Nonparametric Bayesian Methods in Robotics

PROEFSCHRIFT

ter verkrijging van de graad van doctor
aan de Universiteit Leiden,
op gezag van de rector magnificus,
prof. mr. C.J.J.M. Stolker,
volgens besluit van het College voor Promoties
te verdedigen op . . . 2018
in de aula van de Universiteit
klokke . . .

door

Anne Cornelis van Rossum,
geboren op 30 december 1980
te Dirksland

Samenstelling van de promotiecommissie

SIKS Dissertation Series No. ...

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 ....

“ *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

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 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 vision and robotics. 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).

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 does not take into account real-world statistics.

Both generalization and specificity are formalized by a Bayesian model. A Bayesian model is general because it can be solved with general inference methods. One of such general inference methods is a Markov-Chain Monte Carlo method. It does not know anything about real-world statistics. A Bayesian model is also specific in that it can incorporate application-specific know-how by the definition of priors.

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. 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 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.

The problem statement is divided into 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 estimate the number of lines simultaneously with line fitting in computer vision?

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 exists out 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 in the case of 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, (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. By applying Bayesian methodology to these tasks, the inference method becomes optimal in an information-theoretic sense. Also frequentist statisticians agree that nonparametric Bayesian models are consistent in the sense that they approach the underlying true distribution (Wasserman, 1998). There is no need to search for another method to infer lines in a line detection task. If someone would find a method that outperforms a Bayesian method it is either (1) because the signal or noise has not been correctly modeled after all, or (2) because the method overfits with respect to the available data. If approximations are used with respect to optimal Bayesian inference (either variational approximations or Markov-Chain Monte Carlo), there are theoretical guarantuees on convergence.

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. Our approach is to develop subsequently approximations using more sophisticated samplers, so that the theoretical guarantuees on convergence are preserved.

Due to the fact that the models are optimal by construction, there are no experiments required to address the optimality in particular. However, experiments are still required to establish whether the models make sense. Yet, the methodology does also have limitations. For instance, we will not search over different noise models and limit priors to a particular hierarchical level.

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 isle 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 2.3.4) 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 the 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 a third nonparametric Bayesian model, particularly aimed at volumetric inference. This chapter answers the third research question.

Chapter 7 applies the hierarchical sampling method to the domain of recommender engines. It estimates simultaneously the number of user types with a fitting procedure for the invididual user. I want to change this chapter to something relevant to point clouds.

Chapter 8 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 probability theory, and in particular, measure theory underlying random processes (Section 2.1). Five random processes are described, the Beta process, the Gamma process, the Dirichlet process, the Pitman-Yor process, and the hierarchical Dirichlet process. The random processes are presented as a Poisson process with a Lévy measure, a stick-breaking construction, and a sequential presentation (Section 2.2). These representations give rise to different inference methods. Six inference methods are described: Inverse transform sampling, rejection sampling, approximate Bayesian computation, Gibbs sampling, Metropolis-Hastings, and Split-Merge Markov chain Monte Carlo (Section 2.3). 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 (Section 2.4).

2.1 Probability Theory

Modern probability is based on measure theory (Section 2.1.1). 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 (Section 2.1.2). We give three typical examples of Bayesian model compositions, among which an infinite mixture model (Section 2.1.3). A number of processes are described that can be used with (for example as prior distribution) infinite mixture models (Section 2.1.4). We introduce plate notation which visualizes infinite models particularly well (Section 2.1.5). We investigate completely random measures and Lévy measures (Section 2.1.6), exchangebility (Section 2.1.7), and stick-breaking processes (Section 2.1.8), which will form the basis of the processes defined in Section 2.2.

2.1.1 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 notion of size of Euclidean objects to 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 2.1 — *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 $A_1 \cup A_2 \in \Sigma$.

▼ Definition 2.2 — *generated σ-algebra*

A **generated σ-algebra**, with X a set and $B \in 2^X$, is the smallest σ-algebra $\sigma(B)$ that contains all sets of B.

Let $X = \{1,2,3,4\}$ and $B = \{\{1\},\{2,3\}\}$, then the generated σ-algebra is the set $\sigma(B) = \{\varnothing,\{1\},\{2,3\},\{1,4\},\{2,3,4\},\{1,2,3,4\}\}$. Here the sets in B are completed to the sets in $\sigma(B)$ by obeying the requirements of a σ-algebra of closure under complementation and countable unions by addition (e.g., $\{1,4\}$ is added due to closure under completion with respect to $\{2,3\}$).

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 2.3 — *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 2.4 — *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 2.5 — *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 2.6 — *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 2.7 — 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) the *probability measure* (Definition 2.8), (B) the *counting measure* (Definition 2.10), (C) the *Borel measure* (Definition 2.12), (D) the *Lebesgue measure* (Definition 2.17), and (E) the *random measure* (Definition 2.18). 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: 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 2.8).

▼ Definition 2.8 — *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 2.1 the probability measure is visualized as a mapping from the probability space to the unit interval $[0, 1]$.

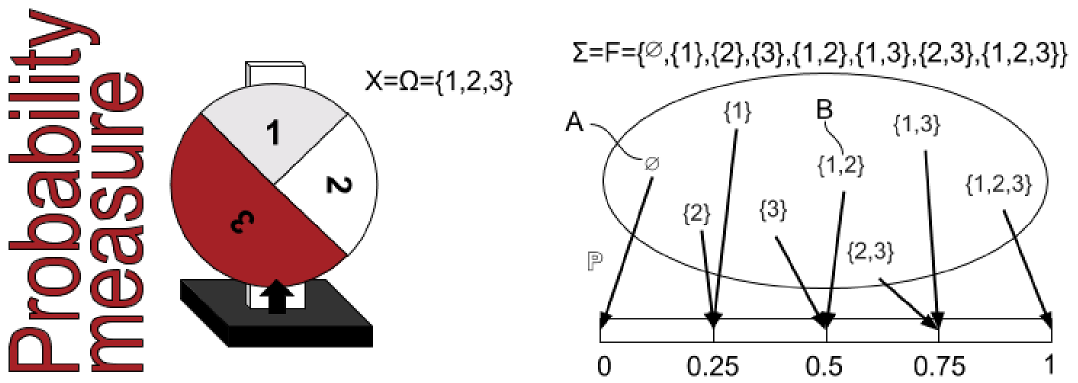


Figure 2.1: A probability measure $\mathbb{P}$ mapping the probability space for 3 events to the unit interval. Left: a turning wheel representing three possible outcomes of which the third is twice as likely as the other two outcomes. Right: a probability measure $\mathbb{P}$ assigned to each outcome. The empty set, $A = \varnothing$, 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 2.9 — *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 $[0, 1]$.

The triple for the probability space $(X, \Sigma, \mathbb{P})$ is also written as $(\Omega, \mathbb{F}, \mathbb{P})$. 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 $\mathbb{F}$.

B: Counting measure

The *counting measure* forms the basis for the definition of discrete probabilities (Schilling, 2005).

▼ Definition 2.10 — *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.

C: Borel measure

The *Borel σ-algebra* defines a σ-algebra for the real line $\mathbb{R}$.

▼ Definition 2.11 — *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 2.12 — *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: $\mu(\emptyset) = 0$;
- 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 2.13 — *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 2.14 — *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.

D: 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 2.15 — *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 2.16 — *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 2.17 — *Lebesgue measure*

A **Lebesgue measure** m on a space $\mathbb{R}^n$ is a Lebesgue outer measure λ with:

- $m(B) = \lambda(B \cup A) + \lambda(B \cup A^c)$.

E: Random measure and random process

A measurable function is defined between two measurable spaces.

▼ Definition 2.18 — *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 variable is a measurable function between two measurable spaces, with as domain a measurable space that is a probability space.

▼ Definition 2.19 — *general random variable*

A **$(\mathbf{X},\Sigma)$-valued random variable** X is a measurable function from probability space $(\Omega,\mathbb{F},\mathbb{P})$ to measurable space (X,Σ).

A random variable is a (X,Σ)-valued random variable with a choice for the codomain and σ-algebra (see Definition 2.20).

▼ Definition 2.20 — *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 $(\mathbb{R},\mathbb{B}_{\mathbb{R}})$.

The codomain is the real line $\mathbb{R}$ and the Borel σ-algebra.

Random variables can be generalized to complex random variables or random elements of any type. A *complex random variable* is a measurable function from Ω to $\mathbb{C}$. A *random elephant* is a measurable function from Ω to a suitable space of elephants (Kingman, 1993).

▼ Definition 2.21 — *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 have encountered a random variable, and a probability measure $\mathbb{P}$ on the original probability space. Now, one might wonder whether probabilities are logically assigned to elements on the measurable space that is the codomain of this random variable. Why does it map to a measurable space and not a measure space actually? This is because (through the σ-algebras of both spaces, or more precisely the random variable itself) the probability measure is *induced* on the target space. This is known as a *probability distribution*:

▼ Definition 2.22 — *probability distribution*

Given a random variable X from $(\Omega,\mathbb{F},\mathbb{P})$ to $(\mathbb{R},\mathbb{B}_\sigma)$, the **probability distribution** μ is the induced probability measure: $\mu(B)=\mathbb{P}(X^{-1}(B))$ for all Borel sets $B\in\mathbb{B}_\sigma$.

The measurable *function* X is inverted: $X^{-1}(\cdot)$. The measure μ exists on $(\mathbb{R},\mathbb{B}_\sigma)$ just as $\mathbb{P}$ exists on $(\Omega,\mathbb{F})$. The notation for the measure μ does not include the original probability space or σ-algebra. The complete notation for the probability distribution μ can be written as a function f of X:

$$f_X(x)=f_{X,(\Omega,\mathbb{F},\mathbb{P}),(\mathbb{R},\mathbb{B}_\sigma)}(x). \tag{2.1}$$

At the left X denotes the random variable, $x \in \Omega$ are the (elementary) outcomes on the sample space Ω. At the right the complete notation adds $\mathbb{F}, \mathbb{P}$ and $\mathbb{R}, \mathbb{B}_\sigma$. The shorthand notation at the left will be used to indicate the real line with a Borel σ-algebra as codomain.

A random variable X is *distributed as* $f_X(x)$, notation:

$$X \sim f_X(x). \tag{2.2}$$

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 2.23 — *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.

2.1.2 Bayesian Inference

Let x be a (S, Σ_S, μ_S)-valued random variable[1], 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 2.24 — *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 2.25 — *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.

[1]The lowercase x is used instead of X in the context of probability distributions as in Eq. 2.1.

▼ Definition 2.26 — *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 2.27 — *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, 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, or parameters given the data. 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 2.28 — *sufficient statistic*

A conditional probability distribution of the data X given a *sufficient statistic* $t = T(X)$ does not depend on parameter θ:

- $P(x|t, \theta) = P(x|t)$

The random variables x and θ define a Bayesian model with observations x and parameters θ.

▼ Definition 2.29 — *Bayesian model*

A **Bayesian model** $f(x, \theta)$ defines a function between observations x and parameters θ with both x and θ random variables.

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 distributed *independent and identically*.

▼ Definition 2.30 — *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 $i \neq j$.

The observations x_i can be distributed in an *exchangeable* sequence in which any order is equally likely.

▼ Definition 2.31 — *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:

$$p(x_0, \ldots, x_{k-1}|\theta) = \prod_{i=0}^{k-1} p(x_i|\theta). \tag{2.3}$$

▼ Definition 2.32 — *likelihood function*

The **likelihood function** is defined as:

$$\mathscr{L}(\theta|x) = p(x|\theta). \tag{2.4}$$

The likelihood of the parameters θ given observations x is the probability of these observations given the parameter values.

The defintion of the likelihood function allows us to find an optimal set of parameter values given the observations. The probability $p(x|\theta)$ can be maximized (Aldrich and Others, 1997).

▼ Definition 2.33 — *maximum likelihood*

Maximum likelihood is defined as:

$$\theta^* \in \operatorname*{argmax}_{\theta} \prod_{i=0}^{k-1} p(x_i|\theta). \tag{2.5}$$

The maximum likelihood method finds the maximum of $\prod_{i=0}^{k-1} p(x|\theta)$ for all possible parameter values θ. The maximum in maximum likelihood 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).

A function $f(\cdot)$ and the logarithm of a function $\log f(\cdot)$ have the same maxima. This is due to the fact that the logarithm is a monotonic function (a monotonically increasing function). The log of a product of logarithms is equal to the sum of the individual logarithms.

▼ Definition 2.34 — *maximum log-likelihood*

Maximum log-likelihood is defined as:

$$\theta^* \in \operatorname*{argmax}_{\theta} \sum_{i=0}^{k-1} \log p(x_i|\theta). \tag{2.6}$$

In the case we have information about the parameters θ we can model this with a probability distribution.

▼ Definition 2.35 — *prior probability distribution*

A **prior probability distribution** defines a probability distribution $p(\theta)$ to 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 2.36 — *maximum a posteriori*

maximum a posteriori estimation is defined as:

$$\theta^* \in \operatorname*{argmax}_{\theta} \sum_{i=0}^{k-1} \log p(x_i|\theta) + \log p(\theta). \tag{2.7}$$

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 2.37 — *Bayesian inference*

Bayesian inference using Bayes' theorem is defined as:

$$f(\theta|x) = p(\theta|x) = \frac{\overbrace{p(x|\theta)}^{\text{likelihood}}\,\overbrace{p(\theta)}^{\text{prior}}}{\underbrace{p(x)}_{\text{normalization constant}}} = \frac{p(x|\theta)p(\theta)}{\int p(x|\theta)p(\theta)d\theta}. \tag{2.8}$$

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 $\propto$.

$$f(\theta|x) \propto p(x|\theta)p(\theta) \tag{2.9}$$

In Bayesian inference $p(\theta|x)$ is calculated. In contrast, in maximum likelihood and maximum a posteriori only parts of Eq. 2.8 are calculated, respectively $p(x|\theta)$ and $p(x|\theta)p(\theta)$. In Section 2.3 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.

There are two supervised learning models, a generative model and a discriminative model. Below we provide their definitions and in Figure 2.2 we give three examples for each model.

▼ Definition 2.38 — *generative model*

A **generative** model defines the joint probability distribution $p(x,\theta)$ and uses Bayes rule to define $p(x|\theta)$.

▼ Definition 2.39 — *discriminative model*

A **discriminative** model defines the conditional probability distribution $p(x|\theta)$ directly.

Figure 2.2 shows three generative and three discriminative models. They are chosen for their structure; from left to right, in both cases, the structure is between the random variables is enriched, first in the form of a sequence structure, then in the form of a graph structure. Figure 2.2 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.

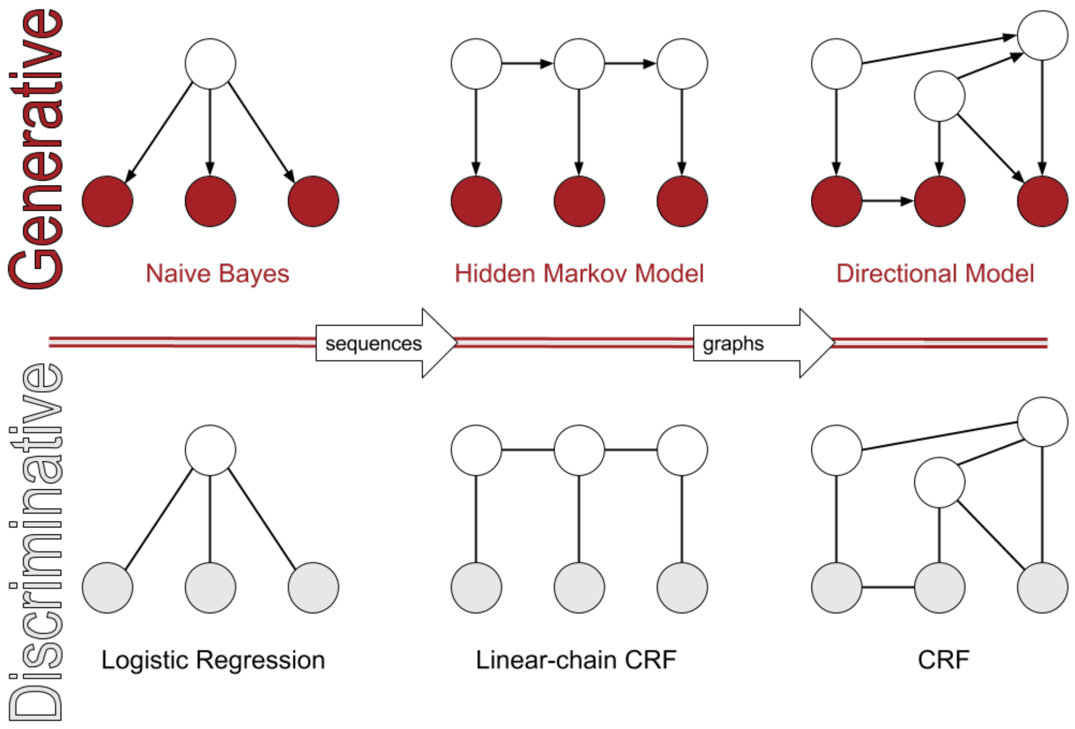


Figure 2.2: Generative models: Naive Bayes Model, Hidden Markov Model, and Directional Model. Discriminative models: Logistic Regression, Linear-chain Conditional Random Fields, and general Conditional Random Fields. Figure adapted from Sutton and McCallum (2011).

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 has lower asymptotic error, but a generative model approaches its asymptotic error faster in the case of a Naive Bayes classifier versus Logistic Regression (Jordan, 2002). However, Xue and Titterington (2008) doubt the existence of such precisely defined regimes. According to them the asymptotic error denotes the error with an increasing number of samples. 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 limit ourselves to generative models.

2.1.3 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 2.40). The finite mixture model is a *sum* over a finite number of probability distributions where each one is weighted (Definition 2.41). The infinite mixture model is a *sum* over an infinite number of probability distributions where each one is weighted (Definition 2.42).

▼ Definition 2.40 — *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 $p(\theta)$:

$$p(\theta|x) \propto p(\theta)\prod_{i=0}^{k-1} p(x_i|\theta). \tag{2.10}$$

A finite mixture model is a sum over a finite number of probability distributions.

▼ Definition 2.41 — *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 $\sum_i w_i = 1$.

$$p(x) = \sum_{i=0}^{k-1} w_i p(x_i). \tag{2.11}$$

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 2.42 — *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 $\sum_i w_i = 1$.

$$p(x) = \sum_{i=0}^{\infty} w_i p(x_i). \tag{2.12}$$

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. In the thesis we will encounter infinite mixture models in Chapters 3 and 4.

A mixture model relates one latent cause (cluster) to each data point. A factorial model or feature allocation model assigns multiple factors to a data point.

▼ Definition 2.43 — *feature model*

A **feature model** is a sum over a number k of sets of probability distributions $p(x_{C_i})$.

$$p(x) = \sum_{i=0}^{k-1} \sum_{j=0}^{C_i} p(x_j). \tag{2.13}$$

A counting model allows each feature to occur multiple times.

▼ Definition 2.44 — *counting model*

A **counting model** is a sum over a number k of sets of probability distributions $p(x_{C_i})$ with each feature occurring a number c_j of times.

$$p(x) = \sum_{i=0}^{k-1} \sum_{j=0}^{C_i} c_j p(x_j). \tag{2.14}$$

2.1.4 General Random Elements

In section 2.1.1 random elements were described in general. Random elements can vary from random vectors, random distributions, random clusters (partitions), to random trees. Table 2.1 describes the random elements and the corresponding examples of random processes in the literature. Below we mention them with the appropriate references.

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.

Table 2.1: 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

2.1.5 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 2.3).

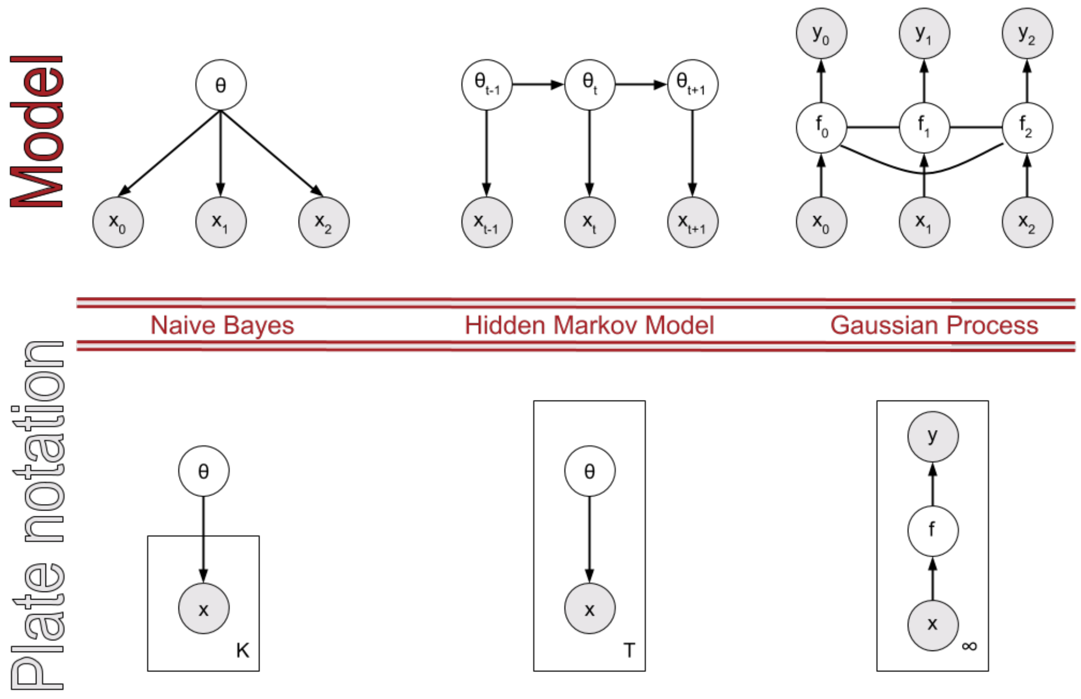


Figure 2.3: Top: graphical model of a Naive Bayes, hidden Markov model, and Gaussian process. Bottom: corresponding plate notation of the Naive Bayes, hidden Markov model, and Gaussian process. Observed variables are denoted by a circle that is shaded.

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.

2.1.6 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 2.45 — *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 2.2: 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).

2.1.7 Exchangeability

Here we recall Definition 2.31 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 2.46 — *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

$$p(x_0, \ldots, x_{k-1}) = \int_{\Sigma_X(X)} \prod_{i=0}^{k-1} p(x_i|\theta)\mu(d\theta) \qquad \forall k \geq 1. \tag{2.15}$$

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 2.3) of exchangeable structures that have a theorem that describes 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 2.3: 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

2.1.8 Stick-breaking Representation

Now 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 2.47 — *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 $\phi \sim GEM(\alpha, \beta)$.

$$w_k \overset{i.i.d.}{\sim} Beta(1-\beta, \alpha + k\beta) \qquad k = 1, \ldots, K \tag{2.16}$$

$$\phi_k = w_k \prod_{i=1}^{k-1}(1 - w_i) \tag{2.17}$$

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.

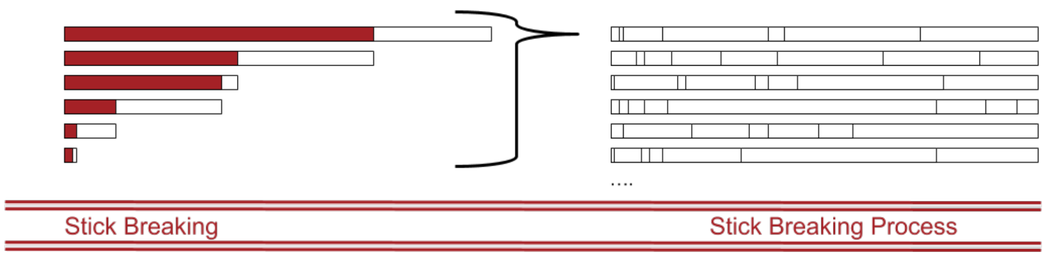


Figure 2.4: The stick-breaking representation. Left: at the first row, the stick is broken at x_0, at the next rows the remaining part of the stick is broken x_i with $i > 0$. Only six iterations are shown. Right: samples of a stick-breaking process. The first row shows the stick ratios from the stick-breaking representation at the left. The next rows show other samples from the same process.

Figure 2.4 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. In this manner generates a stick-breaking process 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 rule. Namely, it is possible to approximate a distribution over partitions by truncating a stick-breaking process. The stick-breaking is then only performed a limited number of times (Kurihara et al., 2007).

In Section 2.2.1 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.

2.2 Five Random Processes

In this section we investigate five random processes. The Dirichlet process (Section 2.2.1), the Beta process (Section 2.2.2), the Gamma process (Section 2.2.3), the Pitman-Yor process (Section 2.2.4), and the hierarchical Dirichlet process (Section 2.2.5). We compare the random processes in Section 2.2.6.

2.2.1 Dirichlet Process

The Dirichlet process is presented as a measure (Section A), is shown to have a sequential representation in the form of the Chinese restaurant process (Section B), and a stick-breaking representation (Section C).

A: Dirichlet Process as a Measure

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

▼ Definition 2.48 — *Dirichlet process*

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

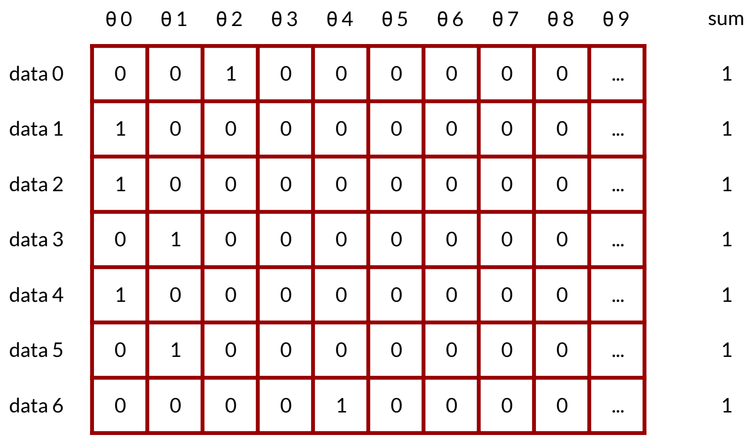
$$X \sim DP(\alpha, H)$$

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:

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

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

The Dirichlet process can be used as a prior for a mixture model (Definition 2.42). This is visualized in (Figure 2.5).



	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 2.5: Matrix representation of a mixture model. At the horizontal axis the latent variables (potentially infinite clusters). At the vertical axis the data items. The rows sum up to one. A data item is only assigned to one cluster.

B: Chinese Restaurant Process

De Finetti's theorem (Definition 2.46) 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 CRP.

▼ Definition 2.49 — *Chinese restaurant process*

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

$$p(z_i = k | z_0, \dots, z_{i-1}) = \begin{cases} \frac{n_k}{\alpha+i} & \text{if } k \leq K_+ \\ \frac{\alpha}{\alpha+i} & \text{if } k > K_+ \end{cases} \tag{2.18}$$

The conditional probability of a cluster assignment z_i for data item y_i given the cluster assignments $z_0, \dots, z_{i-1}$ is proportional to the number of data items n_k assigned to an existing cluster k, or proportional to α for a new cluster.

The Chinese Restaurant Process is visualized in Figure 2.6.

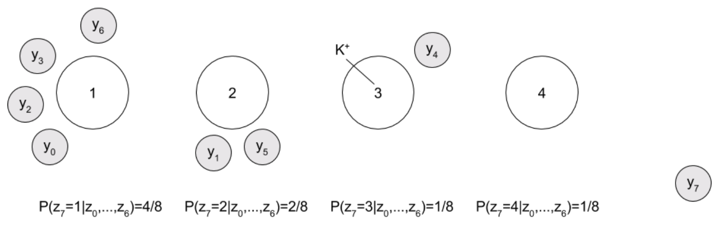


Figure 2.6: The Chinese Restaurant Process with i customers already sitting down. A new customer $y_{i=7}$ arrives and gets assigned, z_i. This is an existing table $\{1,2,3\}$ with a probability proportional to the number of customers n_i sitting at that table: $n_i/(\alpha+i)$, or a new, empty table 4 with probability $1/(\alpha+i)$. In the visualized Chinese restaurant process the dispersion factor $\alpha = 1$.

C: Stick-breaking Representation of the Dirichlet process

▼ Definition 2.50 — *stick-breaking representation of the Dirichlet process*

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

$$\phi_k \sim GEM(\alpha, 0) \tag{2.19}$$

$$\theta_k \sim H \tag{2.20}$$

$$G = \sum_{k=1}^{\infty} \phi_k \delta(\theta, \theta_k) \tag{2.21}$$

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

The weights ϕ_k are sampled from the stick-breaking process $GEM(\alpha, 0)$ (see Definition 2.47). The parameter values θ_k are sampled from the base measure H. To sample from the Dirichlet Process we have to sample these parameters with the given weights.

If the stick-breaking process is used as a prior for a mixture, then the cluster assignments z_i are sampled according to the mixing proportions ϕ:

$$\phi \sim GEM(\alpha, 0) \tag{2.22}$$

$$\theta_k \sim H \tag{2.23}$$

$$G = \sum_{k=1}^{\infty} \phi_k \delta(\theta, \theta_k) \tag{2.24}$$

$$z_i \sim Mult(\phi) \tag{2.25}$$

$$x_i \sim F(\theta_{z_i}) \tag{2.26}$$

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

The Dirichlet process, the Dirichlet process as prior for a mixture model, the Chinese restaurant process and the stick-breaking representation can be compared in Figure 2.7.

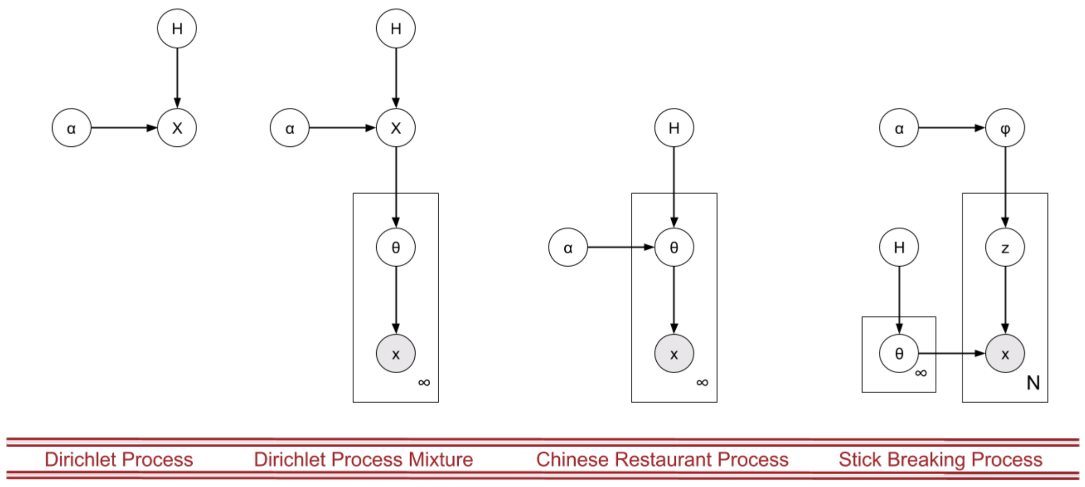


Figure 2.7: From left to right: (1) The Dirichlet process $X \sim DP(\alpha, H)$; (2) The Dirichlet mixture model with X the prior for a sum $\sum_i w_i p(x_i|\theta_i)$; (3) the Chinese restaurant process with X marginalized out; and (4) the Stick-breaking process with a distribution over partition sizes π and indicator variables z_i.

2.2.2 Beta Process

The Beta process is presented as a measure (Section A), is shown to have a sequential representation in the form of the Indian buffet process (Section B), and a stick-breaking representation (Section C).

A: Beta Process as a Measure

A Beta process (Hjort, 1990) is a random process with a countably infinite collection of weighted atoms in a space $(X, \mathbb{B})$ with weights that are in between $[0, 1]$.

▼ Definition 2.51

Let $(X, \mathbb{B})$ be a Borel space, ν a finite measure, and $\alpha > 0$ a scale parameter, then a **Beta process** is a Lévy process on $(X, \mathbb{B})$ with its Lévy measure ν corresponding to the density:

$$\nu(dw) = \alpha w^{-1}(1-w)^{\alpha-1}dw \tag{2.27}$$

with $w > 0$.

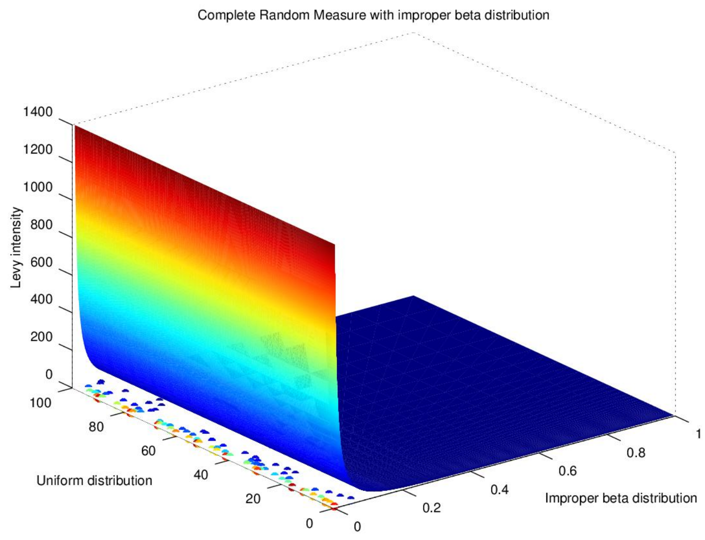


Figure 2.8: A Completely Random Measure with a Lévy intensity defined on the product space $\Omega \otimes (0,1)$. Here Ω is a bounded interval on which the base measure $B_0 = U(0,100)$ is defined. On $(0,1)$ we define an improper beta distribution $\alpha w^{-1}(1-w)^{\alpha-1}$. In this example $\alpha = 10$. This is how a Beta Process can be generated from a nonhomogeneous spatial Poisson point process. This has been visualized before (Jordan, 2010). The image is produced by rejection sampling using a homogeneous Poisson point process at max(ν) over $w = [0.01, 0.9]$. For $w \rightarrow 0$ this maximum would go to ∞ and all points would be rejected. Hence, the points should be denser for w around 0 and should be seen as an approximation of the actual process.

In Figure 2.8 the Beta Process is generated from a Completely Random Measure (see Section 2.1.6) with a Lévy intensity defined on $\Omega \otimes (0,1)$ (Thibaux and Jordan, 2007). In this case Ω is the so-called base measure B_0 and is assumed uniform over a bounded region. The $(0,1)$ space is equipped with an improper Beta distribution. It is called improper or degenerate because the scale parameter of the standard Beta distribution is set to zero. This has the consequence that the integral is infinite: $\nu(\Omega \otimes (0,1)) = \infty$. It is due to the fact that the density $w^{-1}(1-w)^{\alpha-1}$ goes to infinity for $w \rightarrow 0$. That means that a countable infinite number of points can be obtained from the Poisson process.

The Beta process can be used as a prior for a feature or factorial model (Definition 2.43). This is visualized in (Figure 2.9).

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

Figure 2.9: Matrix representation of a factorial model. At the horizontal axis the latent variables (potentially infinite number of features). At the vertical axis the data items. The rows sum up to nonnegative integers. A data item has multiple features.

B: Indian Buffet Process

The Beta process has a sequential representation in the form of the Indian Buffet Process:

▼ Definition 2.52 — *Indian buffet process*

An **Indian buffet process** is a sequential process that is an exchangeable distribution over sparse binary matrices:

$$p(z_{i,j} = k | z_{0,0}, \dots, z_{i-1,K_+}) = \begin{cases} \frac{n_{-i,k}}{i} & \text{if } k \leq K_+ \\ \frac{\lambda^{k_{new}} e^{-\lambda}}{k_{new}!} & \text{if } k > K_+ \end{cases} \tag{2.28}$$

Here $\lambda = \alpha/i$, $k_{new} = K_+ - k$. The i'th data item samples an existing column with a probability of the number of times it has been sampled before divided by its index, $n{-}i, k/i$. It samples a new column with a probability according to a Poisson distribution, $\lambda^{k_{new}} e^{-\lambda}/k_{new}!$. The conditional form of the sequential presentation describes a closed-form solution for Gibbs sampling, section 2.3.4 (Ghahramani and Griffiths, 2005).

C: Stick-breaking Representation of the Beta process

▼ Definition 2.53 — *stick-breaking representation of the Beta process*

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

$$\phi_{k,j} \sim GEM(\alpha, 0) \tag{2.29}$$

$$C_k \sim Poisson(\gamma) \tag{2.30}$$

$$\theta_{k,j} \sim \frac{1}{\gamma} H \tag{2.31}$$

$$G = \sum_{k=1}^{\infty} \sum_{j=1}^{C_i} \phi_{k,j} \delta(\theta, \theta_{k,j}) \tag{2.32}$$

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

The Beta process is used in linguistics (He et al., 2013; Vanhainen and Salvi, 2012), computer vision (Zhou et al., 2011; Gao and Sun, 2013), risk assessment (Li et al., 2014), and medicin (Ross et al., 2014).

2.2.3 Gamma Process

The Gamma process is presented as a measure (Section A), is shown to have a sequential representation in the form of a multi-scoop Indian buffet process (Section B), and a stick-breaking representation (Section C).

A: Gamma Process as a Measure

A Gamma process is a random process with independent gamma distributed increments (Ferguson, 1974). Below we provide a formal definition.

▼ Definition 2.54 — *Gamma process*

Let $(X, \mathbb{B})$ be a Borel space, ν a finite measure, and $\alpha > 0$ a scale parameter, then a **Gamma process** is a Lévy process on $(X, \mathbb{B})$ with its Lévy measure ν corresponding to the density:

$$\nu(dw) = w^{-1} e^{-\alpha w} dw \tag{2.33}$$

with $w > 0$.

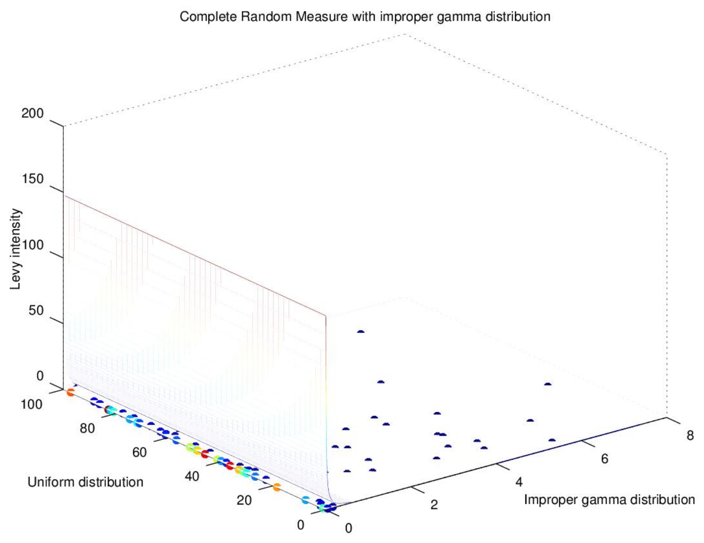


Figure 2.10: A Completely Random Measure with a Lévy intensity defined on the product space $\Omega \otimes \mathbb{R}$. Here Ω is a bounded interval on which the base measure $B_0 = U(0, 100)$ is defined. On $\mathbb{R}$ we define an improper gamma distribution $w^{-1}e^{-\alpha w}$. In this example $\alpha = 1$. This is how a Gamma Process can be generated from a nonhomogeneous spatial Poisson point process. The image is produced by rejection sampling in the same way as Figure 2.8.

The Gamma process can be used as a prior for a counting model (Definition 2.44). This is visualized in (Figure 2.11).

	θ0	θ1	θ2	θ3	θ4	θ5	θ6	θ7	θ8	θ9	sum
data 0	4	0	3	0	0	0	0	0	0	...	7
data 1	1	1	0	0	1	0	0	0	0	...	3
data 2	2	0	0	0	0	0	0	0	0	...	2
data 3	0	0	0	0	0	4	1	0	0	...	5
data 4	0	0	0	2	0	0	0	0	0	...	2
data 5	0	0	0	0	0	4	0	0	6	...	10
data 6	0	0	0	0	0	4	0	0	0	...	4

Figure 2.11: Matrix representation of a counting model. At the horizontal axis the latent variables (potentially infinite number of features). At the vertical axis the data items. Each data item has one or more features and these features can occur multiple times.

B: Multi-Scoop Indian Buffet Process

Find sequential representation. It is an Indian Buffet Process where the customers can pick up multiple scoops from the same dish.

(Zhou et al., 2012)

C: Stick-breaking Representation of the Gamma process

The stick-breaking representation of the Gamma process introduces an additional Gamma distribution that defines the number of times a feature is represented in a data object compared to the Beta process, see Definition 2.55, (Roychowdhury and Kulis, 2015).

▼ Definition 2.55 — *stick-breaking representation of the Gamma process*

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

$$\phi_{k,j} \sim GEM(\alpha, 0) \tag{2.34}$$

$$G_{k,j} \sim Gamma(\alpha + 1, c) \tag{2.35}$$

$$C_k \sim Poisson(\gamma) \tag{2.36}$$

$$\theta_{k,j} \sim \frac{1}{\gamma} H \tag{2.37}$$

$$G = \sum_{k=1}^{\infty} \sum_{j=1}^{C_i} G_{k,j} \phi_{k,j} \delta(\theta, \theta_{k,j}) \tag{2.38}$$

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

The Gamma process is used in risk theory (Dufresne et al., 1991), spatial statistics (Wolpert and Ickstadt, 1998; Rao and Teh, 2009), erosion (Singpurwalla, 1997; Abdel-Hameed, 2012), and finance (Madan and Seneta, 1990; Küchler and Tappe, 2008).

2.2.4 Pitman-Yor Process

The Pitman-Yor process (PYP) introduces another parameter d with respect to the Dirichlet process. It has been developed by Pitman and Yor as the two-parameter Poisson-Dirichlet distribution (Pitman and Yor, 1997). The Pitman-Yor process has the following definition.

▼ Definition 2.56 — *Pitman-Yor process*

A Pitman-Yor process PY over a set S can be used to draw sample paths X:

$$X \sim PY(d, \alpha, H)$$

with $\alpha > -d$ a strength parameter, $0 \leq d < 1$ a discount parameter, and H a measure on S.

The Pitman-Yor process generalizes the Dirichlet process. The Pitman-Yor process has a stick-breaking representation in which sticks are drawn from $GEM(\alpha, \beta)$. The Dirichlet process has a stick-breaking representation in which sticks are drawn from $GEM(\alpha, 0)$, see Def. 2.53.

▼ Definition 2.57

The **stick-breaking representation** of the PYP states that if

$$\phi_k \sim GEM(\alpha, \beta) \tag{2.39}$$

$$\theta_k \sim H \tag{2.40}$$

$$G = \sum_{k=1}^{\infty} \phi_k \delta(\theta, \theta_k) \tag{2.41}$$

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

The Pitman-Yor process is used in quite a few applications, such as language models (Teh et al., 2006), scene segmentation (Sudderth and Jordan, 2009), speech induction (Blunsom and Cohn, 2011), and time series (Bassetti et al., 2014).

2.2.5 Hierarchical Dirichlet Process

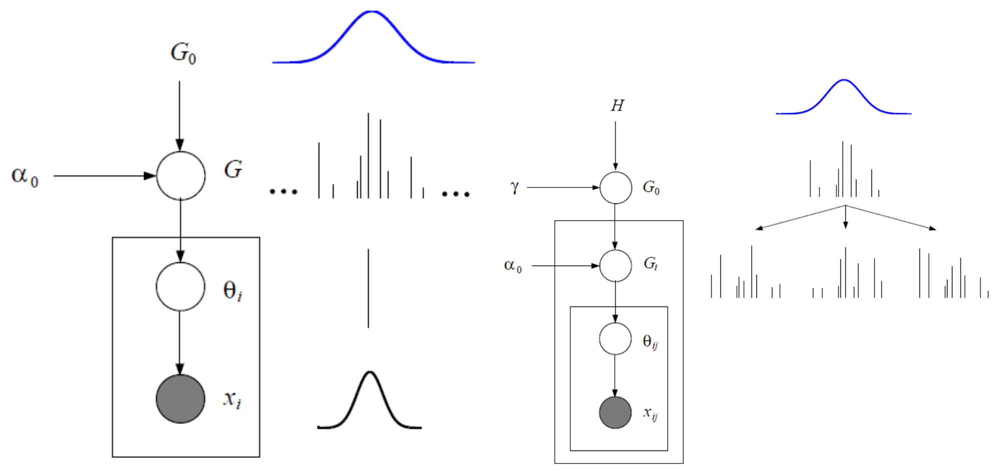
The HDP extends the Dirichlet mixture model with a hierarchical structure (Teh et al., 2006).

▼ Definition 2.58

A Hierarchical Dirichlet process HDP over a set S can be used to draw sample paths X:

$$G_0 \sim DP(\gamma, H)$$
$$X_i \sim DP(\alpha_0, G_0) \text{ for each group } i$$

with a Dirichlet Process with a general γ dispersion parameter and base distribution H as a measure on S of which the generated distributions G_0 are used as base distribution for each group distribution X_i.



(a) Dirichlet Process Mixture Model. Each draw from the process corresponds to a parameter. Each parameter is associated with a distribution (in this case a Gaussian).

(b) Hierarchical Dirichlet Process. Observe that the location of the atoms are fixed through the highest layer G_0. The only freedom left to express by G_i is in the weights of those atoms. This reflects a decomposition in a structural and non-structural component.

Figure 2.12: The difference visualized between a Dirichlet Process mixture and a hierarchical Dirichlet process. It illustrates also that the input of a Dirichlet process does not have to be a continuous function. If it is a continuous distribution it will become a discrete distributed almost surely. If it is a discrete distribution, it will have atoms at the locations where the discrete distribution had its probability mass concentrated.

The hierarchical Dirichlet process uses the outcome of a Dirichlet Process as a starting point to define multiple distributions with atoms at the same locations, while they come equipped with different weights. So, the Dirichlet process on the lower level uses not a continuous distribution as input, but a discrete one, generated by the DP at the top layer. Note, that the Dirichlet Process will *generate* an almost surely (a.s.) discrete distribution, but it can also have a discrete distribution as *prior* H.

2.2.6 Comparison of Random Processes

The Dirichlet process (Section 2.2.1), the Beta process (Section 2.2.2), the Gamma process (Section 2.2.3), the Pitman-Yor process (Section 2.2.4), and the hierarchical Dirichlet process (Section 2.2.5) are not be compared on performance or computationally efficiency. The processes each represent different assumptions on the model structure. The Dirichlet process is suited for clustering problems where observations have to be assigned to a single class. The Beta process is a good model for applications with combinatorial structure: features that are shared among multiple objects. The Gamma process is a model for applications with counting: features are shared among multiple objects and there is a number of features per object. The Pitman-Yor process fits clustering problems where the distribution over clusters sizes obeys a power law. The hierarchical Dirichlet process is a typical example of a process that can model additional structure within a cluster.

2.3 Inference

There will be six inference methods described, all sampling methods. Inverse transform sampling is described in Section 2.3.1. Rejection sampling in Section 2.3.2. Approximate Bayesian computation in Section 2.3.3. Gibbs sampling in Section 2.3.4. Metropolis-Hastings in Section 2.3.5. Split-Merge MCMC in Section 2.3.6. We rapport for every inference method the corresponding algorithm in pseudo code. We compare the inference methods in Section 2.3.7.

2.3.1 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 $p(x = g)$. This 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 $f(x)$

```
1: procedure INVERSE TRANSFORM SAMPLING($f(x)$)        ▷ Distribution to sample from.
2:     $F(x) = P(X \leq x) \quad \forall x \in X$      ▷ Create cumulative distribution function $F(x)$.
3:     for $t = 1 \rightarrow T$ do
4:         $u \sim U(0,1)$                               ▷ Sample from uniform distribution.
5:         $x \sim F^{-1}(u)$                            ▷ Sample $x$ from (the inverse) $F^{-1}(x)$.
6:         $X = X \cup x$
7:     end for
8:     return $X$                                        ▷ $X$ will have the distribution of $f(x)$.
9: 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.

2.3.2 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 $g(x)$.

$$X = S(f(x), g(x)) \tag{2.42}$$

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

Algorithm 2 Rejection sampling for $f(x)$

```
 1: procedure REJECTION SAMPLING($f(x), g(x)$)        ▷ Target and proposal distribution.
 2:     for $t = 1 \to T$ do
 3:         $x^t \sim g(x)$                                ▷ Generate $x^t$ from $g(x)$
 4:         $u \sim U(0,1)$                                ▷ Inverse transform sampling
 5:         $p_0 = f(x)/(Mg(x))$
 6:         if $u < p_0$ then
 7:             $X = X \cup x^t$                           ▷ Accept
 8:         end if
 9:     end for
10:     return $X$                                     ▷ $X$ will have the distribution of $f(x)$
11: end procedure
```

We can use rejection sampling to *sample* from the *posterior* $f(\theta|x)$ given that we know the *exact* likelihood function and that we can *sample* from the prior. 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. Hence 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* from the prior, which we indicate by a tilde, $\sim p(\theta)$.

$$\Theta = S(\sim p(\theta), p(x|\theta), x) \tag{2.43}$$

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

```
 1: procedure REJECTION SAMPLING($p(\theta), p(x|\theta), x$)   ▷ Requires prior, likelihood and
    observations.
 2:     for $t = 1 \to T$ do
 3:         $\theta^t \sim p(\theta)$                          ▷ Generate $\theta^t$ from prior
 4:         $u \sim U(0,1)$                                ▷ Inverse transform sampling
 5:         $p_0 = p(x|\theta)$
 6:         if $u < p_0$ then
 7:             $\Theta = \Theta \cup \theta^t$                ▷ Accept
 8:         end if
 9:     end for
10:     return $\Theta$                        ▷ $\Theta$ will have the distribution of $f(\theta|x)$
11: 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 2.13).

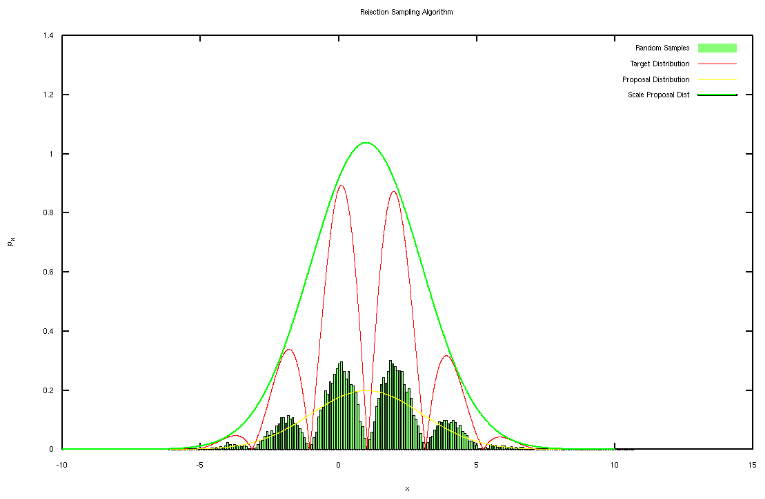


Figure 2.13: A Gaussian is placed over the complex target probability density function. Subsequently the samples that fall in between these two 'envelopes' are rejected. This results in a sampling scheme that follows exactly the more complicated probability density function. Note that if the function 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.

2.3.3 Approximate Bayesian Computation

In approximate Bayesian computation (ABC) (Rubin and Others, 1984) the likelihood function does not need to be calculated[2] (Sisson and Fan, 2011). In contrast, it is assumed that there is a model available that allows to generate observations given the (searched for) parameters. In ABC for each configuration of parameters a set of observations is generated.

$$\Theta = S(\sim p(\theta), X, \sim M(\theta), d(X^t, X), \epsilon) \tag{2.44}$$

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

[2]ABC is also called likelihood-free computation

Algorithm 4 Approximate Bayesian computation

```
1:  procedure APPROXIMATE BAYESIAN COMPUTATION($p(\theta)$, $X$, $M$, $d$, $\epsilon$)    ▷ Requires prior, observations, model, distance function, and threshold.
2:      for $t = 1 \to T$ do
3:          $\theta^t \sim p(\theta)$                                   ▷ Generate $\theta$ from prior
4:          $X^t \sim M(\theta)$                                   ▷ Simulate observations $X^t$ from model $M$
5:          $\rho = d(X^t, X)$      ▷ Calculate distance between simulated and actual observations
6:          if $\rho \leq \epsilon$ then
7:              $\Theta = \Theta \cup \theta^t$                  ▷ Accept $\theta^t$ if distance falls under threshold $\epsilon$.
8:          end if
9:      end for
10:     return $\Theta$                                       ▷ $\Theta$ will have the distribution of $f(\theta|X)$
11: 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.

2.3.4 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 optimizes over one variate in the multivariate probability distribution at a time. The update value is set and fixed. Then, the next variate is chosen in a round-robin like manner.

$$\Theta = S(X, \sim p(\theta_i|\theta_{-i}, X), \sim p(\theta), B) \tag{2.45}$$

The Gibbs algorithm is given in Algorithm 5. Some explanation on the notation is as follows. The multiple parameters in the multivariate probability distribution are denoted by θ. The parameters are denoted individually with θ_i. The set of all parameters except for i is denoted by θ_{-i}. If we sample a parameter we write θ^t with t the iteration or sampling round. The set of parameter samples has capital letter Θ.

Algorithm 5 Gibbs sampling

```
1: procedure GIBBS SAMPLING($p(\theta_i|\theta_{-i},X), p(\theta), X, B$)   ▷ Requires parameters,
   observations and burn-in.
2:     $\theta^0 \sim p(\theta)$                                  ▷ Set parameters to some initial value
3:     for $t = 1 \rightarrow T$ do
4:         for $i = 1 \rightarrow k$ do
5:             $\theta_i^t \sim p(\theta_i^{t-1}|\theta_{-i}^t, X)$   ▷ Generate $\theta_i^t$ from the full conditional probability
6:         end for
7:         $\Theta = \Theta \cup \theta^t$
8:     end for
9:     $\Theta_{B:T} \in \Theta$                                  ▷ Get $\Theta_T$ set, from burn-in $B$ to end of run $T$
10:    $\Theta \sim \Theta_{B:T}$                                 ▷ Sample $\Theta$ from correlated $\Theta_{B:T}$
11:    return $\Theta$
12: 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.

In the physics literature Gibbs sampling is known as Glauber dynamics or the heat bath algorithm. First, observe that Gibbs sampling does not necessary require an actual calculation of the conditional probability in all cases. The obvious exception is for the observations, which are already known. Second, observe that a neat optimization procedure arises when conjugate priors are used. A conjugate prior leads to a posterior distribution that can be described analytically. In such a case it is computationally unnecessary to perform sampling. It is much faster to use the actual available analytic description. This is commonly called collapsed Gibbs sampling.

2.3.5 Metropolis-Hastings Sampling

Metropolis-Hastings (Metropolis et al., 1953) is one of the most well-known Markov chain Monte Carlo (MCMC) algorithms. An MCMC algorithm uses a Markov chain (see Gibbs sampling, Section 2.3.4) 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).

$$\Theta = S(X, \theta^0, Q(\theta^{t+1}|\theta^t), f(\theta, X)) \tag{2.46}$$

Here we need $Q(\theta^{t+1}|\theta^t)$ explicitly as well as samples from it.

Algorithm 6 Metropolis-Hastings sampling

```
 1: procedure METROPOLIS-HASTINGS SAMPLING($\theta^0, X, Q, f$) ▷ Requires initial parameters,
    observations, proposal distribution, and function proportional to desired distribution
 2:     for $t = 1 \to T$ do
 3:         $\theta^{t+1} \sim Q(\theta^{t+1}|\theta^t)$          ▷ Sample from proposal distribution $Q$
 4:         $\alpha = \frac{f(\theta^{t+1}, X^{t+1}) Q(\theta^{t+1}|\theta^t)}{f(\theta^t, X^t) Q(\theta^t|\theta^{t+1})}$          ▷ Calculate acceptance
 5:         $u \sim U(0, 1)$          ▷ Inverse transform sampling
 6:         if $\alpha > u$ then
 7:             $\Theta = \Theta \cup \theta^{t+1}$          ▷ Accept $\theta^{t+1}$
 8:         else
 9:             $\Theta = \Theta \cup \theta^t$          ▷ Reuse previous sample (note, different from rejection)
10:         end if
11:     end for
12:     return $\Theta$          ▷ $\Theta$ will be samples from the distribution $f(\theta|x)$
13: 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.

2.3.6 Split-Merge MCMC Sampling

The discussed sampling methods do not assume much 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 cluster. This has a disadvantage, the procedure in which a cluster is split into two clusters is very slow by moving data points one by one from an old to a new cluster. Much more efficient sampling methods can be designed if we would be able to handle large chunks of cluster assignments at once.

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

```
 1: procedure Split-Merge MCMC Sampling($\theta^0, X, Q, f$)   ▷ Requires initial parameters,
    observations, proposal distribution, and function proportional to desired distribution
 2:     for $t = 1 \rightarrow T$ do
 3:         $i \sim D(0, N-1)$                                  ▷ Sample observation $i$ discretely
 4:         $j \sim D(0, N-1)$                                  ▷ Sample observation $j$ discretely
 5:         if $c_i == c_j$ then
 6:             $c_{old} = c_i$
 7:             $\theta^{t+1}_{c_{new}} \sim Q(\theta^{t+1}|\theta^t)$        ▷ Sample from proposal distribution $Q$
 8:             for $k \in c_{old}$ do
 9:                 $c_k \sim C(c_{old}, c_{new})$                  ▷ Assign to new cluster categorically
10:             end for
11:         else
12:             $c_{merge} = c_i$
13:             for $k \in c_j$ do
14:                 $c_k = c_{merge}$                               ▷ Assign all to first cluster
15:             end for
16:         end if
17:         $\alpha = \frac{f(\theta^{t+1}, X^{t+1}) Q(\theta^{t+1}|\theta^t)}{f(\theta^t, X^t) Q(\theta^t|\theta^{t+1})}$        ▷ Calculate acceptance
18:         $u \sim U(0,1)$                                     ▷ Inverse transform sampling
19:         if $\alpha > u$ then
20:             $\Theta = \Theta \cup \theta^{t+1}$                           ▷ Accept $\theta^{t+1}$
21:         else
22:             $\Theta = \Theta \cup \theta^t$     ▷ Reuse previous sample (note, different from rejection)
23:         end if
24:     end for
25:     return $\Theta$                     ▷ $\Theta$ will be samples from the distribution $f(\theta|x)$
26: 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).

2.3.7 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 2.3.4). If the model becomes more complicated and nonconjugate split-merge sampling will likely accelerate the inference (Section 2.3.6).

2.4 Chapter Conclusions

Chapter 3 describes Gibbs sampling to perform inference over an infinite set of lines. Gibbs sampling requires conditional probabilities. These are given in closed-form because there is a conjugate description of the line parameters given the points that form the lines in this application.

Chapter 4 describes Split-Merge MCMC sampling to perform inference over an infinite set of line segments. The parameters for line segments do not have a conjugate description. Metropolis-Hastings can be used to perform inference over the line segments, but the search space is quite large. The Split-Merge MCMC method performs faster inference than Metropolis-Hastings because it is able to split and merge line segments (with multiple points ascribed to them) at once.

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

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 (Zhang and Kǒsecká, 2007; Gallo et al., 2011; Chen et al., 2001). Second, the noise level is hardcoded 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.

3.1 Infinite Line Model

The Dirichlet process has been previously described as prior for a mixture distribution (in Figure 2.6, see Section 2.2.1). 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).

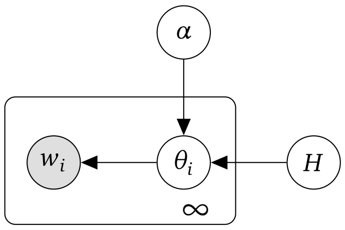


Figure 3.1: The infinite line model using the Chinese restaurant process representation (compare with Figure 2.6). Top: α, the concentration parameter of the Dirichlet process. Bottom, left to right: w_i, the observation, an individual point in a 2D space; θ_i, the parameters (intercept, slope) of the line belonging to observation w_i; H, the base distribution from which line parameter values are sampled.

The infinite line model is visualized in Figure 3.1 using plate notation (see Section 2.1.5). In Section 3.1.1 it is described how θ_i is sampled from H and α. In Section 3.1.2 it is described how w_i is sampled from θ_i. In Section 3.1.3 the prior H for θ_i is described. In Section 3.1.4 it is described how line parameters θ_i can be updated given the data w_i.

3.1.1 Posterior Predictive for a Line given Other Lines

Let us start reiterating the definition of the Dirichlet process. Let H be a distribution over Θ and let α be scalar. The Dirichlet process generates a distribution $G \sim DP(\alpha, H)$:

$$G(\theta_1, \ldots, \theta_\infty) \sim DP(\alpha, H(\theta_1, \ldots, \theta_\infty)). \tag{3.1}$$

A Dirichlet process assigns a Dirichlet distribution to every parameter partition $\Theta_1, \ldots, \Theta_r$:

$$(G(\Theta_1), \ldots, G(\Theta_r)) \sim Dir(\alpha H(\Theta_1), \ldots, \alpha H(\Theta_r)). \tag{3.2}$$

The Dirichlet is conjugate to the categorical:

$$\begin{gathered}(G(\Theta_1), \ldots, G(\Theta_r)) \mid \theta_1, \ldots, \theta_n \sim Dir(\alpha H(\Theta_1) + n_1, \ldots, \alpha H(\Theta_r) + n_r),\\ n_k = \sum_{j=1}^{n} \delta_{\theta_j}(\Theta_k).\end{gathered} \tag{3.3}$$

To study the details about the conjugacy of the Dirichlet distribution with respect to multinomial and categorical distributions we refer to the derivations in Appendix B.

In the above notation, $\delta_{\theta_j}(\Theta_k)$ is a Dirac measure (a generalization of the Dirac delta function), also known as an indicator function. Given a set Θ_k with a σ-algebra over subsets of Θ:

$$\delta_{\theta_j}(\Theta_k) = 1_{\Theta_k}(\theta_k) = \begin{cases} 1 & \text{if } \theta_j \in \Theta_k \\ 0 & \text{if } \theta_j \notin \Theta_k \end{cases}. \tag{3.4}$$

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

$$G(\cdot) \mid \theta_1, \ldots, \theta_n \sim DP\left(\alpha + n, \frac{\alpha}{\alpha + n} H(\cdot) + \frac{n}{\alpha + n} \frac{\sum_{j=1}^{n} \delta_{\theta_j}(\cdot)}{n}\right). \tag{3.5}$$

The posterior base 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 $\alpha + n$. Note that $\delta_{\theta_j}(\cdot)$ is a distribution, the Dirac measure Eq. 3.4.

The posterior predictive for a new parameter θ_{n+1} has the form:

$$P(\theta_{n+1} \in \Theta_k \mid \theta_1, \ldots, \theta_n) = \frac{1}{\alpha + n} \left(\alpha H(\Theta_k) + \sum_{j=1}^{n} \delta_{\theta_j}(\Theta_k) \right). \tag{3.6}$$

In other words, the posterior predictive of θ_{n+1} given the parameters $\theta_1, \ldots, \theta_n$ in Eq. 3.6 has exactly the same form as the posterior base distribution G given the parameters $\theta_1, \ldots, \theta_n$ (Blackwell and MacQueen, 1973) in Eq. 3.5, namely:

$$\theta_{n+1} \mid \theta_1, \ldots, \theta_n \sim \frac{1}{\alpha + n} \left(\alpha H(\theta_{n+1}) + \sum_{j=1}^{n} \delta(\theta_j - \theta_{n+1}) \right). \tag{3.7}$$

A normal Dirac delta function $\delta(\theta_j - \theta_{n+1})$ can be used here, which is only non-zero when θ_j is equal to θ_{n+1}.

Equivalently, if we describe θ_n conditioned on $\theta_1, \ldots, \theta_{n-1}$ we have to run over $n-1$ rather than n parameters:

$$\theta_n \mid \theta_1, \ldots, \theta_{n-1} \sim \frac{1}{\alpha + n - 1} \left(\alpha H(\theta_n) + \sum_{j=1}^{n-1} \delta(\theta_j - \theta_n) \right). \tag{3.8}$$

Due to the exchangeability property we can also consider any other parameter update (Neal, 2000):

$$\theta_i \mid \theta_{-i} \sim \frac{1}{\alpha + n - 1} \left(\alpha H(\theta_i) + \sum_{j \neq i} \delta(\theta_j - \theta_i) \right). \tag{3.9}$$

The notation θ_{-i} means every parameter θ except for the one equal to θ_i.

3.1.2 Likelihood of Data given a Line

The likelihood of data given line parameters is defined to be according to the **Bayesian linear regression** model. The Bayesian linear regression model for a single line (Box and Tiao, 2011) assumes a linear relationship between the independent x_i and dependent variables y_i with Gaussian noise added in the y-direction. The individual points i are drawn from a Normal distribution:

$$y_i \sim \mathcal{N}(x_i \beta, \sigma^2). \tag{3.10}$$

The (column) vector β maps the (row) vector with independent variables x_i to the dependent variable y_i. The noise is normally distributed with standard deviation σ along the dimension of the dependent variable.

In a 2D point cloud the point p is represented by (x_p, y_p). The points are mapped into an intercept-slope representation through defining $X_i = [1 \;\; x_p]$ and $y_i = y_p$. The vector β will then contain the y-intercept as the first value, the slope as the second value.

All observations that belong to the same single line lead to a likelihood function that corresponds to a normally distributed random variable with y and X as parameters:

$$p(y \mid X, \beta, \sigma^2) \propto \sigma^{-n} \exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^T(y - X\beta)\right). \tag{3.11}$$

The dependent variable is now a column vector of values y and each observation has a row of independent variables in X. The vector β and the standard deviation σ are shared across all observations. The term $y - X\beta$ is written out like this:

$$\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} - \begin{pmatrix} 1 & x_1 \\ 1 & x_2 \\ \vdots & \vdots \\ 1 & x_n \end{pmatrix} \begin{pmatrix} \beta_0 \\ \beta_1 \end{pmatrix}. \tag{3.12}$$

Note that Eq. 3.11 has exactly the same form for a single point or for multiple points that belong to the same line. Hence, we have the probability of a point w_i given the line parameters $\theta_k = (\beta_k, \sigma_k)$:

$$F(w_i, \theta_k) = p(w_i \mid \theta_k) = p(w_i \mid \beta_k, \sigma_k^2) = p(y_i \mid X_i, \beta_k, \sigma_k^2). \tag{3.13}$$

To get the full distribution $p(w_i, \beta, \sigma^2)$ we will need also $p(\beta, \sigma^2)$.

3.1.3 Conjugate Prior for a Line

The conjugate prior for the likelihood in Eq. 3.11 is a product of a prior for the standard deviation $p(\sigma)$ and the conditional probability of the line coefficients given the standard deviation $p(\beta \mid \sigma^2)$.

$$p(\sigma^2, \beta) = p(\sigma^2) p(\beta \mid \sigma^2). \tag{3.14}$$

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

$$p(\sigma) \propto (\sigma^2)^{-(\nu_0/2+1)} \exp(-\frac{1}{2\sigma^2} \nu_0 s_0^2). \tag{3.15}$$

This is an $IG(a_0, b_0)$ with $a_0 = \nu_0/2$ and $b_0 = 1/2 \nu_0 s_0^2$. The conditional with respect to the line coefficients has a normal distribution as prior:

$$p(\beta \mid \sigma^2) \propto \sigma^{-n} \exp\left(-\frac{1}{2\sigma^2}(\beta - \mu_0)^T \Lambda_0 (\beta - \mu_0)\right). \tag{3.16}$$

Let us collect $\Lambda_0, \mu_0, a_0, b_0$ into λ_0, we have now a description of our base distribution H:

$$H(\theta_k) = NIG(\theta_k; \lambda_0). \tag{3.17}$$

The Normal-Inverse-Gamma (NIG) is a distribution that combines a Normal and an Inverse Gamma distribution. The line coefficients are sampled from a Normal distribution and the standard deviation is sampled from the Gamma distribution with a_0 and b_0 as hyperparameters.

$$\begin{aligned} \sigma_k = \tau_k^{-1/2} \qquad \tau_k &\sim \mathcal{G}(a_0, b_0), \\ \mu_k &\sim \mathcal{N}(\mu_0, \sigma^2 \Lambda_0^{-1}). \end{aligned} \tag{3.18}$$

3.1.4 Posterior Predictive for a Line given Data

Due to the fact that the NIG is a conjugate prior with respect to the normal distribution (with unknown mean and variance), we have a simplified description for updating the hyperparameters, given a set of observations. The hyperparameters are updated[1] according to (c.f. Denison, 2002):

$$\begin{aligned} \Lambda_n &= \Lambda_0 + X^T X, \\ \mu_n &= \Lambda_n^{-1}(\Lambda_0 \mu_0 + X^T y), \\ a_n &= a_0 + n/2, \\ b_n &= b_0 + 1/2(y^T y + \mu_0^T \Lambda_0 \mu_0 - \mu_n^T \Lambda_n \mu_n). \end{aligned} \tag{3.19}$$

Let us collect $\Lambda_0, \mu_0, a_0, b_0$ into λ and $\Lambda_n, \mu_n, a_n, b_n$ into λ^*. Let us collect a set of our observations and $(X, y)_k$ into w_k. The update for the hyperparameters can then be summarized as:

$$\lambda^* = U_{up}(\lambda, w_k). \tag{3.20}$$

Removing observations does lead to similar updates ("downdates"') for the hyperparameters:

$$\begin{aligned} \Lambda_n &= \Lambda_0 - X^T X, \\ \mu_n &= \Lambda_n^{-1}(\Lambda_0 \mu_0 - X^T y), \\ a_n &= a_0 - n/2, \\ b_n &= b_0 - 1/2(y^T y + \mu_n^T \Lambda_n \mu_n - \mu_0^T \Lambda_0 \mu_0). \end{aligned} \tag{3.21}$$

The downdate for the hyperparameters can then be summarized as:

$$\lambda^* = U_{down}(\lambda, w_k). \tag{3.22}$$

[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.

If we combine this update with sampling θ_k from λ_n according to Eq. 3.17, then we obtain:

$$p(\theta_k \mid \lambda_0, w_k) \propto F(w_k, \theta_k)H(\theta_k; \lambda_0) = p(\theta_k \mid \lambda_n) = NIG(\theta_k; \lambda_n). \tag{3.23}$$

Sampling of $NIG(\theta_k; \lambda_n)$ is as in Eq. 3.18, but with λ_n rather than λ_0.

$$\begin{aligned} \sigma_k = \tau_k^{-1/2} \qquad \tau_k &\sim \mathcal{G}(a_n, b_n), \\ \mu_k &\sim \mathcal{N}(\mu_n, \sigma^2 \Lambda_n^{-1}). \end{aligned} \tag{3.24}$$

Let us integrate over θ (through the function H):

$$Q(w_k, \lambda_0) = \int_\Theta F(w_k, \theta) dH(\theta; \lambda_0). \tag{3.25}$$

3.2 Inference for the Infinite Line Model

The posterior predictive for parameters (see Eq. 3.9) combined with observations w_i is described by:

$$p(\theta_i \mid \theta_{-i}, w_i) \propto r_i H_i(\theta_i) + \sum_{j \neq i} L_{i,j} \delta(\theta_j - \theta_i). \tag{3.26}$$

Eq. 3.26 defines the posterior as proportional (indicated by $\propto$) to three terms. First, the α-weighted posterior r_i for a new *cluster*. Second, the posterior with respect to the prior distribution over the parameter values $H_i(\theta_i)$ given observation w_i. Third, a term that sums over the likelihood of w_i given existing line θ_j, indicated by $L_{i,j}$.

The α-weighted posterior r_i defines the probability that a new cluster will be sampled:

$$r_i = \alpha Q(w_i, \lambda_0) = \alpha \int_\Theta F(w_i, \theta) dH(\theta). \tag{3.27}$$

The posterior $H_i(\theta)$ is the normalized product of the prior distribution $H(\theta_i)$ with the likelihood $F(w_i, \theta_i)$ for a single observation w_i.

$$H_i(\theta_i) \propto H(\theta_i) F(w_i, \theta_i). \tag{3.28}$$

The likelihood $F(w_i, \theta_i)$ we will denote $L_{i,i}$ or, in general, for obervation w_i and line θ_j:

$$L_{i,j} = F(w_i, \theta_j). \tag{3.29}$$

Sampling a new cluster parameter from $H_i(\theta_i)$ is done with probability:

$$p(\theta_{new}) = \frac{r_i}{r_i + \sum_{j \neq i} L_{i,j}}. \tag{3.30}$$

We can use these equations to derive the parameters θ_i with Gibbs sampling.

Algorithm 8 Gibbs sampling over parameters θ_i

```
1:  procedure GIBBS ALGORITHM 1($w, \lambda_0, \alpha$)    ▷ Accepts points $w$, hyperparameters $\lambda_0, \alpha$ and returns $k$ line coordinates
2:      for all $t = 1 : T$ do
3:          for all $i = 1 : N$ do
4:              $r_i = \alpha Q(w_i, \lambda_0)$    ▷ Posterior predictive of $w_i$ given hyper parameters (Eq. 3.27)
5:              for all $j = 1 : N, j \neq i$ do
6:                  $L_{i,j} = F(w_i, \theta_j)$    ▷ Likelihood for a line given observation (Eq. 3.29)
7:              end for
8:              $p(\theta_{new}) = \frac{r_i}{r_i + \sum_{j \neq i} L_{i,j}}$    ▷ Probability of sampling a new parameter (Eq. 3.30)
9:              $u \sim U(0, 1)$
10:             if $p(\theta_{new}) > u$ then    ▷ Sample with probability $p(\theta_{new})$
11:                 $\lambda_n = U_{up}(w_i, \lambda_0)$    ▷ Update hyperparameters with $w_i$ (Eq. 3.20)
12:                 $\theta_i \sim NIG(\theta_i; \lambda_n)$    ▷ Sample $\theta_i$ from NIG (Eq. 3.24)
13:             else
14:                 $\theta_i$ sampled from existing clusters    ▷ Sample old cluster
15:             end if
16:         end for
17:     end for
18:     return summary on $\theta_k$ for $k$ lines
19: end procedure
```

This Gibbs algorithm is earlier described in its general form (see algorithm 1 in Neal, 2000). As shown in Algorithm 8 we perform a loop in which for T iterations each 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 $L_{i,j}$ 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.

3.3 Accelerating Inference for the Infinite Line Model

Gibbs sampling of this model might be accelerated. In Figure 3.2 we use plate notation to show the stick-breaking representation of 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 3.31.

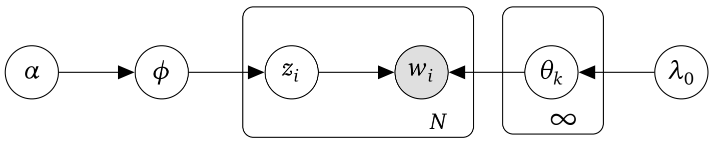


Figure 3.2: The infinite line model in the stick-breaking representation (compare with Figure 3.1). From left to right: α, the concentration parameter of the Dirichlet process; $(\phi_1, \ldots, \phi_k)$, the partition of points over lines; z_i, the assignment parameters that link observation w_i with line k; w_i, the observation, an individual point with x and y coordinates; θ_k, the parameters of line k; λ_0, the base measure from which the line parameter values are sampled.

Algorithm 9 Gibbs sampling over clusters c_k

```
 1: procedure GIBBS ALGORITHM 2($w, \lambda_0, \alpha$)    ▷ Accepts points $w$ and hyperparameters $\lambda_0$ and $\alpha$,
    returns $k$ line coordinates
 2:     for all $t = 1 : T$ do
 3:         for all $i = 1 : N$ do
 4:             $c$ = cluster($w_i$)    ▷ Get cluster $c$ currently assigned to observation $w_i$
 5:             $\lambda_c = U_{down}(w_i, \lambda_c)$    ▷ Adjust cluster hyperparameters on removing $w_i$ (Eq. 3.22)
 6:             $m_c = m_c - 1$    ▷ Adjust cluster size $m_c$
 7:             for all $k = 1 : K$ do
 8:                 $L_k = m_k \mathrm{F}(w_i, \theta_k)$    ▷ Likelihood for cluster $k$ given $w_i$ (Eq. 3.33)
 9:             end for
10:             $r_i = \mathrm{Q}(w_i, \lambda_0)$    ▷ Posterior predictive of $w_i$ given hyper parameters
11:             $p(new) = \frac{r_i}{r_i + \sum_k L_k}$    ▷ Sample new or old?
12:             if $p(new)$ then
13:                 $\lambda_k = U_{up}(w_i, \lambda_0)$    ▷ Update hyperparameters with observation $w_i$
14:                 $\theta_i \sim NIG(\lambda)$    ▷ Sample $\theta_i$ from NIG
15:             else
16:                 $k$ sampled from existing clusters
17:                 $\lambda_k = U_{up}(w_i, \lambda_k)$    ▷ Restore hyperparameters with observation $w_i$
18:             end if
19:             $m_k = m_k + 1$    ▷ Increment cluster size $m_k$
20:         end for
21:         for all $k = 1 : K$ do
22:             $\theta_k \sim NIG(\lambda_k)$    ▷ Sample $\theta_k$ from NIG
23:         end for
24:     end for
25:     return summary on $\theta_k$ for $k$ lines
26: end procedure
```

$$p(c_i = c \text{ and } c_i = c_j \text{ and } i \neq j \mid c_{-i}, w_i, \alpha, \theta) \propto \frac{n_{c,-i}}{\alpha + n - 1} F(w_i \mid \theta_i). \tag{3.31}$$

The probability to sample a new cluster only depends on α and the total number of data items. This is formally described in equation 3.32.

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

Here $\Omega(c)$ denotes all admitted values for c_i. The importance of conjugacy is obvious from Eq. 3.32, it will lead to an analytic form of the integral. The inference method using Eqs. 3.31

and 3.32 is described in Section 3.1.

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:

$$L_k = m_k F(w_i, \theta_k). \tag{3.33}$$

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 "downdate" the corresponding hyperparameters (as previously mentioned). 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.

3.4 Performance of the Infinite Line Model

The infinite line model (see Section 3.1) 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 earlier, 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 3.4.1 we discuss the clustering performance and in Section 3.4.2 we provide two clustering examples.

3.4.1 Clustering Performance

The results of the clustering algorithms are measured using conventional metrics. For instance, we may use the Rand Index. It describes the accuracy of cluster assignments (Rand, 1971) by:

$$R = \frac{a+b}{a+b+c+d}. \tag{3.34}$$

Here a counts the pair of points that belong to the same cluster, both at ground truth as well as after the inference procedure. Likewise b numbers the pair of points that belong to different clusters in both sets. The values c and d describe discrepancies between the ground truth and the results after inference. A Rand Index of one means that there have been no mistakes.

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 3.3 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!

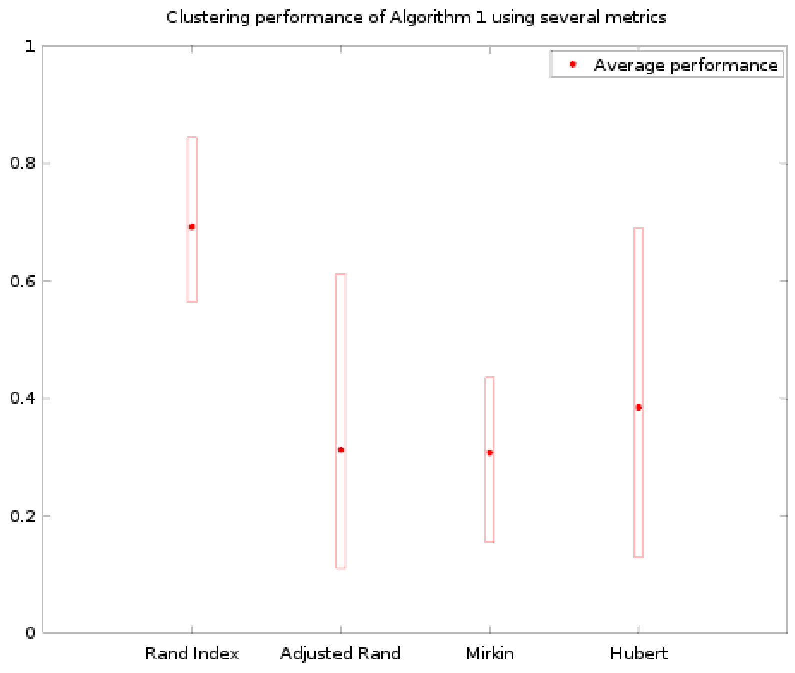


Figure 3.3: The performance of Algorithm 8 with respect to clustering is measured using the Rand Index, the Adjusted Rand Index, the Mirvin metric, and the Hubert metric. A score of 1 means perfect clustering for all metrics, except Mirvin's where 0 denotes perfect clustering.

Algorithm 9 leads to stellar performance measures (Figure 3.4). 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 performance levels. A line seems to form local regions of high probability, making it difficult for points to postulate slightly changed line coordinates.

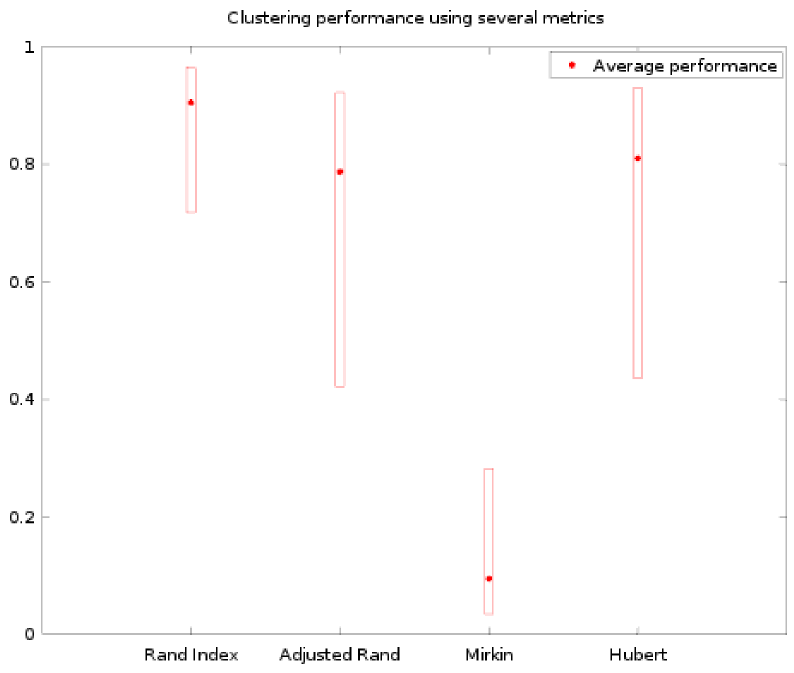


Figure 3.4: The performance of Algorithm 9 with respect to clustering is measured using the Rand Index, the Adjusted Rand Index, the Mirvin metric, and the Hubert metric. A score of 1 means perfect clustering for all metrics, except Mirvin's where 0 denotes perfect clustering.

3.4.2 Two Examples

In the following we show two examples to understand the inference process better. The first example in seen in Figure 3.5. It shows the assignment after a single Gibbs step in Algorithm 8. 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.

The second example in Figure 3.6 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.

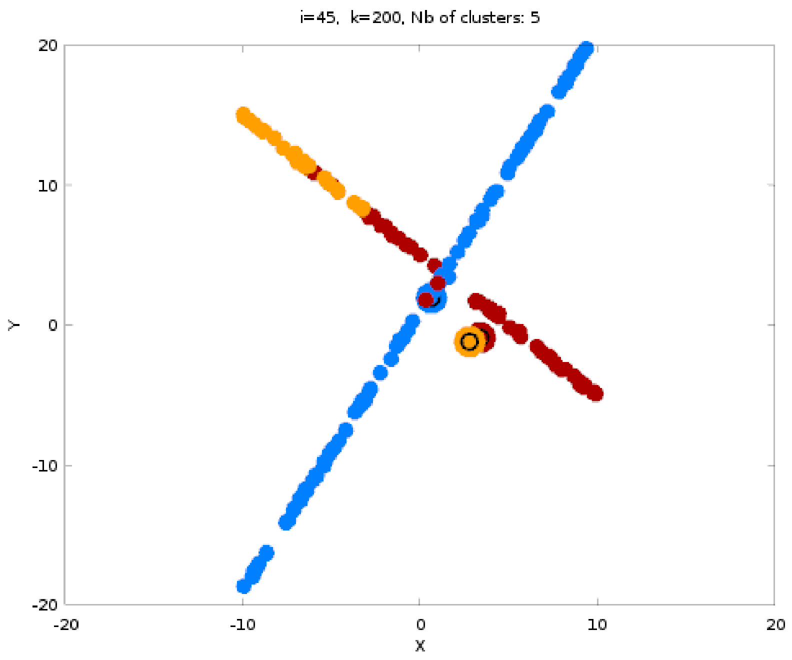


Figure 3.5: One of the Gibbs steps in the inference of two particular lines. The points are roughly distributed according to the lines, but one line exists out of two large clusters. The line coordinates are visualized by a double circle. The x-coordinate is the y-intercept of the line, the y-coordinate is the slope.

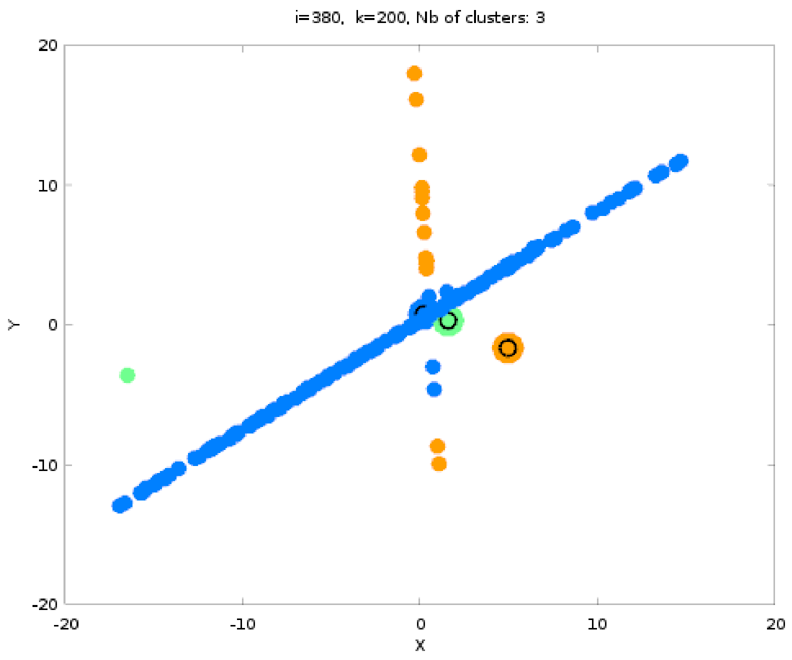


Figure 3.6: The assignment of a line to a single point. In the figure, there are three clusters found, rather than only the two obvious clusters.

1441 3.5 Chapter Conclusions

The infinite line model that is proposed 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 3.4 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., 2016b,a). The Bayesian model is solved through two types of algorithms. Algorithm 8 iterates over all observations and suffers from slow mixing. The individual updates makes it hard to reassign 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. Note, that even optimal inference may occasionally result in misclassifications. The dataset is generated by a random process. Hence, occassionally two lines are generated with almost the same slope and intercept. Points on these lines are impossible to assign to the proper line.

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 quite a trivial 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 limit the lines of infinite length, to line segments. To restrict points on the lines to a uniform distribution of points over a line segment, a symmetric Pareto distribution can be used as prior (see next Chapter). This would subsequently allow for a hierarchical model in which these end points 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 The nonparametric Bayesian model for line estimation (Chapter 3) does not take into account lines that are of finite length. In this chapter, we introduce a Bayesian method to perform inference over such line segments. In this model our prior for the length of the line segment is a symmetric 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 The model is using both a Normal-Inverse-Gamma distribution and a Pareto distribution as priors for an individual line segment (Section 4.1.1). The parameters for the line segments are generated through a Dirichlet process (Section 4.2). The generative Dirichlet process is used to perform inference using Gibbs sampling over auxiliary variables (Section 4.3). The results for inference over line segments are compared with those for lines (Section 4.4). Finally, weak aspects of the current MCMC method are established (Section 4.5). They will form the basis for new inference methods in the next chapters.

4.1 Pareto Pairs

Lines in a two-dimensional space are mathematical objects that can be described by *two* parameters. To restrict a line to a line segment, a total of *four* parameters are required. Two parametrizations will then come to mind. First, a center-point parametrization, in which parameters describe the center of a line segment, the slope of the line through the center, and the size of the line segment. Second, an endpoint parametrization, in which parameters describe the locations of the two endpoints. The two parametrizations are *equivalent*, but generalizations can either be intuitive or cumbersome. The reason is that the generalization to a line segment from a two-dimensional space to a three-dimensional space, requires the endpoints to be positions in a 3D space. Here we see that the center-point parametrization would require a nonintuitive description of the angles in particular directions. In contrast, the generalization to squares and rectangles or shapes with many endpoints, might benefit from the center-point parametrization.

As far as we know there is no statistical description of data points distributed over a line segment that has a conjugate prior form. A line segment itself, however, has a conjugate form! Assume that we have a prior for the location of endpoints on the x-axis. Given the data, we then update the location of the endpoints. By disregarding the distribution of the data points over the segment, we can update the location of the endpoints by a conjugate Bayesian construction.

4.1.1 Pareto Prior

Assume that the data is distributed according to a symmetric uniform distribution. Hence, the likelihood is given by:

$$p(x \mid a) \sim \mathscr{U}(-a, a) = \begin{cases} \frac{1}{2a} & \text{for } x \leq |a| \\ 0 & \text{otherwise} \end{cases}. \tag{4.1}$$

Here the uniform distribution is centered around 0 and extends with size a in both directions. It is possible to shift the entire distribution over a distance b. For now, let us continue with one endpoint at a and one endpoint at $-a$.

A prior for the (endpoints of a) symmetric uniform distribution is a symmetric Pareto distribution:

$$p(a) \sim \mathscr{P}_s(\lambda, k) = \begin{cases} \frac{1}{2}k\lambda^k |a|^{-k-1} & |a| \geq \lambda \\ 0 & \text{otherwise} \end{cases}. \tag{4.2}$$

The factor $\frac{1}{2}$ stems from the fact that the symmetric Pareto distribution is now mirrored across the y-axis. Hence, the probability density is half of that of the normal Pareto distribution for the positive x-axis.

If we would just sample from a symmetric Pareto distribution, we can sample multiple times from the positive x-axis. To actually sample endpoints of segments we have to sample pairs of points.

$$p(a,b) \sim \mathscr{P}_p(\lambda_m, \lambda_n, k) \tag{4.3}$$

This process can be described in two steps. First, we sample a and b from a categorical distribution to decide which one will be the left endpoint and which one the right endpoint. Second, we sample the right endpoint from a normal Pareto distribution and the left endpoint from a mirrored Pareto distribution.

The sampling of Pareto pairs is visualized in Fig. 4.1.

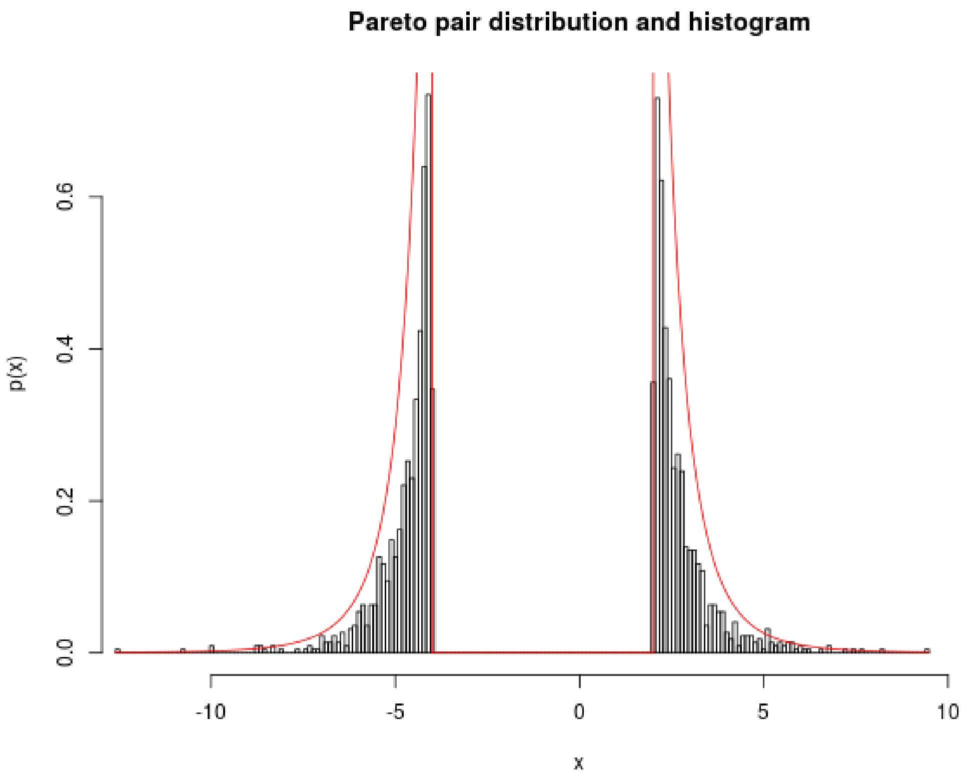


Figure 4.1: Sampling of Pareto pairs. The parameters are $\lambda_m = -4$, $\lambda_n = 2$, $k = 5$, and we have sampled $N = 1000$ pairs. The position parameters λ_m and λ_n define the positions of the endpoints. The shape parameter k defines the variance in the exact positions on both sides. The Pareto pairs always sample endpoints both at the "left" and the "right" (not both at the left or the right side).

4.1.2 Posterior for a Pareto pair

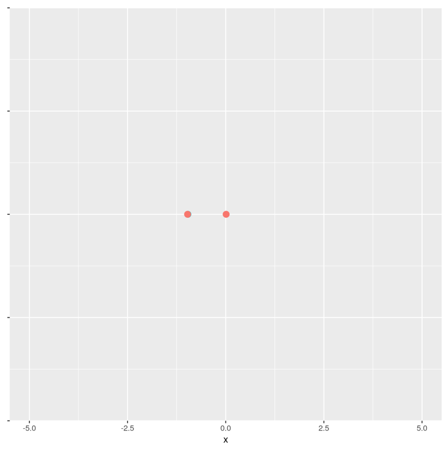
The Pareto distribution is a conjugate prior for the uniform distribution, with updated hyperparameters:

$$p(a \mid D) = \mathscr{P}(c, N + k). \tag{4.4}$$

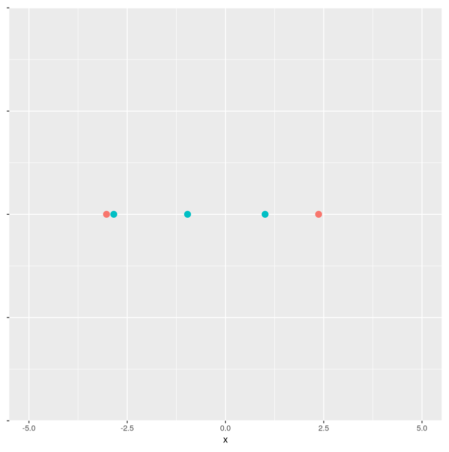
The data is denoted by $D = \{x_0, \dots, x_{N-1}\}$, the parameter k is adjusted with the number of data points N, and the parameter c is the maximum of $\{m, \lambda\}$ with m the maximum value in D.

The posterior for a Pareto pair can be found by sampling in parallel for the endpoint at the "right" and the one at the "left". The endpoint at the right is sampled from a Pareto distribution $\mathscr{P}(c_n, N + k_n)$ with (1) c_n the maximum of the data points D and λ_n, (2) N the number of Pareto pairs, and (3) k_n the scale hyperparameter. The endpoint at the left is sampled fom a Pareto distribution $\mathscr{P}(c_m, N + k_m)$ with (1) c_m the minimum of the data points D and λ_m, (2) N the number of Pareto pairs, and k_m the scale hyperparameter.

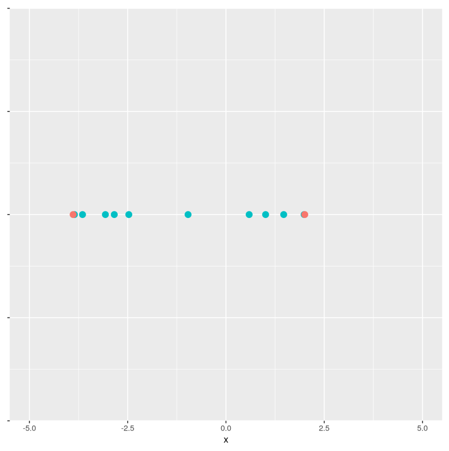
If $k_n \neq -k_m$ the distribution is shifted such that $k'_n = -k'_m$. This makes the form of the probability distribution symmetric with respect to the y axis. In the end, the results are shifted back. This transformation makes sense for pairs of points. We do not want the two scale parameters of the Pareto distribution to influence the symmetry (the shape) of the overall distribution.



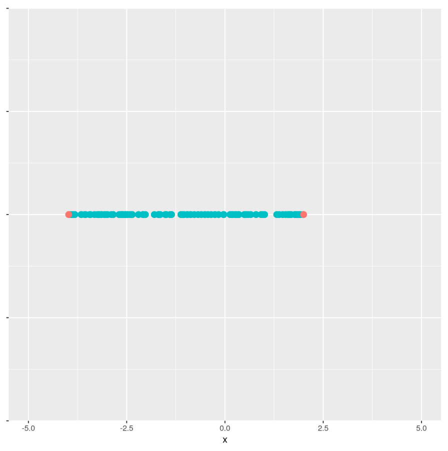
(a) Posterior after 1 data point. The endpoints are still close to zero.



(b) Posterior after 3 data points. The right endpoint is sampled quite far from the data points.



(c) Posterior after 10 data points. The endpoints get sampled closer to the extreme data points.



(d) Posterior after 100 data points. The endpoints are sampled really close to the extrema of the line segment.

Figure 4.2: Consider (1) the data uniformly distributed on a line segment and (2) a symmetric Pareto prior for both endpoints, then we can update the estimate for the endpoints given the data as visualized. Each subfigure shows an adjustment of the endpoints given more data points (1, 3, 10, and 100 data points). The y-axis does not have a significance in these plots.

Sampling from the Pareto distribution is through inverse transform sampling. By sampling from $U(0,1)$ with 1 included, we transform according to $k/U^{1/a}$.

Figure 4.2 shows how the endpoints are updated given the data. An uninformative prior is used. In this case the hyperparameters k_n and k_m are set close to 0, thus the data will wash out the prior immediately. Naturally, it is possible to define large k_n and k_m. In that case it should be noted that the data will never be able to "correct for" this prior: if the true length of the segment is smaller than $|k_n - k_m|$ this will not be inferred properly. Here we note also that the maximum and minimum operators are quite sensitive to outliers as well.

4.2 Generative Process to Create a Line Segment

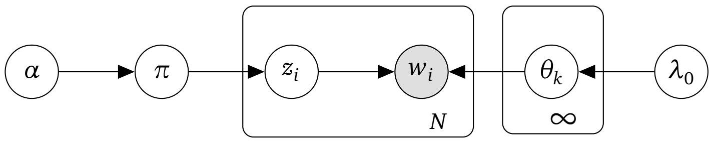


Figure 4.3: The Bayesian linear regression model for multiple line segments in plate notation is the same as for the Infinite Line Model. The Dirichlet process is defined at the left with concentration parameter α. It generates the partitions $(\pi_1, \ldots, \pi_k)$ with assignment parameters z_i that denote which observation w_i belongs to which cluster k. The cluster is summarized through the parameter set θ_k and has λ_0 as its hyperparameter. The parameter set θ_k includes parameters that signify the line itself such as slope and y-intercept, plus the parameters that denote the extent of the segment.

To be able to perform inference over a line segment in a two-dimensional space, we will have to map somehow these points to a one-dimensional space (see Figure 4.2).

In the case of a *line* we can sample θ_i from a Normal-Inverse-Gamma distribution with hyperparameter λ_{temp}. The update for the hyperparameter we obtain in closed form given observations (described in the previous chapter, Sections 3.1.3 and 3.1.4). However, in the case of a line *segment* there is no known conjugate prior available. Here we will introduce a non-conjugate model to describe line segments. It uses a Dirichlet process prior for the assignment of points unto multiple segments (Section 4.2.1). It uses a uniform distribution (Section 4.2.2) to describe the distribution of points across an individual line segment.

4.2.1 Dirichlet Process Prior

Let us reiterate the Dirichlet process for our nonparametric line segment model:

$$\begin{aligned} G &\sim DP(\alpha, H), \\ \theta_i \mid G &\sim G, \\ w_i \mid \theta_i &\sim F(w_i, \theta_i). \end{aligned} \tag{4.5}$$

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 observations on lines.

4.2.2 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 will again use an intercept-slope representation through defining $w_i = (X_i, y_i)$ with $X_i = [1, x_i]$. The column vector $\beta = [\beta_0, \beta_1]$ contains two parameters: the y-intercept β_0 and the slope parameter β_1 (compare Section 3.1.2).

We assume a normally distributed random variable across $y - X\beta$, the same as in the line model (Eq. 3.11):

$$p(y \mid X, \beta, \sigma^2) \propto \sigma^{-n} \exp\left(-\frac{1}{2\sigma^2}(y - X\beta)^T(y - X\beta)\right). \tag{4.6}$$

However, in contrast to the line model we restrict the variable $X_i = [1 \;\; x_i]$ to a subset of all possible values. The data points on the segment are to be generated from a uniform distribution:

$$\begin{aligned} F(y_i, X_i | \beta_i, \sigma_i, a_i, b_i) &= \mathcal{N}(y_i - X_i\beta_i, \sigma_i), \\ X_i &= [1 \;\; x_i], \\ x_i &\sim \mathcal{U}(a_i, b_i). \end{aligned} \tag{4.7}$$

Fig. 4.4 displays the adjustment with points generated uniformly over the line segment.

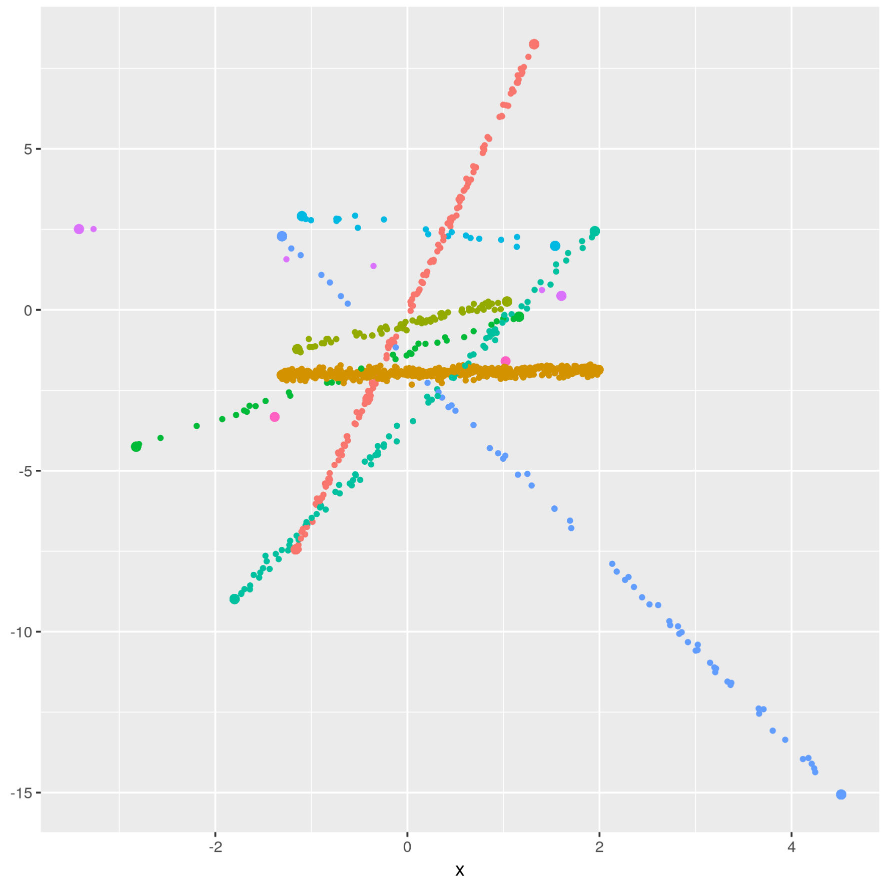


Figure 4.4: Line segments generated through a Dirichlet Process. The line parameters are generated from a Normal-Inverse-Gamma distribution. The endpoints of the line segment are generated from a symmetic Pareto distribution. The points are generated uniformly over the line segments.

Finally, we remark that the description in Eq. 4.7 is not conjugate. This means that inference over line segments requires more complicated sampling strategies than the inference methods in the previous chapter that require conjugacy.

4.3 Inference over a Line Segment

As a sequel to the last remark of section Section 4.2 we are aware that the line segment model is not conjugate. This therefore requires a sampling algorithm that does not make use of conjugacy. One algorithm that does not assume conjuacy is earlier described in its general form by Neal (2000) and is called *Gibbs sampling over auxiliary variables*. The sampling process proposes m new values for the parameters from the hyperparameters. The m 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/m.

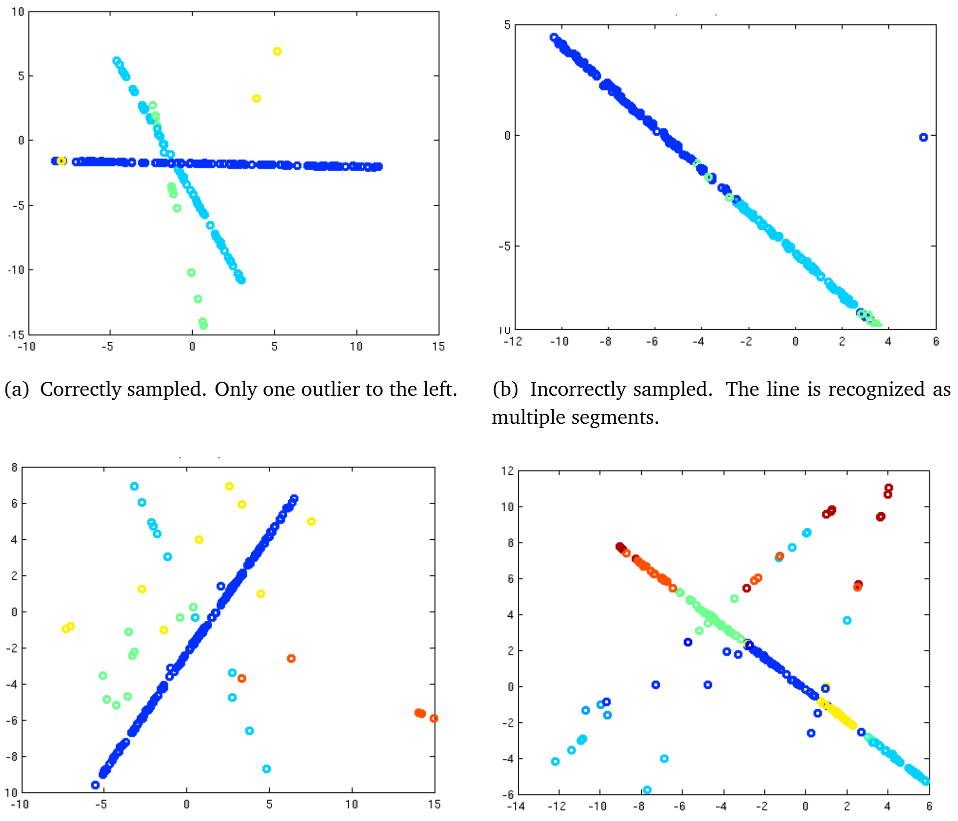
After every data item is assigned a cluster, the cluster parameters themselves are updated given the assigned data items. In a conjugate model the sufficient statistics (Definition 2.28) can be updated at once, given such observations. In a nonconjugate model we will need to update θ_j by sampling from $p(\theta_j \mid y)$.

Algorithm 10 Gibbs sampling over auxiliary variables (a θ_i)

```
1:  procedure GIBBS ALGORITHM WITH AUXILIARY VARIABLES($w, \lambda_0, \alpha$)  ▷ Accepts points $w$,
    hyperparameters $\lambda_0, \alpha$, number of auxiliary variables $m$, and returns $k$ line coordinates.
2:      for all $t = 1 : T$ do
3:          for all $i = 1 : N$ do
4:              for all $j = 1 : m$ do
5:                  $\theta_j \sim NIG(\lambda_0)$                         ▷ Sample $\theta_j$ from NIG.
6:              end for
7:              for all $j = 1 : K + m, j \neq i$ do
8:                  $L_j = \text{likelihood}(w_i, \theta_j)$   ▷ Update likelihood for all $\theta$ except $\theta_i$ given $w_i$.
9:              end for
10:             $P_{-i=1:K} = b \sum_{-i} L_{-i}$              ▷ Calculate probability of existing cluster.
11:             $P_{-i=K:K+m} = b\alpha/mL_m L_{-i}$                ▷ Calculate probability of new cluster.
12:             $\theta_i = \theta_j$ according to above $P_{-i}$          ▷ Sample $\theta_i$ accord. to above prob.
13:             Remove unused clusters.
14:         end for
15:         for all $j = 1 : K$ do
16:             $\theta_j \sim p(\theta_j \mid y)$                                    ▷ Update $\theta_j$.
17:         end for
18:     end for
19:     return summary on $\theta_k$ for $k$ line segments.
20: end procedure
```

4.4 Results

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) Correctly sampled. Only one outlier to the left.

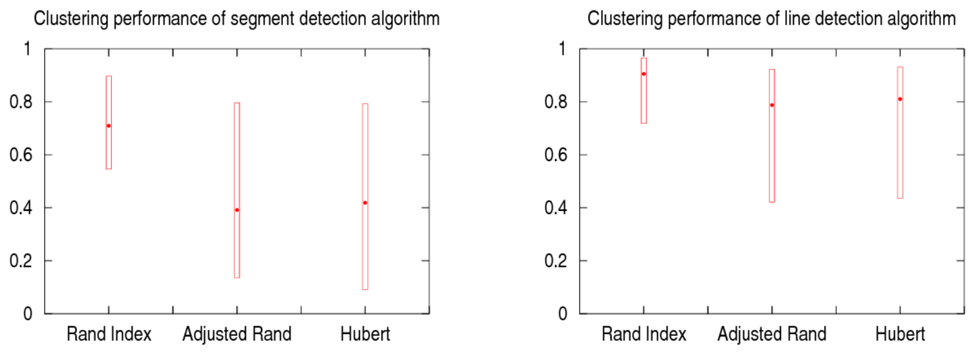
(b) Incorrectly sampled. The line is recognized as multiple segments.

(c) More or less correct. The segments with fewer observations are recognized poorly.

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

Figure 4.5: Bayesian point estimates of the sampling process with varying outcomes.

The results over a larger dataset can be measured with clustering metrics as visualized in Figure 4.6. 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 4.6) compared to line detection (without constraints on segment size).



(a) Segment detection. (b) Line detection.

Figure 4.6: Segment detection performs much 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.

4.5 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. The auxiliary variable Gibbs sampling method converges faster than the ordinary Metropolis-Hastings algorithm by postulating multiple segments rather than only one.

However, the segment estimation problem remains a challenge for the current inference methods. 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 give the chapter conclusions and we 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 clusers. Split-merge samplers are samplers that take into account such structure as already described in Section 2.3.6. Rather than moving data points one by one from an old to a 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.

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 2.3.6).

In the hierarchical models of lines and segments we have used a Dirichlet proces 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 stepsi, 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 assigment of observations y_i unto parameters θ_i.

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

```
 1: 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 $c'$.
 2:     $i \sim U(1,N)$    ▷ Sample $i$ random uniformly over cluster assignments.
 3:     $j \sim U(1,N) \cap i$    ▷ Sample $j$ also random uniformly, but with $j \neq i$.
 4:     $S_R = \{c_i, c_j\}$    ▷ Sampled clusters $c_i, c_j$.
 5:     $S_I = \{c_x\}$ with $c_x \in S_R$ for $x \in \{1,\dots,N\}$    ▷ All data in clusters $c_i, c_j$.
 6:     $S_E = S \cap S_R$    ▷ All data in clusters $c_i, c_j$ excluding $S_R$.
 7:     $N_S = \text{unique}(S_R)$
 8:     if $N_S = 1$ then    ▷ Case: $i, j$ belong to the same cluster.
 9:         $c_i^{(2)} = c_k$ with $c_k \notin \{c_1,\dots,c_N\}$    ▷ Sample new cluster for $c_i^{(2)}$.
10:         $c_j^{(2)} = c_j^{(1)}$    ▷ Keep $c_j$ the same.
11:         $c_e^{(2)} = \text{SPLITPROCEDURE}(S_E, c_i^{(2)}, c_j^{(2)})$    ▷ After $c_i^{(2)}, c_j^{(2)}$ assign $S_E$.
12:         for all $m \notin S_I$ do
13:             $c_m^{(2)} = c_m^{(1)}$    ▷ Data points in clusters other than $c_i, c_j$ are not adjusted.
14:         end for
15:         $c' = \{c_i^{(2)}, c_j^{(2)}, c_e^{(2)}, c_m^{(2)}\}$
16:         $a = a_{split}(c',c)$ according to Eq. ??    ▷ MH acceptance for a split.
17:     else    ▷ Case: $i, j$ belong to different clusters $c_i \neq c_j$ $(N_S = 2)$.
18:         for all $q \in S_I$ do
19:             $c_q^{(1)} = c_j^{(2)}$    ▷ Assign all data points in $c_i$ and $c_j$ to $c_j$.
20:         end for
21:         for all $m \notin S_I$ do
22:             $c_m^{(1)} = c_m^{(2)}$    ▷ Data points in clusters other than $c_i, c_j$ are not adjusted.
23:         end for
24:         $c' = \{c_q^{(1)}, c_m^{(1)}\}$
25:         $a = a_{merge}(c',c)$ according to Eq. ??    ▷ MH acceptance for a merge.
26:     end if
27:     $u \sim U(0,1)$    ▷ Sample $u$ between 0 or 1 uniformly.
28:     if $a < u$ then
29:         $c' = c$    ▷ Reject $c'$ by setting it to $c$
30:     end if
31:     return $c'$, the (updated) cluster assignment vector: $c \to c'$.
32: 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.

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

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
```

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).

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).

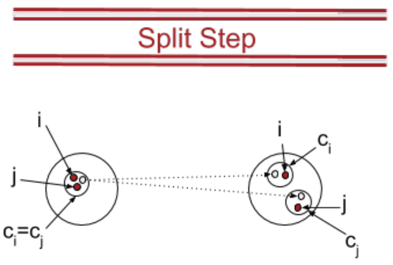


Figure 5.1: A split step. Points i and j are already assigned to separate clusters c_i and c_j. Each next point is assigned to one of the clusters with probability $\frac{1}{2}$, indicated by dotted arrows.

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 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 identies 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.

Algorithm 13 Sequentially Allocated Merge-Split

```
1: 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 $c'_m$.
2:     $T = \text{random\_shuffle}(S)$
3:     for all $m \in T$ do
4:         $p(c_m = c_0|c_0, c_1, \theta_{c_0}, \theta_{c_1}) = \frac{N_0 p(y_k|\theta_0)}{N_0 p(y_k|\theta_0) + N_1 p(y_k|\theta_1)}$
5:         $p(c_m = c_1|c_0, c_1, \theta_{c_0}, \theta_{c_1}) = 1 - p(c_m = c_0|c_0, c_1, \theta_{c_0}, \theta_{c_1})$
6:         $c'_m \sim p(c_m|c_0, c_1, \theta_{c_0}, \theta_{c_1})$
7:     end for
8:     return $c'_m$, the cluster assignment for $S$.
9: 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.

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

5.3 Triadic split-merge sampler

The triadic split-merge sampler uses up to three clusters for a split or merge step (Fig. 5.2).

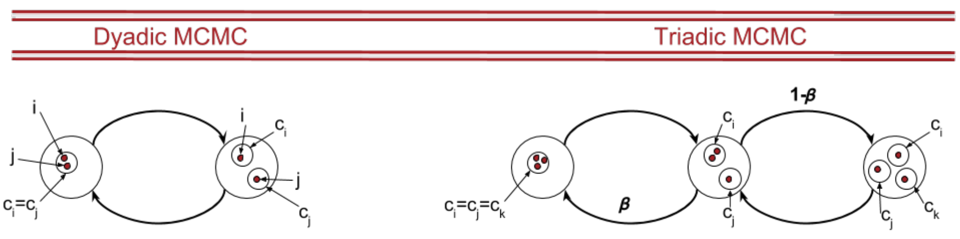


Figure 5.2: Right: dyadic MCMC picks two data items i, j random uniformly. If both are in the same cluster a split towards two clusters is attempted. If both are in distinct clusters a merge towards one cluster is attempted. Left: triadic MCMC picks three data items i, j, k random uniformly. If all three 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 immediate 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.

Algorithm 14 Triadic split-merge sampler

```
 1: 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 $c'$.
 2:     $i \sim U(1,N)$    ▷ Sample $i$ random uniformly over cluster assignments.
 3:     $j \sim U(1,N) \cap i$    ▷ Sample $j$ also random uniformly, but with $j \neq i$.
 4:     $k \sim U(1,N) \cap \{i,j\}$    ▷ Sample $k$ random uniformly, but with $k \neq j$, $k \neq i$.
 5:     $S_R = \{c_i, c_j, c_k\}$    ▷ Sampled clusters $c_i, c_j, c_k$.
 6:     $S_I = \{c_x\}$ with $c_x \in S_R$ for $x \in \{1,\ldots,N\}$    ▷ All data in clusters $c_i, c_j, c_k$.
 7:     $S_E = S_I \cap S_R$    ▷ All data in clusters $c_i, c_j, c_k$ excluding $S_R$.
 8:     $N_S = \text{unique}(S_R)$
 9:     $u \sim U(0,1)$    ▷ Sample $u$ between 0 or 1 uniformly.
10:     if $N_S = 1$ then    ▷ Case: $i, j, k$ belong to the same cluster.
11:         return $c' =$ DYADIC SPLIT-MERGE SAMPER($c$)
12:     else if $N_S = 2$ and $u < \beta$ then    ▷ Case: a cluster with one item and one with two items and $u < \beta$.
13:         return $c' =$ DYADIC SPLIT-MERGE SAMPER($c$)
14:     else if $N_S = 2$ and $u \geq \beta$ then    ▷ Case: a cluster with one item and one with two items and $u \geq \beta$.
15:         $c_i^{(3)} = c_k$ with $c_k \notin \{c_1,\ldots,c_N\}$    ▷ Sample new cluster for $c_i^{(3)}$.
16:         $c_j^{(3)} = c_j^{(2)}$    ▷ Keep $c_j$ the same.
17:         $c_e^{(3)} =$ SPLITPROCEDURE($S_E, c_i^{(3)}, c_j^{(3)}$)    ▷ After $c_i^{(3)}, c_j^{(3)}$ assign $S_E$.
18:         for all $m \notin S_I$ do
19:             $c_m^{(3)} = c_m^{(2)}$    ▷ Data points in clusters other than $c_i, c_j$ are not adjusted.
20:         end for
21:         $c' = \{c_i^{(3)}, c_j^{(3)}, c_e^{(3)}, c_m^{(3)}\}$
22:         $a = a_{split}(c',c)$ according to Eq. ??    ▷ MH acceptance for a split.
23:     else    ▷ Case: $i, j, k$ belong to thee different clusters $c_i \neq c_j \neq c_k$ ($N_S = 3$).
24:         $S_L = S_I \cap \{c_i^{(3)}, c_j^{(3)}\}$    ▷ Data in clusters $c_i, c_j, c_k$ except for $i$ and $j$ itself.
25:         $\{c_i^{(2)}, c_j^{(2)}\} = \text{SAMS}(S_L, c_i^{(3)}, c_j^{(3)})$    ▷ Assign data points in $c_i, c_j, c_k$ to $c_i, c_j$.
26:         for all $m \notin S_L$ do
27:             $c_m^{(2)} = c_m^{(3)}$    ▷ Data points in clusters other than $S_L$ are not adjusted.
28:         end for
29:         $c' = \{c_i^{(2)}, c_j^{(2)}, c_m^{(2)}\}$
30:         $a = a_{merge}(c',c)$ according to Eq. 5.21    ▷ MH acceptance for a merge.
31:     end if
32:     $u \sim U(0,1)$    ▷ Sample $u$ between 0 or 1 uniformly.
33:     if $a < u$ then
34:         $c' = c$    ▷ Reject $c'$ by setting it to $c$
35:     end if
36:     return $c'$, the (updated) cluster assignment vector: $c \rightarrow c'$.
37: end procedure
```

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:

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

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 $N(0, \sigma_0)$ 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 $N(0, \sigma_1)$.

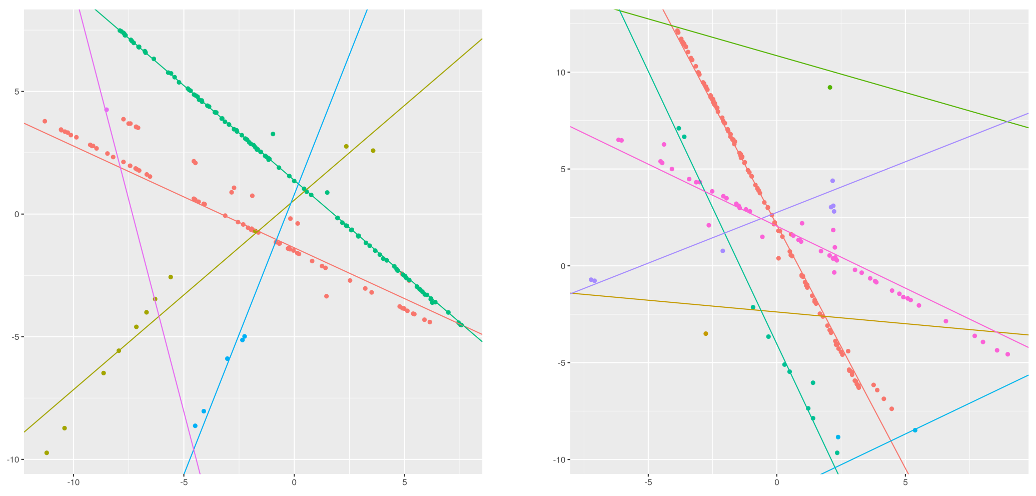


Figure 5.3: Two examples of fitting a mixture of lines to data items scattered over a two-dimensional space. The lines drawn are inferred using the triadic sampler. The lines are not the ground truth, but are meant to demonstrate the typical errors made by fitting methods. Note, for example, that there 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.

5.4.1 Implementation

The sampler is open-source[2] 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 randon 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):

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

[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.

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)).

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

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.

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 5.4 to 5.6 the different metrics are visualized in the form of violin plots.

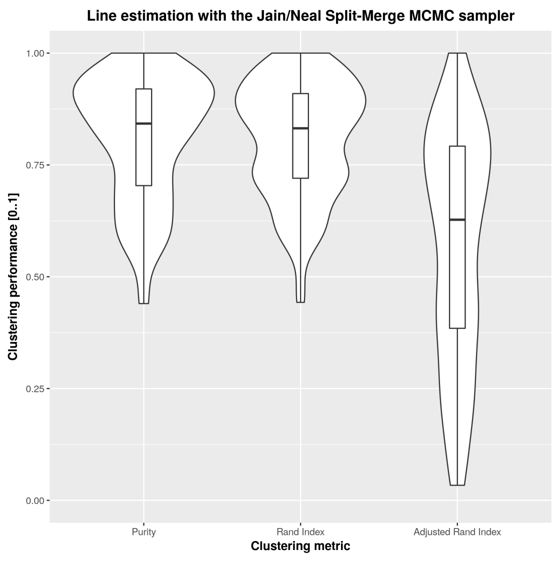


Figure 5.4: Line estimation with the diadic split-merge sampler. The same results as in Table 5.1, but visualized in a violin plot. The distribution over metric values are displayed in a vertical fashion.

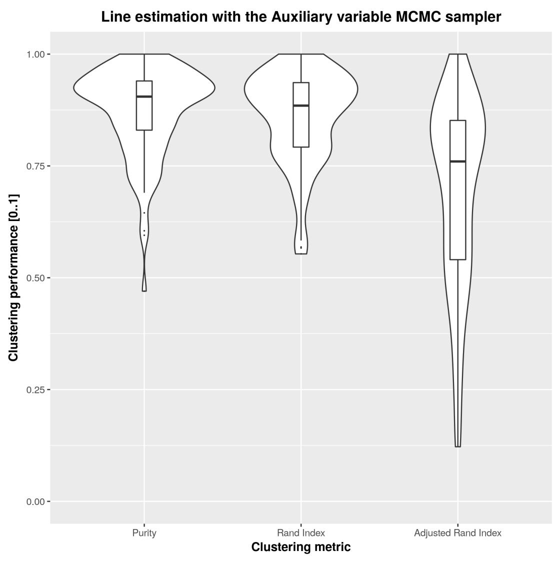


Figure 5.5: Line estimation with the auxiliary variable sampler.

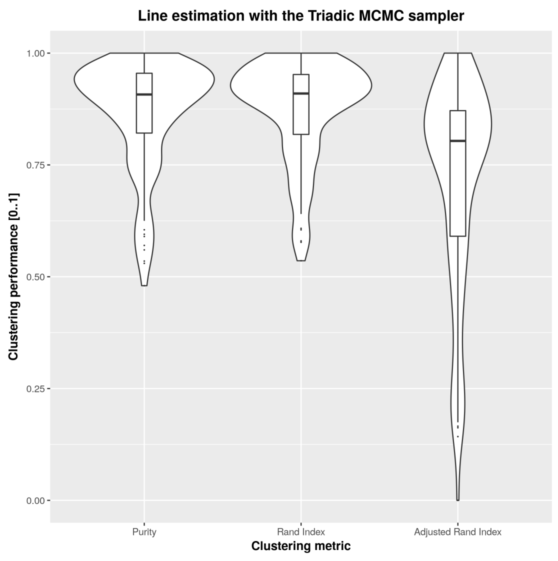


Figure 5.6: Line estimation with the triadic split-merge sampler (our inference method). The values are all shifted up towards one.

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).

This chapter answers our second research question: "How can we estimate the number of lines simultaneously with line fitting in computer vision." 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.

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). We will introduce such a sampler in chapter 6.

CHAPTER 6

Adversarially Trained MCMC Kernels

Contents To use MCMC for volumetric inference it is necessary to be able to accelerate the algorithms even further. Volumetric objects exhibit more structure, which is reflected by symmetry.

Outline We describe MCMC methods that cope with symmetric objects.

6.1 Data-Driven Inference

There are three aspects we would like to address in our inference engine.

The first aspect aims to have structure within our inference engine. The proposal distribution in a Markov chain, although moderately complex in the previous chapter, does not have much knowledge about the model. An artificial border is maintained that does not allow the inference engine to have knowledge about the model. The purpose of this is never articulated in particular. However, it is logical from a separation of concern. Such an inference engine (1) does not need to receive any information about the model and (2) is guaranteed to be general in the sense that it is not tailored to a particular model. This is nicely articulated by Tran et al. (2017) from which we quote.

> “*Many existing probabilistic programming languages treat the inference engine as a black box, abstracted away from the model. These cannot capture probabilistic inferences that reuse the model's representation - a key idea in recent advances in variational inference, generative adversarial networks, and also in more classic inference.*”

The second aspect concerns the data. In MCMC the position for the chain is driven by (1) the prior, (2) the prior and the likelihood, (3) a sequence of priors and likelihood, (4) a sequence of priors, likelihood and proposal distributions, basically anything, except for the data itself. Data-driven approaches would namely destroy the convergence of the Markov chain. To start an MCMC sampler in a data-driven manner and continue in a data-oblivious manner is a possible solution (Zhang and Perez-Cruz, 2017). Even better, it is possible to use a Metropolized independence sampler Jampani et al. (2015). Such a sampler samples independently from the previous state and uses global information. However, to work well its proposal distribution needs to match the target distribution quite well. Although, when combined with a local sampler, it might be sufficient to just be able to match the modes of the target distribution well.

The third aspect concerns the way we build our MCMC engine. The split-merge sampler of the previous chapter has been meticulously designed. If we admit a data-driven approach, we might as well adjust our MCMC engine using training samples. Note, that this training will be across a set of line, box, or scenes mixtures. The MCMC engine will not be able to learn just the parameters of a particular visual object. It will learn how to jump around (optionally, adaptively) from one visual object to the next or from one cluster configuration to the next. In other words, it will be able to teach itself to become a Triadic Split-Merge sampler if that happens to be a good engine. Is it possible to constrain the search through MCMC kernels such that its result is always converging in an MCMC sense? If we aim to learn the transition operator of our Markov chains, there is new literature that makes use of deep nets.

6.2 Learning the Transition Operator

There are multiple methods that can be used in a generative setting. We will discuss the three most prominent ones: (1) generative adversarial networks, (2) variational autoencoders, and (3) infusion training. This is far from an extensive categorization, worth studying are variational walkback (Goyal et al., 2017), stacked generative adversarial networks (Huang et al., 2016), generative latent optimization (Bojanowski et al., 2017), deep learning through the use of non-equilibrium thermodynamics (Sohl-Dickstein et al., 2015), denoising autoencoders, or generative stochastic networks, to name just a few.

6.2.1 Adversarial Training

Adversarial training has been extensively studied since the article on generative adversarial networks by Goodfellow et al. (2014). A particular adversarial setup for training an MCMC has been suggested as well (Song et al., 2017). The generator samples from a Markov chain. A discriminator subsequently needs to judge if its incoming data comes from the generator or if it is sampled from the actual data set. To start the process, the generator can run the chain from the model as well as from the data.

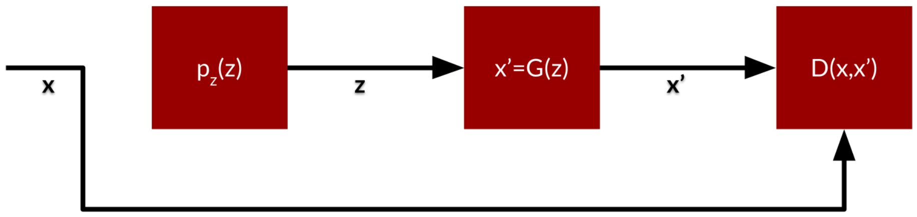


Figure 6.1: Left: $p_z(z)$ is a (prior) random distribution that generates random variables z. Middle: the generator G maps the random variables z to simulated data points x'. Right: the discriminator $D(x,x')$ compares the simulated data x' with the real data x. The generator tries to generate samples in such way that the discriminator has difficulties distinguishing the simulated from the real data.

6.2.2 Variational Autoencoders

Variational autoencoders (Kingma and Welling, 2013; Rezende et al., 2014) are ordinary autoencoders with additional constraints on the latent variables. The latent variables in autoencoder parliance are called the code. In a variational autoencoder the latent variables are forced to approximately describe a 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. 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.

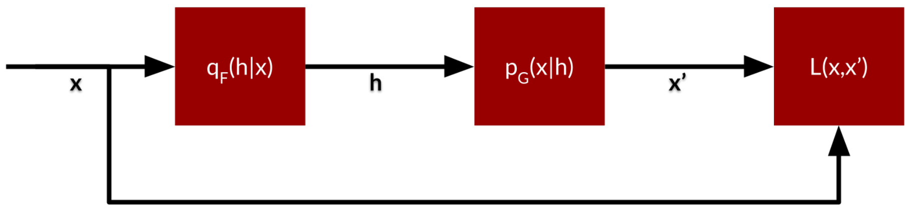


Figure 6.2: Left: $q_F(h|x)$ maps the data x to (hidden) random variables h. Middle: $p_G(x|h)$ maps the hidden random variables to reconstructed data x'. Right: $L(x,x')$ measures the similarity between x and x'.

6.2.3 Infusion Training

The transition operator can also be learned directly through infusion training. In infusion training we gradually adjust totally unstructured noise to a target distribution as well. In this method a particular data point is 'infused' into the Markov chain to bias the model sampling to move towards this particular data point (and not another). In contrast to a generative loss that is a mean squared error, this promises to have less blurry reconstructions.

6.3 Volumetric Models

In image processing autoencoders have been used for 2D shape recognition. To apply the same type of models to 3D point louds, these point clouds are represented through voxels. The application of data-driven deep learning techniques, be it autoencoders, generative adversarial networks, or adversarial autoencoders promises similar good results in these 3D settings than in the current computer vision tasks.

The 3D ShapeNet model (Wu et al., 2015) exists of 3D voxel input that is piped through several stages with an increasing number of filters. The used voxel representation is a binary tensor. It assigns a value of 1 to each voxel that is inside the 3D object mesh and a value 0 to each voxel outside the mesh (empty space). The voxel sizes are fixed as well as the grid size (in this particular model the grid exists of 30x30x30 voxels). The inference model is a Deep Belief Network (DBN). Convolution operators, in the form of filters over small neighbourhoods, are used to reduce the number of model parameters (30x30x30 fully connected would be really many weights). The DBN is used in a supervised setting where shapes are trained with object labels. The model subsequently learns to generate shapes given an object label.

An unsupervised method in the form of a convolutional (volumetric) autoencoder (Sharma et al., 2016) has been applied to the same type of data. This (denoising) autoencoder, coined VConv-DAE maps from an entire voxel grid to another voxel grid. This work uses a combination of standard techniques, a dropout layer, a deconvolution layer, ReLu as well as sigmoid activation, but it is not in particular tailored to 3D point clouds.

Other representations than voxels are used. For example collections of 2D views and transformation parameters (Dosovitskiy et al., 2017). The most interesting are methods that work with raw data, the point cloud themselves. This alleviate the need to process the data and does not inadvertedly increase the data dimensions, for example by artificially introduce voxels where there is no object present.

PointNet (**?**) directly operates on point clouds. To handle the input as a set of points (unordered), it uses a symmetric function over n input vectors and outputs a vector that is invariant to the input order. Typically sum and multiplication operators are such symmetric functions. After input and feature transforms by multi-layer perceptrons, a max pooling operator is used to map the input to a global feature. In the ModelNet40 shape classification benchmark there are more than 12000 CAD models from 40 object categories. PointNet achieves state of the art results compared to volumetric methods for a fraction of the computational costs.

Point clouds are also directly used in so-called deep kd-networks (Klokov and Lempitsky, 2017). A kd-tree is constructed by recursively picking the coordinate axis with the largest range of point coordinates and splitting the set of points into two subsets of equal size. These subsets are recursed into successively. The recursion stops at a particular level, depth D. The kd-networks are purported to outperform for example PointNet amongst other model architectures.

A deep permutation equivariant (for semisupervised learning) and permutation invariant (for supervised learning) network has also bee directly applied to point clouds (Ravanbakhsh et al., 2016). It does not reach the ModelNet40 accuracy levels from PointNet or the kd-networks though.

PointNet++ Qi et al. (2017) introduces hierarchical structure to PointNet. This fits better non-uniform point distributions and seems to surpass kd-nets again on the ModelNet40 task.

Chapter 7

Recommender Engine

Contents The described nonparametric Bayesian models (Chapter 3, 4, 5, and 6) are not limited to computer vision tasks. This chapter describes a recommender engine in which groups of runners are extracted from data collected from social media.

Outline We (1) introduce the form of the data at hand, (2) describe a multi-modal Von Mises-Uniform distribution to model the individual runners, (3) use a Dirichlet Process prior to group people, (4) use the previously described MCMC methods to perform inference, (5) show the results on an artificial and real-world data set, and (6) discuss ways with which the model can be expanded.

7.1 Application

The data of people exercising can be considered binary (someone is either exercising or not in a particular timeslot). We do have however more information available. We know how often people have been exercising in a timeslot. This data has the form as visualized in Table 7.1.

Table 7.1: Example of the type of data about the timing of exercising. A person is represented by row, her preferences by column. There is not a predefined number of users or groups of users.

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
0	0	0	0	4	23	38	9	12	6	2	7	2	3	2	7	5	3	2	0	3	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	11	4	4	2	1	2	0	0	0
0	0	0	0	0	1	0	0	1	2	0	0	0	1	0	2	3	1	1	3	2	0	0	0
0	0	0	0	0	0	5	4	1	3	11	3	7	3	3	4	3	10	23	3	0	0	0	0
0	0	0	0	0	0	8	35	23	12	4	41	14	11	8	7	14	38	36	6	5	1	2	0
0	0	0	0	0	0	0	1	9	2	0	3	0	0	0	1	0	0	0	0	0	0	0	0
0	0	0	0	0	0	29	9	8	7	5	3	0	2	0	1	8	6	1	4	1	2	0	0
0	0	0	0	0	2	14	12	7	1	0	2	0	2	0	3	4	6	4	3	9	2	0	1
0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	3	3	2	0	1	0	0	0
0	0	0	0	0	0	0	1	1	0	1	1	0	0	0	2	1	2	3	0	1	0	0	0
0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	4	5	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	3	2	1	0	1	1	0	0	0	0	3	11	12	0	0	0
0	0	0	0	0	0	0	0	0	2	1	0	0	0	1	0	3	5	4	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	1	18	4	0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	2	0	0	1	9	14	14	2	1	4	15	4	0	0
0	0	0	0	0	0	0	0	2	4	2	1	3	5	2	0	1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	1	0	0	0	1	1	3	3	1	3	2	3	0	0
0	0	0	0	0	0	0	1	2	1	3	2	0	1	2	0	0	0	0	1	2	0	0	0
0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	8	4	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	4	6	2	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	0	1	1	6	2	1	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	2	2	3	12	18	12	12	2	4	3	5	6	43	37	16	1	2	0

The first row defines the time of day, starting at midnight till the last timeslot from 23.00 till 00.00. In this particular case it shows that nobody is running from midnight to four o'clock in the morning, understandably so.

7.2 Model of Individuals

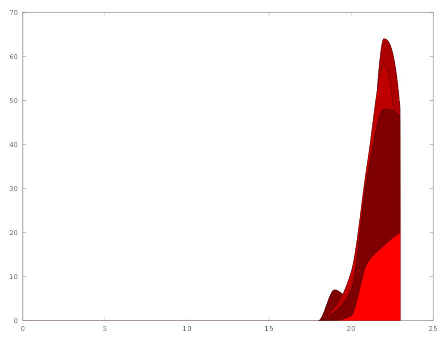
7.2.1 Multi-modal Normal-Uniform Distribution Model

We first postulate the likelihood function for the moments at which people exercise through the day as defined in Eq. 7.1.

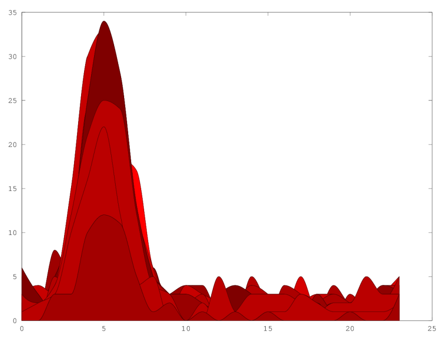
$$f(x|\theta) = w_0\mathcal{U}(a,b) + \sum_{i=1}^{2} w_i\mathcal{N}(\mu_i,\sigma_i) \tag{7.1}$$

The likelihood (Eq. 7.1 is built up out of three probability density functions: one Uniform distribution $\mathcal{U}(a,b)$ with a and b as parameters and two Normal distributions $\mathcal{N}(\mu_i,\sigma_i)$ with mean μ_i and σ_i. The distributions are weighted by the factors w_0, w_1, w_2. The collection of parameters for the likelihood function is referred to by $\theta = \{a, b, \mu_1, \sigma_1, \mu_2, \sigma_2, w_0, w_1, w_2\}$.

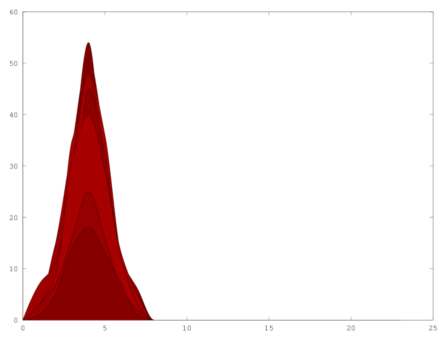
This probability density function $f(x|\theta)$ will have the form as in Fig. 7.1. The uniform distribution generates values here between 00:00 and 24:00. There are on top of that the two Normal distributions that form peaks at certain moments during the day.



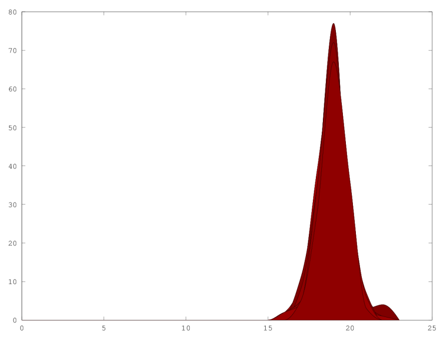
(a) People who like to run late in the evening.



(b) People who run before work, early in the morning and occasionally during the day.



(c) People who exclusively run before work.



(d) People who run right after dinner.

Figure 7.1: The likelihood function for the moments at which people decide to exercise during the day. On the horizontal axis time, on the vertical axis the frequency of exercising.

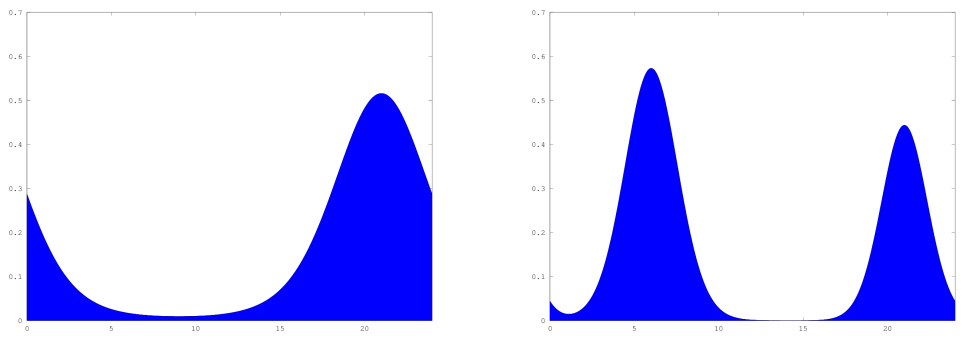
There is something noticable in Fig. 7.1, namely that the 24 hours of a day cause the Normal distribution to be cut off. Especially in Fig. 7.1 (a) there should be some considerable likelihood of running in the wee hours of the morning between 00:00 and 01:00.

7.2.2 Multi-modal Von-Mises-Uniform Distribution Model

There are several options to define a distribution over a limited range T. A so-called wrapped distribution is a distribution defined over the unity circle. By just multiplying it with $T/(2pi)$ it can be used to define a probability density function over a day ($T = 24$).

$$f(x|\theta) = w_0 \mathscr{U}(a,b) + \sum_{i=1}^{2} w_i \mathscr{V}\mathscr{M}(\mu_i, \kappa_i) \tag{7.2}$$

The likelihood (Eq. 7.2) is again built up out of three probability density functions: one Uniform distribution $\mathscr{U}(a,b)$ with a and b as parameters and two Von Mises distributions $\mathscr{V}\mathscr{M}(\mu_i, \kappa_i)$ with mean μ_i and κ_i. The parameters μ_i will be scaled and shifted with $[a,b]$ so all variables within this range fall on the unity circle. The parameter κ_i plays the same role as σ_i for the Normal distribution. The distributions are weighted by the factors w_0, w_1, w_2. The collection of parameters for the likelihood function is referred to by $\theta = \{a, b, \mu_1, \kappa_1, \mu_2, \kappa_2, w_0, w_1, w_2\}$.



(a) People who like to run late in the evening

(b) People who run before work, early in the morning and later in the evening.

Figure 7.2: The improved likelihood function for the moments at which people decide to exercise during the day using the Von Mises distribution. On the horizontal axis time, on the vertical axis the frequency of exercising.

The likelihood with Von Mises distributions rather than Normal distributions is visualized in Fig. 7.2. The Von Mises distributions capture behavior in deviations from a standard exercise time. It does not take into account structured deviations, for example running late from work every Thursday, or weekends, or a person either running in the morning or in the evening, but never both at the same day.

7.2.3 Hyperparameters

The parameters θ for each user j are either fixed or generated from prior distributions. The hyperparameters a and b for the Uniform distribution are set to 0 and 24. People can run potentionally any time of the day. The hyperparameters μ_i, κ_i for the Von Mises distributions are generated from a Uniform-Exponential distribution (Eq. 7.3). The Uniform distribution reflects the fact that if people run on regular times, this time can be any time of the day. The Exponential distribution defines a prior on how much people deviate from such a regular time to exercise.

$$\begin{aligned} f(\mu_i|a,b) &= \mathcal{U}(a,b) \\ f(\kappa_i|\lambda) &= \mathcal{E}(\lambda) \end{aligned} \tag{7.3}$$

The Uniform distribution generates μ_i between a and b. The parameter κ_i is generated from an Exponential distribution with hyperparameter λ. If λ is set to be small (< 0.5) we have a high likelihood that κ can be large and we have pronounced peaks. In contrary, if λ is set to be large, the Von Mises distribution likely approaches the Uniform distribution due to a higher chance of sampling a small value for κ.

The weights we sample from a normalized product of a zero-deflated Bernoulli distribution and a Dirichlet distribution (Eq. 7.4).

$$f(w_i|p,\alpha_i) = \mathscr{B}(p)\mathscr{D}(\alpha_i)/Z \tag{7.4}$$

The Bernoulli distribution samples zeros and ones with probability of $p = 0.5$, leading to 3-vectors like 001, 101, etc. The distribution is corrected in such way that the chance to sample 000 is zero. The Bernoulli distribution only gives weights $w_i = 0$ or $w_i = 1$, hence it is multiplied with a Dirichlet distribution. The Dirichlet samples a 3-vector with weights between zero and one where the weights $\sum_i w_i = 1$. A symmetric Dirichlet distribution with $\alpha = 1$ is similar to the Uniform distribution over the simplex. It is set slightly more towards favoring particular distributions with $\alpha_i = 1/3$ (we assume that people sample one or two distributions, and rarely from all three). The product with the zero-deflated Bernoulli distribution is made up to sum up to one again by normalizing the result.

We can combine Eq. 7.3 and Eq. 7.4 in Eq. 7.5:

$$\theta \sim \mathscr{U}(a,b)\mathscr{E}(\lambda)\mathscr{B}(p)\mathscr{D}(\alpha_i)/Z \tag{7.5}$$

To sample the parameters θ this is the base distribution we will encounter in the next section 7.3.

7.3 Model of Groups

Each person's exercise schedule is represented by a Von-Mises-Uniform distribution. People that are similar do have exercise schedules that can be represented by the same Von-Mises-Uniform distribution. To group similar schedules we define a nonparametric discrete distribution over a potentially infinite number of groups with each person assigned to a group.

A Dirichlet Process (Eq. 7.6) is a distribution over distributions that can be used as a prior for such a nonparametric discrete distribution.

$$DP(\alpha, H) \tag{7.6}$$

The Dirichlet Process has (1) a hyperparameter α, which defines the likelihood that there are many clusters versus few clusters (although it doesn't say anything about its actual count), and (2) a base distribution H, the distribution that generates θ (Eq.7.5).

7.4 Inference

The implementation of the model makes use of Gibbs sampling with auxiliary variables Jain and Neal (2007).

Details on this algorithm can be found in Jain and Neal (2007) and previous work of the authors van Rossum et al. (2016b,c).

Algorithm 15 Gibbs sampling over auxiliary variables

```
 1: procedure GIBBS ALGORITHM WITH AUXILIARY VARIABLES($w, \lambda_0, \alpha$)    ▷ Accepts schedule $w$,
    hyperparameters $\lambda_0, \alpha$, number of auxiliary variables $m$, and returns $k$ groups
 2:     for all $t = 1 : T$ do
 3:         for all $i = 1 : N$ do
 4:             for all $j = 1 : m$ do
 5:                 $\theta_j \sim H(\lambda_0)$    ▷ Sample $\theta_j$ from base distribution $H$ in Eq. 7.5
 6:             end for
 7:             for all $j = 1 : K + m, j \neq i$ do
 8:                 $L_j = \text{likelihood}(w_i, \theta_j)$    ▷ Update likelihood for all theta (except $\theta_i$) given $w_i$
 9:             end for
10:             $P_{-i=1:K} = b \sum_{-i} L_{-i}$    ▷ Calculate probability of existing cluster
11:             $P_{-i=K:K+m} = b\alpha / m L_m L_{-i}$    ▷ Calculate probability of new cluster
12:             $\theta_i = \theta_j$ according to above $P_{-i}$    ▷ Sample $\theta_i$ accord. to above prob
13:             Remove unused clusters
14:         end for
15:         for all $j = 1 : K$ do
16:             $\theta_j \sim p(\theta_j \mid y)$    ▷ Update $\theta_j$
17:         end for
18:     end for
19:     return summary on $\theta_k$ for $k$ groups of runners
20: end procedure
```

7.5 Results

The algorithm is run first on an artificial dataset of which we know the ground truth (Sect. 7.5.1) and next on the real-world dataset from Twitter (Sect. 7.5.2).

7.5.1 Artificial Dataset

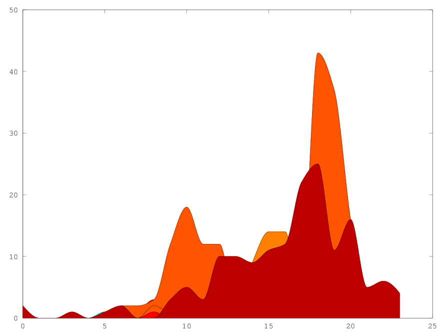
The algorithm has been used on a self-generated dataset (generated from the probability density function as in Fig. 7.2). In this case we have the ground truth that establishes which exercise schedule comes from which probability density function. Using this ground truth we can calculate how often our algorithm makes a mistake, grouping a person with people that belong to another group. The results with this dataset are perfect (Rand Index equal to one: perfect clustering). The results are of the form of Table 7.2 (except by a permutation of indices), hence are not shown (the indices have no intrinsic meaning).

Table 7.2: Top row: A sequence of cluster indices indicates the ground truth. Each cluster index represents a multi-modal Von-Mises-Uniform distribution with different parameters θ. Bottom row: A sequence of cluster indices that are the result of the described algorithm. Each cluster index represents again a multi-modal Von-Mises distribution. Errors would be represented by an inconsistent mapping from the top row to the bottom row.

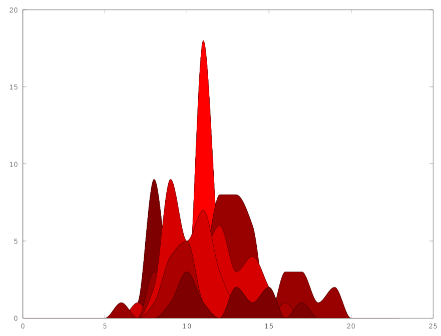
1	4	4	5	2	2	2	1	3	2	1	5	1	1	4	4	3	1	3	1	2	...
4	5	5	1	2	2	2	4	3	2	4	1	4	4	5	5	3	4	3	4	2	...

7.5.2 Real-world Dataset

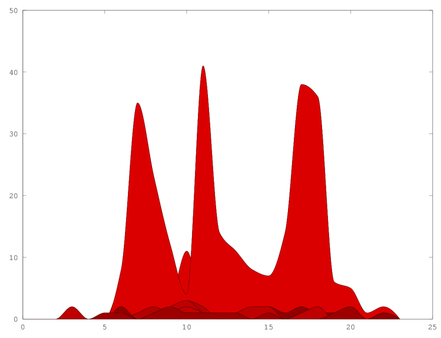
The results when we actually use the collected Twitter dataset can be best visualized (Fig. 7.3). This dataset consists of around 4000 moments at which people decide to run. The dataset is subsequently filtered on regulars, people that at least have run a few times.



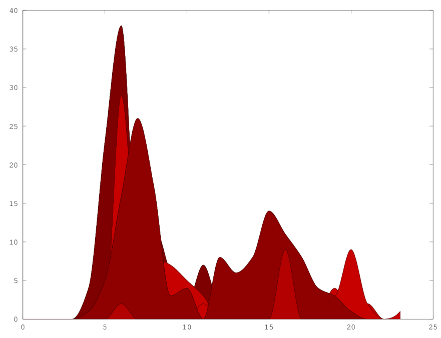
(a) People who predemominally seem to like to run in the early evening.



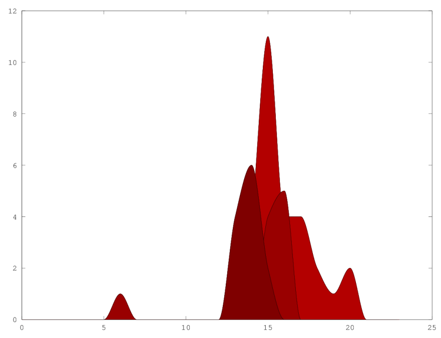
(b) People who run during the day, not later than at 20:00 o'clock.



(c) People who seem to run all the time (interesting collection though!).



(d) People who run early in the morning.



(e) People who run around 16:00 o'clock and almost never run at another time.

Figure 7.3: After running the algorithm, the above figures show the different categories of runners that have been found.

The results in Fig. 7.3 show that there is a categorization of people indeed. After the algorithm has done its work it is possible to assign a certain label in a post-hoc manner, e.g. "People who run around 16:00 o'clock and almost never run at another time.".

7.6 Discussion

The postulated model allows to reason about groups of people performing exercising, without predefining what these groups constitute apart from very general characteristics such as that there might be preferred times of day to exercise. A Dirichlet process is used as a nonparametric Bayesian model in which both the number of groups and the assignment of people exercising are learnt from the data.

There are several directions in which this research can be extended. First, more data would be very helpful. Only a limited number of people are posting consistently their training data online. Tapping into the data of current fitness promoting companies would allow the model to wash out the prior a bit more and adjust to the data. Second, we also collected weather data over this time period and also expect the day of the week to have significant influence. When there will be more data available these are logically dimensions to include in the dataset. Third, it would be interesting to study if people relate to the group of people they have been categorized with. Can this help or support their exercise regime?

Chapter 8

Discussion and Conclusions

References

M Abdel-Hameed. Optimal replacement policies for devices subject to a gamma wear process. *The theory and applications of reliability*, pages 397–412, 2012.

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

David J Aldous. *Exchangeability and related topics*. Springer, 1985.

John Aldrich and Others. R.A. Fisher and the making of Maximum Likelihood 1912-1922. *Statistical Science*, 12(3):162–176, 1997.

Charles E Antoniak. Mixtures of Dirichlet processes with applications to Bayesian nonparametric problems. *The annals of statistics*, pages 1152–1174, 1974.

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

Federico Bassetti, Roberto Casarin, and Fabrizio Leisen. Beta-product dependent Pitman–Yor processes for Bayesian inference. *Journal of Econometrics*, 180(1):49–72, 2014.

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

Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In *European conference on computer vision*, pages 404–417. Springer, 2006.

Christopher M Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag New York, 2016.

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

Phil Blunsom and Trevor Cohn. A hierarchical Pitman-Yor process HMM for unsupervised part of speech induction. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 865–874. Association for Computational Linguistics, 2011.

Piotr Bojanowski, Armand Joulin, David Lopez-Paz, and Arthur Szlam. Optimizing the latent space of generative networks. *arXiv preprint arXiv:1707.05776*, 2017.

Robert C Bolles and Martin 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.

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

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

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

Michael Bryant and Erik 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.

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

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

Jason Chang and John W Fisher III. Parallel sampling of dp mixture models using sub-cluster splits. In *Advances in Neural Information Processing Systems*, pages 620–628, 2013.

Haifeng Chen, Peter Meer, and David 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.

David B Dahl. An Improved Merge-Split Sampler for Conjugate Dirichlet Process Mixture Models. Technical report, University of Wisconsin–Madison, November 2003.

David B Dahl. Sequentially-Allocated Merge-Split Sampler for Conjugate and Nonconjugate Dirichlet Process Mixture Models. Technical report, Texas A&M University, November 2005.

Navneet Dalal and Bill 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.

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

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

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

David GT Denison. *Bayesian methods for nonlinear classification and regression*, volume 386. John Wiley & Sons, 2002.

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

Alexey Dosovitskiy, Jost Tobias Springenberg, Maxim Tatarchenko, and Thomas Brox. Learning to generate chairs, tables and cars with convolutional networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(4):692–705, 2017.

François Dufresne, Hans U Gerber, and Elias S W Shiu. Risk theory with the gamma process. *Astin Bulletin*, 21(02):177–192, 1991.

David B Dunson, Ya Xue, and Lawrence Carin. The matrix stick-breaking process. *Journal of the American Statistical Association*, 2012.

Richard Durstenfeld. Algorithm 235: Random Permutation. *Communications of the ACM*, 7(7):420, July 1964. ISSN 0001-0782. doi: 10.1145/364520.364540. URL `http://doi.acm.org/10.1145/364520.364540`.

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

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

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

Stefano Favaro, Yee Whye Teh, and Others. MCMC for normalized random measure mixture models. *Statistical Science*, 28(3):335–359, 2013.

William Feller. An introduction to probability theory and its applications. Vol. I. 1950.

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

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

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

Wolfgang Förstner and Eberhard 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.

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

David Heaver Fremlin. *Measure theory*, volume 4. Torres Fremlin, 2000.

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

Qing-Bin Gao and Shi-Liang Sun. Human activity recognition with beta process hidden Markov models. In *Machine Learning and Cybernetics (ICMLC), 2013 International Conference on*, volume 2, pages 549–554. IEEE, 2013.

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

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

Anirudh Goyal, Nan Rosemary Ke, Surya Ganguli, and Yoshua Bengio. Variational walkback: Learning a transition operator as a stochastic recurrent net. *arXiv preprint arXiv:1711.02282*, 2017.

Max Halperin and G L Burrows. The Effect of Sequential Batching for AcceptanceâĂŤRejection Sampling Upon Sample Assurance of Total Product Quality. *Technometrics*, 2(1):19–26, 1960.

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

Li He, Hairong Qi, and Russell Zaretzki. Beta process joint dictionary learning for coupled feature spaces with application to single image super-resolution. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 345–352, 2013.

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

Fred 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.

Paul V.C. Hough. Method and Means for Recognizing Complex Patterns, Dec 1962. URL `https://www.google.com/patents/US3069654`. Patent US 3069654 A.

Xun Huang, Yixuan Li, Omid Poursaeed, John Hopcroft, and Serge Belongie. Stacked generative adversarial networks. *arXiv preprint arXiv:1612.04357*, 2016.

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

Michael C Hughes, Emily Fox, and Erik 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.

Tommi S Jaakkola, David Haussler, and Others. Exploiting generative models in discriminative classifiers. *Advances in neural information processing systems*, pages 487–493, 1999.

Sonia Jain and Radford 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`.

Sonia Jain and Radford M Neal. Splitting and Merging Components of a Nonconjugate Dirichlet Process Mixture Model. *Bayesian Analysis*, 2(3):445–472, 2007.

Varun Jampani, Sebastian Nowozin, Matthew Loper, and Peter 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.

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

Dominik Joho, Gian Diego Tipaldi, Nikolas Engelhard, Cyrill Stachniss, Wolfram Burgard, Martin Senk, Felix Faber, Maren Bennewitz, Clemens Eppner, Attila Görög, and Others. 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.

Michael I Jordan. Hierarchical models, nested models and completely random measures. *Frontiers of Statistical Decision Making and Bayesian Analysis: in Honor of James O. Berger. New York: Springer*, 2010.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

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.

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. 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.

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

Roman Klokov and Victor Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 863–872. IEEE, 2017.

David Knowles, Zoubin Ghahramani, and Konstantina 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.

Daphne Koller and Nir 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.

Uwe Küchler and Stefan Tappe. Bilateral Gamma distributions and processes in financial mathematics. *Stochastic Processes and their Applications*, 118(2):261–283, 2008.

Kenichi Kurihara, Max Welling, and Yee Whye Teh. Collapsed Variational Dirichlet Process Mixture Models. In *IJCAI*, volume 7, pages 2796–2801, 2007.

Pierre-Simon Laplace. *Théorie analytique des probabilités*. V. Courcier, 1820.

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

Zhidong Li, Bang Zhang, Yang Wang, Fang Chen, Ronnie Taib, Vicky Whiffin, and Yi Wang. Water pipe condition assessment: a hierarchical beta process approach for sparse incident data. *Machine learning*, 95(1):11–26, 2014.

Antonio Lijoi and Igor 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).

David 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.

Steven 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.

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

Dilip B Madan and Eugene Seneta. The variance gamma (VG) model for share market returns. *Journal of business*, pages 511–524, 1990.

Nicholas Metropolis, Arianna W. Rosenbluth, Marshall N. Rosenbluth, Augusta H. Teller, and Edward Teller. Equation of State Calculations by Fast Computing Machines. *The Journal of Chemical Physics*, 21(6):1087–1092, 1953. doi: 10.1063/1.1699114.

Radford M Neal. Defining Priors for Distributions Using Dirichlet Diffusion Trees. Technical report, University of Toronto, Toronto, Ontario, Canada, 2001.

Radford M Neal. Markov Chain Sampling Methods for Dirichlet Process Mixture Models. *Journal of Computational and Graphical Statistics*, 9(2):249–265, 2000.

Peter Orbanz and Daniel 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.

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

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114, 2017.

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

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

Vinayak Rao and Yee W Teh. Spatial normalized gamma processes. In *Advances in neural information processing systems*, pages 1554–1562, 2009.

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian processes for machine learning*, volume 2. 2006.

Siamak Ravanbakhsh, Jeff Schneider, and Barnabas Poczos. Deep learning with sets and point clouds. *arXiv preprint arXiv:1611.04500*, 2016.

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

James C Ross, Peter J Castaldi, Michael H Cho, and Jennifer G Dy. Dual beta process priors for latent cluster discovery in chronic obstructive pulmonary disease. In *Proceedings of the 20th ACM SIGKDD* *international conference on Knowledge discovery and data mining*, pages 155–162. ACM, 2014.

Daniel M Roy and Yee W Teh. The mondrian process. In *Advances in neural information processing systems*, pages 1377–1384, 2009.

Anirban Roychowdhury and Brian Kulis. Gamma processes, stick-breaking, and variational inference. In *Artificial Intelligence and Statistics*, pages 800–808, 2015.

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

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

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

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

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

Abhishek Sharma, Oliver Grau, and Mario Fritz. Vconv-dae: Deep volumetric shape learning without object labels. In *European Conference on Computer Vision*, pages 236–250. Springer, 2016.

Jianbo Shi and Carlo 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.

Nozer Singpurwalla. Gamma processes and their generalizations: an overview. In *Engineering probabilistic design and maintenance for flood protection*, pages 67–75. Springer, 1997.

Scott A Sisson and Yanan 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.

Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *arXiv preprint arXiv:1503.03585*, 2015.

Jiaming Song, Shengjia Zhao, and Stefano Ermon. A-nice-mc: Adversarial training for mcmc. *arXiv preprint arXiv:1706.07561*, 2017.

Guillaume Stawinski, Arnaud Doucet, and Patrick 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.

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

Erik B Sudderth and Michael I Jordan. Shared segmentation of natural scenes using dependent Pitman-Yor processes. In *Advances in Neural Information Processing Systems*, pages 1585–1592, 2009.

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

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

Yee Whye Teh, Michael I Jordan, Matthew J Beal, and David M Blei. Hierarchical Dirichlet processes. *Journal of the American statistical association*, 101(476), 2006.

Romain Thibaux and Michael I Jordan. Hierarchical beta processes and the Indian buffet process. In *International Conference on Artificial Intelligence and Statistics*, pages 564–571, 2007.

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

Dustin Tran, Matthew D Hoffman, Rif A Saurous, Eugene Brevdo, Kevin Murphy, and David M Blei. Deep probabilistic programming. *arXiv preprint arXiv:1701.03757*, 2017.

Anne C van Rossum, Hai Xiang Lin, Johan Dubbeldam, and H Jaap van den Herik. Fundamentals of nonparametric bayesian line detection. In *International Conference on Pattern Recognition Applications and Methods*, pages 175–193. Springer, 2016a.

Anne C. van Rossum, Hai Xiang Lin, Johan Dubbeldam, and H. Jaap van den 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.

Anne C. van Rossum, Hai Xiang Lin, Johan Dubbeldam, and H. Jaap van den 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.

Anne C. van Rossum, Hai Xiang Lin, Johan Dubbeldam, and H. Jaap van den Herik. Triadic split-merge sampler. Nov 2017.

Niklas Vanhainen and Giampiero Salvi. Word Discovery with Beta Process Factor Analysis. In *INTERSPEECH*, pages 799–802, 2012.

Wei Wang and Stuart 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. URL `http://dl.acm.org/citation.cfm?id=3020847.3020940`.

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

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

Robert L Wolpert and Katja Ickstadt. Poisson/gamma random field models for spatial statistics. *Biometrika*, 85(2):251–267, 1998.

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

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong 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.

Jing-Hao Xue and D Michael Titterington. Comment on âĂIJOn discriminative vs. generative classifiers: A comparison of logistic regression and naive BayesâĂİ. *Neural processing letters*, 28(3): 169–187, 2008.

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

Michael M Zhang and Fernando Perez-Cruz. Accelerated inference for latent variable models. *arXiv preprint arXiv:1705.07178*, 2017.

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

Mingyuan Zhou, Hongxia Yang, Guillermo Sapiro, David B Dunson, and Lawrence Carin. Dependent hierarchical beta process for image interpolation and denoising. In *International conference on artificial intelligence and statistics*, pages 883–891, 2011.

Mingyuan Zhou, Lauren A Hannah, David B Dunson, and Lawrence Carin. Beta-negative binomial process and poisson factor analysis. *Journal of Machine Learning Research*, 2012.

APPENDIX A

PROBABILISTIC CONCEPTS

TODO: What follows down here is old and has to be adapted to the measure-theoretic realizations.

Let us introduce the notation of expectation for random variable X:

$$E_p[X] = \sum_i^k p(X = x_i)x_i \tag{A.1}$$

For the continuous case, if $f_X(x)$ is a properly defined probability density function, the expected value becomes:

$$E_f[X] = \int_{-\infty}^{\infty} x f_X(x) dx \tag{A.2}$$

From the context it can be seen that the object X at the left is a different object than x at the right. The former is random variable (a one-dimensional function, an (in)finite vector), the latter is a value of a random variable (a scalar). The term dx is a measure, in this case it assigns volume to *subsets of random variables values*. A proper notation would incorporate this aspect, but not much will be gained by creating such complex notations.

Let us also introduce the conditional probability:

$$p(X|Y) = \frac{p(X,Y)}{p(Y)} \tag{A.3}$$

Suppose X is a discrete random variable, then the object $p(X)$ is a vector of finite size k, $p(Y)$ is a vector of finite size l, and $p(X|Y)$ as well as $p(X,Y)$ are matrices of size $k \times l$. A conditional probability hence 'divides' a matrix by a vector. It is a proper measure if $p(Y) \neq 0$ (for all values of - or events in - Y).

In for example importance sampling (Sect. **??**), the expectation is taken over a function of a random variable. A function, if measurable, can be taken the expectation over using the so-called law of the unconscious statistician:

$$E_f[g(X)] = \int_{-\infty}^{\infty} g(x) f_X(x) dx \tag{A.4}$$

Here $g(X)$ is a general measurable function, and not restricted to a probability density function.

The notation above is an indefinite integral. We can approach this integral by Monte Carlo integration (Sect. **??**).

$$E_f[g(X)] = \frac{1}{k}\sum_{i=1}^{k} g(x_i) \qquad \text{with} \qquad x_i \sim f_X(x) \tag{A.5}$$

Rather than summing over $f_X(x_i)$, we now sample $x_i \sim f_X(x)$.

A.1 Common Inequalities

A.1.1 Markov's Inequality

Markov's inequality comes up with an upper bound for the probability that an non-negative random variable X exceeds some constant positive threshold a.

$$p(X \geq a) \leq \frac{E[X]}{a} \tag{A.6}$$

The proof in classical probability theory uses an indicator variable:

$$aI(X \geq a) \leq X \tag{A.7}$$

Here $I(X \geq a) = 1$ if the event $X \geq a$ occurs, setting the left-hand side to a (which is of course smaller than X). And $I(X \geq a) = 0$ on the event $X < a$, which is naturally smaller than the non-negative X.

Expectations obey the inequality: if $(X \leq Y)$, then $E[X] \leq E[Y]$, hence:

$$E[aI(X \geq a)] \leq E[X] \tag{A.8}$$

And because expectations add up linearly:

$$E[aI(X \geq a)] = aE[I(X \geq a)] = a(1 \cdot p(X \geq a) + 0 \cdot p(X < a)) = a \cdot p(X \geq a) \tag{A.9}$$

So, we have Markov's inequality combining Eq. A.8 and A.9:

$$a \cdot p(X \geq a) \leq E[X] \tag{A.10}$$

A.1.2 Chebyshev's Inequality

Now, Chebyshev's inequality defines in a simular way (Chebyshev was a teacher of Markov[1]) an upper bound on the deviation from the mean for a random variable. Recall the definition of the variance of X and assume it is finite:

$$Var(X) = E[(X - E[X])^2] = \sigma^2 \tag{A.11}$$

Consider now the random variable $(X - E[X])^2$ and constant $a = (\sigma k)^2$ and write down Markov's inequality:

$$p((X - E[X])^2 \geq (\sigma k)^2) \leq \frac{E[(X - E[X])^2]}{(\sigma k)^2} \tag{A.12}$$

Taking the square root of the unequality at the left, and using the definition of σ^2 at the right, leads to:

$$p(|X - E[X]| \geq \sigma k) \leq \frac{1}{k^2} \tag{A.13}$$

A.1.3 Weak Law of Large Numbers

Chebyshev's inequality can be used to prove the weak law of large numbers. Given that we have a series of random variables, all with the same finite expectation, $E[X_i] = \mu$, then this law states that the sample average $\bar{X} = \frac{1}{n}(X_1 + \cdots + X_n)$ converges in probability towards the expected value:

$$\bar{X} \xrightarrow{P} \mu \qquad \text{for} \qquad n \to \infty \tag{A.14}$$

We can use the independence assumption between variables X_i to write down the variance and expectation of $\bar{X}$:

$$Var(\bar{X}) = Var(\frac{1}{n}(X_1 + \cdots + X_n)) = \frac{1}{n^2} Var(X_1 + \cdots + X_n) = \frac{\sigma^2}{n} E[\bar{X}] = \mu \tag{A.15}$$

[1] There were many mathematically gifted Markov's. This is Andrey Andreyevich Markov Sr., known from the Markov chains and Markov processes. Jr. is known from Markov's principle, Markov's rule and the Markov algorithm.

And now we can apply Chebyshev's inequality on $\bar{X}$:

$$p(|\bar{X}-\mu| \geq \epsilon) \leq \frac{\sigma^2}{n\epsilon^2} \tag{A.16}$$

Convergence in probability towards X is the case if for all ϵ:

$$\lim_{n\to\infty} p(|\bar{X}-X| \geq \epsilon) = 0 \tag{A.17}$$

This is the case for $n \to \infty$ indeed.

A.1.4 Strong Law of Large Numbers

The strong law incorporates the weak law. Rather than convergence *in probability*, it states convergence *almost surely* towards the expected value.

$$\bar{X} \xrightarrow{a.s.} \mu \qquad \text{for} \qquad n \to \infty \tag{A.18}$$

The strong law states that with probability 1, for any $\epsilon > 0$, the inequality $|X-\mu| < \epsilon$ holds for large enough n. The weak law states only that the average $\bar{X}$ is likely near μ, but $|X-\mu| \geq \epsilon$ can still happen, even for large n.

A.1.5 Common Distributions

One of the probability distributions that is interesting to us is the beta-distribution. A normal distribution might be a reasonable prior for a continuous variable such as human heights in a population. If this variable however is itself a probability, a reasonable prior is the beta-distribution. The beta-distribution can be described as:

$$f(x,\alpha,\beta) = \frac{1}{B(\alpha,\beta)} x^{a-1}(1-x)^{\beta-1} = \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{a-1}(1-x)^{\beta-1} \tag{A.19}$$

Here B is the beta function and Γ is the gamma function, the continuous extension of the factorial function: $\Gamma(n) = (n-1)!$. Naturally, there are many of such extensions. The gamma function extends the factorial in a specific sense. It obeys the recurrence relation $f(x+1) = xf(x)$ with $f(1) = 1$. Its description is defined with an improper integral:

$$\Gamma(t) = \int_0^\infty x^{t-1}e^{-x}dx \tag{A.20}$$

The expected value of a random variable X with a beta-distribution:

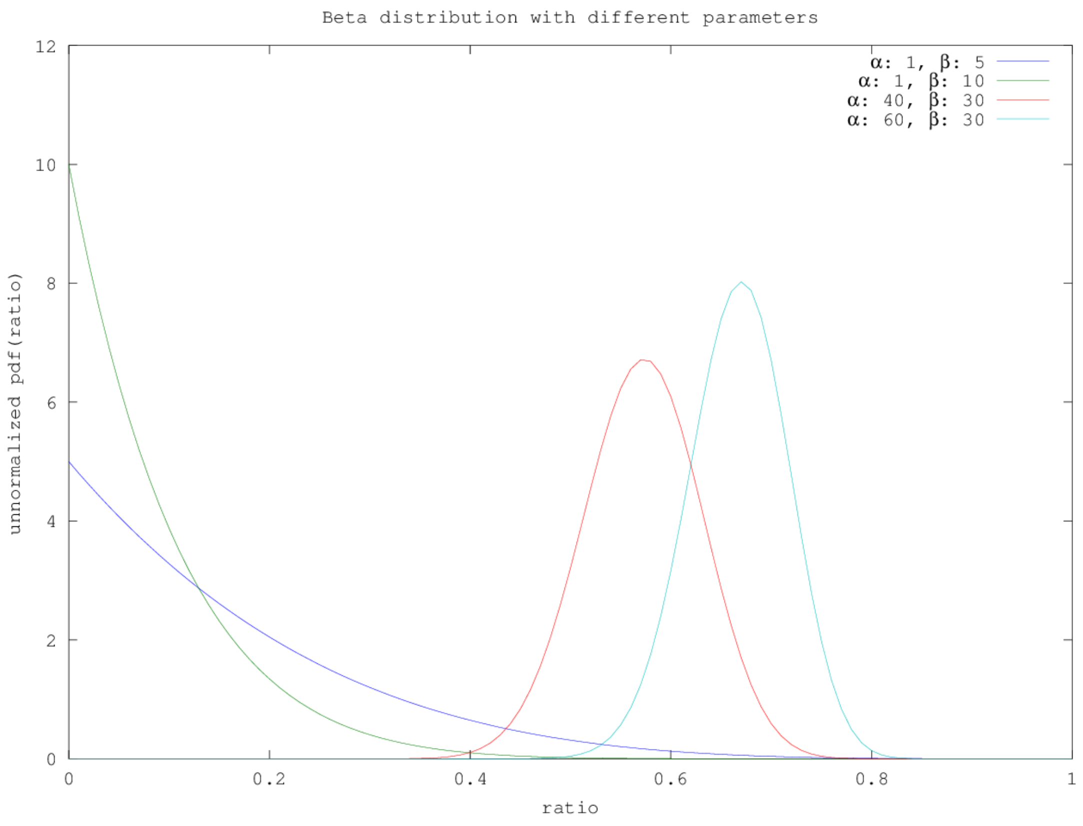


Figure A.1: The beta-distribution with different parameters. The x-axis is the quantity modelled, for example a ratio between wins and losses in a soccer season. The y-axis is the corresponding unnormalized density function. Setting $\alpha = 1$ shows a monotonically decreasing density function. Having a variable α allows probability mass to shift to the end.

$$E_f[X] = \int_0^1 x f(x,\alpha,\beta)dx = \int_0^1 \frac{\Gamma(\alpha+\beta)}{\Gamma(\alpha)\Gamma(\beta)} x^{\alpha}(1-x)^{\beta-1}dx = \frac{\alpha}{\alpha+\beta} \tag{A.21}$$

Let us illustrate the effects of the parameters of the beta-distribution.

The stick-breaking presentation shows that a Dirichlet process consists of beta processes with $\alpha = 1$. The Pitman-Yor process has α left variable. Informally, the location where the stick will be broken (iteratively) for the Dirichlet process is 'quite close to the beginning' with high probability. In the case α becomes larger, the breaking can also occur likely at the end of the stick. This means for breaking a stick, say 20 times, the distribution of stick lengths for the Pitman-Yor process has a much larger support. The difference between the large and small sticks is much larger.

Appendix B

Dirichlet-Multinomial Interpretations

There are multiple interpretations of the compound compound "Dirichlet-multinomial" distribution. We will derive different related compound distributions and show how each of these compound distributions can be used as a likelihood function that is conjugate to the Dirichlet distribution. Let us recall the Dirichlet distribution (Eq. B.1).

$$p(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_i \theta_i^{\alpha_i - 1}. \tag{B.1}$$

The normalizing constant $1/B(\alpha)$ contains the Beta function:

$$B(\alpha) = \frac{\prod_i \Gamma(\alpha_i)}{\Gamma\left(\sum_i \alpha_i\right)}. \tag{B.2}$$

Regarding notation, the Γ function can be written as a factorial $\Gamma(x+1) = x!$. Steps like $\Gamma(1+x) = x\Gamma(x)$ and $\frac{\Gamma(x+n)}{\Gamma(x)} = x^n$ will not be made explicit in the following text.

We would like to find a conjugate distribution for the Dirichlet that will result in a Dirichlet distribution after performing Bayesian inference. The shape of this conjugate distribution is of the form:

$$p(z|\theta) = C \prod_i \theta_i^{z_i}. \tag{B.3}$$

Here C is a normalization constant that depends on the interpretation of the random variable z.

There are three ways to interpret the Dirichlet-multinomial, namely as (1) a compound Dirichlet-categorical distribution (Appendix B.1), (2) a compound Dirichlet-multinomial distribution (Appendix B.2), and as (3) compound Dirichlet with N categorical distributions (Appendix B.3). Each of these interpretations are different. However, if they are used in a Bayesian context, they will all result in a posterior that is a Dirichlet distribution.

B.1 Dirichlet-Categorical

Often, the Dirichlet-multinomial is actually not a compound Dirichlet and *multinomial* distribution, but a compound Dirichlet and *categorical* distribution. Informally, the categorical distribution is a distribution over categories. More formally, it is a generalization of a Bernoulli distribution that assigns probabilities to a random variable with two possible outcomes to a random variable with more than two possible outcomes. The categorical distribution has the form as in Eq. B.4.

$$p(z|\theta) = \prod_i \theta_i^{z_i}. \tag{B.4}$$

This notation defines only *a single* categorical variable, not a set of variables. It is using a 1-of-K encoded representation (cf. Bishop, 2016). Assume a regular 6-faced dice. The face with one pip is represented by $z_0 = [1,0,0,0,0,0]$. The face with two pips $z_1 = [0,1,0,0,0,0]$. There is only one category non-zero, $\sum_j z_{i,j} = 1$. The probabilities of each category are represented by θ_i and are one sixth for a regular dice.

Let us take the product of the Dirichlet distribution in Eq. B.1 and the categorical distribution in Eq. B.4:

$$p(z|\theta)p(\theta|\alpha) = \frac{1}{B(\alpha)} \prod_i \theta_i^{z_i+\alpha_i-1}. \tag{B.5}$$

The compound distribution is created by integrating over θ:

$$p(z|\alpha) = \int p(z|\theta)p(\theta|\alpha)d\theta = \frac{1}{B(\alpha)} \int \prod_i \theta_i^{z_i+\alpha_i-1} d\theta. \tag{B.6}$$

The multivariate Beta function (Eq. B.2) stems from normalizing the Dirichlet distribution and can also be written as an integral over the unnormalized Dirichlet distribution:

$$B(\alpha) = \int \prod_i \theta_i^{\alpha_i-1} d\theta. \tag{B.7}$$

Now we can use the 1-of-K representation to write the integral as follows:

$$\int \prod_i \theta_i^{z_i+\alpha_i-1} d\theta = B(\alpha+z). \tag{B.8}$$

The compound distribution of the Dirichlet-categorical henceforth can be written as:

$$p(z|\alpha) = \int p(z|\theta)p(\theta|\alpha)d\theta = \frac{B(\alpha+z)}{B(\alpha)}. \tag{B.9}$$

We can now perform Bayes' rule that shows the benefit of the closed-form description of the compound distribution in the conjugate context:

$$p(\theta|z,\alpha) = \frac{p(z|\theta,\alpha)p(\theta|\alpha)}{p(z|\alpha)} \tag{B.10}$$

In this case the likelihood $p(z|\theta,\alpha) = p(z|\theta)$ has observations z that only depend on θ (and θ depends on α).

$$p(\theta|z,\alpha) = \frac{p(z|\theta)p(\theta|\alpha)}{p(z|\alpha)} \tag{B.11}$$

Hence the posterior of the Dirichlet prior with the categorical likelihood is a Dirichlet posterior (use Eq. B.5 and Eq. B.9 in Eq. B.11).

$$p(\theta|z,\alpha) = \frac{1}{B(\alpha+z)} \prod_i \theta_i^{z_i+\alpha_i-1}. \tag{B.12}$$

Note again that we run i over the entries in our categorical variable z represented as a vector. This is different from a multinomial distribution over a set of variables!

B.2 Dirichlet-Multinomial Distribution

In case of an actual multinomial distribution, the topic of consideration are *counts* of z. Let's write the counts as $n(z)$.

$$p(z|\theta) = \frac{(\sum_k n(z_k))!}{\prod_k (n(z_k)!)} \prod_k \theta_k^{n(z_k)}. \tag{B.13}$$

We now run over k unique variables, not over a vectorized categorical variable. The following example clarifies the multinomial. Suppose we have a bag with k colored balls. The multinomial is the random variable that assigns a probability to the vector $[n(z_0), \ldots, n(z_{k-1})]$ with $n(z_i)$ the number of balls of color i. The distribution also represents the task of throwing a k-faced dice n times and counting the number of times $n(z)$ category z is occuring.

The product of the Dirichlet distribution with the multinomial can be performed along the same lines as in Appendix B.1.

$$p(z|\theta)p(\theta|\alpha) = \frac{\sum_k n(z_k)\Gamma\left(\sum_k n(z_k)\right)}{\prod_k n(z_k)\prod_k \Gamma(n(z_k))} \frac{1}{B(\alpha)} \prod_k \theta_k^{n(z_k)+\alpha_k-1}. \tag{B.14}$$

This we can simplify to:

$$p(z|\theta)p(\theta|\alpha) = \frac{\sum_k n(z_k)}{\prod_k n(z_k)} \frac{1}{B(n(z_k))} \frac{1}{B(\alpha)} \prod_k \theta_k^{n(z_k)+\alpha_k-1}. \tag{B.15}$$

We will get the following result for the compound distribution (using Eq. B.7 with the Beta function as known result for the integral):

$$p(z|\alpha) = \int p(z|\theta)p(\theta|\alpha)d\theta = \frac{\sum_k n(z_k)}{\prod_k n(z_k)} \frac{1}{B(n(z_k))} \frac{1}{B(\alpha)} B(\alpha + n(z_k)). \tag{B.16}$$

The Dirichlet prior with the multinomial likelihood results in a Dirichlet posterior as well:

$$p(\theta|z,\alpha) = \frac{p(z|\theta)p(\theta|\alpha)}{p(z|\alpha)} = \frac{1}{B(\alpha + n(z_k))} \prod_k \theta_k^{n(z_k)+\alpha_k-1}. \tag{B.17}$$

B.3 Dirichlet-N Categorical Distributions

The third interpretation of a Dirichlet-multinomial distribution is the one in which the term "multinomial" refers to a distribution of a *sequence* of categorical variables. Recall that the multinomial assigns probabilities to the *number* of extracted balls (in an experiment getting n balls out of a bag with k ball types). A sequence of categorical variables has a form for its probability distribution that has no normalization factor:

$$p(z|\theta) = \prod_k \theta_k^{z_k}. \tag{B.18}$$

Here k runs over the categories. Note that this form is exactly the same as that of Eq. B.4. The compound distribution will henceforth have the same form as the Dirichlet-categorical distribution:

$$p(z|\alpha) = \int p(z|\theta)p(\theta|\alpha)d\theta = \frac{B(\alpha + z)}{B(\alpha)}. \tag{B.19}$$

The Dirichlet prior with the n-categorical distribution as likelihood results in a Dirichlet posterior:

$$p(\theta|z,\alpha) = \frac{p(z|\theta)p(\theta|\alpha)}{p(z|\alpha)} = \frac{1}{B(\alpha+z)} \prod_k \theta_k^{z_k+\alpha_k-1}. \tag{B.20}$$

APPENDIX C

GIBBS SAMPLING

Notation:

$$\int dF(x) = F(x) \tag{C.1}$$

A mixture model:

$$L(x) = \int dF(x)\mu(x) \tag{C.2}$$

If we have a particular form of $F(x)$, namely it admits a decomposition of a sum of individual values x_i:

$$F(x) = \sum_i \delta_{x_i} = \delta(x = x_0) + \delta(x = x_1) + \dots \tag{C.3}$$

Then our mixture model can be written as:

$$L(x) = \int dF(x)\mu(x) = \sum_i \mu(x_i) \tag{C.4}$$

Let $x_0 = 3$, $x_1 = 4$, $\mu(x) = x^2$, then $L(x) = 3^2 + 4^2 = 25$.

Walker with $P = F$, $\mu(x) = N(y|\theta)$, $i = j$ and giving each θ_j a weight ω_j:

$$f_P(y) = \int dP(\theta)N(y|\theta) \qquad P = \sum_j \omega_j \delta_{\theta_j} \tag{C.5}$$

Then:

$$f_{\omega,j}(y) = \sum_j \omega_j N(y|\theta_j) \tag{C.6}$$

The likelihood:

$$L(w \mid \alpha, \lambda_0) = p(\phi \mid \alpha) \prod_{i=0}^{N-1} \int p(w_i \mid \theta_i, \phi) dG_0(\theta_i) \tag{C.7}$$

Here the index runs over all data points w_i. Each data point corresponds to a line with parameters θ_i. Here the parameters θ_i and θ_j for data point w_i and w_j can be the same and thus reflect the same line. The index for θ runs over the N data points, not over the K lines.

The distribution G_0 does have hyperparameters λ_0.

We can also group all data points that belong to the same line k together by reordering the product terms:

$$L(w \mid \alpha, \lambda_0) = p(\phi \mid \alpha) \prod_k \prod_{i:z_i=k} \int p(w_i \mid \theta_i, \phi) dG_0(\theta_i) \tag{C.8}$$

Here the factors that belong to line k are multiplied. The index still runs over the data points.

It is also possible not to limit the second product to only the data points i that are assigned to line k.

$$L(w \mid \alpha, \lambda_0) = p(\phi \mid \alpha) \prod_k \prod_i p(z_i|\phi) \int p(w_i \mid \theta_i, \phi) dG_0(\theta_i) \tag{C.9}$$

Now, we are gonna introduce the stick-breaking sum, which turns our integral into a discrete sum.

$G = \sum_l p_l \delta_{Z_l}$ with Z_l iid from G_0 and p_l defined as a product of beta distributions.

$$L(w \mid \alpha, \lambda_0) = p(\phi \mid \alpha) \sum_k \prod_i p(z_i \mid \phi) p(w_i \mid \theta_k) p(\theta_k \mid \lambda_0) \tag{C.10}$$

The first term at the right hand side, $p(\phi \mid \alpha)$, generates the partition ϕ by the Dirichlet process with concentration parameter α. The second term $p(z_i \mid \phi)$ defines indices $z_0, \ldots, z_N$ to link observations $w_0, \ldots, w_N$ with the parameters $\theta_0, \ldots, \theta_K$. The probability $p(w_i \mid \theta_k)$ corresponds to the likelihood equations 3.10 and 3.11 with w_i the tuple of x_i and y_i and θ_k the line parameters σ_k^2 and β_k. The probability $p(\theta_k \mid \lambda_0)$ corresponds to the prior from equation 3.14. The parameters θ_k (that is, σ_k^2 and β_k) are generated from hyperparameters λ_0. The hyperparameters $\lambda_0 = \{\mu_0, \Lambda_0, a, b\}$ are the parameters from the Normal-Inverse-Gamma prior.

The Dirichlet process can be used as a mixture model (Antoniak, 1974; Escobar and West, 1995; MacEachern and Müller, 1998) in which it generates (non-unique) parameters that subsequently generate observations:

$$\begin{aligned} G &\sim DP(\alpha, G_0) \\ \theta_i \mid G &\sim G \\ w_i \mid \theta_i &\sim F(\theta_i) \end{aligned} \tag{C.11}$$

Here F describes the mapping from parameters θ_i to observations w_i. It is possible to integrate over G and sample the parameters directly from the base distribution G_0.

It is possible to integrate over G and get a description in the form of conditionals over the parameters (Blackwell and MacQueen, 1973):

$$\theta_{n+1} \mid \theta_1 \dots \theta_{n-1} \sim \frac{1}{\alpha+n}(\alpha G_0 + \sum_{i=1}^{n} \delta_{\theta_i}) \tag{C.12}$$

C.1 Gibbs Sampling of Parameters

Algorithm, we will draw $\theta_i|\theta_{-i}, y_i$ for all i.

And that continuously. So, that's how we get theta.

Gibbs sampling requires the conditional probabilities of all entities involved (Geman and Geman, 1984). Gibbs sampling just as other Markov chain Monte Carlo methods generates a sequence of correlated samples. Subsequently, if necessary, the Maximum A Posteriori estimation of a value can be found through picking the mode (most common occurring value) of a parameter.

The derivation of the conditional probabilities of parameters with respect to the remaining parameters has been described in the literature (Neal, 2000). Such a derivation uses an important property of the Dirichlet process, namely that it is the conjugate prior of the multinomial distribution. Thanks to conjugacy the following equations have closed-form descriptions. The conditional probabilities are sampled from the base distribution G_0 and the other parameters θ_i in the following way:

$$\theta_{n+1} \mid \theta_1 \dots \theta_{n-1} \sim \frac{1}{\alpha+n}(\alpha G_0 + \sum_{i=1}^{n} \delta_{\theta_i}) \tag{C.13}$$

If we include the observations themselves, we need to include the likelihood as well:

$$\theta_i \mid \theta_{-i}, w_i \sim C \left\{ \sum_{j,j\neq i} F(w_i, \theta_j)\delta_{\theta_j} + \alpha H_i \int F(w_i, \theta) dG_0(\theta) \right\} \tag{C.14}$$

The constant C is a normalization factor to make the above a proper probability density (summing to one). The entity H_i is the posterior density of θ given G_0 as prior and y_i as observation. The notation θ_{-i} describes the set of all parameters Θ with θ_i excluded. The integral over $dG_0(\theta)$ is a Lebesgue-Stieltjes integral that weighs the contribution of $F(w_i, \theta)$ with the base distribution $G_0(\theta)$.

Equation C.14 can be used to perform inference directly with all (non-unique) parameters θ_i tied to observations w_i. Details on inference will be provided in Sect. **??**.

Appendix D

Glossary

burn-in running a Markov chain Monte Carlo for a while before starting to sample from it, so the results are not depending on its initial random starting position. 43

collapsed Gibbs sampling Rao-Blackwellized Gibbs sampling. Certain sampling steps are replaced by steps where one or more variables are integrated out. This is can be thanks to analytic descriptions that arise from the use of conjugate priors. 43

slow mixing a high degree of correlation between subsequent samples in a Markov chain leading to a long time before the chain converges. 57

List of Figures

List of Tables

Acronyms

ABC	approximate Bayesian computation
a.s.	almost surely
BP	Beta process
CRP	Chinese restaurant process
DMM	Dirichlet mixture model
DP	Dirichlet process
GP	Gamma process
GEM	Griffiths, Engen, and McCloskey
HDP	hierarchical Dirichlet process
IBP	Indian buffet process
ILM	infinite line model
MAP	maximum a posteriori
MCMC	Markov chain Monte Carlo
ML	maximum likelihood
MLL	maximum log-likelihood
NB	naive Bayes
NIG	Normal-Inverse-Gamma
PYP	Pitman-Yor process
SAMS	sequentially allocated merge-split

Summary

Summary....

Samenvatting

Samenvatting...

Acknowledgments

I want to thank...

Curriculum Vitae

Anne van Rossum ....

Publications

The investigations performed during my Ph.D. research resulted in the following publications.

- A.C. van Rossum. ....

SIKS Dissertation Series

1998

1 Johan van den Akker (CWI[1]) *DEGAS - An Active, Temporal Database of Autonomous Objects*

2 Floris Wiesman (UM) *Information Retrieval by Graphically Browsing Meta-Information*

3 Ans Steuten (TUD) *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective*

4 Dennis Breuker (UM) *Memory versus Search in Games*

5 Eduard W. Oskamp (RUL) *Computerondersteuning bij Straftoemeting*

1999

1 Mark Sloof (VU) *Physiology of Quality Change Modelling; Automated Modelling of Quality Change of Agricultural Products*

2 Rob Potharst (EUR) *Classification using Decision Trees and Neural Nets*

3 Don Beal (UM) *The Nature of Minimax Search*

4 Jacques Penders (UM) *The Practical Art of Moving Physical Objects*

5 Aldo de Moor (KUB) *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems*

6 Niek J.E. Wijngaards (VU) *Re-Design of Compositional Systems*

7 David Spelt (UT) *Verification Support for Object Database Design*

8 Jacques H.J. Lenting (UM) *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation*

2000

1 Frank Niessink (VU) *Perspectives on Improving Software Maintenance*

2 Koen Holtman (TU/e) *Prototyping of CMS Storage Management*

3 Carolien M.T. Metselaar (UvA) *Sociaal-organisatorische Gevolgen van Kennistechnologie; een Procesbenadering en Actorperspectief*

4 Geert de Haan (VU) *ETAG, A Formal Model of Competence Knowledge for User Interface Design*

5 Ruud van der Pol (UM) *Knowledge-Based Query Formulation in Information Retrieval*

6 Rogier van Eijk (UU) *Programming Languages for Agent Communication*

7 Niels Peek (UU) *Decision-Theoretic Planning of Clinical Patient Management*

8 Veerle Coupé (EUR) *Sensitivity Analyis of Decision-Theoretic Networks*

9 Florian Waas (CWI) *Principles of Probabilistic Query Optimization*

10 Niels Nes (CWI) *Image Database Management System Design Considerations, Algorithms and Architecture*

11 Jonas Karlsson (CWI) *Scalable Distributed Data Structures for Database Management*

2001

1 Silja Renooij (UU) *Qualitative Approaches to Quantifying Probabilistic Networks*

2 Koen Hindriks (UU) *Agent Programming Languages: Programming with Mental Models*

[1]Abbreviations: SIKS - Dutch Research School for Information and Knowledge Systems; CWI - Centrum voor Wiskunde en Informatica, Amsterdam; EUR - Erasmus Universiteit, Rotterdam; KUB - Katholieke Universiteit Brabant, Tilburg; KUN - Katholieke Universiteit Nijmegen; OU - Open Universiteit; RUL - Rijksuniversiteit Leiden; RUN - Radboud Universiteit Nijmegen; TUD - Technische Universiteit Delft; TU/e - Technische Universiteit Eindhoven; UL - Universiteit Leiden; UM - Universiteit Maastricht; UT - Universiteit Twente, Enschede; UU - Universiteit Utrecht; UvA - Universiteit van Amsterdam; UvT - Universiteit van Tilburg; VU - Vrije Universiteit, Amsterdam.

3 Maarten van Someren (UvA) *Learning as Problem Solving*

4 Evgueni Smirnov (UM) *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets*

5 Jacco van Ossenbruggen (VU) *Processing Structured Hypermedia: A Matter of Style*

6 Martijn van Welie (VU) *Task-Based User Interface Design*

7 Bastiaan Schonhage (VU) *Diva: Architectural Perspectives on Information Visualization*

8 Pascal van Eck (VU) *A Compositional Semantic Structure for Multi-Agent Systems Dynamics*

9 Pieter Jan 't Hoen (RUL) *Towards Distributed Development of Large Object-Oriented Models, Views of Packages as Classes*

10 Maarten Sierhuis (UvA) *Modeling and Simulating Work Practice BRAHMS: a Multiagent Modeling and Simulation Language for Work Practice Analysis and Design*

11 Tom M. van Engers (VU) *Knowledge Management: The Role of Mental Models in Business Systems Design*

2002

1 Nico Lassing (VU) *Architecture-Level Modifiability Analysis*

2 Roelof van Zwol (UT) *Modelling and Searching Web-based Document Collections*

3 Henk Ernst Blok (UT) *Database Optimization Aspects for Information Retrieval*

4 Juan Roberto Castelo Valdueza (UU) *The Discrete Acyclic Digraph Markov Model in Data Mining*

5 Radu Serban (VU) *The Private Cyberspace Modeling Electronic Environments Inhabited by Privacy-Concerned Agents*

6 Laurens Mommers (UL) *Applied Legal Epistemology; Building a Knowledge-based Ontology of the Legal Domain*

7 Peter Boncz (CWI) *Monet: A Next-Generation DBMS Kernel For Query-Intensive Applications*

8 Jaap Gordijn (VU) *Value Based Requirements Engineering: Exploring Innovative E-Commerce Ideas*

9 Willem-Jan van den Heuvel (KUB) *Integrating Modern Business Applications with Objectified Legacy Systems*

10 Brian Sheppard (UM) *Towards Perfect Play of Scrabble*

11 Wouter C.A. Wijngaards (VU) *Agent Based Modelling of Dynamics: Biological and Organisational Applications*

12 Albrecht Schmidt (UvA) *Processing XML in Database Systems*

13 Hongjing Wu (TU/e) *A Reference Architecture for Adaptive Hypermedia Applications*

14 Wieke de Vries (UU) *Agent Interaction: Abstract Approaches to Modelling, Programming and Verifying Multi-Agent Systems*

15 Rik Eshuis (UT) *Semantics and Verification of UML Activity Diagrams for Workflow Modelling*

16 Pieter van Langen (VU) *The Anatomy of Design: Foundations, Models and Applications*

17 Stefan Manegold (UvA) *Understanding, Modeling, and Improving Main-Memory Database Performance*

2003

1 Heiner Stuckenschmidt (VU) *Ontology-Based Information Sharing in Weakly Structured Environments*

2 Jan Broersen (VU) *Modal Action Logics for Reasoning About Reactive Systems*

3 Martijn Schuemie (TUD) *Human-Computer Interaction and Presence in Virtual Reality Exposure Therapy*

4 Milan Petkovic (UT) *Content-Based Video Retrieval Supported by Database Technology*

5 Jos Lehmann (UvA) *Causation in Artificial Intelligence and Law – A Modelling Approach*

6 Boris van Schooten (UT) *Development and Specification of Virtual Environments*

7 Machiel Jansen (UvA) *Formal Explorations of Knowledge Intensive Tasks*

8 Yong-Ping Ran (UM) *Repair-Based Scheduling*

9 Rens Kortmann (UM) *The Resolution of Visually Guided Behaviour*

10 Andreas Lincke (UT) *Electronic Business Negotiation: Some Experimental Studies on the Interaction between Medium, Innovation Context and Cult*

11 Simon Keizer (UT) *Reasoning under Uncertainty in Natural Language Dialogue using Bayesian Networks*

12 Roeland Ordelman (UT) *Dutch Speech Recognition in Multimedia Information Retrieval*

13 Jeroen Donkers (UM) *Nosce Hostem – Searching with Opponent Models*

14 Stijn Hoppenbrouwers (KUN) *Freezing Language: Conceptualisation Processes across ICT-Supported Organisations*

15 Mathijs de Weerdt (TUD) *Plan Merging in Multi-Agent Systems*

16 Menzo Windhouwer (CWI) *Feature Grammar Systems - Incremental Maintenance of Indexes to Digital Media Warehouse*

17 David Jansen (UT) *Extensions of Statecharts with Probability, Time, and Stochastic Timing*

18 Levente Kocsis (UM) *Learning Search Decisions*

2004

1 Virginia Dignum (UU) *A Model for Organizational Interaction: Based on Agents, Founded in Logic*

2 Lai Xu (UvT) *Monitoring Multi-party Contracts for E-business*

3 Perry Groot (VU) *A Theoretical and Empirical Analysis of Approximation in Symbolic Problem Solving*

4 Chris van Aart (UvA) *Organizational Principles for Multi-Agent Architectures*

5 Viara Popova (EUR) *Knowledge Discovery and Monotonicity*

6 Bart-Jan Hommes (TUD) *The Evaluation of Business Process Modeling Techniques*

7 Elise Boltjes (UM) *Voorbeeld$_{IG}$ Onderwijs; Voorbeeldgestuurd Onderwijs, een Opstap naar Abstract Denken, vooral voor Meisjes*

8 Joop Verbeek (UM) *Politie en de Nieuwe Internationale Informatiemarkt, Grensregionale Politiële Gegevensuitwisseling en Digitale Expertise*

9 Martin Caminada (VU) *For the Sake of the Argument; Explorations into Argument-based Reasoning*

10 Suzanne Kabel (UvA) *Knowledge-rich Indexing of Learning-objects*

11 Michel Klein (VU) *Change Management for Distributed Ontologies*

12 The Duy Bui (UT) *Creating Emotions and Facial Expressions for Embodied Agents*

13 Wojciech Jamroga (UT) *Using Multiple Models of Reality: On Agents who Know how to Play*

14 Paul Harrenstein (UU) *Logic in Conflict. Logical Explorations in Strategic Equilibrium*

15 Arno Knobbe (UU) *Multi-Relational Data Mining*

16 Federico Divina (VU) *Hybrid Genetic Relational Search for Inductive Learning*

17 Mark Winands (UM) *Informed Search in Complex Games*

18 Vania Bessa Machado (UvA) *Supporting the Construction of Qualitative Knowledge Models*

19 Thijs Westerveld (UT) *Using generative probabilistic models for multimedia retrieval*

20 Madelon Evers (Nyenrode) *Learning from Design: facilitating multidisciplinary design teams*

2005

1 Floor Verdenius (UvA) *Methodological Aspects of Designing Induction-Based Applications*

2 Erik van der Werf (UM) *AI techniques for the game of Go*

3 Franc Grootjen (RUN) *A Pragmatic Approach to the Conceptualisation of Language*

4 Nirvana Meratnia (UT) *Towards Database Support for Moving Object data*

5 Gabriel Infante-Lopez (UvA) *Two-Level Probabilistic Grammars for Natural Language Parsing*

6 Pieter Spronck (UM) *Adaptive Game AI*

7 Flavius Frasincar (TU/e) *Hypermedia Presentation Generation for Semantic Web Information Systems*

8 Richard Vdovjak (TU/e) *A Model-driven Approach for Building Distributed Ontology-based Web Applications*

9 Jeen Broekstra (VU) *Storage, Querying and Inferencing for Semantic Web Languages*

10 Anders Bouwer (UvA) *Explaining Behaviour: Using Qualitative Simulation in Interactive Learning Environments*

11 Elth Ogston (VU) *Agent Based Matchmaking and Clustering - A Decentralized Approach to Search*

12 Csaba Boer (EUR) *Distributed Simulation in Industry*

13 Fred Hamburg (UL) *Een Computermodel voor het Ondersteunen van Euthanasiebeslissingen*

14 Borys Omelayenko (VU) *Web-Service configuration on the Semantic Web; Exploring how semantics meets pragmatics*

15 Tibor Bosse (VU) *Analysis of the Dynamics of Cognitive Processes*

16 Joris Graaumans (UU) *Usability of XML Query Languages*

17 Boris Shishkov (TUD) *Software Specification Based on Re-usable Business Components*

18 Danielle Sent (UU) *Test-selection strategies for probabilistic networks*

19 Michel van Dartel (UM) *Situated Representation*

20 Cristina Coteanu (UL) *Cyber Consumer Law, State of the Art and Perspectives*

21 Wijnand Derks (UT) *Improving Concurrency and Recovery in Database Systems by Exploiting Application Semantics*

2006

1 Samuil Angelov (TU/e) *Foundations of B2B Electronic Contracting*

2 Cristina Chisalita (VU) *Contextual issues in the design and use of information technology in organizations*

3 Noor Christoph (UvA) *The role of metacognitive skills in learning to solve problems*

4 Marta Sabou (VU) *Building Web Service Ontologies*

5 Cees Pierik (UU) *Validation Techniques for Object-Oriented Proof Outlines*

6 Ziv Baida (VU) *Software-aided Service Bundling - Intelligent Methods & Tools for Graphical Service Modeling*

7 Marko Smiljanic (UT) *XML schema matching – balancing efficiency and effectiveness by means of clustering*

8 Eelco Herder (UT) *Forward, Back and Home Again - Analyzing User Behavior on the Web*

9 Mohamed Wahdan (UM) *Automatic Formulation of the Auditor's Opinion*

10 Ronny Siebes (VU) *Semantic Routing in Peer-to-Peer Systems*

11 Joeri van Ruth (UT) *Flattening Queries over Nested Data Types*

12 Bert Bongers (VU) *Interactivation - Towards an e-cology of people, our technological environment, and the arts*

13 Henk-Jan Lebbink (UU) *Dialogue and Decision Games for Information Exchanging Agents*

14 Johan Hoorn (VU) *Software Requirements: Update, Upgrade, Redesign - towards a Theory of Requirements Change*
15 Rainer Malik (UU) *CONAN: Text Mining in the Biomedical Domain*
16 Carsten Riggelsen (UU) *Approximation Methods for Efficient Learning of Bayesian Networks*
17 Stacey Nagata (UU) *User Assistance for Multitasking with Interruptions on a Mobile Device*
18 Valentin Zhizhkun (UvA) *Graph transformation for Natural Language Processing*
19 Birna van Riemsdijk (UU) *Cognitive Agent Programming: A Semantic Approach*
20 Marina Velikova (UvT) *Monotone models for prediction in data mining*
21 Bas van Gils (RUN) *Aptness on the Web*
22 Paul de Vrieze (RUN) *Fundaments of Adaptive Personalisation*
23 Ion Juvina (UU) *Development of Cognitive Model for Navigating on the Web*
24 Laura Hollink (VU) *Semantic Annotation for Retrieval of Visual Resources*
25 Madalina Drugan (UU) *Conditional log-likelihood MDL and Evolutionary MCMC*
26 Vojkan Mihajlovic (UT) *Score Region Algebra: A Flexible Framework for Structured Information Retrieval*
27 Stefano Bocconi (CWI) *Vox Populi: generating video documentaries from semantically annotated media repositories*
28 Borkur Sigurbjornsson (UvA) *Focused Information Access using XML Element Retrieval*

2007

1 Kees Leune (UvT) *Access Control and Service-Oriented Architectures*
2 Wouter Teepe (RUG) *Reconciling Information Exchange and Confidentiality: A Formal Approach*
3 Peter Mika (VU) *Social Networks and the Semantic Web*
4 Jurriaan van Diggelen (UU) *Achieving Semantic Interoperability in Multi-agent Systems: a dialogue-based approach*
5 Bart Schermer (UL) *Software Agents, Surveillance, and the Right to Privacy: a Legislative Framework for Agent-enabled Surveillance*
6 Gilad Mishne (UvA) *Applied Text Analytics for Blogs*
7 Natasa Jovanovic' (UT) *To Whom It May Concern - Addressee Identification in Face-to-Face Meetings*
8 Mark Hoogendoorn (VU) *Modeling of Change in Multi-Agent Organizations*
9 David Mobach (VU) *Agent-Based Mediated Service Negotiation*
10 Huib Aldewereld (UU) *Autonomy vs. Conformity: an Institutional Perspective on Norms and Protocols*
11 Natalia Stash (TU/e) *Incorporating Cognitive/Learning Styles in a General-Purpose Adaptive Hypermedia System*
12 Marcel van Gerven (RUN) *Bayesian Networks for Clinical Decision Support: A Rational Approach to Dynamic Decision-Making under Uncertainty*
13 Rutger Rienks (UT) *Meetings in Smart Environments; Implications of Progressing Technology*
14 Niek Bergboer (UM) *Context-Based Image Analysis*
15 Joyca Lacroix (UM) *NIM: a Situated Computational Memory Model*
16 Davide Grossi (UU) *Designing Invisible Handcuffs. Formal investigations in Institutions and Organizations for Multi-agent Systems*
17 Theodore Charitos (UU) *Reasoning with Dynamic Networks in Practice*
18 Bart Orriens (UvT) *On the development and management of adaptive business collaborations*
19 David Levy (UM) *Intimate relationships with artificial partners*
20 Slinger Jansen (UU) *Customer Configuration Updating in a Software Supply Network*
21 Karianne Vermaas (UU) *Fast diffusion and broadening use: A research on residential adoption and usage of broadband internet in the Netherlands between 2001 and 2005*
22 Zlatko Zlatev (UT) *Goal-oriented design of value and process models from patterns*
23 Peter Barna (TU/e) *Specification of Application Logic in Web Information Systems*
24 Georgina Ramírez Camps (CWI) *Structural Features in XML Retrieval*
25 Joost Schalken (VU) *Empirical Investigations in Software Process Improvement*

2008

1 Katalin Boer-Sorbán (EUR) *Agent-Based Simulation of Financial Markets: A modular, continuous-time approach*
2 Alexei Sharpanskykh (VU) *On Computer-Aided Methods for Modeling and Analysis of Organizations*
3 Vera Hollink (UvA) *Optimizing hierarchical menus: a usage-based approach*
4 Ander de Keijzer (UT) *Management of Uncertain Data - towards unattended integration*
5 Bela Mutschler (UT) *Modeling and simulating causal dependencies on process-aware information systems from a cost perspective*
6 Arjen Hommersom (RUN) *On the Application of Formal Methods to Clinical Guidelines, an Artificial Intelligence Perspective*
7 Peter van Rosmalen (OU) *Supporting the tutor in the design and support of adaptive e-learning*
8 Janneke Bolt (UU) *Bayesian Networks: Aspects of Approximate Inference*
9 Christof van Nimwegen (UU) *The paradox of the guided user: assistance can be counter-effective*

10 Wauter Bosma (UT) *Discourse oriented Summarization*

11 Vera Kartseva (VU) *Designing Controls for Network Organizations: a Value-Based Approach*

12 Jozsef Farkas (RUN) *A Semiotically oriented Cognitive Model of Knowledge Representation*

13 Caterina Carraciolo (UvA) *Topic Driven Access to Scientific Handbooks*

14 Arthur van Bunningen (UT) *Context-Aware Querying; Better Answers with Less Effort*

15 Martijn van Otterlo (UT) *The Logic of Adaptive Behavior: Knowledge Representation and Algorithms for the Markov Decision Process Framework in First-Order Domains*

16 Henriette van Vugt (VU) *Embodied Agents from a User's Perspective*

17 Martin Op't Land (TUD) *Applying Architecture and Ontology to the Splitting and Allying of Enterprises*

18 Guido de Croon (UM) *Adaptive Active Vision*

19 Henning Rode (UT) *From document to entity retrieval: improving precision and performance of focused text search*

20 Rex Arendsen (UvA) *Geen bericht, goed bericht. Een onderzoek naar de effecten van de introductie van elektronisch berichtenverkeer met een overheid op de administratieve lasten van bedrijven*

21 Krisztian Balog (UvA) *People search in the enterprise*

22 Henk Koning (UU) *Communication of IT-architecture*

23 Stefan Visscher (UU) *Bayesian network models for the management of ventilator-associated pneumonia*

24 Zharko Aleksovski (VU) *Using background knowledge in ontology matching*

25 Geert Jonker (UU) *Efficient and Equitable exchange in air traffic management plan repair using spender-signed currency*

26 Marijn Huijbregts (UT) *Segmentation, diarization and speech transcription: surprise data unraveled*

27 Hubert Vogten (OU) *Design and implementation strategies for IMS learning design*

28 Ildiko Flesh (RUN) *On the use of independence relations in Bayesian networks*

29 Dennis Reidsma (UT) *Annotations and subjective machines- Of annotators, embodied agents, users, and other humans*

30 Wouter van Atteveldt (VU) *Semantic network analysis: techniques for extracting, representing and querying media content*

31 Loes Braun (UM) *Pro-active medical information retrieval*

32 Trung B. Hui (UT) *Toward affective dialogue management using partially observable markov decision processes*

33 Frank Terpstra (UvA) *Scientific workflow design; theoretical and practical issues*

34 Jeroen de Knijf (UU) *Studies in Frequent Tree Mining*

35 Benjamin Torben-Nielsen (UvT) *Dendritic morphology: function shapes structure*

2009

1 Rasa Jurgelenaite (RUN) *Symmetric Causal Independence Models*

2 Willem Robert van Hage (VU) *Evaluating Ontology-Alignment Techniques*

3 Hans Stol (UvT) *A Framework for Evidence-based Policy Making Using IT*

4 Josephine Nabukenya (RUN) *Improving the Quality of Organisational Policy Making using Collaboration Engineering*

5 Sietse Overbeek (RUN) *Bridging Supply and Demand for Knowledge Intensive Tasks - Based on Knowledge, Cognition, and Quality*

6 Muhammad Subianto (UU) *Understanding Classification*

7 Ronald Poppe (UT) *Discriminative Vision-Based Recovery and Recognition of Human Motion*

8 Volker Nannen (VU) *Evolutionary Agent-Based Policy Analysis in Dynamic Environments*

9 Benjamin Kanagwa (RUN) *Design, Discovery and Construction of Service-oriented Systems*

10 Jan Wielemaker (UvA) *Logic programming for knowledge-intensive interactive applications*

11 Alexander Boer (UvA) *Legal Theory, Sources of Law & the Semantic Web*

12 Peter Massuthe (TU/e, Humboldt-Universtät zu Berlin) *Operating Guidelines for Services*

13 Steven de Jong (UM) *Fairness in Multi-Agent Systems*

14 Maksym Korotkiy (VU) *From ontology-enabled services to service-enabled ontologies (making ontologies work in e-science with ONTO-SOA)*

15 Rinke Hoekstra (UvA) *Ontology Representation - Design Patterns and Ontologies that Make Sense*

16 Fritz Reul (UvT) *New Architectures in Computer Chess*

17 Laurens van der Maaten (UvT) *Feature Extraction from Visual Data*

18 Fabian Groffen (CWI) *Armada, An Evolving Database System*

19 Valentin Robu (CWI) *Modeling Preferences, Strategic Reasoning and Collaboration in Agent-Mediated Electronic Markets*

20 Bob van der Vecht (UU) *Adjustable Autonomy: Controling Influences on Decision Making*

21 Stijn Vanderlooy (UM) *Ranking and Reliable Classification*

22 Pavel Serdyukov (UT) *Search For Expertise: Going beyond direct evidence*

23 Peter Hofgesang (VU) *Modelling Web Usage in a Changing Environment*

24 Annerieke Heuvelink (VU) *Cognitive Models for Training Simulations*

25 Alex van Ballegooij (CWI) *"RAM: Array Database Management through Relational Mapping*"

26 Fernando Koch (UU) *An Agent-Based Model for the Development of Intelligent Mobile Services*

27 Christian Glahn (OU) *Contextual Support of social Engagement and Reflection on the Web*

28 Sander Evers (UT) *Sensor Data Management with Probabilistic Models*

29 Stanislav Pokraev (UT) *Model-Driven Semantic Integration of Service-Oriented Applications*

30 Marcin Zukowski (CWI) *Balancing vectorized query execution with bandwidth-optimized storage*

31 Sofiya Katrenko (UvA) *A Closer Look at Learning Relations from Text*

32 Rik Farenhorst and Remco de Boer (VU) *Architectural Knowledge Management: Supporting Architects and Auditors*

33 Khiet Truong (UT) *How Does Real Affect Affect Affect Recognition In Speech?*

34 Inge van de Weerd (UU) *Advancing in Software Product Management: An Incremental Method Engineering Approach*

35 Wouter Koelewijn (UL) *Privacy en Politiegegevens; Over geautomatiseerde normatieve informatie-uitwisseling*

36 Marco Kalz (OUN) *Placement Support for Learners in Learning Networks*

37 Hendrik Drachsler (OUN) *Navigation Support for Learners in Informal Learning Networks*

38 Riina Vuorikari (OU) *Tags and self-organisation: a metadata ecology for learning resources in a multilingual context*

39 Christian Stahl (TUE, Humboldt-Universität zu Berlin) *Service Substitution – A Behavioral Approach Based on Petri Nets*

40 Stephan Raaijmakers (UvT) *Multinomial Language Learning: Investigations into the Geometry of Language*

41 Igor Berezhnyy (UvT) *Digital Analysis of Paintings*

42 Toine Bogers (UvT) *Recommender Systems for Social Bookmarking*