brain Network Features for Classification of Epilepsy and PNES*

35 Lisa Facey-Shaw (OUN) *Gamification with digital badges in learning programming*

36 Kevin Ackermans (OUN) *Designing Video-Enhanced Rubrics to Master Complex Skills*

37 Jian Fang (TUD) *Database Acceleration on FPGAs*

38 Akos Kadar (OUN) *Learning visually grounded and multilingual representations*

2020

1 Armon Toubman (UL) *Calculated Moves: Generating Air Combat Behaviour*

2 Marcos de Paula Bueno (UL) *Unraveling Temporal Processes using Probabilistic Graphical Models*

3 Mostafa Deghani (UvA) *Learning with Imperfect Supervision for Language Understanding*

4 Maarten van Gompel (RUN) *Context as Linguistic Bridges*

5 Yulong Pei (TUE) *On local and global structure mining*

6 Preethu Rose Anish (UT) *Stimulation Architectural Thinking during Requirements Elicitation - An Approach and Tool Support*

7 Wim van der Vegt (OUN) *Towards a software architecture for reusable game components*

8 Ali Mirsoleimani (UL) *Structured Parallel Programming for Monte Carlo Tree Search*

9 Myriam Traub (UU) *Measuring Tool Bias and Improving Data Quality for Digital Humanities Research*

10 Alifah Syamsiyah (TUE) *In-database Preprocessing for Process Mining*

11 Sepideh Mesbah (TUD) *Semantic-Enhanced Training Data AugmentationMethods for Long-Tail Entity Recognition Models*

12 Ward van Breda (VU) *Predictive Modeling in E-Mental Health: Exploring Applicability in Personalised Depression Treatment*

13 Marco Virgolin (CWI) *Design and Application of Gene-pool Optimal Mixing Evolutionary Algorithms for Genetic Programming*

14 Mark Raasveldt (CWI/UL) *Integrating Analytics with Relational Databases*

15 Konstantinos Georgiadis (OUN) *Smart CAT: Machine Learning for Configurable Assessments in Serious Games*

16 Ilona Wilmont (RUN) *Cognitive Aspects of Conceptual Modelling*

17 Daniele Di Mitri (OUN) *The Multimodal Tutor: Adaptive Feedback from Multimodal Experiences*

18 Georgios Methenitis (TUD) *Agent Interactions & Mechanisms in Markets with Uncertainties: Electricity Markets in Renewable Energy Systems*

19 Guido van Capelleveen (UT) *Industrial Symbiosis Recommender Systems*

20 Albert Hankel (VU) *Embedding Green ICT Maturity in Organisations*

21 Karine da Silva Miras de Araujo (VU) *Where is the robot?: Life as it could be*

22 Maryam Masoud Khamis (RUN) *Understanding complex systems implementation through a modeling approach: the case of e-government in Zanzibar*

23 Rianne Conijn (UT) *The Keys to Writing: A writing analytics approach to studying writing processes using keystroke logging*

24 Lenin da Nobrega Medeiros (VUA/RUN) *How are you feeling, human? Towards emotionally supportive chatbots*

25 Xin Du (TUE) *The Uncertainty in Exceptional Model Mining*

26 Krzysztof Leszek Sadowski (UU) *GAMBIT: Genetic Algorithm for Model-Based mixed-Integer opTimization*

27 Ekaterina Muravyeva (TUD) *Personal data and informed consent in an educational context*

28 Bibeg Limbu (TUD) *Multimodal interaction for deliberate practice: Training complex skills with augmented reality*

29 Ioan Gabriel Bucur (RUN) *Being Bayesian about Causal Inference*

30 Bob Zadok Blok (UL) *Creatief, Creatieve, Creatiefst*

31 Gongjin Lan (VU) *Learning better – From Baby to Better*

32 Jason Rhuggenaath (TUE) *Revenue management in online markets: pricing and online advertising*

33 Rick Gilsing (TUE) *Supporting service-dominant business model evaluation in the context of business model innovation*

34 Anna Bon (MU) *Intervention or Collaboration? Redesigning Information and Communication Technologies for Development*

35 Siamak Farshidi (UU) *Multi-Criteria Decision-Making in Software Production*

2021

1 Francisco Xavier Dos Santos Fonseca (TUD) *Location-based Games for Social Interaction in Public Space*

2 Rijk Mercuur (TUD) *Simulating Human Routines:Integrating Social Practice Theory in Agent-Based Models*

3 Seyyed Hadi Hashemi (UvA) *Modeling Users Interacting with Smart Devices*

4 Ioana Jivet (OU) *The Dashboard That Loved Me: Designing adaptive learning analytics for self-regulated learning*

5 Davide Dell'Anna (UU) *Data-Driven Supervision of Autonomous Systems*

6 Daniel Davison (UT) *"Hey robot, what do you think?" How children learn with a social robot*

7 Armel Lefebvre (UU) *Research data management for open science*

8 Nardie Fanchamps (OU) *The Influence of Sense-Reason-Act Programming on Computational Thinking*

9 Cristina Zaga (UT) *The Design of Robothings. Non-Anthropomorphic and Non-Verbal Robots to Promote Childrenś Collaboration Through Play*

10 Quinten Meertens (UvA) *Misclassification Bias in Statistical Learning*

11 Anne van Rossum (UL) *Nonparametric Bayesian Methods in Robotic Vision*