THESIS FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

On uncertainty estimation in machine learning

Jakob Lindqvist

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden, 2024

On uncertainty estimation in machine learning

Jakob Lindqvist
ISBN 978-91-8103-044-0

Doktorsavhandlingar vid Chalmers tekniska högskola
Ny serie nr 5502
ISSN 0346-718X

Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg, Sweden
Phone: +46 (0)31 772 1000

Printed by Chalmers Digitaltryck
Gothenburg, Sweden, April 2024

For Andromeda

On uncertainty estimation in machine learning

Jakob Lindqvist
Department of Electrical Engineering
Chalmers University of Technology

Abstract

This thesis explores the representation of probabilistic machine learning models, focusing on representing and quantifying uncertainty. The included publications consider a range of topics, with the common theme of finding methods for representing probability distributions and uncertainty with machine learning models.

The thesis can be divided into three main parts. First, we introduce regularisation methods that improve the convergence of iterated Gaussian smoothers. By interpreting smoothing as an optimisation problem, we develop Levenberg-Marquardt regularisation and line-search methods. These extensions, which require minimal additional computation, provide accurate Gaussian approximations for a richer set of functions and input data compared to existing methods.

The second part addresses uncertainty estimation directly. It presents a general model distillation method for ensembles that efficiently condenses the ensembles' knowledge while retaining their ability to decompose prediction uncertainties into epistemic and aleatoric components. The same type of uncertainty decomposition is vital for us to propose a generalised active learning formulation, in which unlabelled data points are selected based on mutual information between model parameters and noisy labels. The resulting active learning method can leverage the trade-off between dataset size and label quality within a fixed annotation budget.

Finally, the third part considers methods for representing and estimating more complex distributions using generative models. We study the theoretical properties of several important parameter estimation methods for unnormalised models, e.g., energy-based models. We prove connections between importance sampling, contrastive divergence and noise contrastive estimation, thereby establishing a more coherent framework. We use a technique previously limited to energy-based models to propose an improved sampling method for composed diffusion models. By approximating the energy difference between two samples as a line integral of the score function, we achieve a Metropolis-Hastings-like correction step for the score parameterised diffusion models. This enables us to construct improved MCMC sampling methods for standard diffusion models, which previously required an energy parameterisation.

Keywords: Probabilistic machine learning, uncertainty estimation

List of Publications

This thesis is based on the following publications:

[A] **Jakob Lindqvist**, Simo Särkkä, Ángel F. García-Fernández, Matti Raitoharju, Lennart Svensson, "Levenberg–Marquardt and line-search iterated posterior linearisation smoothing". Submitted to Signal Processing, April 2023.

[B] **Jakob Lindqvist***, Amanda Olmin*, Fredrik Lindsten, Lennart Svensson, "A General Framework for Ensemble Distribution Distillation". Machine Learning for Signal Processing, 2020.

[C] **Jakob Lindqvist**, Amanda Olmin, Lennart Svensson, Fredrik Lindsten, "Generalised Active Learning with Annotation Quality Selection". Machine Learning for Signal Processing, 2023.

[D] Amanda Olmin*, **Jakob Lindqvist***, Lennart Svensson, Fredrik Lindsten, "On the connection between Noise-Contrastive Estimation and Contrastive Divergence". Accepted for publication in the International Conference on Artificial Intelligence and Statistics, 2024.

[E] Anders Sjöberg*, **Jakob Lindqvist***, Magnus Önnheim, Mats Jirstrand, Lennart Svensson, "MCMC-Correction for Diffusion Model Composition: Energy Approximation using Diffusion Models". Manuscript.

Other publications by the author, not included in this thesis, are:

[F] A. Olmin, **J. Lindqvist**, L. Svensson, F. Lindsten, "Active Learning with Weak Supervision for Gaussian Processes". International Conference on Neural Information Processing, 2022.

[G] D. Hagerman, R. Naeem, **J. Lindqvist**, C. Lindström, F. Kahl, L. Svensson, "SwInception – Local Attention Meets Convolutions". Accepted for publication in the International Conference on Pattern Recognition and Artificial Intelligence, 2024.

Acknowledgments

First, I would like to thank my closest collaborators: my main supervisor, Lennart Svensson, whose support has been constant throughout my PhD studies; my co-supervisor, Fredrik Lindsten; and my research partners, Amanda Olmin and Anders Sjöberg, with whom collaboration was always easy.

I would also like to thank my colleagues at Chalmers, in the Signal Processing group and beyond, for making Chalmers a truly enjoyable workplace with minimal code of conduct violations. I would particularly like to thank my long-term mates in the corner office: Ebbe, Jacob, and Adam. Thank you for all the heated discussions and ethnic desserts.

Finally, I would like to thank my darling wife, my Louise, for always supporting me, even now when she clearly has more important things to care about.

This research was financially supported by the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

Acronyms

RHS:	Right Hand Side
IID:	Independent and Identically Distributed
MSE:	Mean Squared Error
PMF:	Probability Mass Function
PDF:	Probability Density Function
MAP:	Maximum A Posteriori
ML:	Maximum Likelihood
RTS:	Rauch-Tung-Striebel
EKF:	Extended Kalman Filter

EKS:	Extended Kalman Smoother
IPLF:	Iterated Posterior Linearisation Filter
IPLS:	Iterated Posterior Linearisation Smoother
LM:	Levenberg–Marquardt
IS:	Importance Sampling
ULA:	Unadjusted Langevin Algorithm
SSM:	State Space Model
MC:	Monte-Carlo
MCMC:	Markov Chain Monte-Carlo
CIS:	Conditional Importance Sampling
GD:	Gradient Descent
SGD:	Stochastic Gradient Descent
DAG:	Directed Acyclic Graph
ReLU:	Rectified Linear Unit
CNN:	Convolutional Neural Network
AL:	Active Learning
MI:	Mutual Information
EBM:	Energy-Based Model
ML-IS:	Maximum Likelihood - Importance Sampling
NCE:	Noise-Contrastive Estimation
CD:	Contrastive Divergence
NCE:	Noise-Contrastive Divergence
CNCE:	Conditional Noise-Contrastive Divergence

RNCE:	Ranking Noise-Contrastive Divergence
SM:	Score Matching
DSM:	Denoising Score Matching

Contents

Part I

Overview

CHAPTER 1

Introduction

Machine learning, the ability of computers to learn from data, has a profound impact on our daily lives and society at large. Interactions with machine learning models now occur daily, for almost everybody. The models detect pedestrians around your car, answer online support questions and even create art.

Machine learning is a way of solving problems that is different from other forms of mathematical modelling. Modelling historically meant figuring out underlying logical and physical relationships, reducing and simplifying the problem until it fits into a neat, manageable equation. A model, thus, is something that specifies how something works, or at least how an idealised version of the phenomenon works. Data are, of course, still important, but their role here is to inspire you to figure out the true model: the proverbial falling apple gently encouraging your head to think harder.

Machine learning, on the other hand, does not concern itself with how things work, but rather how they are, and data becomes the guiding principle. Here, a model mimics data and how it actually does it is less important. From this point of view, the model becomes a black box: it somehow takes an input and produces an output and the problem is transformed into finding ways to

tune the model to fit the data. This view of modelling has allowed us to solve problems that were previously considered more fitting for a science fiction novel.

However, for many applications, we do not simply want a black-box guess of the output. When your car tells you it is safe to cross a street, you want that guess to be confident. Therefore, it is desirable to use a probabilistic model, which can predict a distribution over all the possible outcomes rather than a single guess. This way, we can quantify the uncertainty in the model's prediction and make better decisions.

This thesis studies how machine learning models can represent probabilistic models and thereby provide quantifiable and accurate uncertainty descriptions.

1.1 Contributions of the thesis

The papers included in this thesis explore a range of theories and methods related to probabilistic machine learning.

In paper A, we develop regularisation methods for improving the convergence of iterated Gaussian smoothers by incorporating methods from the optimisation literature. In the paper, iterated posterior linearisation smoothing is connected to Gauss–Newton optimisation, which enables two regularising extensions for this important family of smoothing methods: Levenberg–Marquardt regularisation and line-search methods. Our proposed extensions require little extra computation and we show in experiments that they outperform non-regularised smoothers in highly non-linear settings.

Papers B and C deal directly with uncertainty estimation. Paper B develops a method for efficiently decomposing the total uncertainty in a prediction into its epistemic and aleatoric parts. This is achieved by proposing a novel method of model distillation for ensembles that preserves the uncertainty decomposition.

Paper C uses the same kind of uncertainty decomposition to propose a general active learning (AL) procedure which selects unlabelled data points based on the mutual information between model parameters and noisy labels. By allowing the AL method to annotate labels of different qualities, we obtain a more general formulation, which can find the optimal balance of data set size and label quality, for a fixed annotation budget.

Papers D and E study models that can represent complex distributions.

Paper D investigates the properties of different methods for parameter estimation in unnormalised models, such as energy-based models. Although the commonly used methods appear to be conceptually different, we show strong connections between three different classes of methods. The connections provide a foundation for a more coherent framework for unnormalised parameter estimation. They also give theoretical insights into when the methods give unbiased gradient estimates and how to select their proposal distributions.

Paper E proposes a method for improved sampling in score-based diffusion models. We leverage the connection between diffusion and energy-based models to enable sampling with a Metropolis–Hastings-like correction while keeping the more efficient score-based parameterisation of the diffusion model.

1.2 Outline of the thesis

The thesis is divided into two parts. The rest of part I aims to review the theory on which the included papers rely and is organised as follows. Chapter 2 introduces Bayesian inference, with a particular focus on inference in state-space models and sampling methods for approximate inference. Chapter 3 presents the basics of machine learning and how it can be used for probabilistic modelling. Chapter 4 reviews different ways of representing probability distributions with machine learning models and how they can be used to quantify different types of uncertainty. Part I ends with a summary of the included papers in chapter 5 and concluding remarks in 6.

Part II consists of the papers on which this thesis is based.

CHAPTER 2

Bayesian inference

This chapter provides an introduction to Bayesian inference. It presents Bayes' rule (section 2.1) and the special case of Bayesian inference in state-space models (section 2.2). From there, a brief review of approximate inference methods, relevant to this thesis, is given in section 2.3.

2.1 Bayes' rule and its components

Bayesian statistics is a branch of statistical analysis, which has a different philosophical interpretation of modelling than its more well-known frequentist counterpart. From the Bayesian perspective, a probability is interpreted as the certainty of an event in a single trial, not the relative frequency of an event in an infinite number of trials [1]. Another way to formulate it is that a parameter of interest $\boldsymbol{\theta}$, is not interpreted as a single true but unknown value. Instead, the variable itself is considered a random variable, following some probability distribution. The parameter $\boldsymbol{\theta}$ can be a single value but is typically a vector with multiple elements.

Bayesian inference is the framework that provides the methods for reasoning about the distribution of the parameter. Derived from the laws of probability

and probability calculus, it provides a principled way of working with uncertainty. In essence, Bayesian inference is the process starting from a point of uncertainty and, from there, incorporating observations to update our belief about the distribution of $\boldsymbol{\theta}$.

The information about $\boldsymbol{\theta}$, before any data is observed, is encoded in the prior distribution $p(\boldsymbol{\theta})$. The prior distribution is based on beliefs about the relative probability of certain values of $\boldsymbol{\theta}$, which do not depend on observed data. Thus, the prior can be used to model expert experience or physical limitations to the problem, within the probabilistic framework. In the lack of strong prior beliefs, it is also possible to use non-informative priors, which can be selected in several ways [1, p. 61].

The observations (the data) are related to the parameter through a conditional distribution $p(\boldsymbol{x} \mid \boldsymbol{\theta})$, called the likelihood function, which models the distribution of the observations, given a certain value of the parameters. It is common to model the data as conditionally independent, given $\boldsymbol{\theta}$, so that the likelihood of multiple observations is $p(\{\boldsymbol{x}_n\}_{n=1}^N \mid \boldsymbol{\theta}) = \prod_{j=1}^N p(\boldsymbol{x}_n \mid \boldsymbol{\theta})$. This is often written as $p(\mathcal{D} \mid \boldsymbol{\theta})$, where $\mathcal{D}$ is a collection of the data used to estimate $\boldsymbol{\theta}$.

Bayesian inference aims to compute the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x})$, i.e. to state our belief about the distribution of $\boldsymbol{\theta}$, after observing data $\boldsymbol{x}$. The relationship between the posterior, prior and likelihood functions is given by Bayes' rule [1, p. 7]

$$p(\boldsymbol{\theta} \mid \boldsymbol{x}) = \frac{p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\boldsymbol{x})}. \tag{2.1}$$

This rule is trivially derived from the definition of conditional probability, yet it is the cornerstone of Bayesian inference. Virtually any problem in Bayesian inference can be reduced to the trials and tribulations of using Bayes' rule to calculate the posterior distribution.

The denominator in the RHS of (2.1) is called the marginal probability (or marginal likelihood) and can be computed by marginalising the joint probability

$$p(\boldsymbol{x}) = \int p(\boldsymbol{x}, \boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} = \int p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta}; \tag{2.2}$$

if $\boldsymbol{\theta}$ takes discrete values the integration is replaced by summation. The marginal probability is independent of $\boldsymbol{\theta}$ and, therefore, acts as a normalisation constant, ensuring that $p(\boldsymbol{\theta} \mid \boldsymbol{x})$ is a valid probability distribution. An

important consequence of this is that the posterior distribution is proportional to the product of the likelihood and prior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x}) \propto p(\boldsymbol{x} \mid \boldsymbol{\theta})p(\boldsymbol{\theta})$.

Example 2.1.1 Conjugate likelihood. Consider a single parameter θ with a Gaussian prior and likelihood

$$\begin{aligned} p(\theta) &= \mathcal{N}\left(\theta; m_0, s_0^2\right), \\ p(x \mid \theta) &= \mathcal{N}\left(x; \theta, 1/2\right), \end{aligned}$$

with known prior hyperparameters $m_0 = 1$ and $s_0^2 = 1/2$. Assume that $x = 1$ is observed, then Bayes' rule provides the updated belief about θ

$$p(\theta \mid x = 1) = \frac{\mathcal{N}\left(1; \theta, 1/2\right)\mathcal{N}\left(\theta; 0, 1\right)}{\int \mathcal{N}\left(1; \theta, 1/2\right)\mathcal{N}\left(\theta; 0, 1\right) \, \mathrm{d}\theta}.$$

Fortunately, for this example, the posterior is also Gaussian and its mean and variance can be calculated analytically. When the prior and the posterior have the same functional form for a certain likelihood function, we say that we have a conjugate prior (for that specific likelihood function) [2, p. 117]. Since a conjugate prior enables exact inference, the prior is often selected with respect to (w.r.t.) the likelihood function, such that they conjugate.

For this problem, the posterior mean and variance can either be calculated by hand, or they can be obtained from a general update rule for conjugate distributions [2, p. 91]:

$$\begin{aligned} p(\theta \mid x = 1) &= \mathcal{N}\left(\theta; m_1, s_1^2\right), \text{where} \\ m_1 &= s_1\left(\frac{m_0}{s_0^2} + \frac{1}{\sigma^2}\right) = 2/3, \\ s_1 &= \left(\frac{1}{s_0^2} + \frac{1}{\sigma^2}\right) = 1/3. \end{aligned} \tag{2.3}$$

The left part of figure 2.1 shows how the posterior is updated from the prior. It shows the gist of Bayesian inference, namely that after seeing $x = 1$, we are not completely confident that $\theta = 1$, instead, the posterior is a compromise between the observation and the prior belief.

For multiple observations, Bayes' rule can either be used with all data simultaneously, by computing the joint likelihood $p(\{x_n\}_{n=1}^N \mid \theta)$, or it can be used recursively where the current posterior becomes the prior for the next

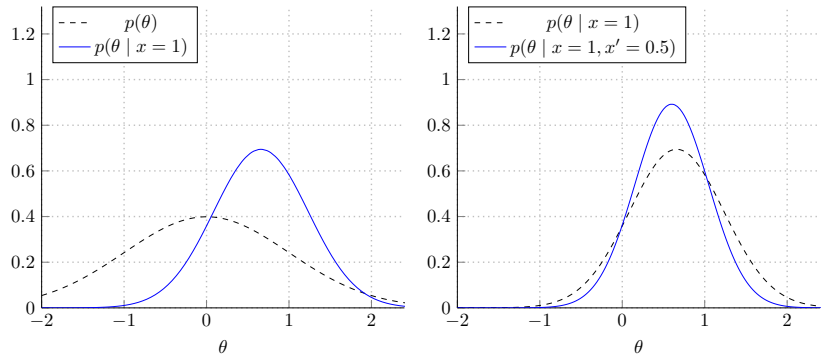

Figure 2.1: Bayesian inference for univariate Gaussians. The figure on the left shows the prior and posterior distribution after observing $x = 1$. The figure on the right shows the posterior distribution after another sample $x' = 1/2$ has been observed. Here the posterior in the left figure acts as the prior.

observation. Going back to the example above, assume that, after computing the posterior $p(\theta \mid x = 1)$, another observation $x' = 0.5$ is made. Then, m_1, s_1^2 acts as the prior parameters in (2.3) and the new posterior parameters become $m_2 = 3/5, s_2^2 = 1/5$. The prior and posterior distribution for the second observation is shown on the right in figure 2.1. Note that after receiving more information, the mean is again shifted slightly and the uncertainty about θ is further reduced.

Having access to the full posterior distribution over $\boldsymbol{\theta}$, as opposed to a single point estimate, is clearly useful. We just saw how it can be used to recursively update our belief about the system we want to model, and it can also be used to infer the distribution of future observations, using the posterior predictive distribution

$$p(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathbb{E}_{\boldsymbol{\theta} \sim p(\boldsymbol{\theta} \mid \boldsymbol{x})} \left[p(\boldsymbol{x}' \mid \boldsymbol{\theta}) \right] = \int p(\boldsymbol{x}' \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{x}) \, \mathrm{d}\boldsymbol{\theta}. \tag{2.4}$$

Additionally, it provides a principled way for us to reason about the uncertainties in the random variable we seek to estimate, which is one of the reasons why Bayesian inference is appealing.

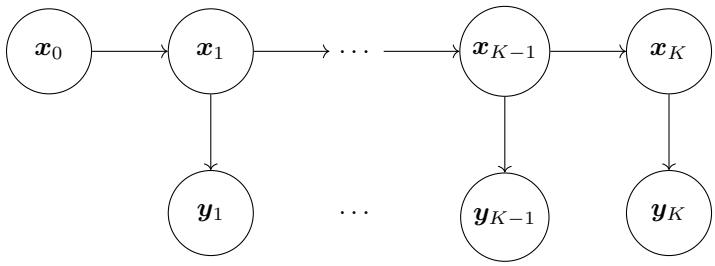

Figure 2.2: Graphical representation of an SSM. The graph encodes the conditional independencies of the model, where the current state $\boldsymbol{x}_k$ only depends on $\boldsymbol{x}_{k-1}$ and the measurement $\boldsymbol{y}_k$ only depends on $\boldsymbol{x}_k$.

We have now seen an example of exact Bayesian inference with conjugate priors, and the next section will present another, more useful, instance of exact inference. However, computing the posterior distribution in closed form is not generally possible. This is because the integral in (2.2) is intractable in general, meaning that there is no closed-form expression for the posterior distribution. Therefore, practical Bayesian inference typically deals with finding approximate posteriors or with doing inference without a properly normalised distribution.

2.2 Inference in state space models

A state space model (SSM) is a stochastic process defined by a sequence of random variables $\boldsymbol{x}_{0:K} \coloneqq (\boldsymbol{x}_0, \boldsymbol{x}_1, \dots \boldsymbol{x}_K)$ (called states) and corresponding measurements $\boldsymbol{y}_{1:K} \coloneqq (\boldsymbol{y}_1, \boldsymbol{y}_2, \dots \boldsymbol{y}_K)$ [3, p. 91]. They are commonly used to model an evolving random variable (the state), which can only be observed through noisy measurements, such as tracking a vehicle's position and velocity from GPS measurements or estimating the true number of individuals infected by some disease from hospitalisation rates [4]–[6]. In the context of Bayesian inference, the sequence of states $\boldsymbol{x}_{0:K}$ corresponds to the unknown random variable we seek to estimate (i.e., $\boldsymbol{\theta}$) and the sequence of measurements $\boldsymbol{y}_{1:K}$ are the observations (i.e., $\boldsymbol{x}$).

An SSM has a particular structure of the conditional independencies of the random variables in the SSM, which can be represented by a graphical model shown in figure 2.2, The figure encodes two important types of conditional

independencies. Firstly, the sequence $\boldsymbol{x}_{0:K}$ has a Markovian property, meaning that the distribution of the next state in the sequence only depends on the current state:

$$p(\boldsymbol{x}_k \mid \boldsymbol{x}_{0:k-1}) = p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}),\ k = 0, \ldots, K. \tag{2.5}$$

The distribution in the RHS models the stochastic dynamics of the process and is called the dynamic model (also process or motion model). Secondly, a measurement at time step k only depends on the state at time step k

$$p(\boldsymbol{y}_k \mid \boldsymbol{y}_{1:k-1}, \boldsymbol{y}_{k+1:K}, \boldsymbol{x}_{0:K}) = p(\boldsymbol{y}_k \mid \boldsymbol{x}_k),\ k = 1, \ldots, K, \tag{2.6}$$

and it is called the measurement model. Intuitively, the state $\boldsymbol{x}_k$ summarises everything we need to know about the system at time step k, which explains the conditional independencies in (2.5) and (2.6).

The SSM is specified by the dynamic and measurement model, together with a prior distribution $p(\boldsymbol{x}_0)$. For processes with additive Gaussian noise, it is common to formulate the SSM on the following form:

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{f}(\boldsymbol{x}_{k-1}) + \boldsymbol{q}_{k-1}, & \boldsymbol{q}_{k-1} &\sim \mathcal{N}(0, \boldsymbol{Q}), \\ \boldsymbol{y}_k &= \boldsymbol{h}(\boldsymbol{x}_k) + \boldsymbol{r}_k, & \boldsymbol{r}_k &\sim \mathcal{N}(0, \boldsymbol{R}), \end{aligned} \tag{2.7}$$

where $\boldsymbol{f}$ is called the dynamic function and $\boldsymbol{h}$ the measurement function. It is perfectly possible to have different functions (and covariance matrices) at every time step, but for notational brevity, we assume the same dynamic and measurement models at every time step.

Following the recipe for Bayesian inference, we seek to calculate the posterior distribution of the states $\boldsymbol{x}_{0:K}$ given the measurements $\boldsymbol{y}_{1:K}$, where the properties of the SSM enable a simple factorisation of the distributions:

$$\begin{aligned} p(\boldsymbol{x}_{0:K} \mid \boldsymbol{y}_{1:K}) &= \frac{p(\boldsymbol{y}_{1:K} \mid \boldsymbol{x}_{0:K}) p(\boldsymbol{x}_{0:K})}{p(\boldsymbol{y}_{1:K})} \\ &= \frac{\left(\prod_{k=1}^{K} p(\boldsymbol{y}_k \mid \boldsymbol{x}_k) p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1})\right) p(\boldsymbol{x}_0)}{p(\boldsymbol{y}_{1:K})}. \end{aligned} \tag{2.8}$$

Like all problems in Bayesian inference, this posterior is only tractable to compute for some special cases of dynamic and measurement models. However,

the complexity of computing the posterior in (2.8) also grows with K. For real-time problems, such as tracking a car over an extended period, this would quickly become infeasible.

For this reason, inference in state space models often considers variants of the above problem, which simplifies the inference objective to posterior distributions which can be computed recursively, and with constant computational complexity. Two such important variants are Bayesian filtering and smoothing, which are used widely in diverse applications.

Filtering and smoothing in state space models

The basic idea of creating simplified inference problems is to relax the requirement of computing the joint posterior distribution over the full sequence of states $\boldsymbol{x}_{0:K}$. Instead, the aim is to compute marginal posterior distributions at each time step. The marginal distributions are less useful than knowing the full joint posterior, but they can still be used to solve several important problems, among them filtering and smoothing.

For filtering, the goal is to calculate the distribution $p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k})$. That is, the distribution of the random variable $\boldsymbol{x}_k$, given all measurements up to and including time step k. This is useful for online applications, where new measurements are observed in real-time and the goal is to find the best possible estimate of the current state $\boldsymbol{x}_k$, given currently available information.

At any time step $k = 1, \ldots, K$, this marginal distribution can be computed recursively in two steps. First is the prediction step, which computes the predicted distribution of $\boldsymbol{x}_k$, given the measurements up until time step $k-1$, $\boldsymbol{y}_{1:k-1}$, given by the Chapman-Kolmogorov equation [3, p. 94]:

$$p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k-1}) = \int p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}) p(\boldsymbol{x}_{k-1} \mid \boldsymbol{y}_{1:k-1}) \, \mathrm{d}\boldsymbol{x}_{k-1}. \tag{2.9}$$

Next, the filtering distribution is computed from the predicted distribution above and using Bayes' rule to include the measurement at time step k. This is called the update step:

$$\begin{aligned} p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) &= \frac{1}{Z_k} p(\boldsymbol{y}_k \mid \boldsymbol{x}_k) p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k-1}), \\ Z_k &= \int p(\boldsymbol{y}_k \mid \boldsymbol{x}_k) p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k-1}) \, \mathrm{d}\boldsymbol{x}_k, \end{aligned} \tag{2.10}$$

is a normalisation constant. By starting from the prior distribution $\boldsymbol{x}_0$, the marginal distribution $p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k})$ can be computed recursively by iterating the prediction and update step up until time step k.

For smoothing, the target is instead the marginal distribution $p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:K})$, i.e., the distribution of $\boldsymbol{x}_k$, given the full measurement sequence $\boldsymbol{y}_{1:K}$ Contrary to filtering, smoothing estimates the random variable $\boldsymbol{x}_k$ by also including future observations, made after time step k. Smoothing is, therefore, an offline method, which updates the belief about $\boldsymbol{x}_k$, using more information than the filtering method.

The marginal distributions for the smoothing method can be computed using the backward recursive equations [3, p. 254]

$$p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:K}) = p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) \int \frac{p(\boldsymbol{x}_{k+1} \mid \boldsymbol{x}_k)\; p(\boldsymbol{x}_{k+1} \mid \boldsymbol{y}_{1:K})}{p(\boldsymbol{x}_{k+1} \mid \boldsymbol{y}_{1:k})} \,\mathrm{d}\boldsymbol{x}_{k+1}, \tag{2.11}$$

where $p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k})$ is the filtering distribution at time step k and $p(\boldsymbol{x}_{k+1} \mid \boldsymbol{y}_{1:k})$ is the predicted distribution at time step $k+1$, given by (2.10) and (2.9) respectively. Thus, by first performing filtering of the entire sequence, the smoothing distributions can be computed backwards, starting with $p(\boldsymbol{x}_K \mid \boldsymbol{y}_{1:K})$ and following the recursion down to time step $k = 1$.

The Kalman filter and Rauch–Tung–Striebel smoother

Bayesian filtering and smoothing leverages the structure of the SSM to derive efficient, recursive, equations for computing the marginal posterior distributions, but the integrals in (2.9)-(2.11) are intractable for general dynamic and measurement models. There is, however, an important special class of SSMs for which the filtering and smoothing distributions can be computed analytically.

With affine motion and measurement models and additive Gaussian noise, the SSM in (2.7) becomes

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{b} + \boldsymbol{q}_{k-1}, & \boldsymbol{q}_{k-1} &\sim \mathcal{N}(0, \boldsymbol{Q}), \\ \boldsymbol{y}_k &= \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{c} + \boldsymbol{r}_k, & \boldsymbol{r}_k &\sim \mathcal{N}(0, \boldsymbol{R}). \end{aligned} \tag{2.12}$$

If the initial prior is also Gaussian $p(\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_0; \boldsymbol{m}_0, \boldsymbol{P}_0)$, the marginal distributions are Gaussian at all time steps and the SSM in (2.12) can also be

expressed as conditional distributions

$$\begin{aligned} p(\boldsymbol{x}_k \mid \boldsymbol{x}_{k-1}) &= \mathcal{N}\left(\boldsymbol{x}_k; \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{b}, \boldsymbol{Q}\right), \\ p(\boldsymbol{y}_k \mid \boldsymbol{x}_k) &= \mathcal{N}\left(\boldsymbol{x}_k; \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{c}, \boldsymbol{R}\right). \end{aligned} \tag{2.13}$$

For this special case, the marginal distributions are normally distributed and can be fully described by their respective mean vectors and covariance matrices. Furthermore, exact inference in the form of filtering and smoothing, is tractable for these SSMs.

For the filtering problem, this results in the Kalman filter (KF) [7], where the predicted and filtering distributions are given by

$$\begin{aligned} p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k-1}) &= \mathcal{N}\left(\boldsymbol{x}_k; \boldsymbol{m}_k^-, \boldsymbol{P}_k^-\right), \\ p(\boldsymbol{x}_k \mid \boldsymbol{y}_{1:k}) &= \mathcal{N}\left(\boldsymbol{x}_k; \boldsymbol{m}_k, \boldsymbol{P}_k\right), \end{aligned} \tag{2.14}$$

and the parameters of these distributions can be computed with the Kalman filter prediction step

$$\begin{aligned} \boldsymbol{m}_k^- &= \boldsymbol{F}\boldsymbol{m}_{k-1} + \boldsymbol{b}, \\ \boldsymbol{P}_k^- &= \boldsymbol{F}\boldsymbol{P}_{k-1}\boldsymbol{F}^\top + \boldsymbol{Q}, \end{aligned} \tag{2.15}$$

and update step

$$\begin{aligned} \boldsymbol{S}_k &= \boldsymbol{H}\boldsymbol{P}_k^-\boldsymbol{H}^\top + \boldsymbol{R}, \\ \boldsymbol{K}_k &= \boldsymbol{P}_k^-\boldsymbol{H}^\top\boldsymbol{S}_k^{-1}, \\ \boldsymbol{m}_k &= \boldsymbol{m}_k^- + \boldsymbol{K}_k\left(\boldsymbol{y}_k - \boldsymbol{H}\boldsymbol{m}_k^- - \boldsymbol{c}\right), \\ \boldsymbol{P}_k &= \boldsymbol{P}_k^- - \boldsymbol{K}_k\boldsymbol{S}_k^{-1}\boldsymbol{K}_k^\top, \end{aligned} \tag{2.16}$$

where the recursion goes from $k = 1$ to $k = K$.

Similarly, the mean and covariance of the smoothing distributions can be evaluated in closed form, by using the parameters of the predicted and filtering distributions above, and performing the backward recursion

$$\begin{aligned} \boldsymbol{G}_k &= \boldsymbol{P}_k\boldsymbol{F}\left(\boldsymbol{P}_{k+1}^-\right)^{-1}, \\ \boldsymbol{m}_k^s &= \boldsymbol{m}_k + \boldsymbol{G}_k\left(\boldsymbol{m}_{k+1}^s - \boldsymbol{m}_{k+1}^-\right), \\ \boldsymbol{P}_k^s &= \boldsymbol{P}_k + \boldsymbol{G}_k\left(\boldsymbol{P}_{k+1}^s - \boldsymbol{P}_{k+1}^-\right)\boldsymbol{G}_k^\top. \end{aligned} \tag{2.17}$$

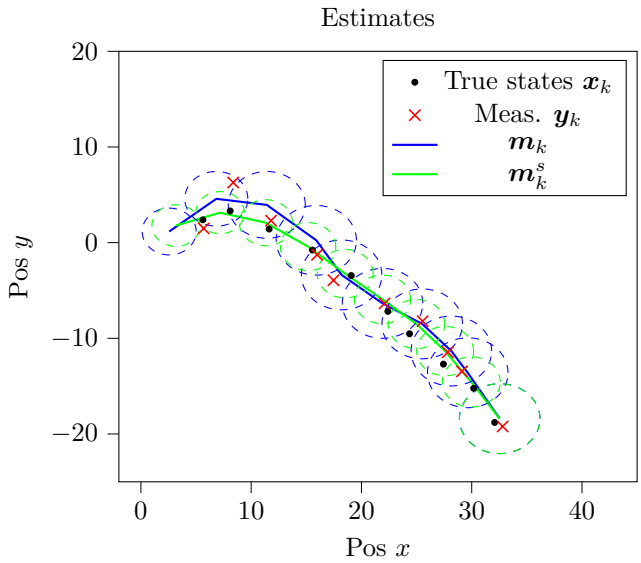

Figure 2.3: Filtering and smoothing trajectories for affine motion and measurement models. The trajectories are based on the noisy measurement sequence $\boldsymbol{y}_{1:K}$, while the true states $\boldsymbol{x}_{0:K}$ are unknown. The filtering estimates (in blue) are represented by the filtered mean $\boldsymbol{m}_k$ and the smoothing estimates (in green) are represented by the smoothed mean $\boldsymbol{m}_k^s$. The dashed lines show the level curves of the marginal PDFs, three standard deviations away from the mean. The RTS smoother produces better estimates than the Kalman filter but also bases its estimates on all measurements.

The recursion starts with $\boldsymbol{m}_K^s = \boldsymbol{m}_K$ and $\boldsymbol{P}_k^s = \boldsymbol{P}_K$. This method is called the Rauch–Tung–Striebel smoother [8].

An example of filtering and smoothing trajectories is shown in figure 2.3.

General Gaussian filters and smoothers

Constraining the SSM to only affine models with additive Gaussian noise can seem restrictive, but there are ways to extend both the Kalman filter and the RTS smoother to SSMs which have non-linear functions $\boldsymbol{f}$ and $\boldsymbol{h}$, called general Gaussian filtering or smoothing. Replacing the non-linear functions

with affine approximations, results in an approximate affine SSM, for which exact inference is possible with the Kalman Filter and the RTS smoother. The approximated affine SSM is of the same form as in (2.7)

$$\begin{aligned} \boldsymbol{x}_k &= \boldsymbol{F}\boldsymbol{x}_{k-1} + \boldsymbol{b} + \boldsymbol{\gamma}_{k-1} + \boldsymbol{q}_{k-1}, \quad \boldsymbol{\gamma} \sim \mathcal{N}(0, \boldsymbol{\Gamma}), \boldsymbol{q}_{k-1} \sim \mathcal{N}(0, \boldsymbol{Q}), \\ \boldsymbol{y}_k &= \boldsymbol{H}\boldsymbol{x}_k + \boldsymbol{c} + \boldsymbol{\lambda}_{k-1} + \boldsymbol{r}_k, \qquad \boldsymbol{\lambda} \sim \mathcal{N}(0, \boldsymbol{\Lambda}), \boldsymbol{r}_k \sim \mathcal{N}(0, \boldsymbol{R}), \end{aligned} \tag{2.18}$$

but with the difference that the parameters of the affine transformation $(\boldsymbol{F}, \boldsymbol{b}, \boldsymbol{\Gamma})$ and $(\boldsymbol{H}, \boldsymbol{c}, \boldsymbol{\Lambda})$ are now estimated by linearisation at every k, where the random variables $\boldsymbol{\gamma}_k$ and $\boldsymbol{\lambda}_k$ model the linearisation error. Note that the linearisation parameters implicitly depend on k.

The most straightforward method of finding an affine approximation is to use the first-order Taylor expansion, here exemplified by the measurement function $\boldsymbol{h}(\cdot)$ linearised around some arbitrary $\boldsymbol{x}'$

$$\boldsymbol{h}(\boldsymbol{x}_k) \approx \boldsymbol{h}(\boldsymbol{x}') + \boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}')(\boldsymbol{x}_k - \boldsymbol{x}') = \underbrace{\boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}')}_{\boldsymbol{H}(\boldsymbol{x}')}\boldsymbol{x}_k + \underbrace{\boldsymbol{h}(\boldsymbol{x}') - \boldsymbol{J}_{\boldsymbol{h}}(\boldsymbol{x}')\boldsymbol{x}'}_{\boldsymbol{c}(\boldsymbol{x}')}, \tag{2.19}$$

where $\boldsymbol{J}_{\boldsymbol{h}}$ is the Jacobian of $\boldsymbol{h}$, evaluated at $\boldsymbol{x}'$. Note that the parameters of the affine transformation, the matrix $\boldsymbol{H}(\boldsymbol{x}')$ and vector $\boldsymbol{c}(\boldsymbol{x}')$, are now functions of the linearisation point $\boldsymbol{x}'$, but we will henceforth drop this dependency to keep the notation simpler. An affine approximation of the dynamic function $\boldsymbol{f}$ is derived analogously.

One filtering method based on using this analytical linearisation to approximate a non-linear SSM is the extended Kalman filter (EKF), and the corresponding smoother is called the extended Kalman smoother (EKS) (or conversely, the extended RTS smoother). At any time step k, the EKF uses the current best approximation of the state $\boldsymbol{x}_k$, as the linearisation point. This means linearising around $\boldsymbol{m}_{k-1}$, the filtered mean at time step $k-1$, for the prediction step, and the predicted mean $\boldsymbol{m}_k^-$, for the update step. The EKS on the other hand, uses the current filtered mean $\boldsymbol{m}_k$, at time k, since it is available from the filtering pass. The linearisation error is assumed to be zero, i.e. $\boldsymbol{\gamma}$ and $\boldsymbol{\lambda}$ are assumed to have mean zero and zero variance.

The Taylor expansion in (2.19) achieves the best possible affine approximation of the measurement function, but only locally around that particular linearisation point. The EKF therefore, will use a good approximation around the mode of the estimated distribution of the state, but all other points where

$\boldsymbol{x}_k$ has support, have no impact on the linearisation. This is evident from the fact that the linearisation does not depend on the estimated covariance matrix.

There is an alternative method, which aims to find the linearisation, that is best on average w.r.t. the whole distribution over the state $\boldsymbol{x}_k$. This method is called statistical linear regression (SLR). Suppose that our current belief about the random variable $\boldsymbol{x}_k$ is described by the distribution $\pi(\boldsymbol{x}_k) = \mathcal{N}(\boldsymbol{x}_k; \boldsymbol{\mu}, \boldsymbol{\Sigma})$. With SLR, the affine approximation of $\boldsymbol{h}(\boldsymbol{x}_k)$, based on this distribution, is selected such that [9]:

$$
\begin{aligned}
(\boldsymbol{H}, \boldsymbol{c}) &= \underset{(\boldsymbol{H}', \boldsymbol{c}')}{\operatorname{argmin}} \, \mathbb{E}_{\boldsymbol{x}_k \sim \pi} \left[\left(\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{H}' \boldsymbol{x}_k - \boldsymbol{c}' \right)^\top \left(\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{H}' \boldsymbol{x}_k - \boldsymbol{c}' \right) \right], \\
\boldsymbol{\Gamma} &= \mathbb{E}_{\boldsymbol{x}_k \sim \pi} \left[\left(\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{H} \boldsymbol{x}_k - \boldsymbol{c} \right) \left(\boldsymbol{h}(\boldsymbol{x}_k) - \boldsymbol{H} \boldsymbol{x}_k - \boldsymbol{c} \right)^\top \right]. \qquad (2.20)
\end{aligned}
$$

That is, $\boldsymbol{H}$ and $\boldsymbol{c}$ constitute the affine approximation of $\boldsymbol{h}$, with the smallest mean squared error (MSE) and $\boldsymbol{\Gamma}$ is the covariance matrix of the linearisation error. The parameters $(\boldsymbol{H}, \boldsymbol{c}, \boldsymbol{\Gamma})$ implicitly depend on the distribution π, but it is omitted for brevity.

The SLR parameters can be expressed in the following way [10]:

$$
\begin{aligned}
\boldsymbol{H} &= \boldsymbol{\Psi}^\top \boldsymbol{\Sigma}^{-1}, \qquad (2.21) \\
\boldsymbol{c} &= \bar{\boldsymbol{h}} - \boldsymbol{H} \boldsymbol{\mu}, \qquad (2.22) \\
\boldsymbol{\Gamma} &= \boldsymbol{\Phi} - \boldsymbol{H} \boldsymbol{\Sigma} \boldsymbol{H}^\top, \qquad (2.23)
\end{aligned}
$$

where

$$
\begin{aligned}
\bar{\boldsymbol{h}} &= \int \boldsymbol{h}(\boldsymbol{x}) \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \\
\boldsymbol{\Psi} &= \int (\boldsymbol{x} - \boldsymbol{\mu})(\boldsymbol{h}(\boldsymbol{x}) - \bar{\boldsymbol{h}})^\top \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \\
\boldsymbol{\Phi} &= \int (\boldsymbol{h}(\boldsymbol{x}) - \bar{\boldsymbol{h}})(\boldsymbol{h}(\boldsymbol{x}) - \bar{\boldsymbol{h}})^\top \pi(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}. \qquad (2.24)
\end{aligned}
$$

The SLR moments in (2.24) are not tractable for most non-linear functions. There are, however, multiple ways of approximating the integrals above, using, for instance, sigma-point methods [11]–[14].

Example 2.2.1 Comparing linearisation methods. Let the prior belief

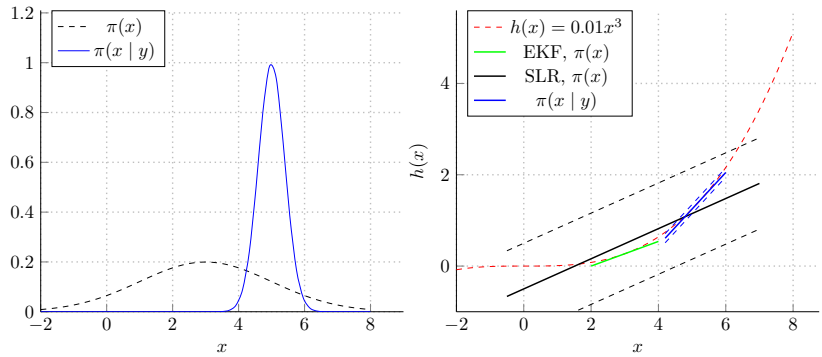

Figure 2.4: Different linearisation methods of the measurement function $h(x)$. The left figure shows the prior (black dashed) and posterior distribution (blue) of the state x. In the right figure, the green line shows the first-order Taylor approximation of h around the prior mean. The black line shows the SLR linearisation w.r.t. to the prior distribution, and the dashed lines represent the linearisation error. The blue line shows the SLR linearisation w.r.t. to the posterior distribution. (Adapted from [10]).

about a scalar state, $\pi(x)$, be a Gaussian centred in $x = 3$ and with high uncertainty and $h(x) = 0.01x^3$, the non-linear measurement function. Figure 2.4 shows a comparison between the EKF and SLR linearisations. From the figure, it is evident that the linearisation methods have very different behaviour. The analytical linearisation of the EKF is perfect at $x = 3$, which is the prior mean, but underestimates the rapid growth for larger values of x. It further assumes that the linearisation is perfect, and thus adds no extra uncertainty to compensate for the approximation. The SLR linearisation is not only a better fit across the support of $\pi(x)$, but it also displays high uncertainty due to the non-linearity of h.

Consider now the effect of observing a rather certain measurement, $y = 1.5$, such that the exact posterior distribution $\pi(x \mid y = 1.5)$ is as the blue line on the left in figure 2.4. Since the EKF linearisation has such a small slope, its update step would have to shift x to the right, in order to explain the measurement, resulting in a poor estimate of the posterior distribution. The SLR, on the other hand, would fare better since the increased uncertainty

would help to explain an unexpected measurement.

Despite the benefit of SLR linearisation, it still seems suboptimal to linearise w.r.t. to the prior $\pi(\boldsymbol{x}_k)$. Ideally, the linearisation would be based on the posterior distribution $\pi(\boldsymbol{x}_k \mid \boldsymbol{y}_k)$, i.e., including the information contained in the measurement. This is, of course, not possible since $\pi(\boldsymbol{x}_k \mid \boldsymbol{y}_k)$ needs the linearisation to be computed. However, there is an approximate method for achieving posterior linearisation, and that is to iterate the update step.

Iterative general Gaussian filters and smoothers

The idea in iterative general Gaussian filtering (smoothing) is to iteratively refine the affine approximations, by linearising around better and better estimates of the mean and covariances of the states. For iterative smoothing, which is of most concern for this thesis, the following two steps are iterated:

$$\begin{aligned} \Theta_{1:K}^{(i+1)} &= \text{Linearisation}\left(\boldsymbol{m}_{1:K}^{(i)}, \boldsymbol{P}_{1:K}^{(i)}\right), \\ (\boldsymbol{m}_{1:K}^{(i+1)}, \boldsymbol{P}_{1:K}^{(i+1)}) &= \text{RTS smoothing}\left(\Theta_{1:K}^{(i+1)}\right), \end{aligned} \tag{2.25}$$

where $\Theta_{1:K}^{(i+1)} = \left(\boldsymbol{F}_{1:K}^{(i+1)}, \boldsymbol{b}_{1:K}^{(i+1)}, \boldsymbol{\Gamma}_{1:K}^{(i+1)}, \boldsymbol{H}_{1:K}^{(i+1)}, \boldsymbol{c}_{1:K}^{(i+1)}, \boldsymbol{\Lambda}_{1:K}^{(i+1)}\right)$ are the parameters of the affine approximation at iteration $i+1$ and $\boldsymbol{m}_{1:K}^{(i)}$, $\boldsymbol{P}_{1:K}^{(i)}$ are the sequences of smoothed means and covariances and iteration i. The initial estimates $\boldsymbol{m}_{1:K}^{(0)}, \boldsymbol{P}_{1:K}^{(0)}$ are obtained from the standard general Gaussian smoothing from the previous section.

This procedure is then iterated until the estimates (hopefully) converge, such that the final linearisation is done w.r.t an accurate approximation of the true posterior distribution [15]. The procedure applies to both linearisation schemes above, resulting in the IEKF (IEKS) [16], [17] for the analytical linearisation and the iterated posterior linearisation filter (IPLF) and smoother (IPLS) for the SLR linearisation [10], [18]. The IPLF and IPLS are more accurately described as families of methods since the different methods of estimating the SLR moments in (2.24) give rise to a sub-genre of different filters and smoothers.

The iterative methods typically perform much better than their non-iterative counterparts [16], but they are not guaranteed to converge, especially in highly non-linear settings. However, some modifications to the IEKS have been

proposed to obtain an algorithm that converges for a wider family of problems [19]–[21]. Paper A in this thesis studies related modifications of the IPLS algorithm.

2.3 Approximate inference through sampling

For problems where it is not possible to compute the posterior in closed form, approximate inference is necessary. In the general Gaussian smoothers, we have already seen an example of this, where the approximation is to assume that the posterior distribution is Gaussian. This approximation technique is not restricted to non-linear SSMs, but is widely used, see e.g., [22, p. 341]. There are, however, problems for which a Gaussian approximation of the posterior is unsuitable. For these scenarios, it is common to relax the requirement of finding a parametric approximation to the posterior and instead do inference based on sampling from the posterior. The following section introduces a selection of such sampling-based methods, for increasingly complicated posteriors.

A typical inference problem is to estimate expected values w.r.t. a posterior distribution. One example of this is computing the posterior predictive distribution in (2.4), which is defined as the expected value of the likelihood $p(\boldsymbol{x}' \mid \boldsymbol{\theta})$ w.r.t. $p(\boldsymbol{\theta} \mid \boldsymbol{x})$, and there are many more such applications.

To make the description more general, let $p(\boldsymbol{z})$ be some target distribution (e.g., the posterior) and $g(\boldsymbol{z})$ some function of $\boldsymbol{z}$, for which we want to estimate the expected value

$$g^\star \coloneqq \mathbb{E}_{p(\boldsymbol{z})}\left[g(\boldsymbol{z})\right] = \int g(\boldsymbol{z}) p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z}. \tag{2.26}$$

For approximate Bayesian inference, $\boldsymbol{z}$ would be the parameter of interest $\boldsymbol{\theta}$ and $p(\boldsymbol{z})$ the posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{x})$.

Consider the case when the integral in (2.26) is intractable, but it is possible to sample from $p(\boldsymbol{z})$. Then, $g^\star$ can be approximated using Monte Carlo (MC) methods [23]–[25], which estimate the expected value as a sample average. An interpretation of this is that $p(\boldsymbol{z})$ is approximated with Dirac delta functions

in the samples $\boldsymbol{z}_j \sim p(\boldsymbol{z})$, to get the approximation

$$\begin{aligned} g^\star &= \int g(\boldsymbol{z})p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} \approx \int g(\boldsymbol{z}) \frac{1}{J} \sum_{j=1}^{J} \delta(\boldsymbol{z} - \boldsymbol{z}_j) \, \mathrm{d}\boldsymbol{z} \\ &= \frac{1}{J} \sum_{j=1}^{J} \int g(\boldsymbol{z}) \delta(\boldsymbol{z} - \boldsymbol{z}_j) \, \mathrm{d}\boldsymbol{z} = \frac{1}{J} \sum_{j=1}^{J} g(\boldsymbol{z}_j) \coloneqq \hat{g}. \end{aligned} \tag{2.27}$$

The estimator $\hat{g}$ is unbiased, which means that it has the correct expected value $\mathbb{E}_{p(\boldsymbol{z})}[\hat{g}] = \mathbb{E}_{p(\boldsymbol{z})}[g(\boldsymbol{z})]$. Furthermore, the variance of the estimator

$$\mathrm{Var}_{p(\boldsymbol{z})}[\hat{g}] = \frac{1}{J} \mathbb{E}_{p(\boldsymbol{z})} \left[\left(g(\boldsymbol{z}) - \mathbb{E}_{p(\boldsymbol{z})}[g(\boldsymbol{z})] \right)^2 \right] \tag{2.28}$$

decreases with the number of samples J and does not depend on the dimensionality of $\boldsymbol{z}$ [2, p. 524].

Importance sampling

For more complicated distributions, it might be impossible to directly sample from $p(\boldsymbol{z})$, but the PDF can be evaluated point-wise. A naive way to estimate $g^\star$ in this case is to uniformly discretise the input space and compute the average of $g(\boldsymbol{z})p(\boldsymbol{z})$ over all bins. This approach would quickly become infeasible, as the number of bins would grow exponentially with the dimension of $\boldsymbol{z}$, and it is likely that the product $g(\boldsymbol{z})p(\boldsymbol{z})$ will be close to zero for many $\boldsymbol{z}$-values, which will essentially not contribute to the estimate.

A better solution is to try to obtain approximate samples from $p(\boldsymbol{z})$. This can be achieved with importance sampling (IS), which introduces a proposal distribution $q(\boldsymbol{z})$ to reformulate the MC approximation in (2.27) as an expected value over q

$$\begin{aligned} g^\star &= \int g(\boldsymbol{z})p(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} = \int g(\boldsymbol{z}) \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} q(\boldsymbol{z}) \, \mathrm{d}\boldsymbol{z} \\ &= \mathbb{E}_{q(\boldsymbol{z})} \left[g(\boldsymbol{z}) \frac{p(\boldsymbol{z})}{q(\boldsymbol{z})} \right] \approx \frac{1}{J} \sum_{j=1}^{J} w(\boldsymbol{z}_j) g(\boldsymbol{z}_j) \coloneqq \hat{g}_{IS}, \end{aligned} \tag{2.29}$$

where $\boldsymbol{z}_j \sim q(\boldsymbol{z})$ are now sampled from the proposal distribution. The weight

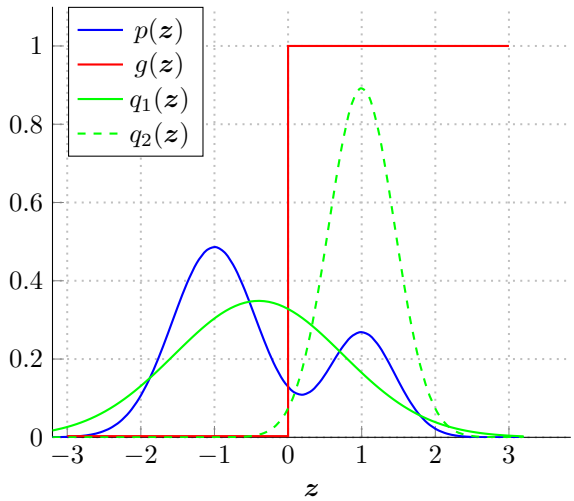

Figure 2.5: Example of IS with two choices of proposal distribution q_1 and q_2. The proposal q_1 (solid green) is the best Gaussian approximation of the posterior distribution (blue), but since the product $g(\boldsymbol{z})p(\boldsymbol{z})$ is zero on parts of the support of $p(z)$, the alternative proposal q_2 (dashed green) is a more suitable choice.

$w(\boldsymbol{z}_j) = \frac{p(\boldsymbol{z}_j)}{q(\boldsymbol{z}_j)}$ corrects the bias introduced by sampling from $q(\boldsymbol{z})$ rather than $p(\boldsymbol{z})$.

The selection of the proposal distribution q then becomes an important design decision. First, its support must be larger than that of $p(\boldsymbol{z})$, i.e. $q(\boldsymbol{z}) > 0, \forall \boldsymbol{z} : p(\boldsymbol{z}) > 0$, for the weights to be defined. It must also be relatively easy to sample from and evaluate, to be useful for this purpose. A secondary concern is that a poor choice of q can increase the variance of the estimator $\hat{g}_{IS}$ significantly. If the support of q differs from the support of p, it will produce many samples with low weights, which do not contribute much to the sum in (2.29), essentially reducing the effective sample size. Further complicating the choice of proposal distribution is that q should ideally produce samples where the product $g(\boldsymbol{z})p(\boldsymbol{z})$ is large. Figure 2.5 shows a simplified scenario where g greatly influences the choice of proposal distribution.

IS can also be used when $p(\boldsymbol{z})$ can only be evaluated up to some constant,

i.e, when

$$p(\boldsymbol{z}) = \frac{\tilde{p}(\boldsymbol{z})}{Z}, \text{ where}$$
$$Z = \int \tilde{p}(\boldsymbol{z}') \, \mathrm{d}\boldsymbol{z}', \tag{2.30}$$

and $\tilde{p}(\boldsymbol{z})$ can be evaluated, but the normalisation constant Z is intractable. Recall from Bayes' rule that the posterior is proportional to the product of the prior and likelihood, but the normalisation constant (the marginal likelihood) is often intractable. Therefore, this scenario is common in Bayesian inference.

For an unnormalised distribution $\tilde{p}(\boldsymbol{z})$, IS can be used to estimate the normalisation constant itself, by reusing samples from the proposal:

$$\begin{aligned} Z &= \int \frac{\tilde{p}(\boldsymbol{z}')}{q(\boldsymbol{z}')} q(\boldsymbol{z}') \, \mathrm{d}\boldsymbol{z}' = \mathbb{E}_{q(\boldsymbol{z}')}\left[\frac{\tilde{p}(\boldsymbol{z}')}{q(\boldsymbol{z}')}\right] \\ &\approx \frac{1}{J} \sum_{\ell=1}^{J} \underbrace{\frac{\tilde{p}(\boldsymbol{z}_\ell)}{q(\boldsymbol{z}_\ell)}}_{\tilde{w}(\boldsymbol{z}_\ell)} \coloneqq \hat{Z}_{IS}. \end{aligned} \tag{2.31}$$

The expected value $g^\star$ can now be approximated using (2.29), replacing $w(\boldsymbol{z}_j)$ with approximate self-normalised weights:

$$g^\star \approx \hat{g}_{IS} \approx \sum_{j=1}^{J} \frac{\tilde{w}(\boldsymbol{z}_j)}{\sum_{\ell=1}^{J} \tilde{w}(\boldsymbol{z}_\ell)} g(\boldsymbol{z}_j). \tag{2.32}$$

IS can also be used to approximately sample from $p(\boldsymbol{z})$, even in the case when the PDF is unnormalised. The idea is to draw J i.i.d samples $\boldsymbol{z}_j \sim q(\boldsymbol{z})$ and compute the corresponding weights $w(\boldsymbol{z}_j)$ for $j = 1, \ldots, J$, computed either as $\frac{p(\boldsymbol{z})}{q(\boldsymbol{z})}$ or with self-normalisation. The weights are used to define a categorical distribution $\hat{p}_{IS}(\boldsymbol{z}) = \text{Categorical}\left([\boldsymbol{z}_1, \ldots, \boldsymbol{z}_J]; [w(\boldsymbol{z}_1), \ldots, w(\boldsymbol{z}_J)]\right)$. Re-weighing the samples draw from q, by sampling from $\hat{p}_{IS}(\boldsymbol{z})$ (with replacement), corrects for the difference between $q(\boldsymbol{z})$ and $p(\boldsymbol{z})$, and will produce approximate samples from $p(\boldsymbol{z})$, with exact sampling in the limit $J \to \infty$ [2, p. 535].

Markov chain Monte Carlo sampling

The usefulness of important sampling-based methods depends on the ability to find a good proposal distribution. This can be difficult for complex target distributions, in particular high-dimensional ones. For such distributions, Markov chain Monte Carlo (MCMC) sampling can be more suitable. Where IS directly produces approximate samples from the target distribution, MCMC samplers instead generate sequences of dependent samples, which converge to the desired target distribution. This enables powerful sampling methods, which can produce approximate samples from complex distributions and MCMC methods are used in a wide range of applications, from statistical physics, computational biology and approximate Bayesian inference [25]–[27].

A Markov chain consists of a sequence of random variables

$$\boldsymbol{z}_{0:T} \coloneqq (\boldsymbol{z}_0, \boldsymbol{z}_1, \ldots, \boldsymbol{z}_T),$$

called states, an initial distribution $p(\boldsymbol{z}_0)$ and transition probabilities $p(\boldsymbol{z}_\tau \mid \boldsymbol{z}_{\tau-1}), \tau = 1, \ldots, T$ (also called a kernel). The whole sequence of states is generated by starting in an initial state $\boldsymbol{z}_0 \sim p(\boldsymbol{z}_0)$ and then stepping through the chain, by sequentially sampling a new state given the previous one.

The joint distribution over the sequence $\boldsymbol{z}_{0:T}$ is specified by the transition probability since a Markov chain fulfils the conditional independence property [23, p. 22]

$$\begin{aligned} p(\boldsymbol{z}_\tau \mid \boldsymbol{z}_{\tau-1}, \ldots, \boldsymbol{z}_0) &= p(\boldsymbol{z}_\tau \mid \boldsymbol{z}_{\tau-1}),\ \tau = 1, \ldots, T, \\ \Rightarrow p(\boldsymbol{z}_{0:T}) &= p(\boldsymbol{z}_0) \prod_{\tau=1}^{T} p(\boldsymbol{z}_\tau \mid \boldsymbol{z}_{\tau-1}) \end{aligned} \tag{2.33}$$

This property is called a Markov property and can be represented with a chain-like directed graph, shown in figure 2.6. The conditional independence means that knowing the value in the previous state is sufficient for determining the distribution of the current state.

Due to the Markov property, the marginal distribution for $\boldsymbol{z}_\tau$ can be expressed in terms of the marginal distribution of the previous state $\boldsymbol{z}_{\tau-1}$

$$p(\boldsymbol{z}_\tau) = \mathbb{E}_{p(\boldsymbol{z}_{\tau-1})}\left[p(\boldsymbol{z}_\tau \mid \boldsymbol{z}_{\tau-1})\right] = \int p(\boldsymbol{z}_\tau \mid \boldsymbol{z}_{\tau-1}) p(\boldsymbol{z}_{\tau-1})\, \mathrm{d}\boldsymbol{z}_{\tau-1}. \tag{2.34}$$

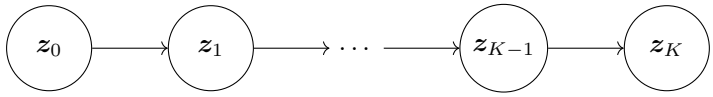

Figure 2.6: Example of a Markov chain. The directed edges encode the conditional independence, where the distribution of $\boldsymbol{z}_\tau$ only depends on the previous state $\boldsymbol{z}_{\tau-1}$.

If a distribution $p^\star(z)$ fulfills the property

$$p^\star(\boldsymbol{z}') = \mathbb{E}_{p^\star(\boldsymbol{z})}\left[p(\boldsymbol{z}' \mid \boldsymbol{z})\right], \tag{2.35}$$

then it is called invariant w.r.t. to the Markov chain with transition probability $p(\boldsymbol{z}' \mid \boldsymbol{z})$. This means that a step in the Markov chain will not change the marginal distribution of $\boldsymbol{z}'$ if the original sample is drawn from $p^\star(\boldsymbol{z})$. Another way of stating it is that the Markov chain has converged to $p^\star$ since further steps in the chain will not change the marginal distribution. Therefore $p^\star$ is also called the stationary distribution.

If a Markov chain has the property that $p(\boldsymbol{z}_T)$ converges to an invariant distribution $p^\star$ as $T \to \infty$, regardless of the initial distribution $p(\boldsymbol{z}_0)$, then it is called ergodic. This means that an ergodic Markov chain can start in any state $\boldsymbol{z}_0$ and if it takes sufficiently many steps, then $\boldsymbol{z}_T$ will be approximately drawn from $p^\star$.

The idea in MCMC is to define an ergodic Markov chain, which has the target distribution as its invariant distribution. The Markov chain can then be used to (approximately) sample from the target distribution and the samples can be used to estimate expected values of the form in (2.27). The caveat is that the chain may have to be run for many steps before it converges to the target distribution. Furthermore, the samples $\boldsymbol{z}_{0:T}$ will be auto-correlated, which will adversely affect the variance of the MC estimate [23, p. 24]. The correlation issue can be mitigated by running multiple Markov chains in parallel. Parallel chains do not directly improve the convergence, but will at least produce more samples in the time it takes to run a single chain.

It is typically not difficult to construct invariant and ergodic Markov chains [28], [29]. The challenge is rather to achieve good mixing, which means the ability to quickly reach the target distribution from an arbitrary starting distribution, while also keeping the auto-correlation low. Furthermore, it can

also be difficult to determine when the chain has run for sufficiently many steps to approximate the target distribution [28]. The following sections present a few MCMC methods, relevant to this thesis.

Metropolis–Hastings

One way to ensure an invariant distribution $p^\star(\boldsymbol{z})$ w.r.t. to a Markov chain with transition probability $p(\boldsymbol{z}' \mid \boldsymbol{z})$ is to require that the two distributions satisfy the detailed balance condition [23, p. 24]:

$$p^\star(\boldsymbol{z})p(\boldsymbol{z}' \mid \boldsymbol{z}) = p^\star(\boldsymbol{z}')p(\boldsymbol{z} \mid \boldsymbol{z}'). \tag{2.36}$$

This condition can be interpreted as a form of symmetry, where the pair $(\boldsymbol{z}', \boldsymbol{z})$ can be generated in two ways, either by starting in $\boldsymbol{z}$ and taking a step in the Markov chain to $\boldsymbol{z}'$ or reversing the step and going backwards in the chain from $\boldsymbol{z}'$ to $\boldsymbol{z}$.

It is easy to verify that detailed balance implies the invariance condition in (2.35):

$$\begin{aligned}\mathbb{E}_{p^\star(\boldsymbol{z})}\left[p(\boldsymbol{z}' \mid \boldsymbol{z})\right] &= \int p(\boldsymbol{z}' \mid \boldsymbol{z})p^\star(\boldsymbol{z}) \,\mathrm{d}\boldsymbol{z} = \int p(\boldsymbol{z} \mid \boldsymbol{z}')p^\star(\boldsymbol{z}') \,\mathrm{d}\boldsymbol{z} \\ &= p^\star(\boldsymbol{z}') \int p(\boldsymbol{z} \mid \boldsymbol{z}') \,\mathrm{d}\boldsymbol{z} = p^\star(\boldsymbol{z}').\end{aligned} \tag{2.37}$$

The Metropolis–Hastings (MH) method is an MCMC variant, which provides a flexible way of constructing valid MCMC kernels [30]–[32]. The kernel is based on a proposal distribution $q(\boldsymbol{z}' \mid \boldsymbol{z})$ (which may depend on the step τ). The proposal distribution can be chosen with few restrictions, and a valid MCMC kernel is instead ensured by adding a rejection step in the sampling. Given a current state $\boldsymbol{z}$, a proposal sample $\boldsymbol{z}' \sim q(\boldsymbol{z}' \mid \boldsymbol{z})$ is randomly either accepted or rejected with an acceptance probability $\alpha(\boldsymbol{z}', \boldsymbol{z}) \in [0, 1]$, i.e.,

$$\boldsymbol{z}_\tau = \begin{cases} \boldsymbol{z}' \sim q(\boldsymbol{z}' \mid \boldsymbol{z}_{\tau-1}), & \text{with prob. } \alpha(\boldsymbol{z}', \boldsymbol{z}_{\tau-1}), \\ \boldsymbol{z}_{\tau-1}, & \text{with prob. } 1 - \alpha(\boldsymbol{z}', \boldsymbol{z}_{\tau-1}). \end{cases} \tag{2.38}$$

In effect, the sampling process is a random walk, where some of the proposed steps are rejected. To achieve good mixing, the proposal distribution should be chosen such that the random walk explores the state space efficiently, yet

still maintains a reasonably high acceptance rate (what this value actually is, is subject to debate [33]–[35]).

The detailed balance condition can be used to derive the relative acceptance probability for the two ways the pair $(\boldsymbol{z}', \boldsymbol{z})$ could have been generated:

$$p(\boldsymbol{z})p(\boldsymbol{z}' \mid \boldsymbol{z}) = p(\boldsymbol{z}')p(\boldsymbol{z} \mid \boldsymbol{z}') \Rightarrow \frac{p(\boldsymbol{z}')}{p(\boldsymbol{z})} = \frac{p(\boldsymbol{z}' \mid \boldsymbol{z})}{p(\boldsymbol{z} \mid \boldsymbol{z}')} = \frac{q(\boldsymbol{z}' \mid \boldsymbol{z})\alpha(\boldsymbol{z}', \boldsymbol{z})}{q(\boldsymbol{z} \mid \boldsymbol{z}')\alpha(\boldsymbol{z}, \boldsymbol{z}')}$$
$$\Rightarrow \frac{\alpha(\boldsymbol{z}', \boldsymbol{z})}{\alpha(\boldsymbol{z}, \boldsymbol{z}')} = \frac{p(\boldsymbol{z}')q(\boldsymbol{z} \mid \boldsymbol{z}')}{p(\boldsymbol{z})q(\boldsymbol{z}' \mid \boldsymbol{z})}. \tag{2.39}$$

The optimal acceptance probability is a compromise between ergodicity and stationarity. Ideally, new samples should always be accepted since we want to explore the state space, but this may not produce the correct invariant distribution $p^\star$. The MH method uses the acceptance probability

$$\alpha(\boldsymbol{z}', \boldsymbol{z}) = \min\left(\frac{p(\boldsymbol{z}')q(\boldsymbol{z} \mid \boldsymbol{z}')}{p(\boldsymbol{z})q(\boldsymbol{z}' \mid \boldsymbol{z})}, 1\right), \tag{2.40}$$

which can be shown to fulfil the detailed balance condition [2, p. 541]. It is, in fact, the optimal choice in the sense that it gives the highest possible acceptance probability while still maintaining the correct invariant distribution. Note that the normalisation constant of the target distribution cancels, and it is only necessary to evaluate the unnormalised distribution (see (2.30)).

Score-based sampling

In the MH method, new states are proposed by a random walk, which can render exploration inefficient. To improve on this, several hybrid Monte Carlo methods have been proposed, which use information about the target distribution to propose new samples. They are inspired by simulations of dynamical systems, which use a specific probability distribution $q(z) \propto \exp(-E(\boldsymbol{z}))$ (called the canonical distribution) over the states, where $E(\boldsymbol{z})$ is the energy in the state $\boldsymbol{z}$. We will, however, introduce them as methods to sample from a general target distribution $p(\boldsymbol{z})$.

The main idea is to construct a proposal distribution, based on the derivative of $p(\boldsymbol{z})$, such that proposed samples are biased towards regions of higher probability under the target distribution. Specifically, we define the score

function as

$$\boldsymbol{s}(\boldsymbol{z}) = \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}), \tag{2.41}$$

assuming that this derivative can be evaluated.

The definition of the score function comes from the fact that for the special case $\log p(\boldsymbol{z}) = -E(\boldsymbol{z})$, the target distribution $p(\boldsymbol{z})$ is equal to the canonical distribution $q(\boldsymbol{z})$. A more practical reason, however, is that by taking the logarithm of $p(\boldsymbol{z})$, we only need access to the unnormalised distribution, since the normalisation constant does not depend on $\boldsymbol{z}$, i.e.,

$$\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}), = \nabla_{\boldsymbol{z}} \log \tilde{p}(\boldsymbol{z}) - \nabla_{\boldsymbol{z}} \log Z = \nabla_{\boldsymbol{z}} \log \tilde{p}(\boldsymbol{z}). \tag{2.42}$$

A proposal distribution can then be constructed, where we start in $\boldsymbol{z}$, take a step along the direction of the score $\boldsymbol{s}(\boldsymbol{z})$ and add noise. The simplest form of score-based MCMC sampling is the unadjusted Langevin algorithm (ULA) [22, p. 496], which proposes new samples as

$$\boldsymbol{z}' = \boldsymbol{z} + \frac{\delta^2}{2} \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z}) + \delta \boldsymbol{\xi}, \quad \boldsymbol{\xi} \sim \mathcal{N}(0, I). \tag{2.43}$$

Here, δ^2 is a hyperparameter controlling the typical step length. Too small a step will cause slow mixing, whereas too large a step might make the samples fail to converge to the desired target distribution.

From this basic idea, more samplers can be constructed, such as the Metropolis-Adjusted Langevin Algorithm [36], which adds an MH correction step, allowing for larger step lengths and Hamiltonian Monte Carlo, which uses more complex dynamics to produce better proposals [37].

CHAPTER 3

Probabilistic machine learning

3.1 Machine learning

Machine learning is a field of mathematical modelling in which the model is not explicitly derived from mathematical principles but rather learned from data. This is particularly useful for problems where constructing an explicit model may be very difficult or even impossible. Consider, for example, the task of image classification. This task is often trivial for the human mind; indeed, even very young children can take a quick look at a picture and deduce which class it belongs to (say, a flower or a car). To explain how this process works, however, is another thing altogether. That is, to explain how the combination of values of each pixel relates to that picture belonging to the class of flower pictures, for instance, is very challenging.

Machine learning does not attempt to explicitly describe this kind of relation. Instead, it introduces a general function $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, defined by a parameter vector $\boldsymbol{\theta}$ that maps input values (e.g., images) $\boldsymbol{x} \in \mathcal{X}$ to output values (e.g., class labels) $\boldsymbol{y} \in \mathcal{Y}$. Each parameter vector defines a specific function, so technically $f_{\boldsymbol{\theta}}$ is a whole class of functions. Ideally, the unknown true model is a member of the model class $f_{\boldsymbol{\theta}}$, such that there exists an optimal parameter

vector $\boldsymbol{\theta}^\star$ for which $f_{\boldsymbol{\theta}^\star}$ describes the true relation between input and output. This is almost never the case, but if $f_{\boldsymbol{\theta}}$ is sufficiently expressive, meaning that it can model complex functions, the best model in our class of functions may provide a reasonable approximation to the true model.

The true mapping between input and output is unknown, but we generally have at our disposal a data set of training examples $\mathcal{D}$. What the data set contains, depends on the particular problem we study. It can, for example, be only input data $\boldsymbol{x}$ or be pairs of input data and a corresponding true label (often called the ground truth) $(\boldsymbol{x}, \boldsymbol{y})$. Common for all settings is that we assume that the data set $\mathcal{D}$ consists of N independent and identically distributed samples (i.i.d), drawn from the true joint probability distribution $p_{\mathcal{D}}(\cdot)$.

Loss function

To proceed, we would like a principled strategy to use the available data to approximate the unknown function of interest. In machine learning, model parameters are almost always selected such that they minimise a loss function, which quantifies how well the model prediction matches real data.

Let us, for clarity, assume that each data point consists of pairs of input data $\boldsymbol{x}_n$ and label $\boldsymbol{y}_n$, that is $\mathcal{D} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{i=1}^N$. From $(\boldsymbol{x}, \boldsymbol{y})$, assume that we obtain the model prediction as $\hat{\boldsymbol{y}} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) \in \mathcal{Y}$. We define a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$, based on the real and predicted label. The loss function (also called objective function or criterion) maps a prediction $\hat{\boldsymbol{y}}$ and true output $\boldsymbol{y}$ to a non-negative number, encoding the quality of the prediction. The loss is, by convention, interpreted as something that should be minimised; the smaller the loss is, the better the prediction corresponds to the true label. This way, the loss function specifies what problem we are trying to solve. For a regression problem, for example, we would use the squared error $\ell(\hat{\boldsymbol{y}}, \boldsymbol{y}) = \|\hat{\boldsymbol{y}} - \boldsymbol{y}\|_2$, indicating that a good prediction is one that is as close as possible to the true output. Since we are interested in how $\boldsymbol{\theta}$ affects $\hat{\boldsymbol{y}}$ for a given $\boldsymbol{x}$, the loss function is typically written on the form

$$\ell(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) \coloneqq \ell(f_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y}). \tag{3.1}$$

Naturally, we are not interested in the prediction of a single data point. Instead, the model should produce good predictions for all data points, i.e., to

minimise the expected value of the loss w.r.t. to $p_{\mathcal{D}}$, called the risk function $\mathcal{L}(\boldsymbol{\theta})$. This expected value is not possible to compute in general, but it can be estimated as the average loss across the whole data set, which is called the empirical risk function and is an unbiased estimate of $\mathcal{L}(\boldsymbol{\theta})$. This turns the problem of finding the optimal parameters $\boldsymbol{\theta}^\star$ of the model into an optimisation problem:

$$\begin{aligned}\boldsymbol{\theta}^\star &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathcal{L}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \mathbb{E}_{(\boldsymbol{x},\boldsymbol{y})\sim p_{\mathcal{D}}}\left[\ell(\boldsymbol{\theta};\boldsymbol{x},\boldsymbol{y})\right] \\ &\approx \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \hat{\mathcal{L}}(\boldsymbol{\theta}) = \underset{\boldsymbol{\theta}}{\operatorname{argmin}}\ \frac{1}{N}\sum_{n=1}^{N}\ell(\boldsymbol{\theta};\boldsymbol{x}_n,\boldsymbol{y}_n).\end{aligned} \tag{3.2}$$

Note that the term risk is somewhat rare in subsets of the machine learning literature and the term loss is used to denote both the loss function, as well as the risk function (a sin committed by yours truly in some of the papers included in this thesis). Regardless, when we refer to training or learning a model to perform a specific task, we mean selecting parameters that minimise the (empirical) risk.

Optimisation

The choice of method for solving the optimisation problem in (3.2), depends on the loss function, the type of data and most importantly, on the model class $f_{\boldsymbol{\theta}}$. The machine learning models studied in this thesis, are primarily neural networks, which are almost exclusively optimised with gradient descent (GD) methods [2, p. 240]. This is because higher-order derivatives are increasingly computationally expensive to compute as the dimension of the parameter vector grows, making most methods which require higher-order derivatives infeasible.

The basic GD method is a procedure in which the parameters are initialised to some value $\boldsymbol{\theta}^{(0)}$ and then iteratively updated in the negative gradient direction. The update step at iteration step i is:

$$\boldsymbol{\theta}^{(i+1)} = \boldsymbol{\theta}^{(i+1)} - \eta^{(i)}\nabla_{\boldsymbol{\theta}}\hat{\mathcal{L}}(\boldsymbol{\theta}), \tag{3.3}$$

where $\eta^{(i)} \in (0,\infty)$ is a hyperparameter called the learning rate (a hyperparameter is a parameter, external to the model which is not directly optimised

along with $\boldsymbol{\theta}$). Under some mild conditions on $\eta^{(i)}$ the parameter $\boldsymbol{\theta}^{(i)}$ will converge to a local optima of $\hat{\mathcal{L}}$ as i goes to infinity [38].

In practice, it is often computationally infeasible to calculate $\nabla_{\boldsymbol{\theta}}\hat{\mathcal{L}}(\boldsymbol{\theta})$ over all the training data. Instead, the parameter update is done with a noisy gradient estimate based on a random subset of the training data, called a mini-batch, resulting in the stochastic GD method (SGD). The introduced stochasticity in the optimisation process will result in higher variance in the estimates of $\nabla_{\boldsymbol{\theta}}\mathcal{L}(\boldsymbol{\theta})$, but the mini-batch gradient estimates of $\nabla_{\boldsymbol{\theta}}\hat{\mathcal{L}}(\boldsymbol{\theta})$ are unbiased, meaning that the steps, on average, will be in the GD direction [39, p. 191].

3.2 Neural networks

Neural networks is a large class of models which are particularly well-suited for machine learning. Originally, neural networks were modelled on the human brain, but nowadays, this principle rarely guides the development of new types of neural networks. Most neural networks of interest are parametric functions with a particular structure called a directed acyclic graph (DAG) [40], An example of a neural network with a DAG structure is shown in figure 3.1. This is also almost always what people actually mean when they refer to a neural network [2], [41], [42]. One key reason for the prevalence of networks with the DAG structure is that they are convenient to optimise using GD methods, even with a very large number of parameters.

We use a specific terminology to describe a DAG that represents a neural network. The nodes in the graph are called neurons and figure 3.1 shows them arranged in L horizontal layers, the neurons in layer l are associated with bias vector $\boldsymbol{b}^{[l]}$. The edges that connect neurons from one layer to the next are each associated with a scalar value, and together they constitute a weight matrix $\boldsymbol{W}^{[l]}$. The set of all weight matrices and bias vectors is the learnable parameters $\boldsymbol{\theta}$ of the network $f_{\boldsymbol{\theta}}$.

The values of neurons in layer l (called activations $\boldsymbol{h}^{[l]}$) are commonly computed by propagating the activations of the previous layer $\boldsymbol{h}^{[l-1]}$ through an affine transformation

$$\boldsymbol{h}^{[l]} = f_{\boldsymbol{\theta}}^{[l]}(\boldsymbol{h}^{[l-1]}) = \sigma^{[l]}\left(\boldsymbol{W}^{[l]}\boldsymbol{h}^{[l-1]} + \boldsymbol{b}^{[l]}\right), \tag{3.4}$$

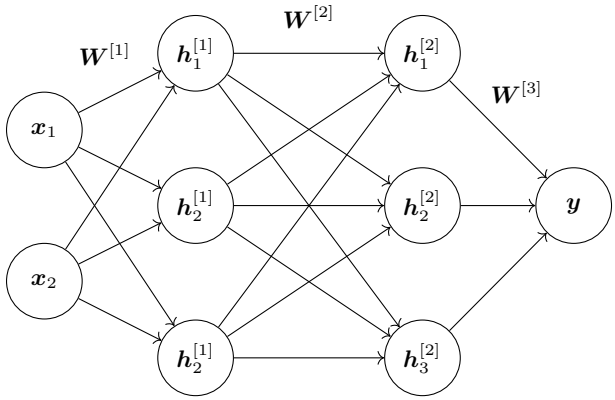

Figure 3.1: A neural network represented by a DAG, here with an input layer, two hidden layers and a single-neuron output layer. The directed edges between the layers represent the flow of information from the input to the output layers. Note that there are neither no edges going backwards to previous layers, nor is there any edges between neurons in the same layer. The directed edges are associated with a weight, where all the edges between layers comprise the weight matrices $\boldsymbol{W}^{[l]}, l = 1, 2, 3$.

followed by an element-wise activation function $\sigma^{[l]}(\cdot)$. The activations in the first layer (called the input layer) are simply the value of the input $\boldsymbol{h}^{[1]} = \boldsymbol{x}$. The whole network then constitutes a mapping from input space to output space:

$$\boldsymbol{y} = f_{\boldsymbol{\theta}}(\boldsymbol{x}) = f_{\boldsymbol{\theta}}^{[L]}(f_{\boldsymbol{\theta}}^{[L-1]}(\dots f_{\boldsymbol{\theta}}^{[1]}(\boldsymbol{x}))). \tag{3.5}$$

The activation functions serve to add non-linearities to the model. Without them the entire network would simply consist of a sequence of nested affine transformations (which itself is an affine transformation). Common activation functions are tanh, the sigmoid function and rectified linear unit (ReLU) [42, p. 168].

The reason that the DAG structure is suitable for GD-based optimisation is that it enables efficient, automatic differentiation of the network output, w.r.t. to its parameters, i.e., to compute $\frac{\partial f_{\boldsymbol{\theta}}(\boldsymbol{x})}{\partial \boldsymbol{\theta}}$, and subsequently the derivatives of the loss function. Specifically, the DAG structure of the network ensures that the derivative of the parameters in layer l can be computed using only the information about the derivatives in layer $l+1$ [41, p. 234]. This method of differentiation, where the derivative information flows backwards through the network, is called backpropagation [43].

Consequently, we can compose different layers into neural networks of arbitrary complexity and, as long as we specify how to differentiate the output of each layer w.r.t. to its input, backpropagation provides an algorithm to automatically compute the derivatives of the entire network. With these derivatives, SGD can be used to train networks with billions of parameters, making backpropagation one of the most important factors behind the success of neural networks.

Therefore, a substantial part of the research in the machine learning field has been dedicated to constructing new types of layers and new ways of composing them to solve problems of ever-increasing complexity. The functional form of the network is called its architecture. New architectures are devised to create models with specific properties. Notable examples are convolutional neural networks (CNN), which were developed for image data to have translationally equivariant layers and transformers which model unordered set-to-set mappings. For the purposes of this thesis, specific model architectures are not that important and we mostly treat the neural networks as black-box mappings from input to output.

3.3 A probabilistic view of machine learning

Whether the real world is fundamentally predictable is a philosophical question that this thesis does not settle. You could perhaps predict the outcome of a coin toss if you had access to all information about the situation, but in terms of modelling, it is often more useful to predict the probability of different outcomes, rather than guessing on a specific one. It is, therefore, natural to frame machine learning in a probabilistic setting.

An important benefit of the probabilistic view is that it provides a principled way of modelling uncertainties. This is essential, for instance when a model's predictions are used to make decisions in autonomous systems. Consider a model used in a self-driving car, which predicts whether the road ahead is free of obstacles. Here, a prediction with full confidence and a 51% prediction should definitely lead to different control decisions from the car's computer, but if the model can only give a binary yes or no output, then these scenarios are identical as far as the model is concerned.

Probabilistic machine learning can mean many different things. It has, for instance, been a long-standing goal to develop Bayesian neural networks in which the network parameters are treated as a random variable and to use Bayes' rule in (2.1) to infer not just a point estimate $\boldsymbol{\theta}^\star$, but the full posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$. However, inference methods have struggled to keep up with the fast-increasing complexity of the neural networks they are supposed to estimate. A compromise is to develop methods to approximately sample parameters from $p(\boldsymbol{\theta} \mid \mathcal{D})$, such as ensembles, MC-dropout and stochastic weight averaging (SWA) [44]–[46].

Probabilistic matching learning can also mean to use neural networks to represent probability distributions. Here, the distinction between probabilistic and ordinary machine learning is much vaguer, since some machine learning tasks, like classification, have typically been posed as probabilistic, while others, like regression, have not. Probabilistic models can model more complex problems by combining classification and regression tasks in a probabilistic way. Examples of this are object detection, which combines classification and regression to predict the likely presence and location of objects [47] and semantic segmentation which in essence is a joint classification of every part of the input data [48], [49].

There are also tasks which are exclusive to the field of probabilistic machine learning, such as generative modelling. Here, major breakthroughs have been

made in recent years, notably in text and image generation.

Maximum likelihood estimation

One core component in most probabilistic machine learning is maximum likelihood (ML) estimation, which is the process of selecting the model parameters that maximise the likelihood of the input data. As a loss function, it is formulated in terms of the negative logarithm of the likelihood function (NLL)

$$\ell^{\text{NLL}}(\boldsymbol{\theta}; \boldsymbol{x}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}), \tag{3.6}$$

where $p_{\boldsymbol{\theta}}$ constitutes the probabilistic model, i.e., a probability distribution, parameterised by $\boldsymbol{\theta}$.

ML estimation differs from the Bayesian inference we introduced in chapter 2 in a few important ways. Here, $\boldsymbol{\theta}$ is not a random variable, it is simply a variable that we find through optimisation. We can make it more similar to Bayesian inference, by introducing a prior distribution over $\boldsymbol{\theta}$ and instead optimise the product $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})p(\boldsymbol{\theta})$. From Bayes' rule, we know that this objective is proportional to the posterior distribution of $\boldsymbol{\theta}$ and the outcome of the optimisation would instead be the Maximum a posteriori (MAP) estimate. However, we note that both the ML and MAP estimates are point estimates, where the ultimate objective in Bayesian inference is to provide the full posterior distribution over the random variable $\boldsymbol{\theta}$.

While exact Bayesian inference may be more appealing in principle, it is often intractable and then ML (or MAP) estimation is a more practical choice. We will now introduce two important problems where ML estimation is used.

Classification

Classification is a machine learning task that is naturally represented in a probabilistic way. At first glance, the task of predicting a class label $y \in \{1, 2, \ldots, C\}$ for a given input $\boldsymbol{x} \in \mathcal{X}$ may come across as ill-suited for a model which outputs a continuous value, as an object's class can not be in between a car and a flower. Instead, we construct a model which outputs a conditional categorical PMF $p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x}) = \text{Categorical}\left(\boldsymbol{y}; \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x})\right)$, where $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is a probability vector, such that each element in the vector $\boldsymbol{\pi}_{\boldsymbol{\theta},c}(\boldsymbol{x}) \in [0, 1], c = 1, \ldots, C$ and $\sum_{c'}^{C} \boldsymbol{\pi}_{\boldsymbol{\theta},c} = 1$. Note, that with this formulation, the model output

is differentiable w.r.t. $\boldsymbol{\theta}$, enabling the use of GD-based methods.

The probability vector $\boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{x})$ is usually computed in two steps. First, a learnable model (e.g., a neural network) maps input to an unconstrained vector $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathbb{R}^C$, then a fixed mapping transforms the output into a valid probability vector. The elements in the output vector $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ are called logits, and each logit is mapped to a value $\boldsymbol{\pi}_{\boldsymbol{\theta},c}(\boldsymbol{x})$, corresponding to the probability of class c. This mapping is called the soft-max function and is defined as

$$\boldsymbol{\pi}_{\boldsymbol{\theta},c}(\boldsymbol{x}) = \frac{\exp(f_{\boldsymbol{\theta}}(\boldsymbol{x}))_c}{\sum_{c'=1}^{C} \exp(f_{\boldsymbol{\theta}}(\boldsymbol{x}))_{c'}}. \tag{3.7}$$

It is easy to verify that the soft-max function produces a valid probability vector $\boldsymbol{\pi}_{\boldsymbol{\theta}}$, and together with $f_{\boldsymbol{\theta}}(\boldsymbol{x})$, it completes the probabilistic model:

$$p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x}) = \text{Categorical}\left(y; \text{soft-max}\left(f_{\boldsymbol{\theta}}(\boldsymbol{x})\right)\right). \tag{3.8}$$

The parameters $\boldsymbol{\theta}$ are estimated by optimising the NLL loss in (3.6) to learn $\boldsymbol{\theta}$:

$$\ell^{\text{NLL}}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) = -\log p(y \mid \boldsymbol{x}) = -\log \boldsymbol{\pi}_{\boldsymbol{\theta},y}(\boldsymbol{x}). \tag{3.9}$$

Regression

A regression model, which predicts a continuous output $\boldsymbol{y}$ from an input $\boldsymbol{x}$, need not be probabilistic. It is perfectly fine to have a model $f_{\boldsymbol{\theta}} : \mathcal{X} \to \mathcal{Y}$, with $\mathcal{Y}$ being some continuous space, and optimise $\boldsymbol{\theta}$ based on, for instance, the squared error. It can, however, easily be converted to a probabilistic model, for example by assuming that $\boldsymbol{y}|\boldsymbol{x}$ is normally distributed, and modifying the model architecture such that $f_{\boldsymbol{\theta}}(\boldsymbol{x})$ now outputs a mean vector $\boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x})$ and a scalar $\log \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})$:

$$p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{y}; \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}), \sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x}) I\right). \tag{3.10}$$

Note that the model here outputs $\log \sigma^2_{\boldsymbol{\theta}} \in \mathbb{R}$ so that exponentiation ensures a positive variance. It is also possible to parameterise the model such that it outputs a square matrix $A_{\boldsymbol{\theta}}(\boldsymbol{x})$ and define $\Sigma_{\boldsymbol{\theta}}(\boldsymbol{x}) = A_{\boldsymbol{\theta}}(\boldsymbol{x}) A_{\boldsymbol{\theta}}(\boldsymbol{x})^\top$ to predict a full, positive semi-definite, covariance matrix.

We can use the NLL loss to learn $\boldsymbol{\theta}$

$$\ell^{\mathrm{NLL}}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{y}) = -\log p(y \mid \boldsymbol{x}) = \frac{1}{2\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})} \left\| \boldsymbol{y} - \boldsymbol{\mu}_{\boldsymbol{\theta}}(\boldsymbol{x}) \right\|_2^2 + \log \sigma_{\boldsymbol{\theta}} + C, \quad (3.11)$$

where C is a constant that does not depend on $\boldsymbol{\theta}$. Such constants are typically omitted from loss functions, as they do not change the location of optima. We can make two observations about the NLL loss. Firstly, the NLL loss for this probabilistic model is similar to the squared error loss used in standard regression. The main difference is that the probabilistic model allows for reducing the loss by increasing the extra output $\sigma^2_{\boldsymbol{\theta}}(\boldsymbol{x})$ for regions in $\mathcal{X}$ where the variance in the output is high, but the loss simultaneously penalises large values of $\sigma^2_{\boldsymbol{\theta}}$ overall. This forces the model to prioritise where it predicts high variance, which has a regularising effect. Secondly, both the classification and regression problem are optimised using the NLL loss. By framing them as problems of predicting probability distributions, we have essentially merged the two problem classes and have a common well-defined loss function. This is an appealing property of the probabilistic framework. Henceforth, we will refer to models which predict the distribution of a variable $\boldsymbol{y}$ given an input $\boldsymbol{x}$ as probabilistic predictive models.

3.4 Quantifying uncertainty

The probabilistic framework provides methods for quantifying the uncertainty in a model's output. This enables a principled way of reasoning about and understanding the confidence of predictions, a vital feature for (among others) security-critical applications, e.g., autonomous driving and medical diagnoses. This section introduces a selection of metrics for quantifying the uncertainty in a random variable, the similarity of two probability distributions and the information between random variables.

Among the metrics quantifying the uncertainty in a random variable, the variance (or covariance)

$$\operatorname{Var}[y] \coloneqq \mathbb{E}\left[(y - \mathbb{E}[y])^2\right] = \mathbb{E}\left[y^2\right] - \mathbb{E}[y]^2, \quad (3.12)$$

is perhaps the best known and it is usually the choice for continuous random variables. For discrete random variables, it is common to use the entropy (or

Shannon entropy to distinguish it from the entropy used in physics) [50, p. 13]

$$\mathbb{H}_p\left[\boldsymbol{y}\right] \coloneqq -\sum_{\boldsymbol{y}\in\mathcal{Y}} p(\boldsymbol{y}) \log p(\boldsymbol{y}) = -\mathbb{E}_p\left[\log p(\boldsymbol{y})\right]. \tag{3.13}$$

For continuous variables, it is possible to replace the summation in the RHS with integration over all possible values of $\boldsymbol{y}$ (when the integral is defined), but this results in the differential entropy which has slightly different properties than the standard, discrete entropy [50, p. 224]. It is, therefore incorrect, although common, to conflate the two.

There are also ways of quantifying the similarity between two probability distributions. The Kullback–Leibler (KL) divergence, for instance, is used to compare distributions as the relative entropy

$$\mathrm{KL}\left[p\|q\right] \coloneqq \mathbb{E}_p\left[\log\left(\frac{p}{q}\right)\right], \tag{3.14}$$

which has the properties that it is always non-negative and is zero, only when $p = q$ [50, p. 20]. It is tempting to interpret the KL divergence as some distance metric between p and q, but it should be noted that it is not a true distance as it is not symmetric, i.e., $\mathrm{KL}\left[p\|q\right] \neq \mathrm{KL}\left[q\|p\right]$ in general.

From the KL divergence, we can also introduce the concept of mutual information (MI). The MI between the random variables $\boldsymbol{x}$ and $\boldsymbol{y}$ measures the information gained about $\boldsymbol{x}$ if we observe $\boldsymbol{y}$ (or vice versa). It is based on comparing the difference between the joint distribution and the product of the marginals [50, p. 18]:

$$\mathrm{MI}\left[\boldsymbol{x};\boldsymbol{y}\right] \coloneqq \mathrm{KL}\left[p(\boldsymbol{x},\boldsymbol{y})\|p(\boldsymbol{x})p(\boldsymbol{y})\right]. \tag{3.15}$$

if $\boldsymbol{x}$ and $\boldsymbol{y}$ are independent, then there is no mutual information between them. This can be seen from the definition: for independent $\boldsymbol{x}$ and $\boldsymbol{y}$, the joint distribution will factorise as the product of the marginal distribution, which leads to the KL divergence being zero since the compared distributions are identical. Note that the MI, in contrast to the KL divergence, is indeed symmetric. The MI can also be expressed in terms of (relative) entropies:

$$\mathrm{MI}\left[\boldsymbol{x};\boldsymbol{y}\right] = \mathbb{H}\left[\boldsymbol{x}\right] - \mathbb{H}\left[\boldsymbol{x} \mid \boldsymbol{y}\right] = \mathbb{H}\left[\boldsymbol{y}\right] - \mathbb{H}\left[\boldsymbol{y} \mid \boldsymbol{x}\right], \tag{3.16}$$

which follows from the definitions of the KL divergence and the entropy [50, p. 19]. This relationship also holds for continuous variables but instead uses the differential entropy.

3.5 Uncertainty decomposition

The uncertainty in a prediction can be further analysed by decomposing it into different types of uncertainty. This can be useful as a general tool for better understanding the black-box predictions of the model but it also has more practical uses, such as identifying difficult samples for model training.

Suppose we have obtained a posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$ by using some data $\mathcal{D}$ to update our belief about some model parameters $\boldsymbol{\theta}$. Predictions on new data $\boldsymbol{x}$ are made with the posterior predictive distribution $p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D})$ in (2.4). The uncertainty in this prediction can be decomposed into two components, epistemic and aleatoric uncertainty.

Epistemic uncertainty is the uncertainty in $\boldsymbol{y}$ which stems from uncertainty about the parameter $\boldsymbol{\theta}$, encoded by $p(\boldsymbol{\theta} \mid \mathcal{D})$. This uncertainty can be reduced by observing more data, which will essentially make the posterior distribution sharper. Aleatoric uncertainty, on the other hand, is uncertainty which cannot be reduced, i.e., the inherent noise in $\boldsymbol{y}$, even when we know $\boldsymbol{\theta}$, encoded by $p(\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta})$.

The canonical example is someone trying to cheat you, by gambling with a possibly biased coin. You can reduce the uncertainty about the coin's bias by tossing it many times and estimating the probability of it coming up heads. This experimentation reduces the epistemic uncertainty. After many trials, you may be reasonably sure that the coin is indeed biased, with the probability of heads being roughly 40%. You, therefore, make the bet that the next toss will be tails. There is still a 40% chance of you losing the bet, due to the aleatoric uncertainty, and you cannot be entirely confident in your prediction, even if you observe an infinite number of coin tosses.

We can use the uncertainty metrics previously introduced in this section to quantify the different types of uncertainty in $p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D})$. The total uncertainty in a prediction is

$$U_{\text{tot}} = I\left[\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}\right] = I\left[\mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}\right]\right], \tag{3.17}$$

where I is some uncertainty metric, like the ones defined above. The aleatoric uncertainty is the uncertainty that remains when we condition on the true value of $\boldsymbol{\theta}$, which on average is

$$U_{\text{ale}} = \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[I\left[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}\right]\right]. \tag{3.18}$$

The epistemic uncertainty can be quantified by measuring some form of variability in $\boldsymbol{\theta}$, but it is also common to simply model it as the remaining uncertainty $U_{\text{epi}} = U_{\text{tot}} - U_{\text{ale}}$.

Example 3.5.1 Uncertainty decomposition for a continuous variable. Consider a linear model with a scalar output y and normally distributed likelihood and posterior,

$$\begin{aligned} p(\boldsymbol{\theta} \mid \mathcal{D}) &= \mathcal{N}\left(\boldsymbol{\theta}, \mu_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}}\right), \\ p(y \mid \boldsymbol{x}, \boldsymbol{\theta}) &= \mathcal{N}\left(y; \boldsymbol{\theta}^\top \boldsymbol{x}, \sigma^2\right), \end{aligned} \tag{3.19}$$

for which the posterior predictive distribution is

$$\begin{aligned} p(y \mid \boldsymbol{x}, \mathcal{D}) &= \int \mathcal{N}\left(y; \boldsymbol{\theta}^\top \boldsymbol{x}, \sigma^2\right) \mathcal{N}\left(\boldsymbol{\theta}, \mu_{\boldsymbol{\theta}}, \Sigma_{\boldsymbol{\theta}}\right) \, \mathrm{d}\boldsymbol{\theta} \\ &= \mathcal{N}\left(y; \mu_{\boldsymbol{\theta}}^\top \boldsymbol{x}, \sigma^2 + \boldsymbol{x}^\top \Sigma_{\boldsymbol{\theta}} \boldsymbol{x}\right). \end{aligned} \tag{3.20}$$

For continuous random variables, the variance is often a suitable uncertainty metric, which then leads to an uncertainty decomposition according to (3.17) and (3.18)

$$\begin{aligned} U_{\text{tot}} &= \operatorname{Var}\left[y \mid \boldsymbol{x}, \mathcal{D}\right] = \sigma^2 + \boldsymbol{x}^\top \Sigma_{\boldsymbol{\theta}} \boldsymbol{x}, \\ U_{\text{ale}} &= \mathbb{E}_{p(\boldsymbol{\theta}|\mathcal{D})}\left[\operatorname{Var}\left[y \mid \boldsymbol{x}, \boldsymbol{\theta}\right]\right] = \sigma^2, \\ U_{\text{epi}} &= \boldsymbol{x}^\top \Sigma_{\boldsymbol{\theta}} \boldsymbol{x}. \end{aligned} \tag{3.21}$$

That is, the total uncertainty is the variance parameter in the posterior predictive distribution, while the aleatoric uncertainty is the variance parameter of the likelihood, which is not reduced by observing more data. The epistemic uncertainty is a function of the posterior covariance parameter, which can be reduced since more data will decrease the uncertainty in the posterior belief about $\boldsymbol{\theta}$.

Example 3.5.2 Uncertainty decomposition for a discrete variable. For the second example, consider a discrete random variable $\boldsymbol{y}$, where we instead will use the (conditional) entropy as the uncertainty metric. The resulting uncertainty decomposition is

$$\begin{aligned} U_{\text{tot}} &= \mathbb{H}\left[\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}\right] = \mathbb{H}\left[\mathbb{E}_{p(\boldsymbol{\theta}, \mathcal{D})}\left[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}\right]\right], \\ U_{\text{ale}} &= \mathbb{E}_{p(\boldsymbol{\theta} \mid \mathcal{D})}\left[\mathbb{H}\left[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}\right]\right], \\ U_{\text{epi}} &= \mathbb{H}\left[\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}\right] - \mathbb{E}_{p(\boldsymbol{\theta} \mid \mathcal{D})}\left[\mathbb{H}\left[\boldsymbol{y} \mid \boldsymbol{x}, \boldsymbol{\theta}\right]\right]. \end{aligned} \tag{3.22}$$

We see that the total uncertainty is the entropy of the average probability vector, while the aleatoric uncertainty is the average entropy across all probability vectors under $p(\boldsymbol{\theta} \mid \mathcal{D})$. The epistemic uncertainty is again the remaining uncertainty after the aleatoric is accounted for, but it can actually be shown that it corresponds to the MI between $\boldsymbol{y}$ and $\boldsymbol{\theta}$ [50, p. 20].

The ability to decompose the uncertainty in this way is very useful. On its own, it provides some explainability, in that it identifies the source of the uncertainty in a prediction. It can also provide the basis for performing out-of-distribution detection, i.e., identifying data that do not fit into the original data distribution, as the model uncertainty is typically high for such previously unseen data [51], [52].

3.6 Active learning

Up to this point in the thesis, the assumption has been that a suitable data set has been available for training any model. In practice, this is rarely the case and aggregating a useful data set can constitute the lion's share of work to learn a machine learning model. The worst case is when data overall are scarce, but usually, input data $\boldsymbol{x}$ are abundant and it is the labels, or annotations, $\boldsymbol{y}$ which are hard to come by. The labels typically require some form of human intervention, often by manually annotating a large data set, which is a time-consuming process.

The field of active learning (AL) seeks to modify the label acquisition process, in order to make a more principled selection of input data points to annotate. Compared to the baseline of randomly selecting which data points to label, AL specifically seeks to find the data points that are the most useful for estimating the model parameters[53]. This has the potential to be useful for deep learning

models, which need large data sets to train [54], [55].

The standard AL problem formulation is to start with some small labelled data set $\mathcal{D}_{\mathrm{L}} = \{(\boldsymbol{x}_n, \boldsymbol{y}_n)\}_{n=1}^{N}$ and a larger, unlabelled, data set $\mathcal{D}_{\mathrm{U}} = \{\boldsymbol{x}_m\}_{m=1}^{M}$. The goal is to obtain the best possible model $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$, given a fixed annotation budget, which limits the number of data points that can be labelled.

The selection of unlabelled points to annotate is normally determined by an acquisition function $\phi_{\mathcal{D}_{\mathrm{L}}}(\boldsymbol{x})$, which is supposed to measure the usefulness of annotating a data point $\boldsymbol{x}$. A common AL setup is to acquire data iteratively, where at each step a new data point is moved from the unlabelled to the labelled data set:

$$\begin{aligned} &\mathcal{D}_{\mathrm{L}} \leftarrow \mathcal{D}_{\mathrm{L}} \cup (\boldsymbol{x}_{m^\star}, \boldsymbol{y}_{m^\star}),\ \mathcal{D}_{\mathrm{U}} \leftarrow \mathcal{D}_{\mathrm{U}} \setminus \boldsymbol{x}_{m^\star},\ \text{where} \\ &\boldsymbol{x}_{m^\star} = \underset{\boldsymbol{x}_m \in \mathcal{D}_{\mathrm{U}}}{\operatorname{argmax}}\ \phi_{\mathcal{D}_{\mathrm{L}}}(\boldsymbol{x}_m), \end{aligned} \tag{3.23}$$

where $\boldsymbol{y}_{m^\star}$ is obtained by labelling $\boldsymbol{x}_{m^\star}$. After a new data point and label is acquired, the model is re-trained, using the enlarged labelled data set,

The reason for re-training the model at each step is that the acquisition function $\phi_{\mathcal{D}_{\mathrm{L}}}(\boldsymbol{x})$ is typically defined in terms of the predictions of the model $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$. For classification, with a scalar label $y \in \{1, 2, \ldots, C\}$, examples of acquisition functions are

- the least confident sample: $\phi_{\mathcal{D}_{\mathrm{L}}}(\boldsymbol{x}) = 1 - \max_y p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ [56];
- the margin sample: $\phi_{\mathcal{D}_{\mathrm{L}}}(\boldsymbol{x}) = p_{\boldsymbol{\theta}}(y^{(1)} \mid \boldsymbol{x}) - p_{\boldsymbol{\theta}}(y^{(2)} \mid \boldsymbol{x})$, where $y^{(1)}, y^{(2)}$ are the first and second most probable label [57];
- the highest entropy sample: $\phi_{\mathcal{D}_{\mathrm{L}}}(\boldsymbol{x}) = -\sum_{y=1}^{C} p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x}) \log p_{\boldsymbol{\theta}}(y \mid \boldsymbol{x})$ [58].

These examples all use the posterior predictive distribution to define $\phi_{\mathcal{D}_{\mathrm{L}}}$, where the entropy-based acquisition function is equivalent to the total uncertainty in (3.17). More recently, though, it has been common to use the methods of uncertainty decomposition to instead define $\phi_{\mathcal{D}_{\mathrm{L}}}$ in terms of the epistemic uncertainty [59]–[62]. This approach is called disagreement-based AL [53, p. 15] and it aims to select the data point with the highest model uncertainty. The intuition is that it is better to prioritise data points with high epistemic uncertainty since that is the uncertainty which can be reduced by observing more data, as opposed to the sample with the highest total uncertainty, which also contains the irreducible aleatoric uncertainty.

Further motivation for the disagreement-based AL is the connection between the epistemic uncertainty and the MI. Recall the uncertainty decomposition for a classification model in (3.22) and its connection to the MI. By selecting the data point with the highest epistemic uncertainty, we select the sample that maximises the MI between the label and the parameters, which intuitively makes sense in an AL context.

In practice, AL methods typically assume a binary selection process, where a new label is either acquired with perfect annotation quality or is not labelled at all. In Paper C, we propose a generalisation of the standard AL formulation to a more flexible approach that considers the possibility of acquiring labels at lower quality for a lower cost.

CHAPTER 4

Representations of probability distributions

The previous chapter described the foundations of probabilistic machine learning and how neural networks can be used to model simple probability distributions. This chapter provides further methods for representing probability distributions of increasing complexity.

4.1 Parametric families of distributions

The classification and regression examples in Section 3.3 were examples of models in parametric families of distributions, which is arguably the intuitive way of defining probabilistic machine learning models. The learned models predicted the parameters of a probability distribution over $\boldsymbol{y}$ rather than directly mapping input $\boldsymbol{x}$ to output $\boldsymbol{y}$. This means that whatever architecture or parameters were selected for the neural network in question, the model is still a member of the corresponding distribution family.

Using a parametric distribution has many advantages; the resulting distribution is a well-known type of distribution which can be selected to be a good fit for the data distribution, while still allowing for large expressive models to estimate the parameters. Furthermore, a suitable (and often tractable)

loss function is always available in the form of the NLL loss in (3.6). The downside is that as the complexity of the actual data distribution increases, the constrained nature of the parametric families can limit the model's ability to represent the data accurately.

Ensemble models

A common strategy to increase the representative capability of parametric distributions is to combine several of them into ensemble models. An ensemble consists of a collection of individual models, called members, where each member represents the same probability distribution. Ensembles have been shown to improve overall predictive performance, as well as to make predictions more robust [51].

It is common, but not necessary, for all the members to belong to the same parametric family. The ensemble thus consists of a set of parameter vectors $\{\boldsymbol{\theta}_\ell\}_{\ell=1}^{L}$, where L is the number of members in the model (the ensemble size). The main idea behind ensembles is that aggregating several models' predictions will result in improved modelling performance. The aggregation can be done in multiple ways, but a standard approach is to let the ensemble prediction be the average prediction of the members

$$p_{\{\boldsymbol{\theta}_\ell\}_{\ell=1}^{L}}(\boldsymbol{y} \mid \boldsymbol{x}) = \frac{1}{L}\sum_{\ell=1}^{L} p_{\boldsymbol{\theta}_\ell}(\boldsymbol{y} \mid \boldsymbol{x}). \tag{4.1}$$

Consider the example where the true distribution is multimodal. A single member of the normal distribution family would be ill-suited to represent this, but an ensemble of normal distributions, where each member models its respective mode of the target distribution, would be a good fit. A closely related concept is mixture models, which are also based on averaging multiple distributions. However, in these models, the weight of each distribution is typically also considered a learnable parameter.

For the ensemble to be useful, the members should be distinct. Otherwise, a single member would, of course, suffice. To ensure a diverse ensemble, the parameters of each member should be estimated in different ways. This can be achieved by using different kinds of ensemble members, perhaps sourcing them from multiple parametric families, or using different neural network architectures. Another alternative is to train each model on different subsets

of the data. The most convenient way, however, is to randomly initialise the members' parameters and trust that the randomness of the SGD training will produce sufficiently different members. This approach works well in practice [51].

The set of parameters $\{\boldsymbol{\theta}_\ell\}_{\ell=1}^L$ can be interpreted as approximate samples from the posterior distribution $p(\boldsymbol{\theta} \mid \mathcal{D})$, which connects the ensemble's prediction to the approximate Bayesian inference methods discussed in section 2.3 [51], [63]. With this interpretation, the ensemble prediction in (4.1) is an approximation of the posterior predictive distribution in (2.4):

$$p(\boldsymbol{y} \mid \boldsymbol{x}, \mathcal{D}) = \int p(\boldsymbol{y} \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \mathcal{D}) \, \mathrm{d}\boldsymbol{\theta} \approx \frac{1}{L} \sum_{\ell=1}^{L} p_{\boldsymbol{\theta}_\ell}(\boldsymbol{y} \mid \boldsymbol{x}) = p_{\{\boldsymbol{\theta}_\ell\}_{\ell=1}^L}(\boldsymbol{y} \mid \boldsymbol{x}). \tag{4.2}$$

With the Bayesian interpretation, the ensemble can also be used to perform the uncertainty decomposition presented in section 3.5. The total and aleatoric uncertainty are both quantified by expected values over the posterior distribution, which can be approximated by the same MC estimate as the posterior predictive distribution above:

$$\begin{aligned} U_{\text{tot}} &\approx I\left[p_{\{\boldsymbol{\theta}_\ell\}_{\ell=1}^L}(\boldsymbol{y} \mid \boldsymbol{x})\right], \\ U_{\text{ale}} &\approx \frac{1}{L} \sum_{\ell=1}^{L} I\left[p_{\boldsymbol{\theta}_\ell}(\boldsymbol{y} \mid \boldsymbol{x})\right]. \end{aligned} \tag{4.3}$$

The advantages of the ensemble come with an increased cost, both in terms of computation and memory. Not only does it require L times more resources when training the model, but this extends also to making new predictions, where L models need to be stored in memory and evaluated to make a single prediction. Therefore, it is common to perform ensemble distillation, which is the process of learning a separate model that imitates the predictions of the full ensemble.

The typical distillation setup is to introduce a new model which can be larger than an individual ensemble member, but significantly smaller than the whole ensemble. The distilled model is then trained to imitate the ensemble's predictions, and this can typically result in a model that performs better than an identical model trained only on the training data. This is partly due to the ability of the ensemble to accurately quantify uncertainty and partly because

more data can potentially be used as the distillation process does not require labelled data, as the labels are instead generated by the ensemble.

In Paper B, we propose a distillation method which can compress the ensemble model, while retaining the ability to decompose uncertainty.

4.2 Energy-based models

The parametric families of distribution can be too restrictive, even when multiple models are combined into an ensemble. Again, consider a normal distribution $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{y}; \mu_{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{\Sigma}_{\theta}(\boldsymbol{x}))$ can only represent a unimodal distribution, regardless of the functions modelling the distribution's parameters. Similarly, an ensemble struggles if the modes of the true PDF are not well-represented by any parametric distributions.

A less constrained representation is the energy-based model (EBM) [64]. An EBM is a type of unnormalised generative model, which can represent highly complex probability distributions and has seen use in many different fields, such as object detection, [65]–[68].

An EBM uses a scalar non-negative energy function $E_{\boldsymbol{\theta}} : \mathbb{R}^{D_x} \to \mathbb{R}^+$ to represent a probability distribution $p_{\boldsymbol{\theta}}(\boldsymbol{x})$. The term energy stems from physics and the interpretation is that states with low energy are more probable than high-energy ones. That is, the energy function is supposed to take low values in regions with high probability density and high values where the PDF has little or no support[64].

The PDF of an EBM is defined in terms of the energy function:

$$p_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}))}{Z_{\boldsymbol{\theta}}}, \tag{4.4a}$$

$$Z_{\boldsymbol{\theta}} = \int \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}'))\ \mathrm{d}\boldsymbol{x}', \tag{4.4b}$$

where $Z_{\boldsymbol{\theta}}$ is a normalisation constant (also called the partition function) which ensures a valid probability distribution. The requirement to be able to normalise the PDF means that $E_{\boldsymbol{\theta}}$ must be chosen such that $Z_{\boldsymbol{\theta}} < \infty$. Without it, we could simply increase the likelihood of our model by lowering the energy at every point in the domain of $\boldsymbol{x}$. Therefore, $Z_{\boldsymbol{\theta}}$ effectively regularises the energy function, so that it prioritises the regions of the input space where it assigns low energies. Note, that in some fields, the energy function is scaled

with an additional parameter β, but we will assume $\beta = 1$.

Just as with the parametric families of distributions, energy functions can be modelled with neural networks parameterised by $\boldsymbol{\theta}$. The difference is that the EBM assumes no functional form of the energy function, other than that it should map the input $\boldsymbol{x}$ to a non-negative scalar. A sign of this flexibility is that we can obtain several of the parametric distributions in 4.1 as special cases of EBMs.

Example 4.2.1 A normal distribution as a special case of an EBM. An EBM is equivalent to a normal distribution if the energy function is constrained to the specific form

$$E_{\boldsymbol{\theta}}(\boldsymbol{x}) = \frac{1}{2}(\boldsymbol{x} - \mu_{\boldsymbol{\theta}})^\top \boldsymbol{\Sigma}_{\boldsymbol{\theta}}^{-1}(\boldsymbol{x} - \mu_{\boldsymbol{\theta}}). \tag{4.5}$$

Plugging this function into (4.4a) will recover the PDF of a normal distribution with mean $\mu_{\boldsymbol{\theta}}$ and covariance $\boldsymbol{\Sigma}_{\boldsymbol{\theta}}$. For this special case, the normalisation constant $Z_{\boldsymbol{\theta}}$ is known, since the integral in (4.4b) is tractable.

This increased flexibility comes at the expense of the ability to easily evaluate the PDF of an EBM. In particular, it is difficult to normalise an EBM, except for special cases like the above example. By normalising, we mean to compute the normalisation constant $Z_{\boldsymbol{\theta}}$ in (4.4b), such that the total probability is 1. This integral is intractable for general energy functions. Therefore, the PDF of and EBM can only be evaluated up to the constant $\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}))$. However, we saw in section 2.3, that only having access to an unnormalised distribution can still be useful. Using only $E_{\boldsymbol{\theta}}$, we can evaluate the relative probability of samples and even sample from $p_{\boldsymbol{\theta}}$, but first, we need methods to estimate the EBM parameters $\boldsymbol{\theta}$ from data.

ML estimation with importance sampling

In principle, the parameters of $E_{\boldsymbol{\theta}}$ can be estimated through ML estimation with SGD. From a sample of the underlying data distribution $\boldsymbol{x} \sim p_{\mathcal{D}}(\cdot)$, the NLL criterion in (3.6) is evaluated for the EBM in (4.4a) as

$$\ell^{\text{NLL}}(\boldsymbol{\theta}; \boldsymbol{x}) = -\log p_{\boldsymbol{\theta}}(\boldsymbol{x}) = E_{\boldsymbol{\theta}}(\boldsymbol{x}) + \log Z_{\boldsymbol{\theta}}, \tag{4.6}$$

but since we cannot compute $Z_{\boldsymbol{\theta}}$ in closed form, we cannot directly evaluate the NLL, making a gradient search infeasible.

The most straightforward solution is to estimate $Z_{\boldsymbol{\theta}}$ with importance sampling (see section 2.3). Let $q(\boldsymbol{x})$ be a proposal distribution such that it a) covers the support of $p_{\boldsymbol{\theta}}(x)$ and b) is easy both to sample from and to evaluate its PDF. Draw J i.i.d. samples $\boldsymbol{x}_{1:J} \coloneqq [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_J]$ from q and approximate $Z_{\boldsymbol{\theta}}$ using (2.31). In terms of the EBM, the approximation of the normalisation constant is:

$$Z_{\boldsymbol{\theta}} \approx \hat{Z}_{\boldsymbol{\theta}}^{\text{IS}} = \frac{1}{J} \sum_{j=1}^{J} \tilde{w}(\boldsymbol{x}_j), \quad \text{where } \tilde{w}(\boldsymbol{x}_j) = \frac{\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}_j))}{q(\boldsymbol{x}_j)}. \tag{4.7}$$

Approximating $Z_{\boldsymbol{\theta}}$ in (4.6) with $\hat{Z}_{\boldsymbol{\theta}}^{\text{IS}}$, results in an approximate NLL criterion based on $\boldsymbol{x} \sim p_{\mathcal{D}}(\cdot)$ and $\boldsymbol{x}_j \sim q(\cdot), j = 1, \ldots, J$. To simplify the notation we rename $\boldsymbol{x} = \boldsymbol{x}_0$ to define the sequence $\boldsymbol{x}_{0:J} = [\boldsymbol{x}_0, \boldsymbol{x}_1, \ldots, \boldsymbol{x}_J]$ and for the rest of the EBM section, we assume $\boldsymbol{x}_0 \sim p_{\mathcal{D}}(\cdot)$ and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_J$ are drawn from the proposal distribution q. This leads to an approximate NLL criterion, based on IS

$$\ell^{\text{ML-IS}}(\boldsymbol{\theta}; \boldsymbol{x}_{0:J}) = E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) + \log \hat{Z}_{\boldsymbol{\theta}}^{\text{IS}} = E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) + \log \frac{1}{J} \sum_{j=1}^{J} \tilde{w}(\boldsymbol{x}_j). \tag{4.8}$$

The caveat with using the IS approximation is that, although $\hat{Z}_{\boldsymbol{\theta}}^{\text{IS}}$ is an unbiased estimate of $Z_{\boldsymbol{\theta}}$, the gradient $\nabla \log \hat{Z}_{\boldsymbol{\theta}}^{\text{IS}}$ is not [69], [70]. Thus, IS introduces a bias in the gradients, which can adversely impact the convergence of the SGD algorithm [71].

MCMC sampling and contrastive divergence

Another way of estimating the EBM's parameters is to directly estimate the gradient of the EBM NLL criterion in equation (4.6) w.r.t. the parameters.

This gradient can expressed via the identity[64]:

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \ell^{\mathrm{NLL}}(\boldsymbol{\theta}; \boldsymbol{x}_0) &= \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) + \nabla_{\boldsymbol{\theta}} \log Z_{\boldsymbol{\theta}} = \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) + \frac{1}{Z_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} Z_{\boldsymbol{\theta}} \\ &= \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) + \frac{1}{Z_{\boldsymbol{\theta}}} \int \nabla_{\boldsymbol{\theta}} \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}')) \, \mathrm{d}\boldsymbol{x}' \\ &= \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) - \int \frac{\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}'))}{Z_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}') \, \mathrm{d}\boldsymbol{x}' \\ &= \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) - \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x}')} \left[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}') \right]. \end{aligned} \tag{4.9}$$

This equation enforces our intuition that the NLL is improved by decreasing the energy (recall $\boldsymbol{\theta}$ is shifted in the negative gradient direction) for samples from the data distribution $\boldsymbol{x}_0$, while increasing it for other samples (here, drawn from the model distribution $\boldsymbol{x}' \sim p_{\boldsymbol{\theta}}(\cdot)$). Unfortunately, the gradient in (4.9) is yet again intractable since it includes an expected value w.r.t. to $p_{\boldsymbol{\theta}}$ which generally cannot be computed in closed form. The solution is to approximate it, using sampling techniques from section 2.3, for instance with MCMC sampling. Combining MCMC with SGD has some limitations, as SGD optimisation typically requires a large number of small gradient steps to converge and MCMC sampling requires long Markov chains for the sampling distribution to converge, making the optimisation process prohibitively expensive.

Contrastive divergence (CD) is a method that aims to mitigate this issue [72]. CD uses the approximation

$$\begin{aligned}\nabla_{\boldsymbol{\theta}} \ell^{\mathrm{NLL}}(\boldsymbol{\theta}; \boldsymbol{x}_0) &= \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) - \mathbb{E}_{p_{\boldsymbol{\theta}}(\boldsymbol{x}')} \left[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}') \right] \\ &\approx \nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}_0) - \mathbb{E}_{K_{\boldsymbol{\theta}}(\boldsymbol{x}' | \boldsymbol{x}_0)} \left[\nabla_{\boldsymbol{\theta}} E_{\boldsymbol{\theta}}(\boldsymbol{x}') \right], \end{aligned} \tag{4.10}$$

where $K_{\boldsymbol{\theta}}(\boldsymbol{x}' \mid \boldsymbol{x}_0)$ is an MCMC-kernel, initialised at a sample from the data distribution $\boldsymbol{x}_0$, such that $p_{\boldsymbol{\theta}}$ is invariant w.r.t. $K_{\boldsymbol{\theta}}$. The chain is run for k steps (the variants of CD are subsequently called CD-k) and the k is typically very small. Indeed, taking only a single step (i.e., CD-1) is a common choice. Such a short chain is typically not enough to converge and obtain actual samples from $p_{\boldsymbol{\theta}}$, but the idea in CD is that, by starting the MCMC-chain at $\boldsymbol{x}_0$, samples will be close to the modes of $p_{\boldsymbol{\theta}}$, and thus require fewer steps of the chain to obtain good samples from the model distribution. Furthermore, the argument is that, since the gradient step is repeated many times, it is acceptable if the gradient estimate is slightly off and that, on average, it will

update the parameters in the correct direction. There is empirical evidence that parameters estimated with CD have a smaller bias, compared to the ML estimate [73].

Note, that this derivation of the CD gradient estimate differs slightly from when it was first proposed by Hinton [72]. The original derivation is based on a different criterion, but it ultimately results in the same gradient estimate, and the derivation presented here seems to be more common recently [74], [75].

Noise-contrastive estimation

There are also methods which do not use the NLL criterion in (4.6) and instead introduce alternative criteria for EBM parameter estimation. Noise-contrastive estimation (NCE), is a method that changes the problem formulation from model estimation to learning to discriminate between samples from the true data distribution $p_{\mathcal{D}}$ and samples from a noise distribution p_N [76]. That is, the model is presented with either a real data point or noise and is trained to predict which distribution the point originates from. The motivation for introducing this proxy task is that for a model to properly be able to discriminate between real and noisy samples, it has to learn an accurate representation of the underlying data distribution $p_{\mathcal{D}}$.

The original NCE criterion is derived as a simple binary classification problem, for which a latent variable z is introduced with prior

$$p(z) = \text{Bernoulli}\left(z; 1-\eta\right), \quad \eta \in [0,1]. \tag{4.11}$$

Here, z indicates whether a sample is a real or noisy data point, such that

$$\begin{aligned} p_{\mathcal{D},N}(\boldsymbol{x} \mid z=0) &= p_{\mathcal{D}}(\boldsymbol{x}), \\ p_{\mathcal{D},N}(\boldsymbol{x} \mid z=1) &= p_N(\boldsymbol{x}). \end{aligned} \tag{4.12}$$

The parameter η is the prior probability of sampling $z=0$ and is defined in terms of the number of noisy samples J per real data point:

$$\eta = \frac{1}{1+J} \Rightarrow J = \frac{1-\eta}{\eta}. \tag{4.13}$$

That is, $(1-\eta)/\eta$ corresponds to the average number of noise samples for

every real data sample.

To sample from the joint distribution $p_{\mathcal{D},N}(\boldsymbol{x}, z) = p_{\mathcal{D},N}(\boldsymbol{x}|z)p(z)$, we simply sample z from a Bernoulli coin toss to choose whether to sample $\boldsymbol{x}$ from $p_{\mathcal{D}}$ or p_N. We then pretend that we do not know the origin of $\boldsymbol{x}$ and compute the posterior distribution of z, given x:

$$\begin{aligned} p_{\mathcal{D},N}(z \mid \boldsymbol{x}) &= \frac{p_{\mathcal{D},N}(\boldsymbol{x} \mid z)p(z)}{p_{\mathcal{D},N}(\boldsymbol{x} \mid z=0)p(z=0) + p_{\mathcal{D},N}(\boldsymbol{x} \mid z=1)p(z=1)} \\ &= \frac{(1-z)\eta p_{\mathcal{D}}(\boldsymbol{x}) + z(1-\eta)p_N(\boldsymbol{x})}{\eta p_{\mathcal{D}}(\boldsymbol{x}) + (1-\eta)p_N(\boldsymbol{x})}, \end{aligned} \tag{4.14}$$

wherein the last equality, the binary property of z is used to express the posterior in terms of $p_{\mathcal{D}}$ and p_N. The posterior can be rewritten in terms of J, by dividing the numerator and denominator by η:

$$\begin{aligned} p_{\mathcal{D},N}(z \mid \boldsymbol{x}) &= \frac{(1-z)p_{\mathcal{D}}(\boldsymbol{x}) + z\frac{1-\eta}{\eta}p_N(\boldsymbol{x})}{p_{\mathcal{D}}(\boldsymbol{x}) + \frac{1-\eta}{\eta}p_N(\boldsymbol{x})} \\ &= \frac{(1-z)p_{\mathcal{D}}(\boldsymbol{x}) + zJp_N(\boldsymbol{x})}{p_{\mathcal{D}}(\boldsymbol{x}) + Jp_N(\boldsymbol{x})}. \end{aligned} \tag{4.15}$$

The posterior can not be evaluated since the PDF of the true data distribution $p_{\mathcal{D}}$ is not known. To get a usable criterion for estimating $\boldsymbol{\theta}$, $p_{\mathcal{D}}$ is approximated with the PDF of the EBM $p_{\boldsymbol{\theta}}$, to obtain the NCE criterion as the negative log posterior

$$\ell^{\text{NCE}}(\boldsymbol{\theta}; \boldsymbol{x}, z) = -\log p_{\boldsymbol{\theta},N}(z \mid \boldsymbol{x}) = -\log \frac{(1-z)p_{\boldsymbol{\theta}}(\boldsymbol{x}) + zJp_N(\boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}) + Jp_N(\boldsymbol{x})}, \tag{4.16}$$

but the criterion is still evaluated on samples drawn from $p_{\mathcal{D},N}(\boldsymbol{x}, z)$.

Note, that the NCE criterion requires a normalised distribution $p_{\boldsymbol{\theta}}$. In practice, it is common to estimate $Z_{\boldsymbol{\theta}}$ as an additional, learnable, parameter. The learned model is, therefore, a proper normalised PDF, but this also limits the types of distributions that can be approximated. If we aim to model a conditional distribution $p_{\boldsymbol{\theta}}(\boldsymbol{y} \mid \boldsymbol{x})$, then the normalisation depends on the conditioning, i.e., $Z_{\boldsymbol{\theta}}(\boldsymbol{x})$, making it impractical to estimate as additional parameters. Several modifications and extensions to the original NCE criterion, have been proposed, partly to mitigate this [66], [77]–[81]. Two important extensions are conditional NCE (CNCE) [78] and ranking NCE (RNCE) [79].

In CNCE, the original NCE criterion above is extended to allow a noise distribution $p_N(\boldsymbol{x}' \mid \boldsymbol{x})$ conditioned on a sample from the data distribution $\boldsymbol{x} \sim p_{\mathcal{D}}(\cdot)$. The rationale is, that if p_N should closely resemble $p_{\mathcal{D}}$, then an easy way to achieve this is to add noise to real data samples, with

$$p_N(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}(\boldsymbol{x}'; \boldsymbol{x}, \boldsymbol{\Sigma}), \tag{4.17}$$

being an obvious example (the covariance matrix $\boldsymbol{\Sigma}$ must still be manually tuned).

The CNCE criterion is derived analogous to the original NCE criterion, but conditioning on $\boldsymbol{x} \sim p_{\mathcal{D}}(\cdot)$ has important implications. Again, z is introduced as a latent indicator variable, but the prior is assumed to be $p(z) = \text{Bernoulli}(z; 1/2)$. Now, two data points $\boldsymbol{x}$ and $\boldsymbol{x}'$ are sampled, given z:

$$\begin{aligned} p_{\mathcal{D},N}(\boldsymbol{x}, \boldsymbol{x}' \mid z = 0) &= p_{\mathcal{D}}(\boldsymbol{x}) p_N(\boldsymbol{x}' \mid \boldsymbol{x}) \\ p_{\mathcal{D},N}(\boldsymbol{x}, \boldsymbol{x}' \mid z = 1) &= p_{\mathcal{D}}(\boldsymbol{x}') p_N(\boldsymbol{x} \mid \boldsymbol{x}'). \end{aligned} \tag{4.18}$$

Given a pair $(\boldsymbol{x}, \boldsymbol{x}')$, the posterior distribution of z is computed (again, substituting the unknown $p_{\mathcal{D}}$ for $p_{\boldsymbol{\theta}}$)

$$\begin{aligned} p_{\boldsymbol{\theta},N}(z = 0 \mid \boldsymbol{x}, \boldsymbol{x}') &= \frac{p_{\boldsymbol{\theta},N}(\boldsymbol{x}, \boldsymbol{x}' \mid z = 0) p(z = 0)}{p_{\boldsymbol{\theta},N}(\boldsymbol{x}, \boldsymbol{x}' \mid z = 0) p(z = 0) + p_{\boldsymbol{\theta},N}(\boldsymbol{x}, \boldsymbol{x}' \mid z = 1) p(z = 1)} \\ &= \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}) p_N(\boldsymbol{x}' \mid \boldsymbol{x})}{p_{\boldsymbol{\theta}}(\boldsymbol{x}) p_N(\boldsymbol{x}' \mid \boldsymbol{x}) + p_{\boldsymbol{\theta}}(\boldsymbol{x}') p_N(\boldsymbol{x} \mid \boldsymbol{x}')}, \end{aligned} \tag{4.19}$$

the posterior for $z = 1$ is computed analogously.

In contrast to the NCE posterior in (4.15), the CNCE posterior has a factor $p_{\boldsymbol{\theta}}$ in every term in both numerator and denominator. Recall from (4.4b) that the normalisation constant $Z_{\boldsymbol{\theta}}$ is independent of the input data, which means that $p_{\boldsymbol{\theta}}(\boldsymbol{x})$ and $p_{\boldsymbol{\theta}}(\boldsymbol{x}')$ have the same normalisation. Therefore, $Z_{\boldsymbol{\theta}}$ cancels out in the CNCE posterior and the resulting criterion is

$$\begin{aligned} \ell^{\text{CNCE}}(\boldsymbol{\theta}; \boldsymbol{x}, \boldsymbol{x}', z = 0) &= -\log p_{\boldsymbol{\theta},N}(z = 0 \mid \boldsymbol{x}, \boldsymbol{x}') \\ &= -\log \frac{\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x})) p_N(\boldsymbol{x}' \mid \boldsymbol{x})}{\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x})) p_N(\boldsymbol{x}' \mid \boldsymbol{x}) + \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}')) p_N(\boldsymbol{x} \mid \boldsymbol{x}')}, \end{aligned} \tag{4.20}$$

which does not require a normalised PDF $p_{\boldsymbol{\theta}}$. This property means that ℓ^{CNCE}

can be optimised to estimate $\boldsymbol{\theta}$ without having to ever estimate $Z_{\boldsymbol{\theta}}$, but at the same time, the model $p_{\boldsymbol{\theta}}$ will only be learned up to a constant.

The second important NCE extension is ranking NCE (RNCE). It extends the binary classification problem to a categorical classification problem. Whereas the original NCE method controls the ratio of real and noisy samples with the Bernoulli parameter η, RNCE always sample a single real data point $\boldsymbol{x}_0 \sim p_{\mathcal{D}}(\cdot)$ and $\boldsymbol{x}_{1:J}$ from q. From this variation of the sampling process, the RNCE criterion is derived in much the same way as the other two NCE criteria, by calculating the posterior distribution of $\boldsymbol{x}_0$ being the true sample, i.e.

$$\begin{aligned} p_{\mathcal{D},N}(z = 0 \mid \boldsymbol{x}_{0:J}) &= \frac{p_{\mathcal{D},N}(\boldsymbol{x}_{0:J} \mid z = 0)p(z)}{p_{\mathcal{D},N}(\boldsymbol{x}_{0:J})} \\ &= \frac{p_{\mathcal{D}}(\boldsymbol{x}_0)\prod_{j=1}^{J} p_N(\boldsymbol{x}_j)p(z = 0)}{\sum_{j=0}^{J} p_{\mathcal{D}}(\boldsymbol{x}_j)\prod_{\ell \neq j} p_N(\boldsymbol{x}_\ell)p(z = j)}. \end{aligned} \tag{4.21}$$

This expression can be further simplified, where the key step is to divide the numerator and denominator by $\prod_{j=0}^{J} p_N(\boldsymbol{x}_j)$, which will lead to the final RNCE criterion [79], where the $p_{\boldsymbol{\theta}}$ replaces $p_{\mathcal{D}}$:

$$\begin{aligned} \ell^{\mathrm{RNCE}}(\boldsymbol{\theta}; \boldsymbol{x}_{0:J}, z = 0) &= -\log p_{\boldsymbol{\theta},N}(z = 0 \mid \boldsymbol{x}_{0:J}) \\ &= -\log \frac{\exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}_0))/p_N(\boldsymbol{x}_0)}{\sum_{j=0}^{J} \exp(-E_{\boldsymbol{\theta}}(\boldsymbol{x}_j))/p_N(\boldsymbol{x}_j)}. \end{aligned} \tag{4.22}$$

Similarly to the CNCE criterion, the normalisation constant cancels out and does not need to be estimated to evaluate the criterion.

The role of the noise distribution p_N in the NCE criteria and how it should be selected is not a settled question. There are those that claim that the ideal choice is $q = p_{\mathcal{D}}$ [65], [66], [76], [78], [80], while others claim that it should be $q = p_{\boldsymbol{\theta}}$ [82], [83]. While this argument has mostly been explored for the vanilla NCE criterion, Paper D of this thesis provides theoretical arguments for the optimal q for the CNCE and RNCE criteria.

Score matching

Another way to avoid computing the normalisation is to formulate a criterion in terms of the gradient of the distribution we seek to approximate. The

intuition behind this method is that two functions, $f(\boldsymbol{x})$ and $g(\boldsymbol{x})$, which have the same derivatives, must only differ up to some constant, i.e., $f(\boldsymbol{x}) = g(\boldsymbol{x}) + C$. Therefore, we can estimate the parameters of a model distribution by introducing a criterion based only on the derivatives of the model and target distributions.

For an EBM it is specifically useful to base a criterion on the score function, defined in (2.41), as the score function is independent of the normalisation constant $Z_{\boldsymbol{\theta}}$

$$\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(x) = -\nabla_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(x) - \nabla_{\boldsymbol{x}} \log Z_{\boldsymbol{\theta}} = -\nabla_{\boldsymbol{x}} E_{\boldsymbol{\theta}}(x). \tag{4.23}$$

Thus, this form of estimation is called score matching [84], as the EBM parameters are estimated by matching the score functions of the EBM and data distributions, leading to the criterion

$$\mathcal{L}^{\mathrm{SM}}(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}\sim p_{\mathcal{D}}} \left[\frac{1}{2} \left\| \nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}) - \nabla_{\boldsymbol{x}} \log p_{\mathcal{D}}(\boldsymbol{x}) \right\|_2^2 \right]. \tag{4.24}$$

The RHS is equivalent to the Fisher divergence $\mathrm{D_F}\left[p_{\boldsymbol{\theta}} \| p_{\mathcal{D}}\right]$ [85], and SM can, therefore, be interpreted as finding the EBM most similar to the data distribution in terms of this divergence.

To use SM as a parameter estimation method, we need to consider two important properties. The first one is that matching the derivatives can only estimate the model up to some normalisation constant and the resulting model is thus unnormalised. The second thing is that the objective in (4.24) cannot be directly evaluated since the score function of the true data distribution $p_{\mathcal{D}}$ is unknown.

Computing the score of $p_{\mathcal{D}}$ can be avoided by expressing the difference of score functions in (4.24) by expanding the square and partially integrating the only term which depends on both $\boldsymbol{\theta}$ and the derivative of $\log p_{\mathcal{D}}$ [84]. The drawback is that the resulting criterion involves second-order derivatives of $\log p_{\boldsymbol{\theta}}$, which can be prohibitively costly to compute for complex models [86]. The computational complexity can be reduced by instead computing a noisy estimate using Hutchinson's trace estimator [87]. Then, computing the second-order derivatives can be reduced to computing two gradients of scalar functions, reducing the number of times needed to perform backpropagation. Thus, this variant allows for controlling a trade-off between computational

complexity and accuracy of the estimate and it is termed Sliced SM [88].

Another common variant is Denoising SM (DSM), which approximates the criterion in (4.24) by introducing a noisy version of the data distribution [89]

$$q_\sigma(\boldsymbol{x}') = \int q_\sigma(\boldsymbol{x}' \mid \boldsymbol{x}) p_\mathcal{D}(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x}, \quad q_\sigma(\boldsymbol{x}' \mid \boldsymbol{x}) = \mathcal{N}\left(\boldsymbol{x}'; \boldsymbol{x}, \sigma^2 I\right). \tag{4.25}$$

The DSM criterion is based on the Fisher divergence between the model and the perturbed version of the data distribution

$$\mathrm{D_F}\left[p_{\boldsymbol{\theta}}(\boldsymbol{x}') \| q_\sigma(\boldsymbol{x}')\right] = \mathbb{E}_{\boldsymbol{x}' \sim q_\sigma(\boldsymbol{x}')}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}') - \nabla_{\boldsymbol{x}} \log q_\sigma(\boldsymbol{x}')\right\|_2^2\right], \tag{4.26}$$

and it is equivalent to the criterion in (4.24) in the limit of $\sigma \to 0$.

It is not obvious that this noisy Fisher divergence is easier to evaluate, as $q_\sigma(\boldsymbol{x}')$ is also a complex function, based on the unknown data distribution. However, it can be shown that [89, p. 12]

$$\begin{aligned}
&\operatorname*{argmin}_{\boldsymbol{\theta}} \mathrm{D_F}\left[p_{\boldsymbol{\theta}}(\boldsymbol{x}') \| q_\sigma(\boldsymbol{x}')\right] \\
&= \operatorname*{argmin}_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}' \sim q_\sigma(\boldsymbol{x}')}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}') - \nabla_{\boldsymbol{x}} \log q_\sigma(\boldsymbol{x}')\right\|_2^2\right] \\
&= \operatorname*{argmin}_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{p_\mathcal{D}(\boldsymbol{x}) q_\sigma(\boldsymbol{x}' \mid \boldsymbol{x})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}} \log p_{\boldsymbol{\theta}}(\boldsymbol{x}') - \nabla_{\boldsymbol{x}} \log q_\sigma(\boldsymbol{x}' \mid \boldsymbol{x})\right\|_2^2\right]}_{\coloneqq \mathcal{L}^{\mathrm{DSM}}(\boldsymbol{\theta})}.
\end{aligned} \tag{4.27}$$

That is, in DSM we construct a tractable criterion for score matching with the drawback that the parameter estimation is guided towards a perturbed version of the actual target distribution. The choice of noise level σ is a design parameter, where a higher noise level can improve convergence properties but will also result in a noisier target distribution.

It is evident from this section, that there are many methods for estimating the parameters of an EBM. There is no strong consensus in the literature for which is the best choice. Furthermore, each method has additional design choices to consider, such as noise and proposal distributions. In Paper D, we try to improve this situation by proving theoretical connections between IS, NCE and CD. Additionally, these connections are used to motivate the choice

of noise and proposal distribution.

4.3 Diffusion models

Diffusion processes are well-known in the field of stochastic differential [90], but in recent years, they have become very important for machine learning in the form of denoising diffusion probabilistic models [91]–[93]. They have been especially successful in the computer vision domain [94], but also in many others [95], [96].

The idea behind diffusion models is to simplify the sampling problem by reducing it into many small steps. The diffusion model is, in principle, a full sequence of models, each responsible for predicting a less noisy version of its input. This way, diffusion models generate high-quality samples from complex data distributions.

The data for diffusion models comes from a known forward process, which samples corrupted versions of the input data at different noise levels. Let $\boldsymbol{x}_0 \sim p_{\mathcal{D}}(\boldsymbol{x}_0) \coloneqq q(\boldsymbol{x}_0)$ be a sample from the data distribution, then construct a Markov chain of latent variables $\boldsymbol{x}_{1:T} \coloneqq (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_T)$, with transition probabilities

$$q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t I\right). \tag{4.28}$$

The sequence $\beta_t \in [0, 1), \forall t = 1, \ldots, T$ is called the beta-schedule, and is a hyperparameter [93]. At each step in the chain, the mean value of the distribution is scaled towards zero and more noise is added to the sample, such that for large enough T we have $q(\boldsymbol{x}_T) \approx \mathcal{N}(\boldsymbol{x}_T; 0, I)$. Figure 4.1 gives a schematic view of the diffusion process. Note that other parameterisations of the forward processes are possible and somewhat common, but they all follow the same basic idea presented here. Similarly, the choice of Gaussian transition probabilities is common but not necessary.

A useful property of the forward process is that since all transition probabilities are Gaussian, the conditional diffusion probability $q(\boldsymbol{x}_t|\boldsymbol{x}_0)$ can be

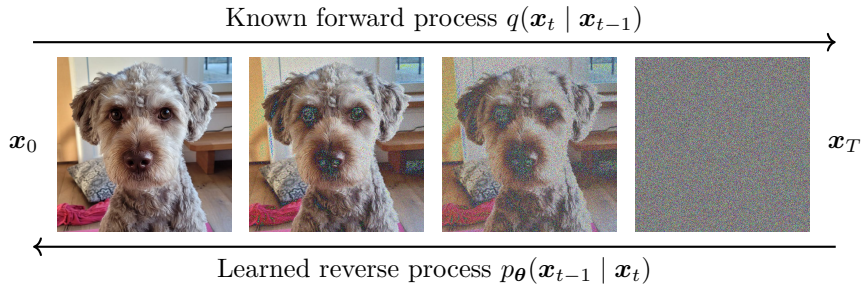

Figure 4.1: A sample $\boldsymbol{x}_0 \sim p_{\mathcal{D}}(\boldsymbol{x}_0) \coloneqq q(\boldsymbol{x}_0)$ is drawn from the data distribution. A known forward process adds small amounts of noise such that (approximately) all information about the input image is removed at $t = T$. The diffusion model learns to reverse this process by gradually denoising the input at every step.

computed in closed form,

$$\begin{aligned} q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) &= \int q(\boldsymbol{x}_{1:t} \mid \boldsymbol{x}_0) \, \mathrm{d}\boldsymbol{x}_{1:t-1} = \int \prod_{s=1}^{t} q(\boldsymbol{x}_t \mid \boldsymbol{x}_{t-1}) \, \mathrm{d}\boldsymbol{x}_{1:t-1} \\ &= \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha})I\right), \end{aligned} \tag{4.29}$$

where $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_t$ [93]. This means that we can sample $\boldsymbol{x}_t \mid \boldsymbol{x}_0$ with the forward process, without having to traverse the entire Markov chain up to t.

The goal with diffusion models is to learn a model, $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$, that removes the noise from a sample at any diffusion step t. Approximate samples from $q(\boldsymbol{x}_0)$ can then be drawn, by starting from $\boldsymbol{x}_T \sim \mathcal{N}(0, I)$ and traversing the chain backwards until $t = 0$. This is called the reverse process.

Knowledge of the forward process helps us choose how to parameterise $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)$. If β_t is sufficiently small, the posterior, *reverse*, distribution $q(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ can be approximated by a Gaussian distribution. Therefore, it is reasonable to use the parameterisation

$$p_\theta(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t), \boldsymbol{\Sigma}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\right). \tag{4.30}$$

To learn the model, we would ideally minimise the NLL under the data

distribution $-\log q(\boldsymbol{x}_0)$, but this is intractable. However, we can derive an upper bound of the NLL by incorporating the latent variables $\boldsymbol{x}_{1:T}$ [93]

$$\begin{aligned}\mathbb{E}_{q(x_0)}\left[-\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0)\right] \leq \mathbb{E}_{q(\boldsymbol{x}_{0:T})}\Big[\\ \mathrm{KL}\left[q(\boldsymbol{x}_T \mid \boldsymbol{x}_0) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}_T)\right] - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0 \mid \boldsymbol{x}_1) \\ + \sum_{t>1} \mathrm{KL}\left[q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right].\Big]\end{aligned} \tag{4.31}$$

The main component of the upper bound is the summation at the end of (4.31), where the distribution $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) = \mathcal{N}\left(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0), \tilde{\boldsymbol{\Sigma}}(\boldsymbol{x}_t, \boldsymbol{x}_0)\right)$ is a Gaussian and its parameters can be computed in closed form. The expressions for the involved Kullback–Leibler divergences in (4.31) can be further simplified if we assume a fixed covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{\theta}} = \tilde{\beta} I$:

$$\begin{aligned}&\mathrm{KL}\left[q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0) \| p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t)\right] \\ &\quad = \int \mathcal{N}\left(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}\right) \log \frac{\mathcal{N}\left(\boldsymbol{x}_{t-1}; \tilde{\boldsymbol{\mu}}, \tilde{\boldsymbol{\Sigma}}\right)}{\mathcal{N}\left(\boldsymbol{x}_{t-1}; \mu_{\boldsymbol{\theta}}, \tilde{\beta} I\right)} \, \mathrm{d}\boldsymbol{x}_{t-1} \\ &\quad = \frac{\|\tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0) - \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2}{2\tilde{\beta}} + C.\end{aligned} \tag{4.32}$$

The loss function that is used in practice, replaces the summation over t in (4.31) with samples $(\boldsymbol{x}_0, \boldsymbol{x}_t) \sim q(\boldsymbol{x}_0) q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)$ (from (4.29)):

$$\mathbb{E}_{q(\boldsymbol{x}_0) q(\boldsymbol{x}_t \mid \boldsymbol{x}_0)} \left[\frac{\|\tilde{\boldsymbol{\mu}}(\boldsymbol{x}_t, \boldsymbol{x}_0) - \mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)\|_2}{2\tilde{\beta}} \right]. \tag{4.33}$$

The actual parameterisation of the model $\mu_{\boldsymbol{\theta}}$ as well as the value of $\tilde{\beta}$ is a design choice. The most common parameterisation is based on the functional form of $q(\boldsymbol{x}_{t-1} \mid \boldsymbol{x}_t, \boldsymbol{x}_0)$ [93], such that the mean of the distribution is parameterised as

$$\mu_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) = \frac{1}{\sqrt{\alpha_t}} \left(\boldsymbol{x}_t - \frac{\beta_t}{\sqrt{1 - \bar{\alpha}_t}} \boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t) \right). \tag{4.34}$$

We recall that $\mu_{\boldsymbol{\theta}}$ represents the mean of $\boldsymbol{x}_{t-1}$, and note that this parameterisation essentially obtains this mean by removing noise from x_t. The learnable model, $\boldsymbol{\epsilon}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)$, is therefore called a noise prediction model, and is typically

parameterised by a neural network.

Diffusion models can be related to EBMs via a connection to denoising score matching [92], [97], [98]. In Paper E, we use this connection to propose Metropolis–Hastings corrected MCMC sampling methods for diffusion models.

CHAPTER 5

Summary of included papers

This chapter provides a summary of the included papers.

5.1 Paper A

Jakob Lindqvist, Simo Särkkä, Ángel F. García-Fernández, Matti Raitoharju, Lennart Svensson
Levenberg–Marquardt and line-search iterated posterior linearisation smoothing
Signal Processing (submitted), Apr 2023,.

Bayesian smoothing in nonlinear state-space models with additive Gaussian noise is commonly done with general Gaussian smoothers. They are a type of smoother which linearises the state space model around some estimate of the state and then applies the Rauch-Tung-Striebel (RTS) smoother to the approximated state-space model. A well-known example is the extended Kalman Smoother (KS), which linearises the state-space model through a first-order Taylor approximation around the prior mean of the state, at time k. Another class of smoothers use statistical linear regression (SLR) to approximate the

state-space model with the optimal linearisation w.r.t. to the full prior distribution, e.g., the unscented RTS smoother. This method of linearisation is more expensive to compute, but it typically improves smoothing performance and does not require a closed-form expression for the Jacobian of the motion and measurement models. The process can then be repeated, with a new linearisation around the estimated states, resulting in iterative versions of the smoothers (e.g., the IEKS and the IPLS) with increased performance. The iterative smoothing is not guaranteed to converge, however, and it is of interest to develop extensions with better convergence properties. The IEKS has previously been shown to be equivalent to Gauss–Newton (GN) optimisation of a cost function based on the nonlinear state space model. Standard regularising techniques from the general optimisation literature can therefore be applied to the IEKS, such as Levenberg–Marquardt regularisation and line-search methods. Developing similar regularised versions of the, typically more performant, IPLS is therefore of interest, but deriving a corresponding GN cost function for the IPLS is more involved, due to the nature of SLR-based linearisation. In this paper, we derive a sequence of cost functions for the IPLS, such that GN optimisation of these is equivalent to IPLS smoothing at each iteration. From this GN connection, we propose the Levenberg–Marquardt-IPLS and line-search-IPLS. We perform numerical simulations to demonstrate the benefit of these extensions in highly nonlinear scenarios.

5.2 Paper B

Jakob Lindqvist*, Amanda Olmin*, Fredrik Lindsten, Lennart Svensson
A General Framework for Ensemble Distribution Distillation
Machine Learning for Signal Processing, 2020.

Ensembles of multiple models are used not only for improved predictive performance but also to provide more reliable uncertainty estimates. In particular, ensembles enable decomposition into aleatoric and epistemic uncertainties. To mitigate the cost of using and storing multiple models, it is common to perform ensemble distillation, which is to compress the full ensemble into a smaller model, by training it to predict similar outputs as the ensemble. In standard distillation, the distilled model typically predicts parameters for the same family of distributions as a single ensemble member. This approach

works well with regards to maintaining model performance but the uncertainty decomposition property is lost in the process. This paper proposes a form of ensemble distribution distillation which preserves the uncertainty decomposition. This is achieved by having the distilled model predict the parameters for another family of distributions, modelling the distribution of parameters predicted by the full ensemble. We develop a framework for this new form of ensemble distillation, the framework is general and applies to several problems, including classification and regression.

5.3 Paper C

Jakob Lindqvist, Amanda Olmin, Lennart Svensson, Fredrik Lindsten
Generalised Active Learning with Annotation Quality Selection
Machine Learning for Signal Processing, 2023,.

Supervised learning methods rely on annotated training data sets of data points and their corresponding label. The annotation process is costly as it almost always requires human labour or long computer simulations. Active learning (AL) are a family of methods which seeks to quantify which data points are the most useful for learning a model, and select those for labelling, thereby achieving the optimal training data set, under a constrained annotation budget. The AL problem is typically formulated as a binary selection decision, to either select a particular data point for labelling or or not. When a data point is selected, it is typically assumed that the label is acquired by a fixed cost and with perfect accuracy, In practical applications, however, it is reasonable to assume that allowing labels to be acquired at different noise levels, ought to affect the annotation cost. Coupled with the fact that model estimation with noisy labels is a well-studied field, we have a generalisation of the standard active learning formulation, where the standard binary acquisition decision is extended to a continuous choice of annotation quality. We derive a concrete criterion for this novel AL approach, based on the mutual information (MI) between model parameters and noisy labels. The usefulness of our formulation is demonstrated with both classification and regression examples, but we find that the complexity of the MI criterion limits its application to more complex models.

5.4 Paper D

Amanda Olmin*, **Jakob Lindqvist***, Lennart Svensson, Fredrik Lindsten
On the connection between Noise-Contrastive Estimation and Contrastive Divergence
Accepted for publication in *International Conference on Artificial Intelligence and Statistics, 2024*.

Unnormalised models, for example, EBMs, can model complex distributions, but the intractable normalisation constant prohibits standard ML estimation techniques. Alternative methods for parameter estimation in unnormalised models therefore remain an active research field. Among the plethora of methods, we can identify two important groups: those that approximate the ML criterion with sampling methods, such as importance sampling (ML-IS) and contrastive divergence (CD), and those that introduce a proxy criterion to avoid having to compute the normalisation constant, namely noise-contrastive estimation (NCE) and its variants, conditional NCE (CNCE) and ranking NCE (RNCE). These two groups are conceptually different and are generally treated as two distinct ways of doing parameter estimation. In this paper we show that both CNCE and RNCE can be viewed as ML estimation methods, connecting the two groups. Specifically, we show that the RNCE criterion is equivalent to ML estimation with a variant of IS, called conditional importance sampling. We also show that both RNCE and CNCE are special cases of CD. By constructing two invariant MCMC kernels for CD, we obtain CD criteria which are equivalent to RNCE and CNCE respectively. These findings not only enable a more coherent framework for unnormalised parameter estimation by connecting the two groups but also provide theoretical arguments for why the noise-contrastive methods outperform ML-IS empirically. Furthermore, the connection allows us to apply existing extensions of CD to NCE, the benefits of which, we demonstrate with numerical experiments.

5.5 Paper E

Anders Sjöberg*, **Jakob Lindqvist***, Magnus Önnheim, Mats Jirstrand, Lennart Svensson
MCMC-Correction for Diffusion Model Composition: Energy Approxi-

mation using Diffusion Models
Manuscript, 2024.

This paper proposes an improved sampling method for diffusion models. The method is based on a connection between diffusion models and energy-based models, which is the fact that both model types can be used to approximate the score function of a marginal distribution in a Markov chain. For the EBM, the score function can be obtained through explicit differentiation of the energy function, whereas the diffusion model's noise prediction is directly proportional to the score. The energy and score parameterised versions of the diffusion model both have appealing properties. The energy parameterisation enables a Metropolis–Hastings correction step in the sampling process, while the more common score parameterisation has more available pre-trained models, which is important when composing different models to sample from new distributions. We propose a method for adding a similar correction step for a score-parameterised model, by estimating an energy difference as a curve integral of the score.

CHAPTER 6

Concluding Remarks and Future Work

This thesis explores a range of theories and methods related to uncertainty estimation in probabilistic machine learning. The included publications contain contributions on various aspects of uncertainty estimation in probabilistic models. Hopefully, the work done within the confines of this thesis has furthered the cause of probabilistic machine learning in some way or, at the very least, showed that it is a topic well worth further exploration.

Paper A proposes an improved version of Bayesian smoothing for SLR-based iterated smoother. This paper stands out somewhat in this thesis since it does not directly involve deep learning. However, the SLR-based smoothers studied in the paper can, in principle, be used with more complex models than those evaluated in our work. A natural first step is to investigate how these methods can be extended to other probabilistic graphical models such as those that appear in simultaneous localisation and mapping (SLAM). Developing hybrid methods with, for instance, a measurement model parameterised by a neural network, with gradients propagated through the Markov chain is also an interesting case for future work.

Papers B and C concern the quantification of uncertainty, both how it is represented and how it can be used in downstream applications. Uncertainty

estimation is a vital part of most active learning algorithms, and we have shown that an alternative formulation can improve performance. However, finding more computationally feasible methods for estimating the value of unlabelled data remains challenging and deserves more attention from our research community.

Papers D and E deal with generative models, arguably the most researched topic in machine learning today. A common theme in these two papers is that their contributions bridge seemingly different techniques: Paper D connects different parameter estimation methods, and Paper E combines sampling techniques for diffusion and energy-based models. These types of contributions are important as they provide tangible improvements and further our understanding of the underlying methods on a more profound level. These papers are but two examples of a growing category of contributions, which blurs the boundaries between the existing forms of deep generative models [98]–[101] and this trend shows little sign of slowing down.

References

[1] A. Gelman, J. B. Carlin, H. S. Stern, D. B. Rubin, A. Vehtari, and D. B. Dunson, *Bayesian data analysis.* (Texts in statistical science series). Chapman & Hall/CRC, 2014, ISBN: 9781439840955.

[2] C. M. Bishop, *Pattern recognition and machine learning.* Springer, 2006.

[3] S. Särkkä and L. Svensson, *Bayesian Filtering and Smoothing* (Institute of Mathematical Statistics Textbooks), 2nd ed. Cambridge University Press, 2023.

[4] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Est. with Appl. to Tracking and Navigation: Theory, Alg. and Software.* John Wiley & Sons, Ltd, Jan. 2004, ISBN: 047141655X.

[5] S. J. Godsill and P. J. W. Rayner, *Digital Audio Restoration.* Springer London, 1998, ISBN: 9781447115618.

[6] S. K. Nanda, G. Kumar, V. Bhatia, and A. K. Singh, "Kalman-based compartmental estimation for covid-19 pandemic using advanced epidemic model," *Biomedical Signal Processing and Control*, vol. 84, p. 104 727, 2023, ISSN: 1746-8094.

[7] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960, ISSN: 0021-9223.

[8] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[9] H. Arasaratnam, S. Haykin, and R. Elliott, "Discrete-time nonlinear filtering algorithms using gauss–hermite quadrature," *Proceedings of the IEEE*, vol. 95, pp. 953–977, Jun. 2007.

[10] Á. F. García-Fernández, L. Svensson, M. R. Morelande, and S. Särkkä, "Posterior linearization filter: Principles and implementation using sigma points," *IEEE Transactions on Signal Processing*, vol. 63, no. 20, pp. 5561–5573, 2015.

[11] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proceedings of the IEEE*, vol. 92, no. 3, pp. 401–422, 2004.

[12] I. Arasaratnam and S. Haykin, "Cubature kalman filters," *IEEE Transactions on Automatic Control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[13] I. Arasaratnam and S. Haykin, "Cubature Kalman smoothers," *Automatica*, vol. 47, no. 10, pp. 2245–2250, 2011, ISSN: 0005-1098.

[14] S. Särkkä, "Unscented Rauch–Tung–Striebel smoother," *Automatic Control, IEEE Transactions on*, vol. 53, May 2008.

[15] A. Gelb, *Applied optimal estimation.* MIT Press, 1974, ISBN: 0262200279.

[16] B. M. Bell and F. W. Cathey, "The iterated Kalman filter update as a Gauss–Newton method," *IEEE Transactions on Automatic Control*, vol. 38, no. 2, pp. 294–297, 1993.

[17] B. M. Bell, "The iterated Kalman smoother as a Gauss–Newton method.," *SIAM Journal on Optimization*, vol. 4, no. 3, pp. 626–636, 1994, ISSN: 10526234.

[18] Á. F. García-Fernández, L. Svensson, and S. Särkkä, "Iterated posterior linearization smoother," *Transactions on Automatic Control*, vol. 62, no. 4, pp. 2056–2063, 2017.

[19] S. Särkkä and L. Svensson, "L-M and line–search extended Kalman smoothers," in *Int. Conf. on Acoustics, Speech and Signal Processing*, 2020.

[20] Y. Chen and D. Oliver, "Levenberg–Marquardt forms of the iterative ensemble smoother for efficient history matching and uncertainty quantification," *Computational Geosciences*, vol. 17, Aug. 2013.

[21] J. Mandel, E. Bergou, S. Gürol, and S. Gratton, "Hybrid Levenberg–Marquardt and weak constraint ensemble Kalman smoother method," *Nonlinear Processes in Geophysics Disc.*, vol. 2, May 2015.

[22] D. J. C. MacKay, *Information theory, inference, and learning algorithms.* Cambridge University Press, 2003, ISBN: 0521642981.

[23] R. M. Neal, *Bayesian Learning for Neural Networks. [electronic resource].* (Lecture Notes in Statistics: 118). Springer New York, 1996, ISBN: 9781461207450.

[24] C. Andrieu, N. de Freitas, A. Doucet, and M. I. Jordan, "An introduction to MCMC for machine learning," *Machine Learning*, vol. 50, no. 1–2, pp. 5–43, 2003.

[25] M. Chen, Q. Shao, and J. Ibrahim, *Monte Carlo Methods in Bayesian Computation* (Springer Series in Statistics). Springer New York, 2001, ISBN: 9780387989358.

[26] A. Gupta and J. B. Rawlings, "Comparison of parameter estimation methods in stochastic chemical kinetic models: Examples in systems biology," *AIChE journal. American Institute of Chemical Engineers*, vol. 60, no. 4, pp. 1253–1268, Apr. 2014, ISSN: 0001-1541.

[27] M. F. Kasim, A. F. A. Bott, P. Tzeferacos, D. Q. Lamb, G. Gregori, and S. M. Vinko, "Retrieving fields from proton radiography without source profiles," *Physical Review E*, vol. 100, no. 3, p. 033 208, Sep. 24, 2019, Publisher: American Physical Society.

[28] A. Gelman and D. B. Rubin, "Inference from iterative simulation using multiple sequences," *Statistical Science*, vol. 7, no. 4, Nov. 1, 1992, ISSN: 0883-4237.

[29] M. K. Cowles and B. P. Carlin, "Markov chain monte carlo convergence diagnostics: A comparative review," *Journal of the American Statistical Association*, vol. 91, no. 434, pp. 883–904, 1996, Publisher: [American Statistical Association, Taylor & Francis, Ltd.], ISSN: 0162-1459.

[30] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, and A. H. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.

[31] W. K. Hastings, "Monte Carlo sampling methods using Markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.

[32] L. Tierney, "Markov Chains for Exploring Posterior Distributions," *The Annals of Statistics*, vol. 22, no. 4, pp. 1701–1728, 1994.

[33] A. Gelman, W. R. Gilks, and G. O. Roberts, "Weak convergence and optimal scaling of random walk Metropolis algorithms," *The Annals of Applied Probability*, vol. 7, no. 1, pp. 110–120, 1997.

[34] G. O. Roberts and J. S. Rosenthal, "Optimal scaling for various metropolis-hastings algorithms," *Statistical Science*, vol. 16, no. 4, pp. 351–367, 2001, ISSN: 08834237.

[35] M. Bédard, "Optimal acceptance rates for metropolis algorithms: Moving beyond 0.234," *Stochastic Processes and their Applications*, vol. 118, no. 12, pp. 2198–2222, Dec. 1, 2008, ISSN: 0304-4149.

[36] G. O. Roberts and O. Stramer, "Langevin diffusions and metropolis-hastings algorithms.," *Methodology & Computing in Applied Probability*, vol. 4, no. 4, pp. 337–357, 2002, ISSN: 13875841.

[37] S. Duane, A. Kennedy, B. J. Pendleton, and D. Roweth, "Hybrid monte carlo," *Physics Letters B*, vol. 195, no. 2, pp. 216–222, 1987, ISSN: 0370-2693.

[38] H. Robbins and S. Monro, "A Stochastic Approximation Method," *The Annals of Mathematical Statistics*, vol. 22, no. 3, pp. 400–407, 1951.

[39] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[40] D. Koller and N. Friedman, *Probabilistic graphical models : principles and techniques*. (Adaptive computation and machine learning). MIT Press, 2009, ISBN: 9780262013192.

[41] C. M. Bishop, *Deep Learning: Foundations and Concepts*. 2024, ISBN: 978-3-031-45468-4.

[42] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[43] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[44] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles," in *NeurIPS*, 2017.

[45] Y. Gal and Z. Ghahramani, "Dropout as a bayesian approximation: Representing model uncertainty in deep learning," in *Proceedings of The 33rd International Conference on Machine Learning*, M. F. Balcan and K. Q. Weinberger, Eds., ser. Proceedings of Machine Learning Research, vol. 48, New York, New York, USA: PMLR, 20–22 Jun 2016, pp. 1050–1059.

[46] P. Izmailov, D. Podoprikhin, T. Garipov, D. Vetrov, and A. Wilson, "Averaging weights leads to wider optima and better generalization," English (US), in *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, ser. 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018, Association For Uncertainty in Artificial Intelligence (AUAI), 2018, pp. 876–885.

[47] G. Hess, C. Petersson, and L. Svensson, "Object detection as probabilistic set prediction," in *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part X*, Springer, Oct. 2022, pp. 550–566.

[48] R. Ranftl, A. Bochkovskiy, and V. Koltun, "Vision transformers for dense prediction," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 12 159–12 168.

[49] R. Strudel, R. Garcia, I. Laptev, and C. Schmid, "Segmenter: Transformer for semantic segmentation," in *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021, pp. 7242–7252.

[50] T. M. Cover and J. A. Thomas, *Elements of information theory*. Wiley-Interscience, 2006, ISBN: 0471748811.

[51] B. Lakshminarayanan, A. Pritzel, and C. Blundell, "Simple and scalable predictive uncertainty estimation using deep ensembles," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6405–6416, ISBN: 9781510860964.

[52] Y. Ovadia, E. Fertig, J. Ren, *et al.*, "Can you trust your model's uncertainty? evaluating predictive uncertainty under dataset shift," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d 'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[53] B. Settles, *Active Learning.* (Synthesis Lectures on Artificial Intelligence and Machine Learning). Morgan & Claypool Publishers, 2012, ISBN: 978-1-60845-726-7.

[54] C. Aggarwal, X. Kong, Q. Gu, J. Han, and P. Yu, "Active learning: A survey," English (US), in *Data Classification*. CRC Press, Jan. 2014, pp. 571–605, Publisher Copyright: © 2015 by Taylor & Francis Group, LLC., ISBN: 9781466586741.

[55] P. Ren, Y. Xiao, X. Chang, *et al.*, "A survey of deep active learning," *ACM Comput. Surv.*, vol. 54, no. 9, Oct. 2021, ISSN: 0360-0300.

[56] D. D. Lewis and J. Catlett, "Heterogeneous uncertainty sampling for supervised learning," in *Machine Learning Proceedings 1994*, W. W. Cohen and H. Hirsh, Eds., San Francisco (CA): Morgan Kaufmann, 1994, pp. 148–156, ISBN: 978-1-55860-335-6.

[57] T. Scheffer, C. Decomain, and S. Wrobel, "Active hidden markov models for information extraction," in *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis*, ser. IDA '01, Berlin, Heidelberg: Springer-Verlag, 2001, pp. 309–318, ISBN: 3540425810.

[58] B. Settles and M. Craven, "An analysis of active learning strategies for sequence labeling tasks," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, ser. EMNLP '08, Honolulu, Hawaii: Association for Computational Linguistics, 2008, pp. 1070–1079.

[59] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel, "Bayesian Active Learning for Classification and Preference Learning," *arXiv e-prints*, arXiv:1112.5745, arXiv:1112.5745, Dec. 2011.

[60] A. Kirsch, J. van Amersfoort, and Y. Gal, "Batchbald: Efficient and diverse batch acquisition for deep B ayesian active learning," in *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d 'Alché-Buc, E. Fox, and R. Garnett, Eds., vol. 32, Curran Associates, Inc., 2019.

[61] N. Tishby, C. Pereira, and W. Bialek, "The information bottleneck method," *Proceedings of the 37th Allerton Conference on Communication, Control and Computation*, vol. 49, Jul. 2001.

[62] L. Wang, M. Chen, M. Rodrigues, D. Wilcox, R. Calderbank, and L. Carin, "Information-theoretic compressive measurement design," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 6, pp. 1150–1164, Jun. 2017, ISSN: 0162-8828.

[63] A. Kendall and Y. Gal, "What uncertainties do we need in bayesian deep learning for computer vision?" In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, *et al.*, Eds., vol. 30, Curran Associates, Inc., 2017.

[64] Y. LeCun, S. Chopra, R. Hadsell, M. Ranzato, F. Huang, and et al., "A tutorial on energy-based learning," *Predicting structured data*, 2006.

[65] F. K. Gustafsson, M. Danelljan, R. Timofte, and T. B. Schön, "How to train your energy-based model for regression," in *British Machine Vision Virtual Conference*, 2020.

[66] R. Gao, E. Nijkamp, D. P. Kingma, Z. Xu, A. M. Dai, and Y. N. Wu, "Flow contrastive estimation of energy-based models," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.

[67] Y. Du, S. Li, J. Tenenbaum, and I. Mordatch, "Improved Contrastive Divergence Training of Energy Based Models," in *International Conference on Machine Learning*, 2021.

[68] P. Florence, C. Lynch, A. Zeng, *et al.*, "Implicit behavioral cloning," in *Conference on Robot Learning*, 2022.

[69] C. A. Naesseth, F. Lindsten, and T. B. Schön, "Elements of sequential Monte Carlo," *Foundations and Trends in Machine Learning*, vol. 12, no. 3, pp. 307–392, Nov. 2019.

[70] C. P. Robert, G. Casella, and G. Casella, *Monte Carlo statistical methods*. Springer, 1999.

[71] L. Bottou, F. E. Curtis, and J. Nocedal, "Optimization methods for large-scale machine learning," *SIAM Review*, vol. 60, no. 2, pp. 223–311, 2018.

[72] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, Aug. 2002.

[73] M. A. Carreira-Perpinan and G. Hinton, "On contrastive divergence learning," in *International Workshop on Artificial Intelligence and Statistics*, 2005.

[74] M. Welling, A. Mnih, and G. E. Hinton, "Wormholes improve contrastive divergence," *Advances in Neural Information Processing Systems*, 2003.

[75] A. U. Asuncion, Q. Liu, A. T. Ihler, and P. Smyth, "Particle filtered MCMC-MLE with connections to contrastive divergence," in *International Conference on Machine Learning*, 2010.

[76] M. Gutmann and A. Hyvärinen, "Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics," English, *Journal of Machine Learning Research*, vol. 13, pp. 307–361, 2012.

[77] M. Pihlaja, M. Gutmann, and A. Hyvärinen, "A family of computationally efficient and simple estimators for unnormalized statistical models," in *Conference on Uncertainty in Artificial Intelligence*, Catalina Island, CA: AUAI Press, 2010, pp. 442–449.

[78] C. Ceylan and M. U. Gutmann, "Conditional noise-contrastive estimation of unnormalised models," in *International Conference on Machine Learning*, 2018.

[79] R. Jozefowicz, O. Vinyals, M. Schuster, N. Shazeer, and Y. Wu, "Exploring the limits of language modeling," *arXiv preprint arXiv:1602.02410*, 2016.

[80] N. Xu, "Self-adapting noise-contrastive estimation for energy-based models," M.S. thesis, Tsinghua University, 2022.

[81] Z. Ma and M. Collins, "Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency," in *Conference on Empirical Methods in Natural Language Processing*, 2018.

[82] I. Goodfellow, "On distinguishability criteria for estimating generative models," in *Workshop Contribution in International Conference on Learning Representations*, 2015.

[83] F. K. Gustafsson, M. Danelljan, and T. B. Schön, "Learning proposals for practical energy-based regression," in *International Conference on Artificial Intelligence and Statistics*, 2022, pp. 4685–4704.

[84] A. Hyvärinen, "Estimation of Non-Normalized Statistical Models by Score Matching," *Journal of Machine Learning Research*, vol. 6, pp. 695–709, 2005.

[85] O. T. Johnson, *Information Theory And The Central Limit Theorem.* World Scientific Publishing Company, 2004, ISBN: 9781860945373.

[86] J. Martens, I. Sutskever, and K. Swersky, "Estimating the hessian by back-propagating curvature," in *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ser. ICML'12, Edinburgh, Scotland: Omnipress, 2012, pp. 963–970, ISBN: 9781450312851.

[87] M. Hutchinson, "A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines," *Communications in Statistics - Simulation and Computation*, vol. 19, no. 2, pp. 433–450, 1990.

[88] Y. Song, S. Garg, J. Shi, and S. Ermon, "Sliced score matching: A scalable approach to density and score estimation," in *Conference on Uncertainty in Artificial Intelligence*, 2019.

[89] P. Vincent, "A connection between score matching and denoising autoencoders," *Neural Computation*, vol. 23, pp. 1661–1674, 2011.

[90] F. C. Klebaner, *Introduction to stochastic calculus with applications.* Imperial College Press, 2012, ISBN: 9781848168329.

[91] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," in *International conference on machine learning*, PMLR, 2015, pp. 2256–2265.

[92] Y. Song and S. Ermon, "Generative modeling by estimating gradients of the data distribution," *Advances in neural information processing systems*, vol. 32, 2019.

[93] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[94] P. Dhariwal and A. Nichol, "Diffusion models beat GANs on image synthesis," *Advances in neural information processing systems*, vol. 34, pp. 8780–8794, 2021.

[95] D. Lüdke, M. Biloš, O. Shchur, M. Lienen, and S. Günnemann, "Add and thin: Diffusion for temporal point processes," in *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[96] K. Wang, Z. Xu, Y. Zhou, *et al.*, *Neural network diffusion*, 2024.

[97] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," in *International Conference on Learning Representations*, 2021.

[98] Y. Du, C. Durkan, R. Strudel, *et al.*, "Reduce, Reuse, Recycle: Compositional generation with energy-based diffusion models and MCMC," in *Proceedings of the 40th International Conference on Machine Learning*, A. Krause, E. Brunskill, K. Cho, B. Engelhardt, S. Sabato, and J. Scarlett, Eds., ser. Proceedings of Machine Learning Research, vol. 202, PMLR, Jul. 2023, pp. 8489–8510.

[99] M. S. Albergo, N. M. Boffi, and E. Vanden-Eijnden, *Stochastic interpolants: A unifying framework for flows and diffusions*, 2023.

[100] S. Yang, Y. Du, B. Dai, D. Schuurmans, J. B. Tenenbaum, and P. Abbeel, "Probabilistic adaptation of black-box text-to-video models," in *The Twelfth International Conference on Learning Representations*, 2024.

[101] R. Gao, Y. Song, B. Poole, Y. N. Wu, and D. P. Kingma, "Learning energy-based models by diffusion recovery likelihood," in *International Conference on Learning Representations*, 2021.