Uncertainty on Asynchronous Time Event Prediction

Marin Biloš*, **Bertrand Charpentier***, **Stephan Günnemann**
Technical University of Munich, Germany
{`bilos, charpent, guennemann`}`@in.tum.de`

Abstract

Asynchronous event sequences are the basis of many applications throughout different industries. In this work, we tackle the task of predicting the next event (given a history), and how this prediction changes with the passage of time. Since at some time points (e.g. predictions far into the future) we might not be able to predict anything with confidence, capturing uncertainty in the predictions is crucial. We present two new architectures, WGP-LN and FD-Dir, modelling the evolution of the distribution on the probability simplex with time-dependent logistic normal and Dirichlet distributions. In both cases, the combination of RNNs with either Gaussian process or function decomposition allows to express rich temporal evolution of the distribution parameters, and naturally captures uncertainty. Experiments on class prediction, time prediction and anomaly detection demonstrate the high performances of our models on various datasets compared to other approaches.

1 Introduction

Discrete events, occurring irregularly over time, are a common data type generated naturally in our everyday interactions with the environment (see Fig. 2a for an illustration). Examples include messages in social networks, medical histories of patients in healthcare, and integrated information from multiple sensors in complex systems like cars. The problem we are solving in this work is: given a (past) sequence of asynchronous events, what will happen next? Answering this question enables us to predict, e.g., what action an internet user will likely perform or which part of a car might fail.

While many recurrent models for asynchronous sequences have been proposed in the past [19, 6], they are ill-suited for this task since they output a *single prediction* (e.g. the most likely next event) only. In an asynchronous setting, however, such a single prediction is not enough since the most likely event can change with the passage of time – even if no other events happen. Consider a car approaching another vehicle in front of it. Assuming nothing happens in the meantime, we can expect different events at *different times in the future*. When forecasting a short time, one expects the driver to start overtaking; after a longer time one would expect braking; in the long term, one would expect a collision. Thus, the expected behavior changes depending on the time we forecast, assuming no events occured in the meantime. Fig. 2a illustrates this schematically: having observed a square and a pentagon, it is likely to observe a square after a short time, while a circle after a longer time. Clearly, if some event occurs, e.g. braking/square, the event at the (then) observed time will be taken into account, updating the temporal prediction.

An ad-hoc solution to this problem would be to discretize time. However, if the events are near each other, a high sampling frequency is required, giving us very high computational cost. Besides, since there can be intervals without events, an artificial 'no event' class is required.

In this work, we solve these problems by directly predicting the entire evolution of the events over (continuous) time. Given a past asynchronous sequence as input, we can predict and evaluate for

---

*Equal contribution

any future timepoint what the next event will likely be (under the assumption that no other event happens in between which would lead to an update of our model). Crucially, the likelihood of the events might change and one event can be more likely than others multiple times in the future. This periodicity exists in many event sequences. For instance, given that a person is currently at home, a smart home would predict a high probability that the kitchen will be used at lunch and/or dinner time (see Fig. 1a for an illustration). We require that our model captures such multimodality.

While Fig. 1a illustrates the evolution of the categorical distribution (corresponding to the probability of a specific event class to happen), an issue still arises outside of the observed data distribution. E.g. in some time intervals we can be *certain* that two classes are *equiprobable*, having observed many similar examples. However, if the model has not seen any examples at specific time intervals during training, we do not want to give a confident prediction. Thus, we incorporate *uncertainty* in a prediction directly in our model. In places where we expect events, the confidence will be higher, and outside of these areas the uncertainty in a prediction will grow as illustrated in
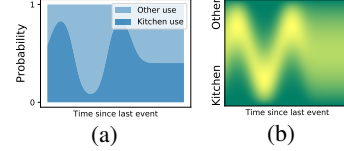

(a)   (b)

Figure 1: (a) An event can be expected multiple times in the future. (b) At some times we should be uncertain in the prediction. Yellow denotes higher probability density.

Fig. 1b. Technically, instead of modeling the evolution of a categorical distribution, we model the *evolution of a distribution on the probability simplex*. Overall, our model enables us to operate with the *asynchronous discrete* event data from the past as input to perform *continuous-time* predictions to the future incorporating the predictions' uncertainty. This is in contrast to existing works as [6, 18].

2 Model Description

We consider a sequence $[e_1, \ldots, e_n]$ of events $e_i = (c_i, t_i)$, where $c_i \in \{1, \ldots, C\}$ denotes the class of the ith event and $t_i \in \mathbb{R}$ is its time of occurrence. We assume the events arrive over time, i.e. $t_i > t_{i-1}$, and we introduce $\tau_i^* = t_i - t_{i-1}$ as the observed time gap between the ith and the $(i-1)$th event. The history preceding the ith event is denoted by $\mathcal{H}_i$. Let $S = \{ \boldsymbol{p} \in [0,1]^C, \sum_c p_c = 1 \}$ denote the set of probability vectors that form the $(C-1)$-dimensional simplex, and $P(\theta)$ be a family of probability distributions on this simplex parametrized by parameters θ. Every sample $\boldsymbol{p} \sim P(\theta)$ corresponds to a (categorical) class distribution.

Given e_{i-1} and $\mathcal{H}_{i-1}$, our goal is to model the evolution of the class probabilities, and their uncertainty, of the next event i over time. Technically, we model parameters $\theta(\tau)$, leading to a distribution P over the class probabilities $\boldsymbol{p}$ for all $\tau \geq 0$. Thus, we can estimate the most likely class after a time gap τ by calculating $\operatorname{argmax}_c \bar{\boldsymbol{p}}(\tau)_c$, where $\bar{\boldsymbol{p}}(\tau) := \mathbb{E}_{\boldsymbol{p}(\tau) \sim P(\theta(\tau))}[\boldsymbol{p}(\tau)]$ is the expected probability vector. Even more, since we do not consider a point estimate, we can get the amount of certainty in a prediction. For this, we estimate the probability of class c being more likely than the other classes, given by $q_c(\tau) := \mathbb{E}_{\boldsymbol{p}(\tau) \sim P(\theta(\tau))}[\mathbb{1}_{\boldsymbol{p}(\tau)_c \geq \max_{c' \neq c} \boldsymbol{p}(\tau)_{c'}}]$. This tells us how certain we are that one class is the most probable (i.e. 'how often' is c the argmax when sampling from P).

Two expressive and well-established choices for the family P are the Dirichlet distribution and the logistic-normal distribution (Appendix A). Based on a common modeling idea, we present two models that exploit the specificities of these distributions: the WGP-LN (Sec. 2.1) and the FD-Dir (Sec. 2.2). We also introduce a novel loss to train these models in Sec. 2.3.

Independent of the chosen model, we have to tackle two core challenges: (1) **Expressiveness.** Since the time dependence of $\theta(\tau)$ may be of different forms, we need to capture complex behavior. (2) **Locality.** For regions out of the observed data we want to have a higher uncertainty in our predictions. Specifically for $\tau \to \infty$, i.e. far into the future, the distribution should have a high uncertainty.

2.1 Logistic-Normal via a Weighted Gaussian Process (WGP-LN)

We start by describing our model for the case when P is the family of logistic-normal (LN) distributions. How to model a compact yet expressive evolution of the LN distribution? Our core idea is to exploit the fact that the LN distribution corresponds to a multivariate random variable whose *logits* follow a *normal distribution* – and a natural way to model the evolution of a normal distribution is a *Gaussian Process*. Given this insight, the core idea of our model is illustrated in Fig. 2: (1) we generate M pseudo points based on a hidden state of an RNN whose input is a sequence, (2) we fit
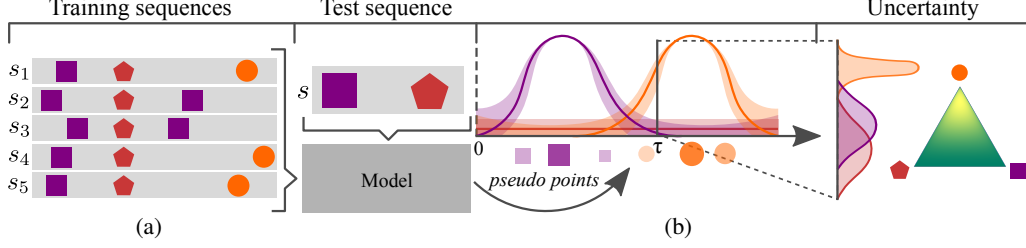
Figure 2: The model framework. (a) During training we use sequences s_i. (b) Given a new sequence of events s the model generates pseudo points that describe $\boldsymbol{\theta}(\tau)$, i.e. the temporal evolution of the distribution on the simplex. These pseudo points are based on the data that was observed in the training examples and weighted accordingly. We also have a measure of certainty in our prediction.

a Gaussian Process to the pseudo points, thus capturing the temporal evolution, and (3) we use the learned GP for estimating the parameters $\boldsymbol{\mu}(\tau)$ and $\boldsymbol{\Sigma}(\tau)$ of the final LN distribution at any specific time τ. Thus, by generating a small number of points we characterize the full distribution.

Classic GP. To keep the complexity low, we train one GP per class c. That is, our model generates M points $(\tau_j^{(c)}, y_j^{(c)})$ per class c, where $y_j^{(c)}$ represents logits. Note that the first coordinate of each pseudo point corresponds to time, leading to the temporal evolution when fitting the GP. Essentially we perform a non-parameteric regression from the time domain to the logit space. Indeed, using a classic GP along with the pseudo points, the parameters θ of the logistic-normal distribution, $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, can be easily computed for any time τ in closed form:

$$\mu_c(\tau) = \boldsymbol{k}_c^T \boldsymbol{K}_c^{-1} \boldsymbol{y}_c, \ \sigma_c^2(\tau) = s_c - \boldsymbol{k}_c^T \boldsymbol{K}_c^{-1} \boldsymbol{k}_c \tag{1}$$

where $\boldsymbol{K}_c$ is the gram matrix w.r.t. the M pseudo points of class c based on a kernel k (e.g. $k(\tau_1, \tau_2) = \exp(-\gamma^2(\tau_1 - \tau_2)^2)$). Vector $\boldsymbol{k}_c$ contains at position j the value $k(\tau_j^{(c)}, \tau)$, and $\boldsymbol{y}_c$ the value $y_j^{(c)}$, and $s_c = k(\tau, \tau)$. At every time point τ the logits then follow a multivariate normal distribution with mean $\boldsymbol{\mu}(\tau)$ and covariance $\boldsymbol{\Sigma} = \text{diag}(\boldsymbol{\sigma}^2(\tau))$.

Using a GP enables us to describe complex functions. Furthermore, since a GP models uncertainty in the prediction depending on the pseudo points, uncertainty is higher in areas far away from the pseudo points. Specifically, it holds for distant future; thus, matching the idea of locality. However, uncertainty is always low around the M pseudo points. Thus M should be carefully picked since there is a trade-off between having high certainty at (too) many time points and the ability to capture complex behavior. Thus, in the following we present an extended version solving this problem.

Weighted GP. We would like to pick M large enough to express rich multimodal functions and allow the model to discard unnecessary points. To do this we generate an additional weight vector $\boldsymbol{w}^{(c)} \in [0, 1]^M$ that assigns the weight $w_j^{(c)}$ to a point $\tau_j^{(c)}$. Giving a zero weight to a point should discard it, and giving 1 will return the same result as with a classic GP. To achieve this goal, we introduce a new kernel function:

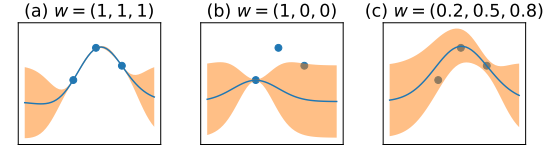
Figure 3: WGP on toy data with different weights. (a) All weights are 1 – classic GP. (b) Zero weights discard points. (c) Mixed weight assignment.
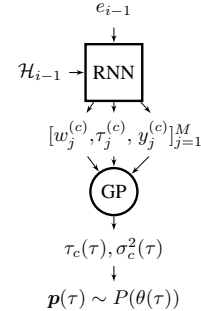
$$k'(\tau_1, \tau_2) = f(w_1, w_2)k(\tau_1, \tau_2) \tag{2}$$

where k is the same as above. The function f weights the kernel k according to the weigths for τ_1 and τ_2. We require f to have the following properties: (1) f should be a valid kernel over the weights, since then the function k' is a valid kernel as well; (2) the importance of pseudo points should not increase, giving $f(w_1, w_2) \leq \min(w_1, w_2)$; this fact implies that a point with zero weight will be discarded since $f(w_1, 0) = 0$ as desired. The function $f(w_1, w_2) = \min(w_1, w_2)$ is a simple choice that fulfills these properties. In Fig. 3 we show the effect of different weights when fitting of a GP (see Appendix B for a more detailed discussion of the behavior of the $\min$ kernel).

To predict μ and σ^2 for a new time τ, we can now simply apply Eq. 1 based on the new kernel k', where the weight for the *query* point τ is 1.

Figure 4: Model diagram

3

To summarize: From a hidden state $h_i = \text{RNN}(e_{i-1}, \mathcal{H}_{i-1})$ we use a a neural network to generate M weighted pseudo points $(w_j^{(c)}, \tau_j^{(c)}, x_j^{(c)})$ per class c. Fitting a Weighted GP to these points enables us to model the temporal evolution of $\mathcal{N}(\mu_c(\tau), \sigma_c^2(\tau))$ and, thus, accordingly of the logistic-Normal distribution. Fig. 4 shows an illustration of this model. Note that the cubic complexity of a GP, due to the matrix inversion, is not an issue since the number M is usually small (< 10), while still allowing to represent rich multimodal functions. Crucially, given the loss defined in Sec. 2.3, our model is fully differentiable, enabling us efficient training.

2.2 Dirichlet via a Function Decomposition (FD-Dir)

Next, we consider the Dirichlet distribution to model the uncertainty in the predictions. The goal is to model the evolution of the concentrations parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_C)^T$ of the Dirichlet over time. Since unlike to the logistic-normal, we cannot draw the connection to the GP, we propose to decompose the parameters of the Dirichlet distribution with expressive (local) functions in order to allow complex dependence on time. Since the concentration parameters $\alpha_c(\tau)$ need to be positive, we propose the following decomposition of $\alpha_c(\tau)$ in the log-space

$$\log \alpha_c(\tau) = \sum_{j=1}^{M} w_j^{(c)} \cdot \mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)}) + \nu \tag{3}$$

where the real-valued scalar ν is a constant prior on $\log \alpha_c(\tau)$ which takes over in regions where the Gaussians are close to 0.

The decomposition into a sum of Gaussians is beneficial for various reasons: (i) First note that the concentration parameter α_c can be viewed as the effective number of observations of class c. Accordingly the larger $\log \alpha$, the more certain becomes the prediction. Thus, the functions $\mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)})$ can describe time regions where we observed data and, thus, should be more certain; i.e. regions around the time $\tau_j^{(c)}$ where the 'width' is controlled by $\sigma_j^{(c)}$. (ii) Since most of the functions' mass is centered around their mean, the locality property is fulfilled. Put differently: In regions where we did not observed data (i.e. where the functions $\mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)})$ are close to 0), the value $\log \alpha_c(\tau)$ is close to the prior value ν. In the experiments, we use $\nu = 0$, thus $\alpha_c(\tau) = 1$ in the out of observed data regions; a common (uninformative) prior value for the Dirichlet parameters. Specifically for $\tau \to \infty$ the resulting predictions have a high uncertainty. (iii) Lastly, a linear combination of translated Gaussians is able to approximate a wide family of functions [4]. And similar to the weighted GP, the coefficients $w_j^{(c)}$ allow discarding unnecessary basis functions.

The basis functions parameters $(w_j^{(c)}, \tau_j^{(c)}, \sigma_j^{(c)})$ are the output of the neural network, and can also be interpreted as weighted pseudo points that determine the regression of Dirichlet parameters $\theta(\tau)$, i.e. $\alpha_c(\tau)$, over time (Fig. 2 & Fig. 4). The concentration parameters $\alpha_c(\tau)$ themselves have also a natural interpretation: they can be viewed as the rate of events after time gap τ.

2.3 Model Training with the Distributional Uncertainty Loss

The core feature of our models is to perform predictions in the future with uncertainty. The classical cross-entropy loss, however, is not well suited to learn uncertainty on the categorical distribution since it is only based on a single (point estimate) of the class distribution. That is, the standard cross-entropy loss for the ith event between the true categorical distribution $\boldsymbol{p}_i^*$ and the predicted (mean) categorical distribution $\overline{\boldsymbol{p}}_i$ is $\mathcal{L}_i^{\text{CE}} = \text{H}[\boldsymbol{p}_i^*, \overline{\boldsymbol{p}}_i(\tau_i^*)] = -\sum_c p_{ic}^* \log \overline{p}_{ic}(\tau_i^*)$. Due to the point estimate $\overline{\boldsymbol{p}}_i(\tau) = \mathbb{E}_{\boldsymbol{p}_i \sim P_i(\theta(\tau))}[\boldsymbol{p}_i]$, the uncertainty on $\boldsymbol{p}_i$ is completely neglected.

Instead, we propose the uncertainty cross-entropy which takes into account uncertainty:

$$\mathcal{L}_i^{\text{UCE}} = \mathbb{E}_{\boldsymbol{p}_i \sim P_i(\theta(\tau_i^*))}[\text{H}[\boldsymbol{p}_i^*, \boldsymbol{p}_i]] = - \int P_i(\theta(\tau_i^*)) \sum_c p_{ic}^* \log p_{ic} \tag{4}$$

Remark that the uncertainty cross-entropy does not use the compound distribution $\overline{\boldsymbol{p}}_i(\tau)$ but considers the expected cross-entropy. Based on Jensen's inequality, it holds: $0 \leq \mathcal{L}_i^{\text{CE}} \leq \mathcal{L}_i^{\text{UCE}}$. Consequently, a low value of the uncertainty cross-entropy guarantees a low value for the classic cross entropy loss, while additionally taking the variation in the class probabilities into account. A comparison between the classic cross entropy and the uncertainty cross-entropy on a simple classification task and anomaly detection in asynchronous event setting is presented in Appendix F.

4

In practice the true distribution p_i^* is often a one hot-encoded representation of the observed class c_i which simplifies the computations. During training, the models compute $P_i(\theta(\tau))$ and evaluate it at the true time of the next event τ_i^* given the past event e_{i-1} and the history $\mathcal{H}_{i-1}$. The final loss for a sequence of events is simply obtained by summing up the loss for each event $\mathcal{L} = \sum_i \mathbb{E}_{p_i \sim P_i(\theta(\tau_i^*))}[\mathrm{H}[p_i^*, p_i]]$.

Fast computation. In order to have an efficient computation of the uncertainty cross-entropy, we propose closed-form expressions. *(1) Closed-form loss for Dirichlet.* Given that the observed class c_i is one hot-encoded by p_i^*, the uncertain loss can be computed in closed form for the Dirichlet:

$$\mathcal{L}_i^{\mathrm{UCE}} = \mathbb{E}_{p_i(\tau_i^*) \sim \mathbf{Dir}(\boldsymbol{\alpha}(\tau_i^*))}[\log p_{c_i}(\tau_i^*)] = \Psi(\alpha_{c_i}(\tau_i^*)) - \Psi(\alpha_0(\tau_i^*)) \tag{5}$$

where Ψ denotes the digamma function and $\alpha_0(\tau_i^*) = \sum_c^C \alpha_c(\tau_i^*)$. *(2) Loss approximation for GP.* For WGP-LN, we approximate $\mathcal{L}_i^{\mathrm{UCE}}$ based on second order series expansion (Appendix C):

$$\mathcal{L}_i^{\mathrm{UCE}} \approx \mu_{c_i}(\tau_i^*) - \log\Big(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\Big) + \frac{\sum_c^C (\exp(\sigma_c^2(\tau_i^*)) - 1) \exp(2\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*))}{2\Big(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\Big)^2} \tag{6}$$

Note that we can now fully backpropagate through our loss (and through the models as well), enabling to train our methods efficiently with automatic differentiation frameworks and, e.g., gradient descent.

Regularization. While the above loss much better incorporates uncertainty, it is still possible to generate pseudo points with high weight values outside of the observed data regime giving us predictions with high confidence. To eliminate this behaviour we introduce a regularization term r_c:

$$r_c = \alpha \underbrace{\int_0^T (\mu_c(\tau))^2 \, d\tau}_{\text{Pushes mean to 0}} + \beta \underbrace{\int_0^T (\nu - \sigma_c^2(\tau))^2 \, d\tau}_{\text{Pushes variance to } \nu} \tag{7}$$

For the WGP-LN, $\mu_c(\tau)$ and $\sigma_c(\tau)$ correspond to the mean and the variance of the class logits which are pushed to prior values of 0 and ν. For the FD-Dir, $\mu_c(\tau)$ and $\sigma_c(\tau)$ correspond to the mean and the variance of the class probabilities where the regularizer on the mean can actually be neglected because of the prior ν introduced in the function decomposition (Eq. 3). In experiments, ν is set to 1 for WGP-LN and $\frac{C-1}{C^2(C+1)}$ for FD-Dir which is the variance of the classic Dirichlet prior with concentration parameters equal to 1. For both models, this regularizer forces high uncertainty on the interval $(0, T)$. In practice, the integrals can be estimated with Monte-Carlo sampling whereas α and β are hyperparameters which are tuned on a validation set.

In [16], to train models capable of uncertain predictions, another dataset or a generative models to access out of observed distribution samples is required. In contrast, our regularizer suggests a simple way to consider out of distribution data which does not require another model or dataset.

3 Point Process Framework

Our models FD-Dir and WGP-LN predict $P(\theta(\tau))$, enabling to evaluate, e.g., $\overline{p}$ after a specific time gap τ. This corresponds to a conditional distribution $q(c|\tau) := \overline{p}_c(\tau)$ over the classes. In this section, we introduce a *point process* framework to generalize FD-Dir to also predict the time distribution $q(\tau)$. This enables us to predict, e.g., the most likely time the next event is expected or to evaluate the joint distribution $q(c|\tau) \cdot q(\tau)$. We call the model FD-Dir-PP.

We modify the model so that each class c is modelled using an inhomogeneous Poisson point process with positive locally integrable intensity function $\lambda_c(\tau)$. Instead of generating parameters $\theta(\tau) = (\alpha_1(\tau), ..., \alpha_C(\tau))$ by function decomposition, FD-Dir-PP generates intensity parameters over time: $\log \lambda_c(\tau) = \sum_{j=1}^M w_j^{(c)} \mathcal{N}(\tau | \tau_j^{(c)}, \sigma_j^{(c)}) + \nu$. The main advantage of such general decomposition is its potential to describe complex multimodal intensity functions contrary to other models like RMTPP [6] (Appendix D). Since the concentration parameter $\alpha_c(\tau)$ and the intensity parameter $\lambda_c(\tau)$ both relate to the number of events of class c around time τ, it is natural to convert one to the other.

Given this C-multivariate point process, the probability of the next class given time and the probability of the next event time are $q(c|\tau) = \frac{\lambda_c(\tau)}{\lambda_0(\tau)}$ and $q(\tau) = \lambda_0(\tau) e^{-\int_0^\tau \lambda_0(s) ds}$ where $\lambda_0(\tau) = \sum_{c=1}^C \lambda_c(\tau)$.

Since the classes are now modelled via a point proc., the log-likelihood of the event $e_i = (c_i, \tau_i^*)$ is:

$$\log q(c_i, \tau_i^*) = \log q(c_i | \tau_i^*) + \log q(\tau_i^*) = \underbrace{\log \frac{\lambda_{c_i}(\tau_i^*)}{\lambda_0(\tau_i^*)}}_{(i)} + \underbrace{\log \lambda_0(\tau_i^*)}_{(ii)} - \underbrace{\int_0^{\tau_i^*} \lambda_0(t) dt}_{(iii)} \quad (8)$$

The terms (ii) and (iii) act like a regularizer on the intensities by penalizing large cumulative intensity $\lambda_0(\tau)$ on the time interval $[t_{i-1}, t_i]$ where no events occurred. The term (i) is the standard cross-entropy loss at time τ_i. Or equivalently, by modeling the distribution $\mathbf{Dir}(\lambda_1(\tau), .., \lambda_C(\tau))$, we see that term (i) is equal to $\mathcal{L}_i^{\text{CE}}$ (see Section 2.3). Using this insight, we obtain our final FD-Dir-PP model: We achieve uncertainty on the class prediction by modeling $\lambda_c(\tau)$ as concentration parameters of a Dirichlet distribution and train the model with the loss of Eq. 8 replacing term (i) by $\mathcal{L}_i^{\text{UCE}}$. As it becomes apparent FD-Dir-PP differs from FD-Dir only in the regularization of the loss function, enabling it to be interpreted as a point process.

4   Related Work

Predictions based on discrete sequences of events regardless of time can be modelled by Markov Models [2] or RNNs, usually with its more advanced variants like LSTMs [11] and GRUs [5]. To exploit the time information some models [15, 19] additionally take time as an input but still output a single prediction for the entire future. In contrast, temporal point process framework defines the intensity function that describes the rate of events occuring over time.

RMTPP [6] uses an RNN to encode the event history into a vector that defines an exponential intensity function. Hence, it is able to capture complex past dependencies and model distributions resulting from simple point processes, such as Hawkes [10] or self-correcting [12], but not e.g. multimodal distributions. On the other hand, Neural Hawkes Process [18] uses continuous-time LSTM which allows specifying more complex intensity functions. Now the likelihood evaluation is not in closed-form anymore, but requires Monte Carlo integration. However, these approaches, unlike our models, do not provide any uncertainty in the predictions. In addition, WGP-LN and FD-Dir can be extended with a point process framework while having the expressive power to represent complex time evolutions.

Uncertainty in machine learning has shown a great interest [9, 8, 14]. For example, uncertainty can be imposed by introducing distributions over the weights [3, 17, 20]. Simpler approaches introduce uncertainty directly on the class prediction by using Dirichlet distribution independent of time [16, 21]. In contrast, the FD-Dir model models complex temporal evolution of Dirichlet distribution via function decomposition which can be adapted to have a point process interpretation.

Other methods introduce uncertainty time series prediction by learning state space model with Gaussian processes [7, 23]. Alternatively, RNN architecture has been used to model the probability density function over time [25]. Compared to these models, the WGP-LN model uses both Gaussian processes and RNN to model uncertainty and time. Our models are based on pseudo points. Pseudo points in a GP have been used to reduce the computational complexity [22]. Our goal is not to speed up the computation, since we control the number of points that are generated, but to give them different importance. In [24] a weighted GP has been considered by rescaling points; in contrast, our model uses a custom kernel to discard (pseudo) points.

5   Experiments

We evaluate our models on large-scale synthetic and real world data. We compare to neural point process models: **RMTPP** [6] and **Neural hawkes process** [18]. Additionally, we use various RNN models with the knowledge of the time of the next event. We measure the accuracy of class prediction, accuracy of time prediction, and evaluate on an anomaly detection task to show prediction uncertainty.

We split the data into train, validation and test set (60%–20%–20%) and tune all models on a validation set using grid search over learning rate, hidden state dimension and L_2 regularization. After running models on all datasets 5 times we report mean and standard deviation of test set accuracy. Details

FD-Dir on 3-G · · · WGP-LN on 3-G · · · FD-Dir on Multi-G · · · WGP-LN on Multi-G
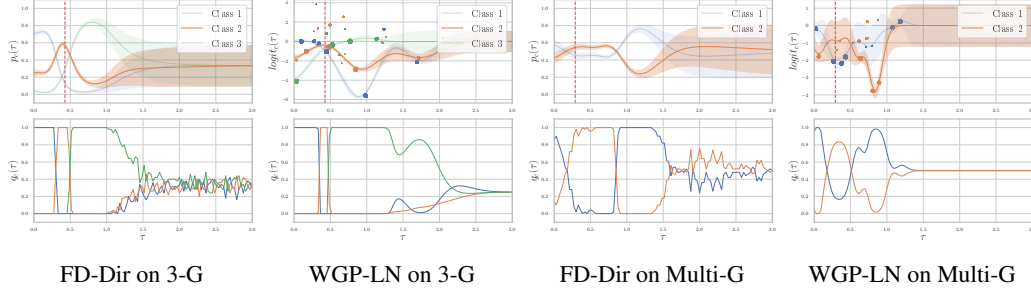
Figure 5: Visualization of the prediction evolution. The red line indicates the true time of the next event for an example sequence. Here, both models predict the orange class, which is correct, and capture the variation of the class distributions over time. Generated points from WGP-LN are plotted with the size corresponding to the weight. For predictions in the far future, both models given high uncertainty.

on model selection can be found in Appendix H.1. The code and further supplementary material is available online.[2]

We use the following data (more details in Appendix G): (1) **Graph.** We generate data from a directed Erdős–Rényi graph where nodes represent the states and edges the weighted transitions between them. The time it takes to cross one edge is modelled with one normal distribution per edge. By randomly walking along this graph we created 10K asynchronous events with 10 unique classes. (2) **Stack Exchange.**[3] Sequences contain rewards as events that users get for participation on a question answering website. After preprocessing according to [6] we have 40 classes and over 480K events spread over 2 years of activity of around 6700 users. The goal is to predict the next reward a user will receive. (3) **Smart Home** [1].[4] We use a recorded sequence from a smart house with 14 classes and over 1000 events. Events correspond to the usage of different appliances. The next event will depend on the time of the day, history of usage and other appliances. (4) **Car Indicators.** We obtained a sequence of events from car's indicators that has around 4000 events with 12 unique classes. The sequence is highly asynchronous, with τ ranging from milliseconds to minutes.

Visualization. To analyze the behaviour of the models, we propose visualizations of the evolutions of the parameters predicted by FD-Dir and WGP-LN.

Set-up: We use two toy datasets where the probability of an event depends only on time. The first one (**3-G**) has three classes occuring at three distinct times. It represents the events in the Fig. 13a. The second one (**Multi-G**) consists of two classes where one of them has two modes and corresponds to the Fig. 1a. We use these datasets to showcase the importance of time when predicting the next event. In Fig. 5, the four top plots show the evolution of the categorical distribution for the FD-Dir and the logits for the WGP-LN with 10 points each. The four bottom plots describe the certainty of the models on the probability prediction by plotting the probability $q_c(\tau)$ that the probability of class c is higher than others, as introduced in Sec. 2. Additionally, the evolution of the dirichlet distribution over the probability simplex is presented in Appendix E.

Results. Both models learn meaningful evolutions of the distribution on the simplex. For the 3-G data, we can distinguish four areas: the first three correspond to the three classes; after that the prediction is uncertain. The Multi-G data shows that both models are able to approximate multimodal evolutions.

Class prediction accuracy. The aim of this experiment is to assess whether our models can correctly predict the class of the next event, given the time at which it occurs. For this purpose, we compare our models against Hawkes and RMTPP and evalute prediction accuracy on the test set.

Results. We can see (Fig. 6) that our models consistently outperform the other methods on all datasets. Results of the other baselines can be found in Appendix H.2.

Time-Error evaluation. Next, we aim to assess the quality of the time intervals at which we have confidence in one class. Even though WGP-LN and the FD-Dir do not model a distribution on

---

[2]https://www.daml.in.tum.de/uncertainty-event-prediction
[3]https://archive.org/details/stackexchange
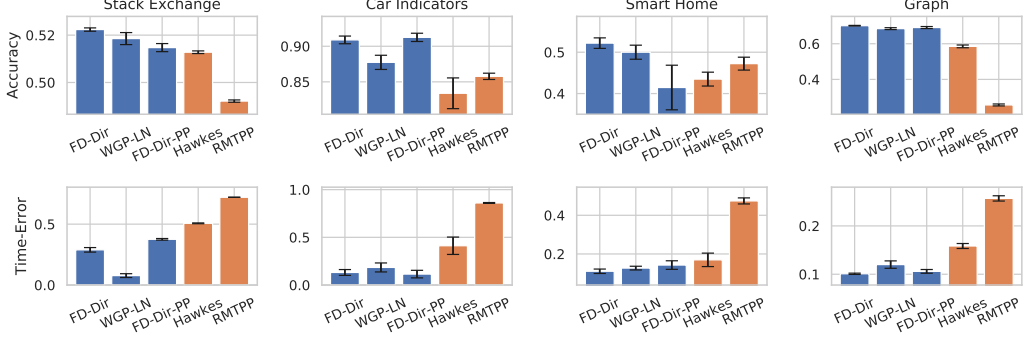[4]https://sites.google.com/site/tim0306/datasets

7

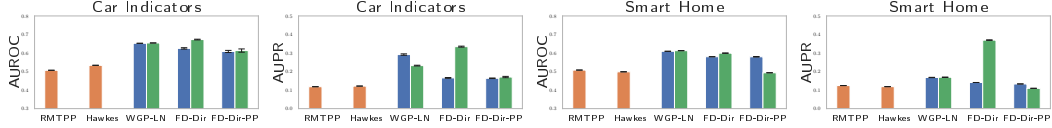Figure 6: Class accuracy (top; higher is better) and Time-Error (bottom; lower is better).

Figure 7: AUROC and APR comparison across dataset on anomaly detection. The orange and blue bars use categorical uncertainty score whereas the green bars use distributional uncertainty.

time, they still have intervals at which we are certain in a class prediction, making the conditional probability a good indicator of the time occurrence of the event.

Set-up. While models predicting a *single* time $\hat{\tau}_i$ for the next event often use the MSE score $\frac{1}{n}\sum_{i=1}^{n}(\hat{\tau}_i - \tau_i^*)^2$, in our case the MSE is not suitable since one event can occur at multiple time points. In the conventional least-squares approach, the mean of the true distribution is an optimal prediction; however, here it is almost always wrong. Therefore, we use another metric which is better suited for multimodal distributions. Assume that a model returns a score function $g_i^{(c)}(\tau)$ for each class regarding the next event i, where a large value means the class c is likely to occur at time τ. We define Time-Error $= \frac{1}{n}\sum_{i=1}^{n}\int \mathbb{1}_{g_i^{(c)}(\tau) \geq g_i^{(c)}(\tau_i^*)}d\tau$. The Time-Error computes the size of the time intervals where the predicted score is larger than the score of the observed time τ_i^*. Hence, a performant model would achieve a low Time-Error if its score function $g_i^{(c)}(\tau)$ is high at time τ^*. As the score function in our models, we use the corresponding class probability $\bar{p}_{ic}(\tau)$.

Results. We can see that our models clearly obtain the best results on all datasets. The point process version of FD-Dir does not improve the performance. Thus, taking also into account the class prediction performance, we recommend to use our other two models. In Appendix H.3 we compare FD-Dir-PP with other neural point process models on time prediction using the MSE score and achieve similar results.

Anomaly detection & Uncertainty. The goal of this experiment is twofold: (1) it assesses the ability of the models to detect anomalies in asynchronous sequences, (2) it evaluates the quality of the predicted uncertainty on the categorical distribution. For this, we use a similar set-up as [16].

Set-up: The experiments consist in introducing anomalies in datasets by changing the occurrence time of 10% of the events (at random after the time transformation described in appendix G). Hence, the anomalies form out-of-distribution data, whereas unchanged events represent in-distribution data. The performance of the anomaly detection is assessed using Area Under Receiver Operating Characteristic (AUROC) and Area Under Precision-Recall (AUPR). We use two approaches: (i) We consider the *categorical uncertainty* on $\bar{p}(\tau)$, i.e., to detect anomalies we use the predicted probability of the true event as the anomaly score. (ii) We use the *distribution uncertainty* at the observed occurrence time provided by our models. For WGP-LN, we can evaluate $q_c(\tau)$ directly (difference of two normal distributions). For FD-Dir, this probability does not have a closed-form solution so instead, we use the concentration parameters which are also indicators of out-of-distribution events. For all scores, i.e $\bar{p}(\tau)_c$, $q_c(\tau)$ and $\alpha_c(\tau)$, a low value indicates a potential anomaly around time τ.

Results. As seen in Fig. 5, the FD-Dir and the WGP-LN have particularly good performance. We observe that the FD-Dir gives better results especially with distributional uncertainty. This might be due to the power of the concentration parameters that can be viewed as number of similar events around a given time.

6 Conclusion

We proposed two new methods to predict the evolution of the probability of the next event in asynchronous sequences, including the distributions' uncertainty. Both methods follow a common framework consisting in generating pseudo points able to describe rich multimodal time-dependent parameters for the distribution over the probability simplex. The complex evolution is captured via a Gaussian Process or a function decomposition, respectively; still enabling easy training. We also provided an extension and interpretation within a point process framework. In the experiments, WGP-LN and FD-Dir have clearly outperformed state-of-the-art models based on point processes; for event and time prediction as well as for anomaly detection.

References

[1] *Activity Recognition in Pervasive Intelligent Environments of the Atlantis Ambient and Pervasive Intelligence series, Atlantis Press*, chapter 8. Springer, 2010.

[2] Ron Begleiter, Ran El-Yaniv, and Golan Yona. On prediction using variable order markov models. *J. Artif. Int. Res.*, 22(1):385–421, December 2004.

[3] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight Uncertainty in Neural Networks. *arXiv e-prints*, page arXiv:1505.05424, May 2015.

[4] Craig Calcaterra and Axel Boldt. Approximating with gaussians. *arXiv e-prints*, page arXiv:0805.3795, 2008.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[6] Nan Du, Hanjun Dai, Rakshit Trivedi, Utkarsh Upadhyay, Manuel Gomez-Rodriguez, and Le Song. Recurrent marked temporal point processes: Embedding event history to vector. In *KDD*. ACM, 2016.

[7] Stefanos Eleftheriadis, Tom Nicholson, Marc Deisenroth, and James Hensman. Identification of gaussian process state space models. In *NIPS*. Curran Associates, Inc., 2017.

[8] Dhivya Eswaran, Stephan Günnemann, and Christos Faloutsos. *The Power of Certainty: A Dirichlet-Multinomial Model for Belief Propagation*, pages 144–152. 06 2017.

[9] Meire Fortunato, Charles Blundell, and Oriol Vinyals. Bayesian Recurrent Neural Networks. *arXiv e-prints*, page arXiv:1704.02798, Apr 2017.

[10] Alan G Hawkes. Spectra of some self-exciting and mutually exciting point processes. *Biometrika*, 58(1):83–90, 1971.

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[12] Valerie Isham and Mark Westcott. A self-correcting point process. *Stochastic Processes and Their Applications*, 8(3):335–347, 1979.

[13] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[14] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *arXiv e-prints*, page arXiv:1612.01474, Dec 2016.

[15] Yang Li, Nan Du, and Samy Bengio. Time-dependent representation for neural event sequence prediction, 2018.

[16] Andrey Malinin and Mark Gales. Predictive Uncertainty Estimation via Prior Networks. *arXiv e-prints*, page arXiv:1802.10501, Feb 2018.

[17] Patrick L. McDermott and Christopher K. Wikle. Bayesian Recurrent Neural Network Models for Forecasting and Quantifying Uncertainty in Spatial-Temporal Data. *arXiv e-prints*, page arXiv:1711.00636, Nov 2017.

[18] Hongyuan Mei and Jason M Eisner. The neural hawkes process: A neurally self-modulating multivariate point process. In *Advances in Neural Information Processing Systems*, 2017.

[19] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. Phased LSTM: accelerating recurrent network training for long or event-based sequences. *CoRR*, abs/1610.09513, 2016.

[20] Hippolyt Ritter, Aleksandar Botev, and David Barber. A scalable laplace approximation for neural networks. In *International Conference on Learning Representations*, 2018.

[21] Peter Sadowski and Pierre Baldi. Neural network regression with beta, dirichlet, and dirichlet-multinomial outputs, 2019.

[22] Edward Snelson and Zoubin Ghahramani. Sparse gaussian processes using pseudo-inputs. In *Advances in neural information processing systems*, 2006.

[23] Ryan Turner, Marc Deisenroth, and Carl Rasmussen. State-space inference and learning with gaussian processes. In Yee Whye Teh and Mike Titterington, editors, *AISTATS*, 2010.

[24] Junfeng Wen, Negar Hassanpour, and Russell Greiner. Weighted gaussian process for estimating treatment effect. 2018.

[25] Kyongmin Yeo, Igor Melnyk, Nam Nguyen, and Eun Kyung Lee. Learning temporal evolution of probability distribution with recurrent neural network, 2018.

Supplementary Materials: Uncertainty on Asynchronous Time Event Prediction

A Distributions

For reference, we give here the definition of the Dirichlet and Logistic-normal distribution.

A.1 Dirichlet distribution

The Dirichlet distribution with concentration parameters $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_K)$, where $\alpha_i > 0$, has the probability density function:

$$f(\boldsymbol{x}; \boldsymbol{\alpha}) = \frac{\prod_{i=1}^{K} \Gamma(\alpha_i)}{\Gamma\left(\sum_{i=1}^{K} \alpha_i\right)} \prod_{i=1}^{K} x_i^{\alpha_i - 1} \tag{A.1}$$

where Γ is a gamma function:

$$\Gamma(\alpha) = \int_0^\infty \alpha^{z-1} e^{-\alpha} dz$$

A.2 Logistic-normal distribution (LN)

The logistic normal distribution is a generalization of the logit-normal distribution for the multidimensional case. If $\boldsymbol{y} \in \mathbb{R}^C$ follows a normal distribution, $\boldsymbol{y} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, then

$$\boldsymbol{x} = \left[\frac{e^{y_1}}{\sum_{i=1}^{C} e^{y_i}}, \ldots, \frac{e^{y_C}}{\sum_{i=1}^{C} e^{y_i}} \right]$$

follows a logistic-normal distribution.

B Behavior of the min kernel

The desired behavior of the min kernel function can easily be illustrated by considering the gram matrix $\boldsymbol{K}$ and vector $\boldsymbol{k}$, which are required to estimate μ and σ^2 for a new time point τ. W.l.o.g. consider M pseudo points $\tau_1, \ldots, \tau_M$ such that $w_1 < \cdots < w_M$. Since the new query point is observed we assign it weight 1. It follows:

$$\boldsymbol{k} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_M \end{bmatrix} \odot \begin{bmatrix} k(\tau_1, \tau) \\ k(\tau_2, \tau) \\ \vdots \\ k(\tau_M, \tau) \end{bmatrix}, \quad \boldsymbol{K} = \begin{bmatrix} w_1 & w_1 & \ldots & w_1 \\ w_1 & w_2 & \ldots & w_2 \\ \vdots & \vdots & \ddots & \vdots \\ w_1 & w_2 & \ldots & w_M \end{bmatrix} \odot \begin{bmatrix} k(\tau_1, \tau_1) & \ldots & k(\tau_1, \tau_M) \\ k(\tau_2, \tau_1) & \ldots & k(\tau_2, \tau_M) \\ \vdots & \ddots & \vdots \\ k(\tau_M, \tau_1) & \ldots & k(\tau_M, \tau_M) \end{bmatrix} \tag{B.2}$$

Assuming $w_1 = 0$ returns $\boldsymbol{k}$ without the first row and $\boldsymbol{K}$ without the first row and column. Plugging them back into equation 1 we can see that the point τ_1 is discarded, as desired. In practice, the weights have values from interval $[0, 1]$ which in turn gives us the ability to *softly discard* points. This is shown in Fig. 3 we can see that the mean line does not have to cross through the points with weights < 1 and the variance can remain higher around them.

C Computation of the approximation for the uncertainty cross-entropy of WGP-LN

Given true categorical distribution $\boldsymbol{p}_i^*$, and predicted $\boldsymbol{p}_i(\tau)$, the uncertainty cross-entropy can be calculated as in Eq. 4. For the WGP-LN model $\boldsymbol{p}_i(\tau) = \text{softmax}(\boldsymbol{z}_i(\tau))$, where $\boldsymbol{z}_i(\tau)$ are logits that come from a Gaussian process and follow a normal distribution $\mathcal{N}(\boldsymbol{\mu}_i(\tau), \boldsymbol{\Sigma}_i(\tau))$,. Therefore, $\exp(\boldsymbol{z}_i(\tau))$ follows a log-normal distribution. We will use this to derive an approximation of the loss.

From now on, we omit τ from the equations. Mean and variance for $\sum_c^C \exp(\boldsymbol{z}_{c_i})$ are then:

$$\mathbb{E}\left[\sum_c^C \exp(\boldsymbol{z}_{c_i})\right] = \sum_c^C \exp(\boldsymbol{\mu}_{c_i} + \boldsymbol{\sigma}_{c_i}^2/2)$$

$$\mathbf{Var}\left[\sum_{c_i}^C \exp(\boldsymbol{z}_{c_i})\right] = \sum_{c_i}^C (\exp(\sigma_{c_i}^2) - 1)\exp(2\boldsymbol{\mu}_{c_i} + \boldsymbol{\sigma}_{c_i}^2) \tag{C.3}$$

The expectation of the cross entropy loss given that logits are following a normal distribution is

$$\mathcal{L}_i^{\text{UCE}} = \mathbb{E}[\mathcal{L}_i^{\text{CE}}] = \mathbb{E}[\log(\exp(\boldsymbol{z}_{c_i}))] - \mathbb{E}\left[\log\left(\sum_c^C \exp(\boldsymbol{z}_{c_i})\right)\right] \tag{C.4}$$

In general, given a random variable x, we can approximate expectation of $\log x$ by performing a second order Taylor expansion around the mean μ:

$$\mathbb{E}[\log x] \approx \mathbb{E}\left[\log\mu + \underbrace{\frac{(\log\mu)'}{1!}(x-\mu)}_{\mathbb{E}[x-\mu]=0} + \frac{(\log\mu)''}{2!}(x-\mu)^2\right]$$

$$\approx \mathbb{E}[\log\mu] - \frac{\mathbf{Var}[x]}{2\mu^2} \tag{C.5}$$

Using C.5 together with C.3 and plugging into C.4 we get a closed-form solution for the loss for event i:

$$\mathcal{L}_i^{\text{UCE}} \approx \mu_{c_i}(\tau_i^*) - \log\left(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\right) + \frac{\sum_c^C (\exp(\sigma_c^2(\tau_i^*)) - 1)\exp(2\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*))}{2\left(\sum_c^C \exp(\mu_c(\tau_i^*) + \sigma_c^2(\tau_i^*)/2)\right)^2} \tag{C.6}$$

D Non Expressiveness of RMTPP intensities

The intensity function has the following form in the RMTPP model [6]:

$$\log\lambda_0(t) = \boldsymbol{v}^T \cdot \boldsymbol{h}_i + w(t - t_i) + b \tag{D.7}$$

The variables $\boldsymbol{v}$, w and b are learned parameters and $\boldsymbol{h}_i$ is given by the hidden state of an RNN. The only dependence on t is $(t - t_i)$. RMTPP is then limited to monotonic intensity functions with respect to time.

E Dirichlet Evolution

Our goal is to model the evolution of a distribution on a probability simplex. Fig. 1b shows this for two classes. In general, we can do the same for multiple classes. Fig. 8 shows an example of the Dirichlet distribution for three classes, and how it changes over time. This evolution is the output of the FD-Dir model trained on the 3-G dataset, created to simulate the car example from Sec. 1 (see also Fig. 13a in Appendix G). The three classes: *overtaking*, *breaking* and *collision* occur independently of each other at three different times. The represent the corners of the triangle in Fig. 8.

We can distinguish three cases: (a) at first we are certain that the most likely class is *overtaking*; (b) as time passes, the most likely class becomes *breaking*, (c) and finally *collision*. After that, we are in the area where we have not seen any data and do not have a confident prediction (d).
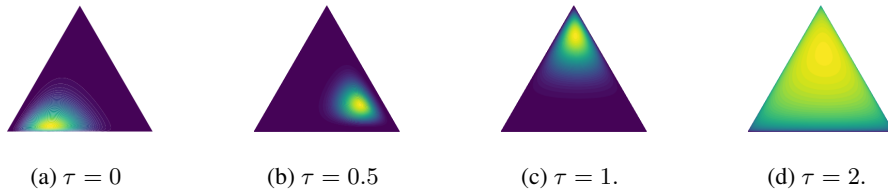

(a) $\tau = 0$ (b) $\tau = 0.5$ (c) $\tau = 1.$ (d) $\tau = 2.$

Figure 8: Dirichlet distribution at different time for the 3-G dataset with $\sigma = 1$.

F Comparison of the classical cross-entropy and the uncertainty cross-entropy

F.1 Simple classification task

In this section, we do not consider temporal data. The goal of this experiment is to show the benefit of the uncertainty cross-entropy compare with the classical cross-entropy loss on a simple classification task. As a consequence, we do not consider RNN in this section. We use a simple two layers neural network to predict the concentration parameters of a Dirichlet distribution from the input vector.

Set-up. The set-up is similar to [16] and consists of two datasets of 1500 instances divided in three equidistant 2-D Gaussians. One dataset contains non-overlapping classes (**NOG**) whereas the other contains overlapping classes (**OG**). Given one input x_i, we train simple two layers neural networks to predict the concentration parameters of a Dirichlet distribution $\mathbf{Dir}(\alpha_1(x_i), \alpha_2(x_i), \alpha_3(x_i))$ which model the uncertainty on the categorical distribution $\boldsymbol{p}(x_i)$. On each dataset, we train two neural networks. One neural network is trained with the classic cross-entropy loss $\mathcal{L}^{\mathrm{CE}}$ which uses only the mean prediction $\bar{\boldsymbol{p}}(x_i)$. The second neural network is trained with the uncertainty cross-entropy loss plus a simple α-regularizer:

$$\mathcal{L}^{\mathrm{UCE}} + \left| \alpha_0(x_i) - \sum_j \mathbb{1}_{x_j \in N_w(x_i)} \right| \tag{F.8}$$

where x_i is the input 2-D vector and $N_w(x_i) = \{x', ||x' - x_i||_2^2 < w\}$ is its euclidean neighbourhood of size w. We set $w = 10^{-5}$ for the non-overlapping Gaussians and $w = 10^{-2}$ for the overlapping Gaussians.

Results. The categorical entropy $-\sum_c p_c(x_i) \log p_c(x_i)$ is a good indicator to know how certain is the categorical distribution $\boldsymbol{p}(x_i)$ at point x_i. A high entropy meaning that the categorical distribution is uncertain. For non overlapping Gaussians (Fig. 9a and 9b), we remark that both losses learn uncertain categorical distribution only on thin borders. However, for overlapping Gaussians (See Fig. 9c and 9d),the uncertainty cross-entropy loss learns more uncertain categorical distributions because of the thicker borders.

Other interesting results are the concentration parameters learned by the two models (Fig. 10, Fig. 11). The classic cross-entropy loss learns very high value for $\alpha_1(x_i), \alpha_2(x_i), \alpha_3(x_i)$ which does match with the true distribution of the data. In contrast, the uncertainty cross-entropy learn meaningful alpha values for both datasets (delimiting the in-distribution areas for α_0 and centred around the classes for the others).
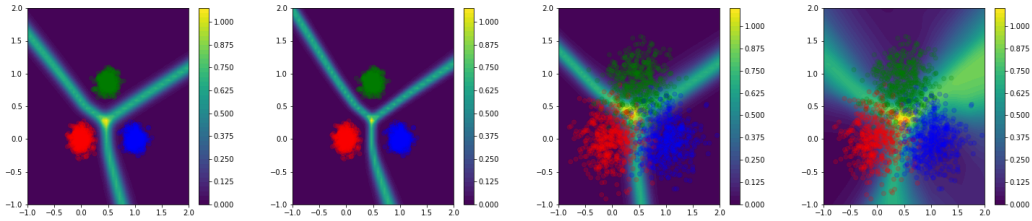
(a) NOG - CE - Cat. Ent.   (b) NOG - UCE - Cat. Ent.   (c) OG - CE - Cat. Ent.   (d) OG - UCE - Cat. Ent.

Figure 9: The Figures 9a and 9b plot the entropy of the categorical distribution learned on a classification task with three non-overlapping Gaussians. They show categorical entropy learned with the classic cross-entropy and learned with the uncertainty cross-entropy. The Figures 9c and 9d plot the entropy of the categorical distribution learned on a classification task with three overlapping Gaussians. They show categorical entropy learned with the classic cross-entropy and learned with the uncertainty cross-entropy.

(a) CE - α_0　　(b) CE - α_1　　(c) CE - α_2　　(d) CE - α_3

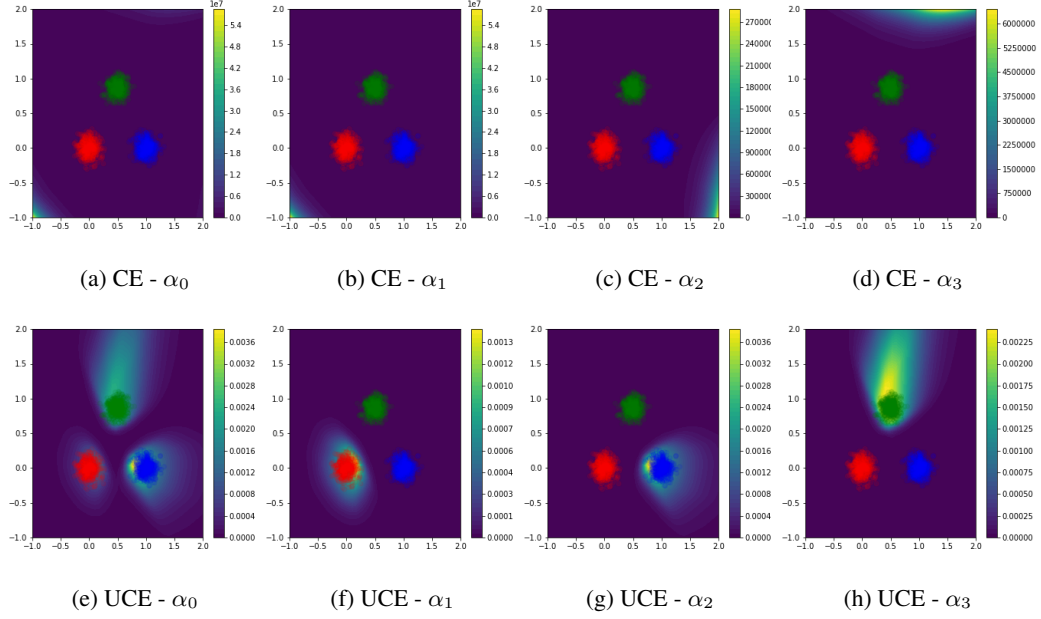(e) UCE - α_0　　(f) UCE - α_1　　(g) UCE - α_2　　(h) UCE - α_3
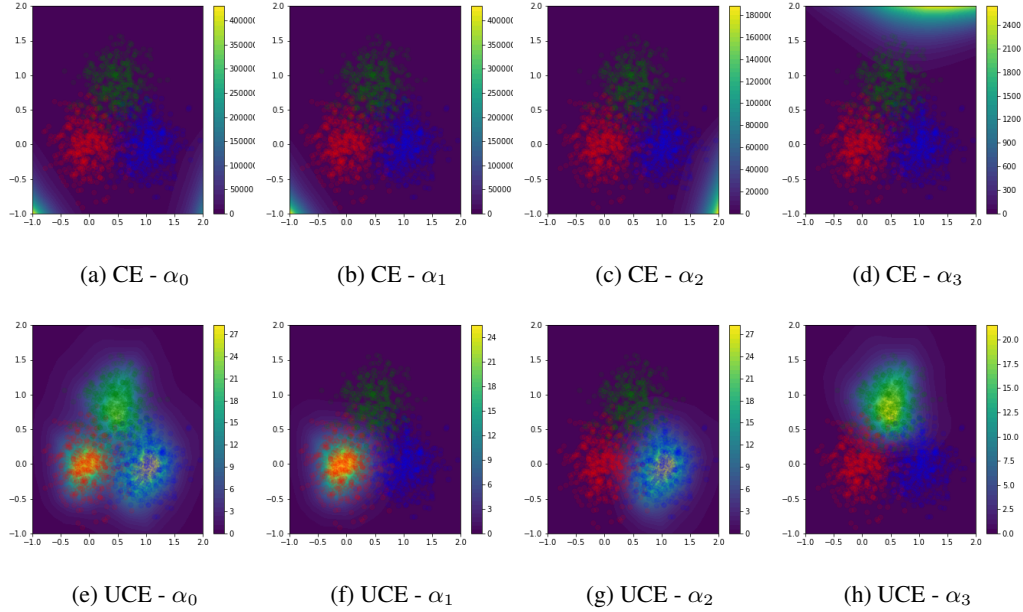
Figure 10: Concentration parameters of the Dirichlet distribution on a classification task with three non-overlapping Gaussians. The figures 10a, 10b, 10c, 10d are α_0, α_1, α_2, α_3 learned with the classic cross-entropy. The figures 10a, 10b, 10c, 10d are α_0, α_1, α_2, α_3 learned with the uncertainty cross-entropy.

(a) CE - α_0　　(b) CE - α_1　　(c) CE - α_2　　(d) CE - α_3

(e) UCE - α_0　　(f) UCE - α_1　　(g) UCE - α_2　　(h) UCE - α_3

Figure 11: Concentration parameters of the Dirichlet distribution on a classification task with three non-overlapping Gaussians. The figures 11a, 11b, 10c, 11d are α_0, α_1, α_2, α_3 learned with the classic cross-entropy. The figures 11a, 11b, 10c, 11d are α_0, α_1, α_2, α_3 learned with the uncertainty cross-entropy.

F.2 Asynchronous Event Prediction

In this section, we consider temporal data. The goal of this experiment is again to show the benefit of the uncertainty cross-entropy compared to the classical cross-entropy in the case of asynchronous event prediction.

Set-up. For this purpose, we use the same set-up describe in the experiment Anomaly detection & Uncertainty. We trained the model FD-Dir with three different type of losses: (1) The classical cross-entropy (CE), (2) The classical cross-entropy with regularization described in section 2.3 (CE + reg) and (3) The classical uncertainty cross-entropy with regularization described in section 2.3 (UCE + reg).
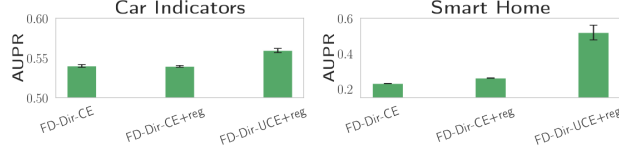

Figure 12: Loss comparison in anomaly detection

Results. The results are shown in Fig. 12. The loss UCE + reg consistently improves the anomaly detection based on the distribution uncertainty.
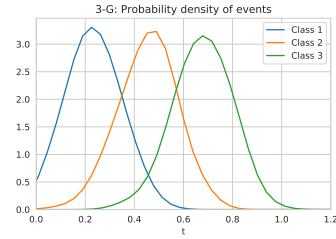
G Datasets

In this section we describe the datasets in more detail. The time gap between two events $\tau_i^* = t_i - t_{i-1}$ is first log-transformed before applying min-max normalization: $\hat{\tau}_i^* = \frac{\tau_i' - \min(\tau_i^{*'})}{(\max(\tau_i^*) - \min(\tau_i^{*'}))}$ with $\tau_i^{*'} = \log(\tau_i^* + \epsilon), \epsilon > 0$.

3-G.   We use $C = 3$ and draw from a normal distribution $P(\tau|c_i) = \mathcal{N}(i+1, 1.)$. This dataset tries to imitate the setting from Fig. 13a as explained in 1. We generate 1000 events. Probability density is shown in figure 13b. Models that are not taking time into account cannot solve this problem. Below is the code. We create the **Multi-G** dataset similarly.
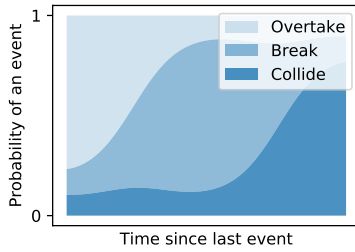
(a) Car example explained in section 1 where probabilities of events to occur change over time

(b) Probability density of events in K-Gaussians dataset. We can see that classes are independent of history.

```
def generate():
    data = np.zeros((1000, 2))
    for i in range(1000):
        i_class = np.random.choice(3, 1)[0]
        time = np.random.normal(i_class + 1, 1.)
        while time <= 0:
            time = np.random.normal(i_class + 1, 1.)
        data[i, 0] = i_class
        data[i, 1] = time
    return data
```

Car Indicators. A sequence contains signals from a single car during one ride. We remove signals that are perfectly correlated giving 6 unique classes in the end. Top 3 classes make up 33%, 32%, and 16% of a total respectively. From figure 14 we can see that the setting is again asynchronous.
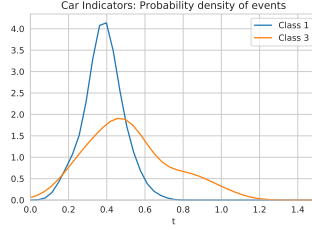

Figure 14: Probability density of events in Car Indicators dataset for 2 selected classes. Time is log-transformed.

Graph. We generate graph G with 10 nodes and 48 edges between them. We assign variables μ and σ to each transition (edge) between events (nodes). The time it takes to make a transition between nodes i and j is drawn from normal distribution $\mathcal{N}(\mu_{ij}, \sigma_{ij}^2)$. By performing a random walk on the graph we create 10 thousand events. This dataset is similar to K-Gaussians with the difference that a model needs to learn the relationship between events together with the time dependency. Parts of the trace are shown in figure 15.
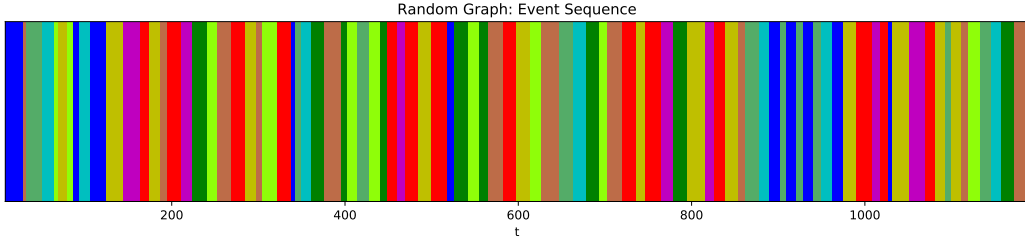

Figure 15: Trace of events for random graph. Different colors represent different classes and width of a single column represents the time that passed.

H   Details of experiments

We test our models (**WGP-LN**, **FD-Dir** and **DPP**) against neural point process models (**RMTPP** and **Hawkes**) and simple baselines (**RNN** and **LSTM** – getting only history as an input, **F-RNN** and **F-LSTM** – having also the real time of the next event as an additional input; thus, they get a strong advantage!). We test on real world (**Stack Exchange**, **Car Indicators** and **Smart Home**) and synthetic datasets (**Graph**). We show that our models consistently outperform all the other models when evaluated with class prediction accuracy and Time-Error.

H.1   Model selection

We apply the same tuning technique to all models. We split all datasets into train–validation–test sets ($60\% - 20\% - 20\%$), use the validation set to select a model and the test set to get final scores. For Stack Exchange dataset we split on users. In all other datasets we split the trace based on time. We search over dimension of a hidden state $\{32, 64, 128, 256\}$, batch size $\{16, 32, 64\}$ and L_2 regularization parameter $\{0, 10^{-3}, 10^{-2}, 10^{-1}\}$. We use the same learning rate 0.001 for all models and an Adam optimizer [13], run each of them 5 times for maximum of 100 epochs with early stopping after 5 consecutive epochs without improvement in the validation loss. For the number of points M we pick 3 for WGP-LN and 20 for FD-Dir. WGP-LN and FD-Dir have additional regularization (Eq. 7) with hyperparameters α and β. For both models we choose $\alpha = \beta = 10^{-3}$. Model with the highest mean accuracy on the validation set is selected. We use GRU cell [5] for both of our models. We trained all models on GPUs (1TB SSD).

Table 1: Class accuracy comparison for all models on all datasets

	Car Indicators	Graph	Smart Home	Stack Exchange
FD-Dir	0.909 ± 0.005	$\mathbf{0.701 \pm 0.002}$	$\mathbf{0.522 \pm 0.013}$	$\mathbf{0.522 \pm 0.001}$
Dir-PP	$\mathbf{0.912 \pm 0.006}$	0.691 ± 0.006	0.415 ± 0.054	0.515 ± 0.002
WGP-LN	0.877 ± 0.010	0.685 ± 0.005	0.500 ± 0.017	0.519 ± 0.003
Hawkes	0.834 ± 0.022	0.585 ± 0.008	0.435 ± 0.017	0.513 ± 0.001
RMTPP	0.858 ± 0.004	0.257 ± 0.005	0.472 ± 0.016	0.492 ± 0.000
F-LSTM	0.855 ± 0.006	0.657 ± 0.002	0.411 ± 0.029	-
F-RNN	0.849 ± 0.013	0.615 ± 0.011	0.472 ± 0.035	-
LSTM	0.858 ± 0.010	0.251 ± 0.008	0.375 ± 0.026	-
RNN	0.838 ± 0.016	0.258 ± 0.008	0.437 ± 0.017	-

Table 2: Time-Error comparison for all models on all datasets

	Car Indicators	Graph	Smart Home	Stack Exchange
FD-Dir	$\mathbf{0.115 \pm 0.040}$	$\mathbf{0.101 \pm 0.001}$	$\mathbf{0.111 \pm 0.011}$	0.289 ± 0.019
WGP-LN	0.184 ± 0.047	0.120 ± 0.008	0.127 ± 0.010	$\mathbf{0.077 \pm 0.016}$
FD-Dir-PP	0.132 ± 0.031	0.106 ± 0.004	0.143 ± 0.022	0.375 ± 0.007
Hawkes	0.412 ± 0.091	0.158 ± 0.005	0.170 ± 0.035	0.507 ± 0.003
RMTPP	0.860 ± 0.004	0.257 ± 0.005	0.474 ± 0.016	0.721 ± 0.001
F-LSTM	0.277 ± 0.118	0.141 ± 0.002	0.209 ± 0.023	-
F-RNN	0.516 ± 0.105	0.146 ± 0.004	0.186 ± 0.011	-
LSTM	0.860 ± 0.010	0.251 ± 0.008	0.376 ± 0.026	-
RNN	0.841 ± 0.016	0.258 ± 0.008	0.439 ± 0.017	-

H.2 Results

Tables 1 and 2, together with Fig. 16 show test results for *all* models on *all* datasets for Class accuracy and Time-Error.
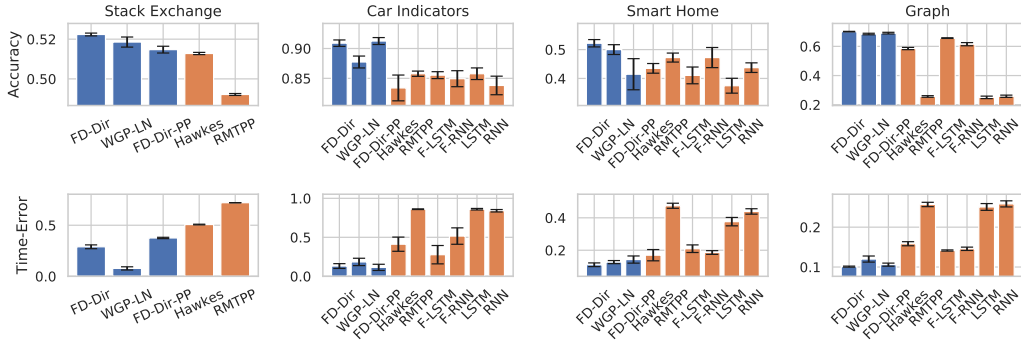

Figure 16: Class accuracy (top) and Time-Error (bottom) comparison across datasets

H.3 Time Prediction with Point Processes

The benefit of the point process framework is the ability to get the point estimate for the time $\hat{\tau}$ of the next event:

$$\hat{\tau} = \int_0^\infty t q(\tau) dt \tag{H.9}$$

where

$$q(\tau) = \lambda_0(\tau) \exp\left(-\int_0^\tau \lambda_0(s) ds\right) \tag{H.10}$$

17

The usual way to evaluate the quality of this prediction is using an MSE score. As we have already discussed in Sec. 5, this is not optimal for our use case. Nevertheless, we did preliminary experiments comparing our neural point process model **FD-Dir-PP** to others. We use **RMTPP** [6] since it achieves the best results. On Car Indicators dataset our model has mean MSE score of 0.4783 while RMTPP achieves 0.4736. At the same time FD-Dir-PP outperforms RMTPP on other tasks (see Sec. 5).