Introducing Probabilistic Bézier Curves for N-Step Sequence Prediction

Ronny Hug, Wolfgang Hübner, Michael Arens
Fraunhofer Institute of Optronics, System Technologies, and Image Exploitation (IOSB)*
Gutleuthausstr. 1, 76275 Ettlingen, Germany
{ronny.hug, wolfgang.huebner, michael.arens}@iosb.fraunhofer.de

Abstract

Representations of sequential data are commonly based on the assumption that observed sequences are realizations of an unknown underlying stochastic process, where the learning problem includes determination of the model parameters. In this context, a model must be able to capture the multi-modal nature of the data, without blurring between single modes. This paper proposes probabilistic Bézier curves ($\mathcal{N}$-Curves) as a basis for effectively modeling continuous-time stochastic processes. The model is based on Mixture Density Networks (MDN) and Bézier curves with Gaussian random variables as control points. Key advantages of the model include the ability of generating smooth multi-mode predictions in a single inference step which reduces the need for Monte Carlo simulation. This property is in line with recent attempts to address the problem of quantifying uncertainty as a regression problem. Essential properties of the proposed approach are illustrated by several toy examples and the task of multi-step sequence prediction. As an initial proof of concept, the model performance is compared to an LSTM-MDN model and recurrent Gaussian processes on two real world use-cases, trajectory prediction and motion capture sequence prediction.

1 Introduction

Probabilistic models of sequential data have a broad range of applications related to representation learning, sequence generation and prediction. Formulated as a sequence learning problem, these tasks can be tackled by learning a model of a presumed underlying stochastic process $\{X_t\}_{t\in T}$, where X_t is a random variable and T is an index set. The model is then learned from given realizations (sample sequences).

This paper focuses on the subtask of multi-step sequence prediction, where the evolution of a probability density function for n discrete time steps should be inferred, given m subsequent observations of a sample sequence. Multi-step predictions of this form are relevant in applications that require ahead-of-time planning, like robot navigation or autonomous driving (Lefèvre, Vasquez, and Laugier 2014). Common approaches for tackling sequence prediction include autoregressive models, like the Gaussian Process regression model (Roberts et al. 2013; Montgomery, Jennings, and Kulahci 2015), and (stochastic) neural networks (Rudenko et al. 2019). While being able to generate (multi-modal) sequence predictions, a drawback shared by many of these models is a computationally intensive inference scheme. Further, the process of prediction is unconstrained in most neural network architectures, leading to the generation of artifacts.

In order to overcome these limitations, this paper proposes a model of a continuous-time stochastic process, which extends on Mixture Density Networks (Bishop 2006) and can be learned from time-discrete realizations. For an initial proof of concept, the index set is given by $t \in [0, 1]$, thus focusing on fixed length sequences. Set into the context of n-step prediction, the generation of multi-modal predictions should be possible without the need of extensive Monte Carlo simulation, thus performing n-step inference with minimum overhead in a single step. To achieve this, sequence prediction, including uncertainty estimation, is treated as a regression problem (Kuleshov, Fenner, and Ermon 2018). The modes of the modeled stochastic process are described in terms of probabilistic, parametric curves termed $\mathcal{N}$-Curves. $\mathcal{N}$-Curves are based on Bézier curves with Gaussian distributions as control parameters. Every point on the $\mathcal{N}$-Curve is a Gaussian random variable, thus resembling the model of a continuous-time stochastic process, which constrains inference in order to generate sufficiently smooth predictions. For generating multi-modal predictions (multiple sequences), a stochastic process consisting of Gaussian mixture random variables can be defined by combining multiple $\mathcal{N}$-Curves into a mixture of $\mathcal{N}$-Curves. Further, by basing the approach on Bézier curves, data of arbitrary dimensionality can be modeled by choosing the control point dimensionality accordingly.

The paper is structured as follows: First, an overview over related work is given in section 2. Next, the $\mathcal{N}$-Curve mixture model is derived in section 3. Then, a neural network-based approach for learning the parameters of an $\mathcal{N}$-Curve mixture from sequence data is presented in section 4. Experiments are conducted on real world data in order to give

*Fraunhofer IOSB is a member of the Fraunhofer Center for Machine Learning.

an initial proof of concept in the context of (multi-modal) n-step sequence prediction (section 5). Section 6 concludes the paper and gives future directions of research.

2 Related Work

For structuring the related work, approaches for 1-step prediction will be presented briefly in 2.1. Next, approaches extended for n-step prediction are presented in section 2.2. Overall, this section addresses approaches that model probability distributions, or stochastic processes, explicitly. Approaches learning distributions in their latent space, like Generative Adversarial Networks (Goodfellow et al. 2014) or Variational Autoencoders (Kingma and Welling 2014), are not considered here.

In accordance with these criteria, sections 2.1 and 2.2 focus on recent extensions of Gaussian Processes and neural networks that generate probabilistic output.

2.1 One-step prediction

In 1-step prediction, the task is to infer the next element in a sequence, given the last m preceding observations of the same sequence. Often, it is used as a building block of n-step prediction by performing 1-step predictions iteratively. Besides that, 1-step prediction is a fundamental component in Bayesian filtering, i.e. its *prediction* step (Särkkä 2013).

Gaussian Process (*GP*) regression (Rasmussen 2003) is a model commonly used in 1-step prediction. Given a collection of sample points of a non-linear function $f(\cdot) : \mathbb{R}^m \rightarrow \mathbb{R}$, a mean function $m(\cdot)$ and a covariance function $k(\cdot, \cdot)$ (*kernel*) the GP yields a multivariate Gaussian prior probability distribution over function space. The Gaussian distribution can be used to determine a conditional predictive distribution over the next element in a sequence given preceding observations (Ellis, Sommerlade, and Reid 2009).

Deep Gaussian Processes (Damianou and Lawrence 2013) extend on the GP framework in order to constitute non-Gaussian, and therefore more complex models. A deep GP is a hierarchy of multiple GPs using non-linear mappings between each layer of the hierarchy. However, the resulting probability densities are intractable and thus require an approximate solution, which can be achieved e.g. by variational approximation (Campbell and Yau 2015).

Bayesian Neural Networks (Bishop 1995, *BNN*) treat all weights and biases as random variables. Bayesian inference is applied during training in order to determine the posterior distribution of the network parameters (a.k.a. *Bayesian Backpropagation*). Due to intractable probability distributions, either Monte Carlo methods (Neal 1992) or approximate inference has to be applied. Common techniques used for approximate inference include variational inference (Blundell et al. 2015), inference based on expectation propagation (Hernández-Lobato and Adams 2015) and Monte Carlo dropout (Gal and Ghahramani 2016).

Mixture Density Networks (Bishop 2006, *MDN*) provide another neural network for probabilistic inference. MDNs are deterministic neural networks, mapping the output of the last layer onto the parameters of a Gaussian (mixture) distribution, thus treating uncertainty estimation as a regression problem. Compared to BNNs, being a deterministic model, these networks are much simpler in terms of inference and computational cost, while still generating probabilistic output. On the downside, MDNs do not allow to make assumptions about model uncertainty in a direct way.

2.2 Multi-step prediction

The models presented in this section build upon the (deep) GP, BNN and MDN models presented in the previous section. The task is to infer the next n subsequent elements of a sequence of interest given m preceding observations.

Recurrent Gaussian Processes. Looking at GPs, the standard GP model can be used for n-step prediction by embedding its 1-step prediction model into a sequential Monte Carlo simulation (Ellis, Sommerlade, and Reid 2009). In case of the deep GP, a recurrent extension has been proposed by Mattos et al. (2015), incorporating a recurrent variational approximation scheme using a state space model-based approach. Being based on the deep GP model, the recurrent deep GP model requires an approximate inference approach for generating predictions. Besides having a computation intensive inference scheme, GP-based approaches grant good control over generated predictions, by explicitly modeling the kernel functions, thus controlling the prior over functions representable by the model. This gives an advantage over most competing neural network-based approaches that generate sequences in a mostly unconstrained fashion. In comparison, the model proposed in this paper also optimizes in function space in order to constrain generated predictions, but grants less control than GP-based approaches as no explicit prior is given.

Bayesian Recurrent Neural Networks (*BRNN*) have been proposed by Fortunato, Blundell, and Vinyals (2017), where the variational Bayesian Backpropagation scheme is adapted for "Backpropagation Through Time". BNNs, and BRNNs respectively, offer robustness to over-fitting, allow probabilistic predictions and provide information about model uncertainty. As a drawback, such models are difficult to train, due to the requirement of approximate inference making the training computationally more intensive and potentially less stable. Further, the need for approximate inference also yields a significant computational overhead when generating predictions.

Recurrent Mixture Density Networks (*RMDN*) are most commonly based on the model proposed in (Graves 2013), where an MDN is stacked on top of an LSTM. The recurrent structure is then used for encoding the observed sequence as well as for generating predictions. It should be noted, that sometimes a temporal convolutional network (Bai, Kolter, and Koltun 2018) is used instead of an LSTM for encoding observations due to its less complex structure. Compared to BRNNs, RMDNs have a simpler structure and thus are easier to train, which is the reason why these models are widely used for sequence prediction (Rudenko et al. 2019). It has to be noted though, that commonly this model (e.g. Alahi et al. 2016; Bartoli et al. 2018) is only used for unimodal predictions. This is most likely due to unimodal n-step prediction being cheap computation-wise, while multimodal n-step prediction requires expensive Monte Carlo

simulation (Hug et al. 2018). Recent approaches actually targeting multi-modal prediction are given by e.g. (Bhattacharyya, Schiele, and Fritz 2018) and (Hug et al. 2018).

3 Modeling stochastic processes using Bézier curves

This section proposes a Bézier curve defined by Gaussian control points capable of describing a stochastic process $\mathcal{G}_T = \{X_t\}_{t\in T}$ with Gaussian random variables $X_t \sim \mathcal{N}(\mu_t, \Sigma_t)$ and index set $T = [0, 1]$. Later on, this concept is extended for modeling random variables following a Gaussian mixture distribution. Throughout this section, simple experiments are conducted to investigate different properties of the proposed model. For these experiments, the model is trained using the approach presented in section 4.

A Bézier curve of degree N

$$B(t, \mathcal{P}) = \sum_{i=0}^{N} b_{i,N}(t) P_i \tag{1}$$

is a polynomial curve constructed by a linear combination of $N + 1$ d-dimensional *control points* $\mathcal{P} = \{P_0, P_1, ...P_N\}$ using the Bernstein polynomials

$$b_{i,N}(t) = \binom{N}{i} (1 - t)^{N-i} t^i \tag{2}$$

for weighting. A curve point is determined by the curve parameter $t \in [0, 1]$.

When modeling a stochastic process $\mathcal{G}_T$ using a parametric curve, each curve point needs to represent a Gaussian distribution. Therefore, a "Gaussian" Bézier curve ψ (abbrev.: $\mathcal{N}$-Curve) is proposed. The $\mathcal{N}$-Curve is an extension of equation (1), where the control points $\mathcal{P}_\mathcal{N} = \{P_0, P_1, ...P_N\}$ are defined to follow a Gaussian distribution with $P_i \sim \mathcal{N}(\mu_i, \Sigma_i)\ \forall P_i \in \mathcal{P}_\mathcal{N}$. The set of mean vectors is denoted as $\mu_\mathcal{P} = \{\mu_0, \mu_1, ..., \mu_N\}$ and the set of covariance matrices $\Sigma_\mathcal{P} = \{\Sigma_0, \Sigma_1, ..., \Sigma_N\}$ respectively. Thus, the $\mathcal{N}$-Curve is defined by a tuple $\psi = (\mu_\mathcal{P}, \Sigma_\mathcal{P})$. Given $\mathcal{P}_\mathcal{N}$, the stochasticity is inherited from the control points to the curve points $B_\mathcal{N}(t, \psi)\ \forall t \in [0, 1]$ by the linear combination of the Gaussian control points, which again follows a Gaussian distribution. Each curve point then defines the parameters of a (multivariate) Gaussian probability distribution

$$B_\mathcal{N}(t, \psi) = (\mu^\psi(t), \Sigma^\psi(t)) \tag{3}$$

with

$$\begin{aligned} \mu^\psi(t) &= \sum_{i=0}^{N} b_{i,N}(t)\mu_i, \text{ and} \\ \Sigma^\psi(t) &= \sum_{i=0}^{N} \left(b_{i,N}(t)\right)^2 \Sigma_i. \end{aligned} \tag{4}$$

Following $Ax+By \sim \mathcal{N}(A\mu_x+B\mu_y, A\Sigma_x A^T+B\Sigma_y B^T)$[1] for $x \sim \mathcal{N}(\mu_x, \Sigma_x)$ and $y \sim \mathcal{N}(\mu_y, \Sigma_y)$, it can directly be seen that $B_\mathcal{N}(t, \psi)$ induces the Gaussian probability density

$$p_t^\psi(x) = p(x|\mu^\psi(t), \Sigma^\psi(t)) = \mathcal{N}(x|\mu^\psi(t), \Sigma^\psi(t)) \tag{5}$$

[1]Following the definition as provided in *The Matrix Cookbook* (Petersen and Pedersen 2008).

at index t.

With respect to the stochastic process $\mathcal{G}_T = \{X_t\}_{t\in T}$, Gaussian distributions at n discrete points in time $\{X_1, ..., X_n\}$ can now be described with $B_\mathcal{N}(t, \psi)$ using n equally distributed values for t, yielding a discrete subset

$$T_* = \{\frac{v}{n-1} | v \in \{0, ..., n-1\}\} = \{t_1, ..., t_n\} \tag{6}$$

of the index set T. Thus, each process index (curve parameter) $t_i \in T_*$ corresponds to its respective sequence index at time $i \in \{1, ..., n\}$. Following this, the Gaussian random variable X_{t_i} at time i is given by

$$X_{t_i} \sim \mathcal{N}(B_\mathcal{N}(t_i, \psi)) = \mathcal{N}(\mu^\psi(t_i), \Sigma^\psi(t_i)) \tag{7}$$

with P_0 and P_n as exact start and end conditions.

Figure 1 depicts a 2-dimensional example for an $\mathcal{N}$-Curve, where an $\mathcal{N}$-Curve with 5 control points and respective covariance ellipses are shown (left). Gaussian random variables X_t along the $\mathcal{N}$-Curve given different values for t are illustrated on the right image. Note that the parametric curve interpolates the mean vectors of all Gaussian distributions through time.

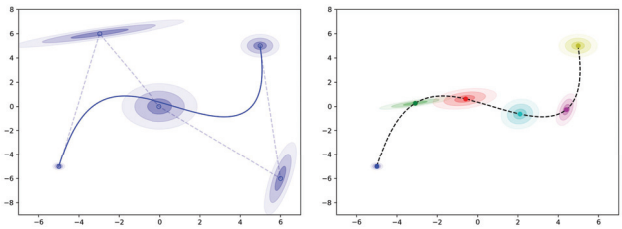

Figure 1: Left: Exemplary 2-dimensional $\mathcal{N}$-Curve (mean curve as given by Gaussian control points). Right: Gaussian distributions along the $\mathcal{N}$-Curve for different t.

Toy example 1: Approximating unimodal processes. This example illustrates the capabilities of the $\mathcal{N}$-Curve model in representing unimodal stochastic processes. As a simple experiment, an unimodal stochastic process with mean values moving along a curve is examined. Samples have been taken around respective mean values at 11 points in time under varying standard deviations to show the impact of the number of control points used. Figure 2 shows the mean curve, standard deviations and samples, as well as sample sequences used to estimate the parameters of an $\mathcal{N}$-Curve. Resulting $\mathcal{N}$-Curves with 5 and 15 control points are depicted in figure 3. It can be seen that the $\mathcal{N}$-Curve model gives a smooth mean curve and compensates noise using the variance of the control points. Further, increasing the number of control points allows to represent the variances more accurately.

3.1 Extending $\mathcal{N}$-Curves for Gaussian mixture probability distributions

While Gaussian probability distributions are a sufficient representation for unimodal sequence data, many real world problems require multi-modal representations. For this case,

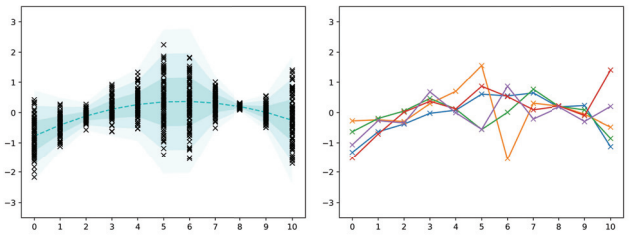

Figure 2: Left: Ground truth mean values, standard deviations and sample points taken from a stochastic process at different points in time. Right: Sample realizations.

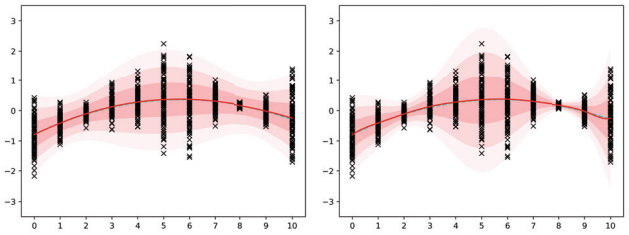

Figure 3: Approximations of given stochastic process using $\mathcal{N}$-Curves with 5 (left) and 15 (right) control points.

a common approach is to use a Gaussian mixture probability distribution $\Upsilon(\{\pi_k\}_{k\in\{1,...,K\}}, \{(\mu_k, \Sigma_k)\}_{k\in\{1,...,K\}})$ defined by K weighted Gaussian components, with

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k), \text{with } \sum_k \pi_k = 1 \qquad (8)$$

as probability density function. In the same way, the concept of $\mathcal{N}$-Curves can be extended to a mixture Ψ of K weighted $\mathcal{N}$-Curves $\{\psi_1, ..., \psi_K\}$ with normalized weights $\pi = \{\pi_1, ..., \pi_K\}$. The random variables X_t at index $t \in T$ then follow the Gaussian mixture distribution

$$X_t \sim \Upsilon(\pi, \{B_{\mathcal{N}}(t, \psi_k)\}_{k\in\{1,...,K\}}). \qquad (9)$$

Accordingly, the probability density at $t \in T$ induced by Ψ is given by

$$p_t^{\Psi}(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu^{\psi_k}(t), \Sigma^{\psi_k}(t)), \qquad (10)$$

with $\mu^{\psi_k}(t)$ and $\Sigma^{\psi_k}(t)$ induced by the Gaussian distribution at $t \in T$ according to the k'th $\mathcal{N}$-Curve, i.e. $(\mu^{\psi_k}(t), \Sigma^{\psi_k}(t)) = B_{\mathcal{N}}(t, \psi_k)$.

Toy example 2: Approximating multi-modal processes. Similar to the unimodal approximation example, each random variable of the stochastic process now follows a bimodal Gaussian mixture distribution and realizations of the process follow one of two possible paths, as shown in the left image of figure 4. This constraint on the realizations introduces a specific structure (two distinct curves) into the training dataset that can be captured by the $\mathcal{N}$-Curve mixture model. The variance is constant for all time steps. Figure 4 shows the ground truth mean and standard deviations

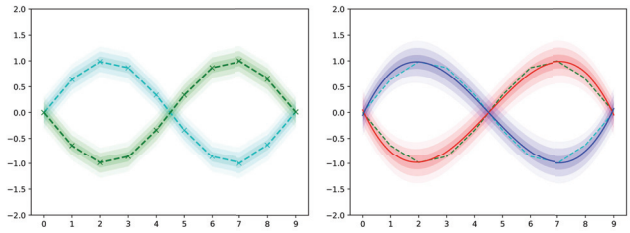

Figure 4: Stochastic process modeling two curves (left) and an approximation given by an $\mathcal{N}$-Curve mixture model using $k = 2$ components (right).

along both curves (left) and the approximation given by an $\mathcal{N}$-Curve mixture model using $k = 2$ components (right). It can be seen that due to the structure in the training data, the $\mathcal{N}$-Curve mixture model is capable of approximating the ground truth distributions.

4 $\mathcal{N}$-Curve Mixture Density Networks

For learning the parameters of an $\mathcal{N}$-Curve mixture from discrete sequence data, using a Mixture Density Network (*MDN*) is proposed. An MDN (Bishop 2006), is, most commonly, a single layer neural network $\Phi(\mathcal{V}) = (\pi, \mu, \Sigma|\mathcal{V})$, that takes an input vector $\mathcal{V}$ and maps it onto the parameters of a mixture of Gaussians, i.e. weights π, mean vectors μ, standard deviations σ and correlations ρ. By generating the parameters from $\mathcal{V}$, the mixture can be conditioned on arbitrary inputs. According to equation (10), an MDN Φ outputs the parameters of an $\mathcal{N}$-Curve mixture, i.e. the weights and Gaussian distribution parameters for each control point $\Phi(\mathcal{V}) = \{(\pi_k, \psi_k)\}_{k\in\{1,...,K\}} = \{(\pi_k, (\mu_{\mathcal{P},k}, \Sigma_{\mathcal{P},k}))\}_{k\in\{1,...,K\}}$. Advantages of using an MDN for learning the $\mathcal{N}$-Curve mixture parameters, rather than other algorithms, like EM, are two-fold. First, the MDN provides an easy approach to learn and process conditional $\mathcal{N}$-Curve mixtures, allowing the model to be used in a conditional inference framework. Second, the MDN can be incorporated easily into (almost) any neural network architecture without the need to control the gradient flow.

Let $\hat{\mathcal{S}} = \{S_1, ..., S_M\}$ be a set of M realizations of a stochastic process with $\mathcal{S}_j = \{x_1^{\mathcal{S}_j}, ..., x_n^{\mathcal{S}_j}\}$ where each $x_i^{\mathcal{S}_j}$ for $i \in \{1, ..., n\}$ is a sample value for the respective random variable X_{t_i} at time i for $t_i \in T_*$ (see 3). In order to simplify training, independence of Gaussian distributions along an $\mathcal{N}$-Curve is assumed. This yields, that the joint probability of the samples $x_i^{\mathcal{S}_j}$ in a sequence $\mathcal{S}_j$ along an $\mathcal{N}$-Curve ψ factorizes, such that

$$p^{\psi}(\mathcal{S}_j) = p^{\psi}(x_1^{\mathcal{S}_j}, ..., x_n^{\mathcal{S}_j}) = \prod_{i=1}^{n} p_{t_i}^{\psi}(x_i^{\mathcal{S}_j}). \qquad (11)$$

Note that $p^{\psi}(\mathcal{S}_j)$ is an unnormalized Gaussian density.

For a single sequence $\mathcal{S}_j$ and an $\mathcal{N}$-Curve ψ, the loss func-

tion is then defined by the negative log-likelihood

$$\begin{aligned}\mathcal{L} &= -log\ p^{\psi}(x_1^{\mathcal{S}_j}, ..., x_n^{\mathcal{S}_j}) \\ &= -\sum_{i=1}^{n} log\ p_{t_i}^{\psi}(x_i^{\mathcal{S}_j})\end{aligned} \tag{12}$$

of the sequence. Therefore, the loss for $\hat{\mathcal{S}}$ is defined as

$$\mathcal{L} = \sum_{j=1}^{M} \left(-\sum_{i=1}^{n} log\ p(x_i^{\mathcal{S}_j} | \mu^{\psi}(t_i), \Sigma^{\psi}(t_i)) \right). \tag{13}$$

Equation (13) can easily be extended for $\mathcal{N}$-Curve mixtures. Given an $\mathcal{N}$-Curve mixture $\Psi = \Phi(\mathcal{V})$, the likelihood of a single training sequence $\mathcal{S}_j$ is now calculated as the weighted linear combination of the likelihood of $\mathcal{S}_j$ for each ψ_k (see equation (11)):

$$p^{\Psi}(\mathcal{S}_j) = \sum_{k=1}^{K} \pi_k p^{\psi_k}(\mathcal{S}_j). \tag{14}$$

Thus, the loss for $\hat{\mathcal{S}}$ can then be defined as

$$\begin{aligned}\mathcal{L} &= \frac{1}{M} \sum_{j=1}^{M} -log \sum_{k=1}^{K} \pi_k p^{\psi_k}(\mathcal{S}_j) \\ &= \frac{1}{M} \sum_{j=1}^{M} -log \sum_{k=1}^{K} exp \left(log \pi_k + \sum_{i=1}^{n} log \left(p_{t_i}^{\psi}(x_i^{\mathcal{S}_j}) \right) \right).\end{aligned} \tag{15}$$

Finally, the $\mathcal{N}$-Curve Mixture Density Network can be trained using a standard gradient descent policy. Here, the gradients are similar to those derived in (Graves 2013), thus this derivation is left out to keep the paper concise.

4.1 Toy example 3: Presence of superfluous mixture components

Next, the behavior of the model in its native formulation is examined in presence of superfluous mixture components. For this, the example data with 2 curves from toy example 2 (section 3.1) is used to train an $\mathcal{N}$-Curve mixture with $k = 6$, i.e. 4 superfluous, components. Preferably, in the resulting model, $\pi = 0$ holds for all 4 unnecessary components, while the remaining components model the two curves in the data with equal weight ($\pi = 0.5$). Figure 5 depicts all $\mathcal{N}$-Curves of the mixture with corresponding weights after training. It can be seen, that $\pi = 0$ only holds for 2 components and 4 components are used to model the data. The components shown in *red* and *yellow* describe one curve and the *cyan* and *magenta* colored components the other. Still, the sum of weights for each curve in the data is approximately equal to 0.5 and the variances of these components are consistent. As a consequence, this behavior could be tackled by e.g. incorporating weight regularization during training.

5 Real world evaluation

In order to provide an initial proof of concept, the performance of the proposed $\mathcal{N}$-Curve MDN is compared to different state-of-the-art sequence prediction models. Further, a qualitative evaluation is presented, inspecting results of the quantitative evaluation and different aspects of the model as partly shown in previous sections.

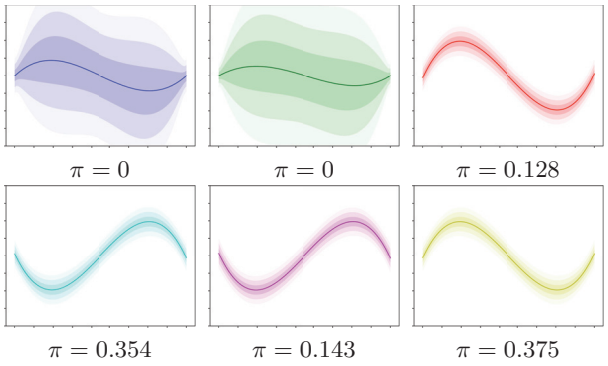

Figure 5: Learned $\mathcal{N}$-Curves for all mixture components. Some superfluous components have been suppressed (blue/green), while multiple similar $\mathcal{N}$-Curves emerge for the two ground truth curves (red/yellow and cyan/magenta).

5.1 Quantitative evaluation

First, the performance of the proposed model is compared to state-of-the-art models in two n-step sequence prediction tasks: trajectory prediction and motion capture sequence prediction. In both tasks, evaluated models need to represent a stochastic process describing $m + n$ time steps, such that given m observations of a process realization, the remaining n steps can be inferred. Following this, the $\mathcal{N}$-Curve MDN needs to map the observation sequence onto an $\mathcal{N}$-Curve mixture, which models the sequence to be predicted. As the $\mathcal{N}$-Curve MDN expects a single input vector $\mathcal{V}$, the observation sequence is encoded using an LSTM network. Note that although m is fixed and a feed-forward network could be used for encoding, the use of an LSTM network yielded better results in both tasks.

Trajectory prediction. An exemplary application domain making use of trajectory prediction is automated video surveillance. Here, predictions are mainly performed on tracklets recorded from a static camera, i.e. sequences consisting of subsequent 2D image coordinates. To highlight the need for multi-modal sequence models, a subset of the Stanford Drone Dataset (Robicquet et al. 2016, *SDD*) is used, as it contains geometrically constrained scenes including e.g. junctions. In order to increase the number of available trajectories, annotations from multiple recordings are combined into a single coordinate system for the SDD scenes *hyang* and *deathcircle*, respectively. Further, positional information is normalized by scaling (x, y)-coordinates to conform $x, y \in [-1, 1]$. The sampling rate is set to 6 Hz. Observation and prediction lengths are fixed to $m = 20$ (3.2s) and $n = 40$ (6.6s).

Here, the $\mathcal{N}$-Curve MDN is compared to two recent approaches focusing on multi-modal trajectory prediction: The Particle LSTM by (Hug et al. 2018) and the "Best of Many Samples" LSTM (abbrev.: LSTM-BMS) by (Bhattacharyya, Schiele, and Fritz 2018). The former embeds an LSTM combined with an MDN into a particle filter cycle in order to produce multi-modal n-step predictions, while the latter utilizes an LSTM to produce multiple sample predictions and is

	Particle LSTM	LSTM-BMS	$\mathcal{N}$-Curve MDN
hyang	0.129 (18.552)	0.147 (-2.574)	**0.088** (**-4.760**)
d-circle	0.366 (15.028)	0.483 (0.389)	**0.314** (**-3.617**)

Table 1: FDE (NLL) values for evaluated models on *hyang* and *deathcircle* (d-circle) scenes taken from the Stanford Drone Dataset.

learned using a "Best of Many Samples" objective function, yielding diverse samples.

For evaluation, a subset of 200 randomly selected tracklets (of length 60) has been used. The results are reported in terms of the *Final Displacement Error* (*FDE*), being the RMSE between the endpoints of the most likely prediction and the respective ground truth trajectory. Additionally, the *Negative Log-Likelihood* (*NLL*, see equation 15) is reported as a metric for probabilistic output. An $\mathcal{N}$-Curve MDN with $k = 3$ components and 6 control points per component is used. For the FDE, the endpoint of the $\mathcal{N}$-Curve with the highest weight $argmax_k\ \pi_k$ is used. In case of the LSTM-BMS[2], 100 sample predictions are generated, which are then clustered using k-means with $k = 3$. Clusters are weighted according to the number of trajectories in each cluster. The endpoint of the mean of the highest weight cluster is used for the FDE. For the NLL, the points of each cluster mean provide the mean values for each time step and the covariance is calculated using the trajectories in each respective cluster. The same procedure is applied for the Particle LSTM using 100 particles.

Results are reported in table 1. It can be seen that the $\mathcal{N}$-Curve MDN performs best in terms of FDE and NLL among the provided models. Larger NLL values for the Particle LSTM are most likely due to the particle filter collapsing onto few particles in regions with small variation, leading to small variances and thus higher NLL values.

Motion capture sequence prediction. For providing a higher dimensional example, sequences consisting of 59-dimensional skeleton description vectors from the CMU motion capture database[3] are used. Here, training is performed on sequences 1 to 4 from subject 35, and testing is performed on sequences 5 to 8 from the same subject. In order to make results comparable to (Mattos et al. 2015) with the code provided by the authors[4], only walking motion is considered. The test set is modified to only contain the first 70 points of each sequence, in order to conform with the fixed sequence length currently necessary for the $\mathcal{N}$-Curve MDN. Further, the data is standardized with zero mean and unitary standard deviation. In contrast to the previous experiment, a control input is given in terms of the y coordinate of the left toes for each time step (during observation and prediction). The observation and prediction lengths are set to $m = 20$ and $n = 50$ time steps. The $\mathcal{N}$-Curve MDN is compared to a simple multilayer perceptron (*MLP*) and the recurrent Gaussian process model (*RGP*) introduced by Mattos et al. (2015).

[2]https://github.com/apratimbhattacharyya18/CGM_BestOfMany

[3]http://mocap.cs.cmu.edu/

[4]https://github.com/zhenwendai/RGP

	MLP	RGP	$\mathcal{N}$-Curve MDN
CMU	0.911	0.822	**0.794**

Table 2: RMSE values for different models on sequences taken from the CMU motion capture database.

For the evaluation, the RMSE over all predicted values in the sequence is reported. Here, the number of components of the $\mathcal{N}$-Curve MDN is set to $k = 1$ using 10 control points. For the MLP, a single hidden layer with 1000 units and $tanh$ activation is used. This MLP directly maps the concatenated observation combined with the control sequence onto the prediction. In case of the RGP model, a 2 hidden layer model with 200 inducing points is used according to the evaluation performed in (Mattos et al. 2015). The results are reported in table 2. Again, the $\mathcal{N}$-Curve MDN performs best among the provided models.

5.2 Qualitative evaluation

Before discussing the smoothing property of the model and the models behavior in presence of superfluous components, qualitative results for both, trajectory prediction and motion capture sequence prediction, are demonstrated. Starting with trajectory prediction, some examples generated by the $\mathcal{N}$-Curve MDN are depicted in figure 6. The model produces diverse predictions, following different possible paths through the scene with probabilities matching the training data. In case of *hyang*, different tracklets just before an intersection show possible predictions in directions given by the pathways. For *deathcircle*, a tracklet entering the roundabout is given, thus predictions leaving at all possible exits are generated. Images (IV) through (VI) of figure 6 depict the individual components of the $\mathcal{N}$-Curve MDN for the first prediction example. It should be noted, that the trained $\mathcal{N}$-Curve MDN represents the entire $m + n$ step sequence, thus the variance for the observation is also shown. As a change in direction (blue and green paths) is less likely, the variance increases towards the end of the prediction, overlapping with the straight path, increasing its probability. This approximately also matches the data: When calculating path probabilities using similar trajectories from the dataset, 15% follow the blue path, 60% the red path and 25% the green path.

For motion capture sequence prediction, the last 3 steps of the observation (cyan) and the mean values for the first 7 steps of the prediction (red) are illustrated in figure 7. As this task is concerned with walking motion, most movement appears at the legs and feet of the skeleton. It can be seen that at the end of the observation sequence, the right foot starts rising and throughout the prediction finishes one step forward, thus correctly approximating the walking motion.

Smoothing. The smoothing property of the $\mathcal{N}$-Curve model is apparent when looking at the representation of single channels in the case of motion capture sequence prediction. Figure 8 shows the 70 step sequences for the *left hand*

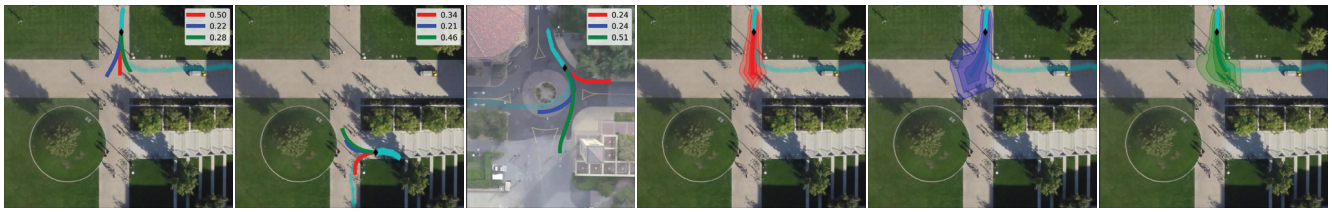

Figure 6: I - III: Predictions generated by a 3-component $\mathcal{N}$-Curve MDN for *hyang* (I and II) and *deathcircle* (III). Observations are shown in (saturated) cyan, ground truth in transparent cyan and predictions in red, green and blue. Predicted path probabilities are indicated in the image legend. IV - VI: Individual components of the first example with variances.

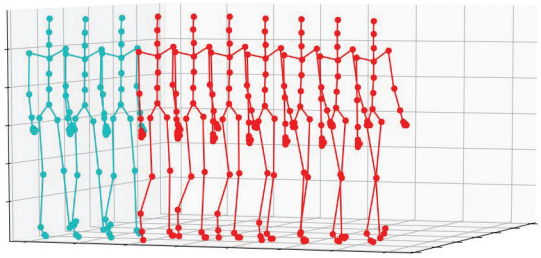

Figure 7: Final 3 steps of observed motion (cyan) and first 7 steps of predicted motion (red).

and *right tibia*. It is clearly visible, that the model learns a smooth mean to represent the entire sequence and tries to cope with noise by varying the variance of the control points.

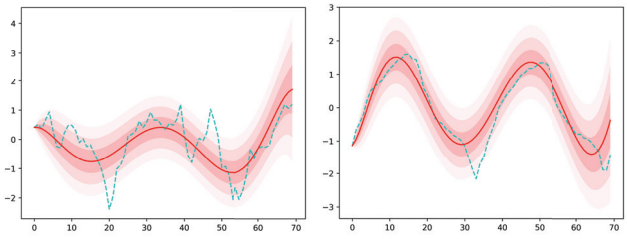

Figure 8: $\mathcal{N}$-Curve approximation of different channels (left hand and right tibia) in motion capture sequence prediction.

Superfluous components. The toy example in 4.1 indicates that, in its native formulation, the model is incapable of completely suppressing superfluous components and overlaps several, nearly equal components, in order to approximate single modes. This behavior can be confirmed looking at the example depicted in figure 9. While the red component is driven towards zero, the blue and green components are quite similar, only differing in the modeled movement speed slightly as indicated by the length of the predicted trajectory. As before, the variances of both components are similar, thus weight regularization could be incorporated during training in order to tackle this behavior.

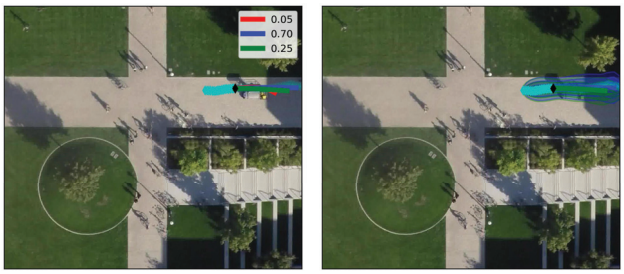

Figure 9: Left: $\mathcal{N}$-Curve mixture model trajectory prediction example with superfluous components. Right: Similar components with variance.

6 Conclusions and Future Works

In this paper, a proof of concept for the $\mathcal{N}$-Curve mixture model, an approach for learning the model of a continuous-time stochastic process defined by Gaussian mixture distributions, has been presented. The approach is based on Bézier curves with Gaussian control points, thus a respective stochastic process is represented by a mixture of parametric, probabilistic curves, termed $\mathcal{N}$-Curves. By using parametric curves and optimizing in function space rather than the d-dimensional space of sequence values, the proposed model is able to generate smooth continuous predictions in a single inference step. Initial experiments show that the presented model is viable for n-step sequence prediction and achieves state-of-the-art performance on different real world tasks. Future work mainly focuses on developing the $\mathcal{N}$-Curve mixture model into a recurrent system, in order to process sequences of variable length (i.e. allow index sets like $t \in \mathbb{R}_0^+$). Further, it should be investigated on the effect of non-linear variance interpolation between control points, due to non-linear weighting.

References

Alahi, A.; Goel, K.; Ramanathan, V.; Robicquet, A.; Fei-Fei, L.; and Savarese, S. 2016. Social lstm: Human trajectory prediction in crowded spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 961–971.

Bai, S.; Kolter, J. Z.; and Koltun, V. 2018. An empirical

evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*.

Bartoli, F.; Lisanti, G.; Ballan, L.; and Del Bimbo, A. 2018. Context-aware trajectory prediction. In *2018 24th International Conference on Pattern Recognition (ICPR)*, 1941–1946. IEEE.

Bhattacharyya, A.; Schiele, B.; and Fritz, M. 2018. Accurate and diverse sampling of sequences based on a "best of many" sample objective. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8485–8493.

Bishop, C. M. 1995. *Neural networks for pattern recognition*. Oxford university press.

Bishop, C. M. 2006. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc.

Blundell, C.; Cornebise, J.; Kavukcuoglu, K.; and Wierstra, D. 2015. Weight uncertainty in neural networks. *arXiv preprint arXiv:1505.05424*.

Campbell, K., and Yau, C. 2015. Bayesian gaussian process latent variable models for pseudotime inference in single-cell rna-seq data. *bioRxiv*.

Damianou, A., and Lawrence, N. 2013. Deep gaussian processes. In *Artificial Intelligence and Statistics*, 207–215.

Ellis, D.; Sommerlade, E.; and Reid, I. 2009. Modelling pedestrian trajectory patterns with gaussian processes. In *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*, 1229–1234. IEEE.

Fortunato, M.; Blundell, C.; and Vinyals, O. 2017. Bayesian recurrent neural networks. *arXiv preprint arXiv:1704.02798*.

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

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, 2672–2680.

Graves, A. 2013. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*.

Hernández-Lobato, J. M., and Adams, R. 2015. Probabilistic backpropagation for scalable learning of bayesian neural networks. In *International Conference on Machine Learning*, 1861–1869.

Hug, R.; Becker, S.; Hübner, W.; and Arens, M. 2018. Particle-based pedestrian path prediction using lstm-mdl models. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2684–2691.

Kingma, D. P., and Welling, M. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Kuleshov, V.; Fenner, N.; and Ermon, S. 2018. Accurate uncertainties for deep learning using calibrated regression. In Dy, J., and Krause, A., eds., *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, 2796–2804. Stockholmsmässan, Stockholm Sweden: PMLR.

Lefèvre, S.; Vasquez, D.; and Laugier, C. 2014. A survey on motion prediction and risk assessment for intelligent vehicles. *ROBOMECH journal* 1(1):1.

Mattos, C. L. C.; Dai, Z.; Damianou, A.; Forth, J.; Barreto, G. A.; and Lawrence, N. D. 2015. Recurrent gaussian processes. *arXiv preprint arXiv:1511.06644*.

Montgomery, D. C.; Jennings, C. L.; and Kulahci, M. 2015. *Introduction to time series analysis and forecasting*. John Wiley & Sons.

Neal, R. M. 1992. Bayesian training of backpropagation networks by the hybrid monte carlo method. Technical report, Citeseer.

Petersen, K. B., and Pedersen, M. S. 2008. The matrix cookbook. *Technical University of Denmark* 7(15):510.

Rasmussen, C. E. 2003. Gaussian processes in machine learning. In *Summer School on Machine Learning*, 63–71. Springer.

Roberts, S.; Osborne, M.; Ebden, M.; Reece, S.; Gibson, N.; and Aigrain, S. 2013. Gaussian processes for time-series modelling. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 371(1984):20110550.

Robicquet, A.; Sadeghian, A.; Alahi, A.; and Savarese, S. 2016. Learning social etiquette: Human trajectory understanding in crowded scenes. In *European conference on computer vision*, 549–565. Springer.

Rudenko, A.; Palmieri, L.; Herman, M.; Kitani, K. M.; Gavrila, D. M.; and Arras, K. O. 2019. Human motion trajectory prediction: A survey. *arXiv preprint arXiv:1905.06113*.

Särkkä, S. 2013. *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks. Cambridge University Press.