ISI Informal Workshop on Learning, Algorithms and Networks
February 18, 2020

# Adversarial Games for generative modeling of Temporally-Marked Event Sequences

Giuseppe Manco
Institute of High Performance Computing and Networks

# Giuseppe Manco

Laboratory of
Advanced Analytics
on Complex Data

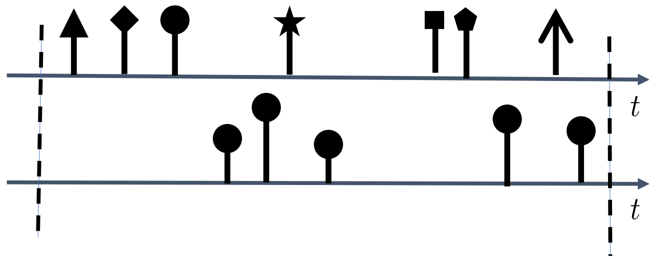
- Director of Research at Institute for high performance computing and networking of National Research Council of Italy
- Head of the ADALab
  - Behavioral Modeling and Scalable Analytics
  - 8 Researchers, 5 fellows


# Temporally Marked Point Processes (TMPPs)

- Data = (noisy) observations of events generated by some complex process
  - Each event refers to a single process (execution) instance and stores at least:
    (1) a time-stamp and (2) a categorical attribute representing the type of the event
  - **Trace** $\mathcal{H}$ = sequence of all events linked to a process instance (stores instance's history)

$$\mathcal{H} = \{ \boldsymbol{e}_1, \boldsymbol{e}_2, \dots, \boldsymbol{e}_m \} \qquad \boldsymbol{e_i} = (\boldsymbol{a}_i, t_i)$$

- Marked Temporal Point Processes' view




**Discrete events in continuous time**

intra-event times may vary over a wider range of temporal scales
(differently from "classic" time-series data)

# Application contexts

- Business Process (or Workflow) Management events
  - execution traces of a process, storing information on performed activities
- Medical events
  - acute incidents, doctor’s visits, tests, diagnoses, and medications
- Consumer behavior
  - purchasing patterns
- “Quantified self” data
  - Wearable devices and apps to record eating, traveling, working, sleeping, waking
- Social media actions
  - previous posts, shares, comments, messages,…
- Smart cities and mobility patterns
  - trajectories, taxi/car/public transportation adoptions, etc.
- Smart industry
  - Optimization of the production, predictive maintenance…

# Goal

- **General goal:** Understand the structural and temporal dynamics of process traces
  - can provide insights on the complex patterns that govern the process
  - can be used to forecast future events

- **Specific objectives:**
  - **Data generation**: Generate new realistic traces from scratch
    - to have a surrogate of real data (due to privacy/scalability constraints), or for simulation analyses
  - **Predict future events:** Given an incomplete trace (partial history of a process instance)

$$\mathcal{H}_{<h} = \{e_1, \dots, e_{h-1}\} \quad \text{with} \quad h \leq m$$

Make forecasts on the subsequent events (structured-prediction / conditional-generation task)

# Approach: Learn a Probabilistic Generative Model

- Probability distribution to learn

$$P(\boldsymbol{e}_h|\mathcal{H}_{<\boldsymbol{h}}) = P(\boldsymbol{a}_h, t_h|\mathcal{H}_{<\boldsymbol{h}})$$

- Possible decompositions
  - Independence

$$P(\boldsymbol{a}_h, t_h|\mathcal{H}_{<\boldsymbol{h}}) = P(\boldsymbol{a}_h\,|\mathcal{H}_{<\boldsymbol{h}})P(t_h|\mathcal{H}_{<\boldsymbol{h}})$$

  - Time is context – dependent

$$P(\boldsymbol{a}_h, t_h|\mathcal{H}_{<\boldsymbol{h}}) = P(\boldsymbol{a}_h\,|\mathcal{H}_{<\boldsymbol{h}})P(t_h|\boldsymbol{a}_h, \mathcal{H}_{<\boldsymbol{h}})$$

# Intermezzo: ML estimation

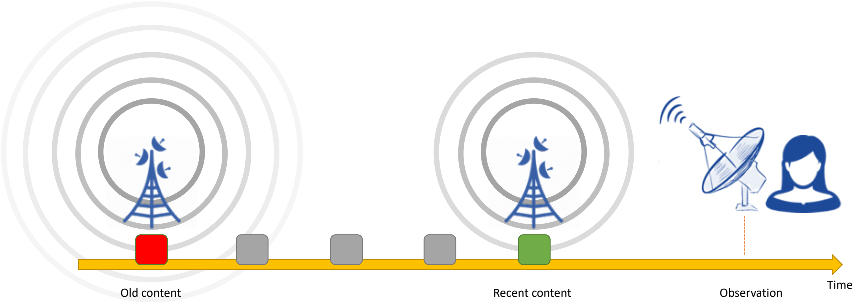
- Given a sample $X = \{ x_1, x_2, \dots, x_n\}$
  - sampled from a true distribution $\mathbb{P}_\mathrm{r}$
- Given a proposal distribution $\mathbb{P}_\theta$ parametrized by $\theta$

- Find the parameter $\hat{\theta}$ that optimizes the likelihood:

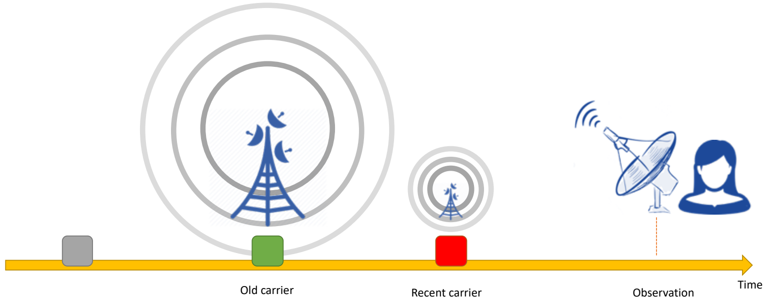
$$\begin{aligned}\hat{\theta} &= \mathrm{argmax}_\theta P_\theta(X) \\ &= \mathrm{argmax}_\theta \prod_i P_\theta(x_i) \\ &= \mathrm{argmax}_\theta \sum_i \log P_\theta(x_i) \\ &= \mathrm{argmax}_\theta \mathbb{E}_{x\sim\mathbb{P}_\mathrm{r}} \log P_\theta(x)\end{aligned}$$

# Parameterized Models

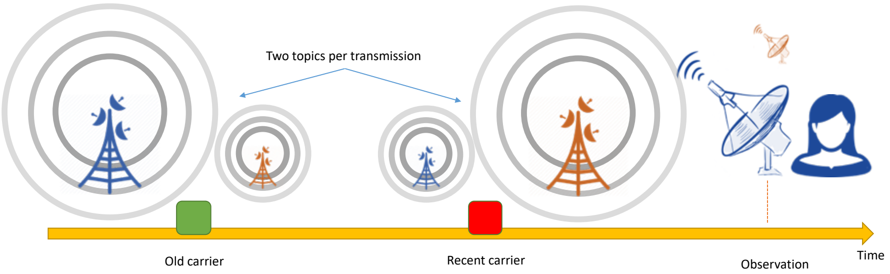
- Three key elements for modeling contagion




Time Dependency




Carrier Dependency




Topic Dependency

G. Manco, E. Ritacco and N. Barbieri, "A Factorization Approach for Survival Analysis on Diffusion Networks," *IEEE TKDE (2019)*

# Auto-Regressive generative models

- Explicit probability model
  - factorized as a product of conditional per-step distributions (chain rule)

$$P(\mathcal{H}) = \prod_{h=1}^{m} P(\boldsymbol{e}_h | \mathcal{H}_{<h})$$

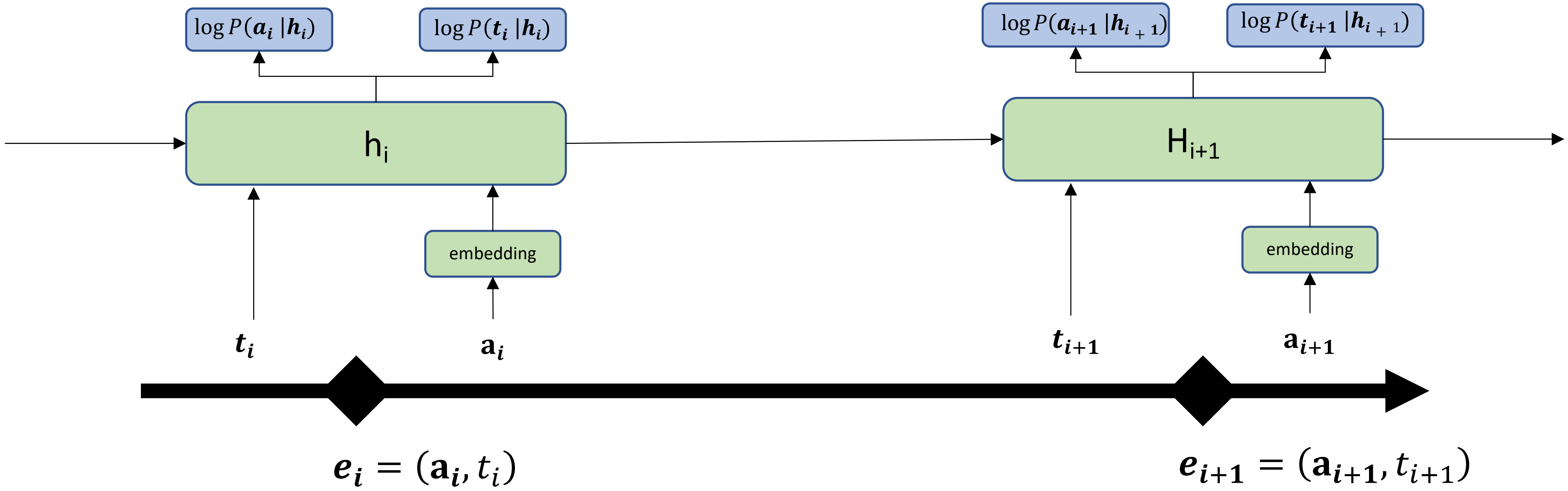
- Conditional probs. are approximated with an NN (usually an RNN)

$$P(\boldsymbol{e}_h | \mathcal{H}_{<\boldsymbol{h}}) \approx f_\theta(\mathbf{s}_h)$$
$$\mathbf{s}_h = RNN(\boldsymbol{e}_{h-1}, \mathbf{s}_{h-1})$$

- ***Learning***: tractable **maximum-likelihood** (**ML**) training
  - optimizes exact likelihood
- Inference:
  - generates a suffix (or an entire sequence) via incremental auto-regression

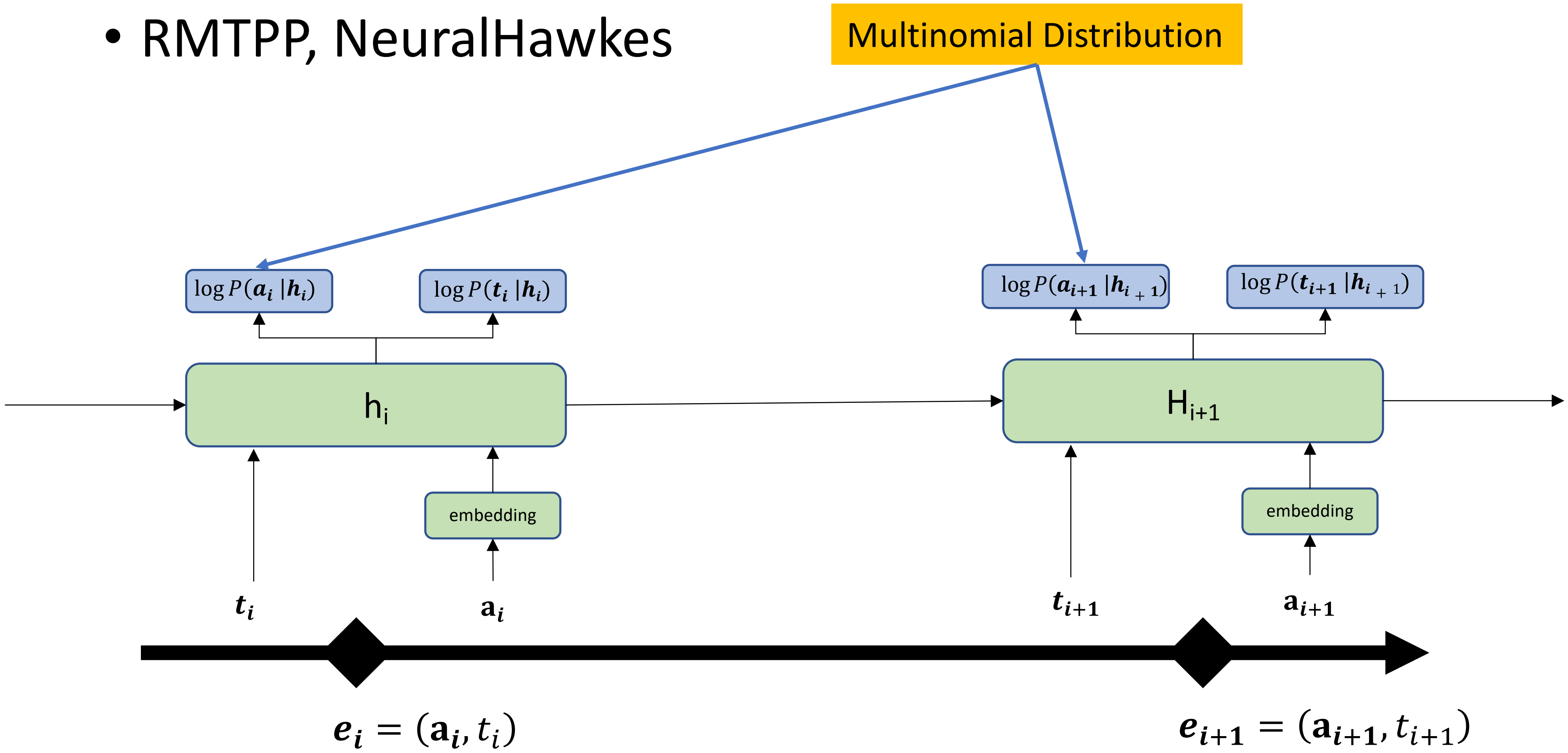
# Examples

- RMTPP, NeuralHawkes [Du et al 2016] [Mei 2017]



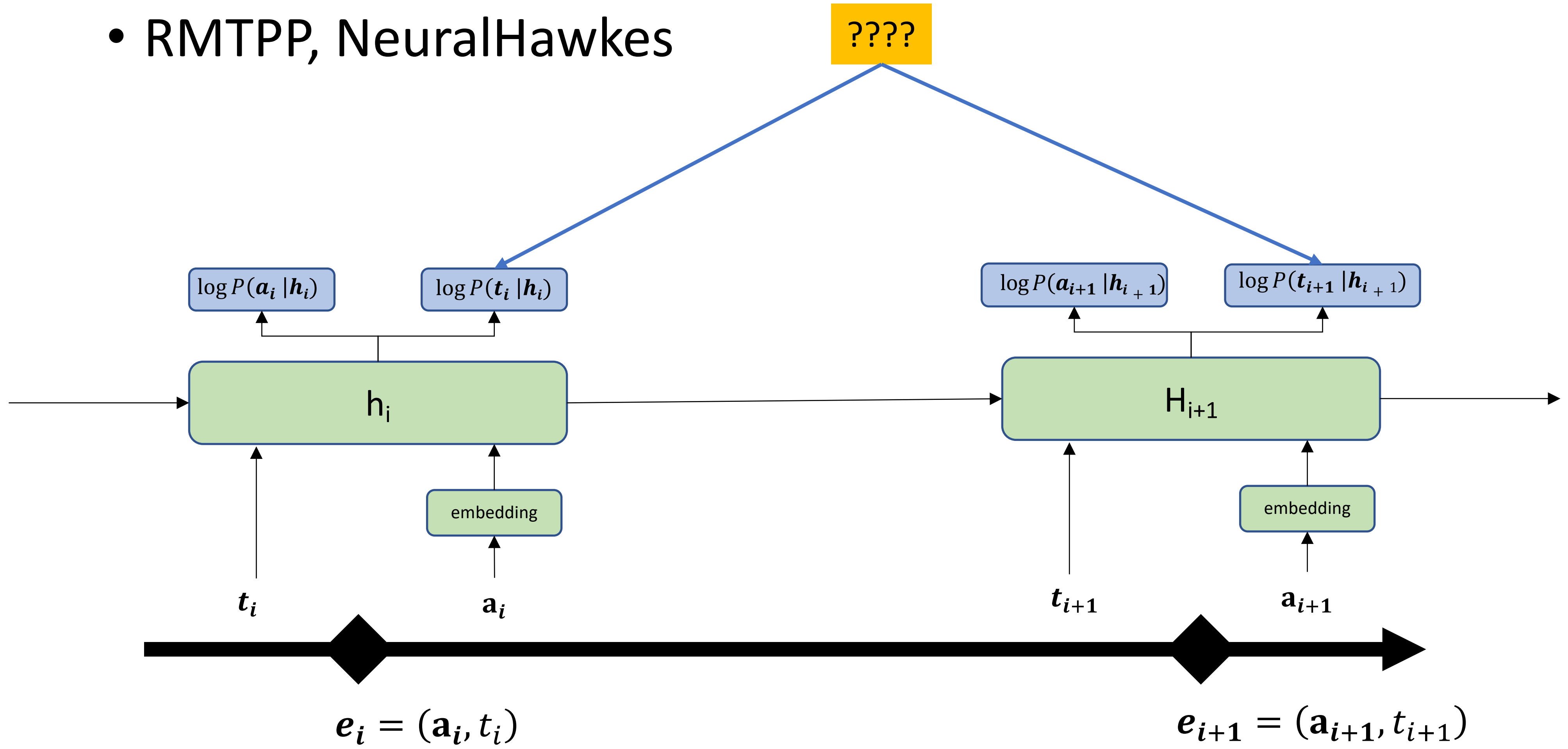
# Examples

- RMTPP, NeuralHawkes

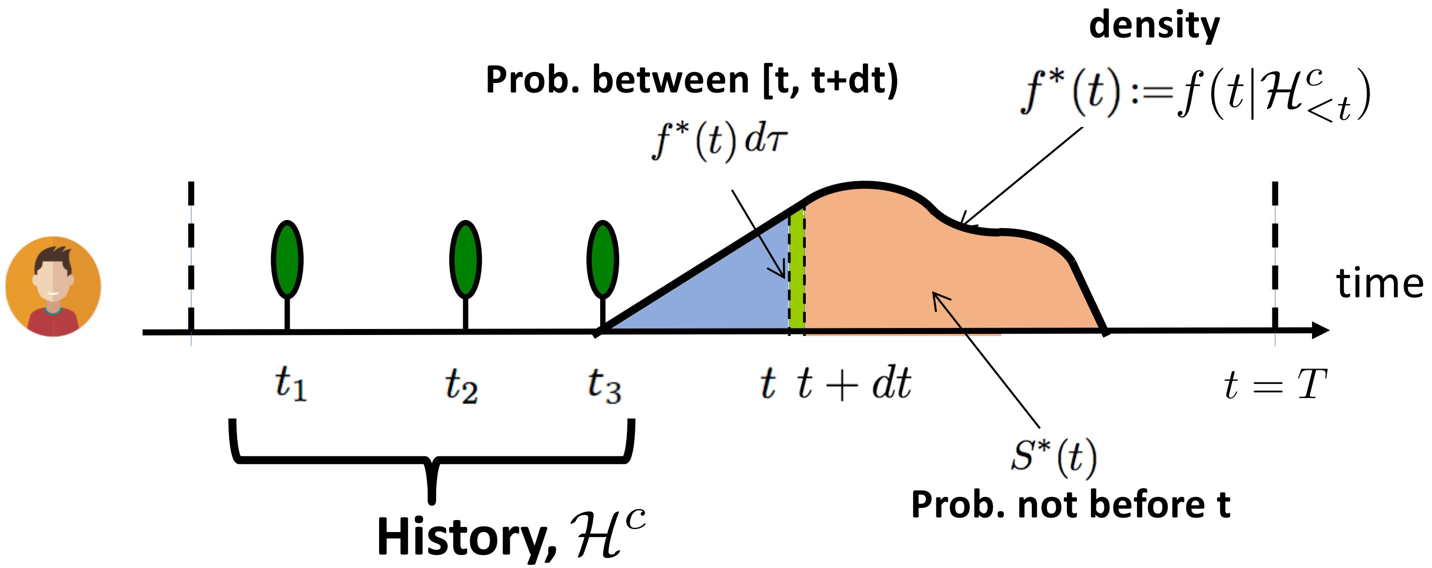


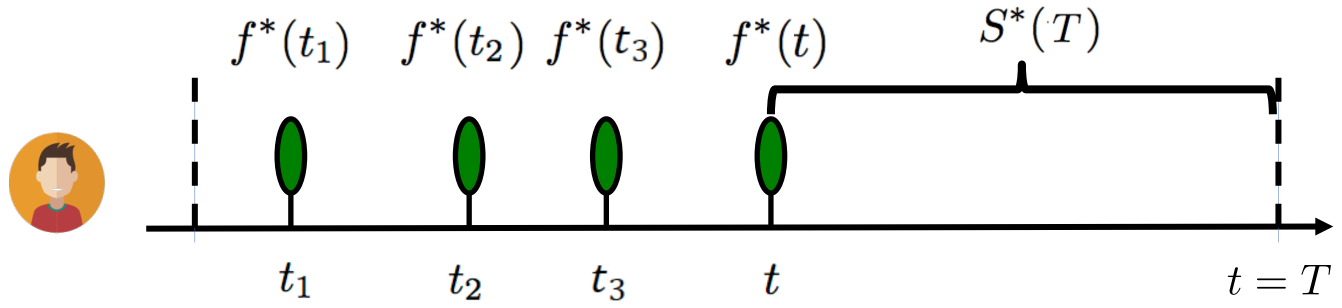
# Examples

- RMTPP, NeuralHawkes



# Model time as a random variable



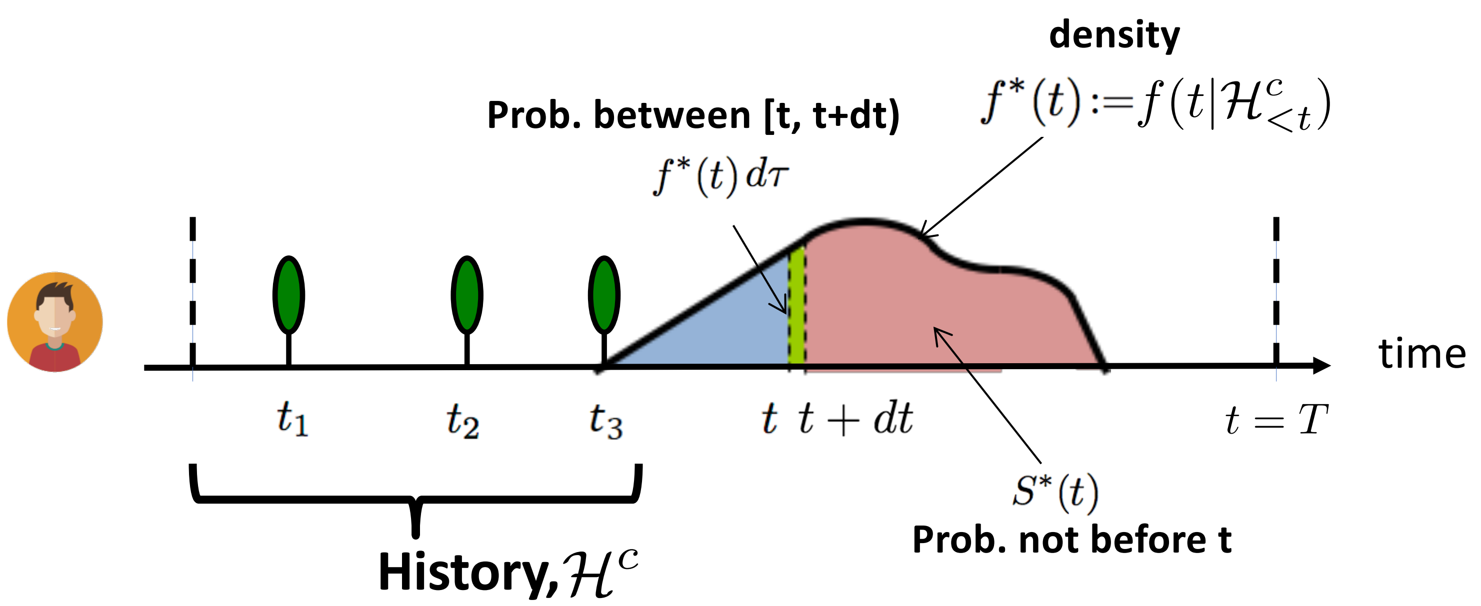




**Likelihood of a timeline:** $f^*(t_1)\ f^*(t_2)\ f^*(t_3)\ f^*(t)\ S^*(T)$



Source: Gomez-Rodriguez, Valera: Learning with Temporal Point Processes, ICML2018

# Intensity function




**Intensity:**

**Probability between [t, t+dt) but not before t**

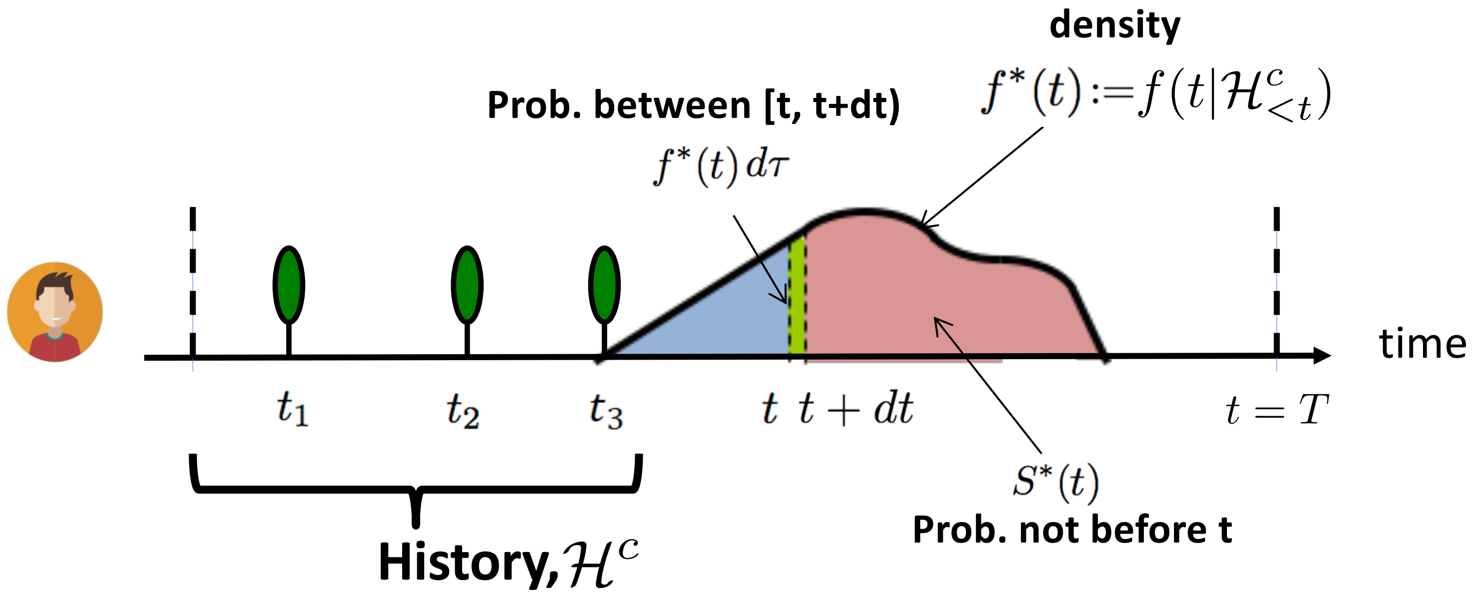
$$\lambda^*(t) = \lim_{\Delta t \to 0} \frac{P(t<T\leq \Delta t | T>t)}{\Delta t} = \frac{f^*(t)}{S^*(t)}$$

$\lambda^*(t)$ **It is a rate = # of events / unit of time**



Source: Gomez-Rodriguez, Valera: Learning with Temporal Point Processes, ICML2018

# Intensity function




**Intensity:**

**Probability between [t, t+dt) but not before t**

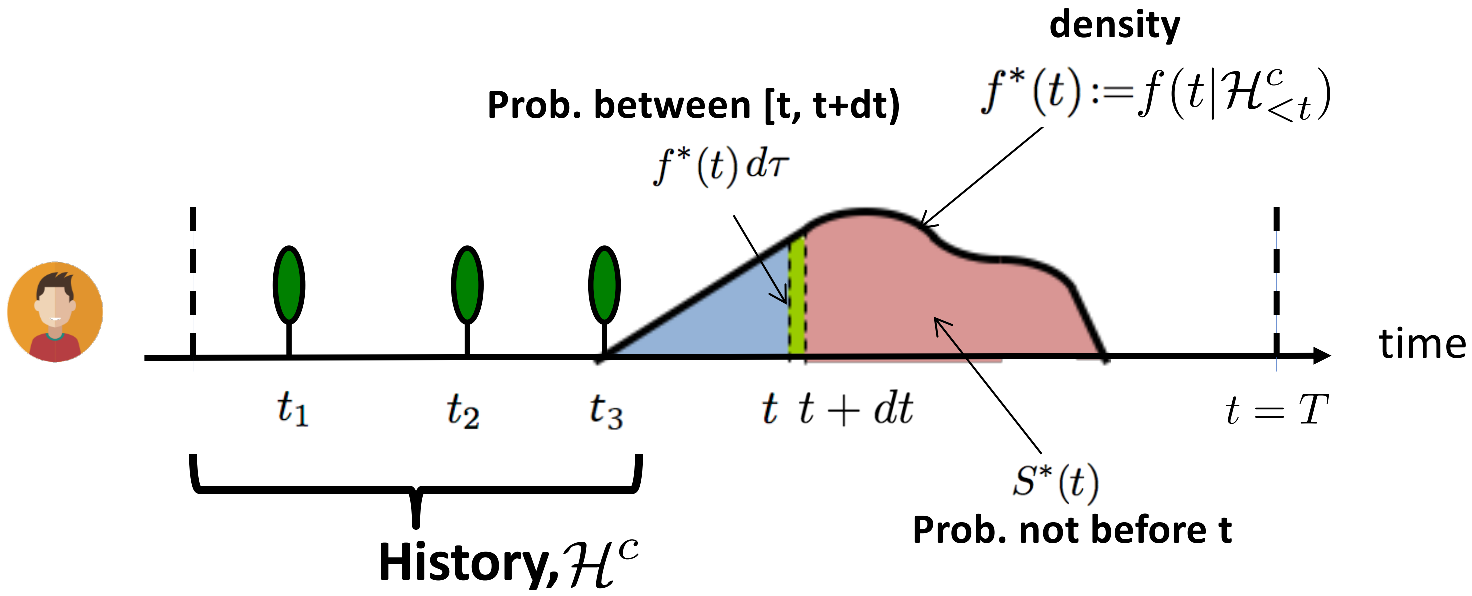
$$f^*(t) = \lambda^*(t)S^*(t)$$

$\lambda^*(t)$ **It is a rate = # of events / unit of time**



Source: Gomez-Rodriguez, Valera: Learning with Temporal Point Processes, ICML2018

# Intensity function




**Intensity:**

**Probability between [t, t+dt) but not before t**

$$S^*(t) = exp\left(-\int_0^t \lambda^*(t)du\right)$$

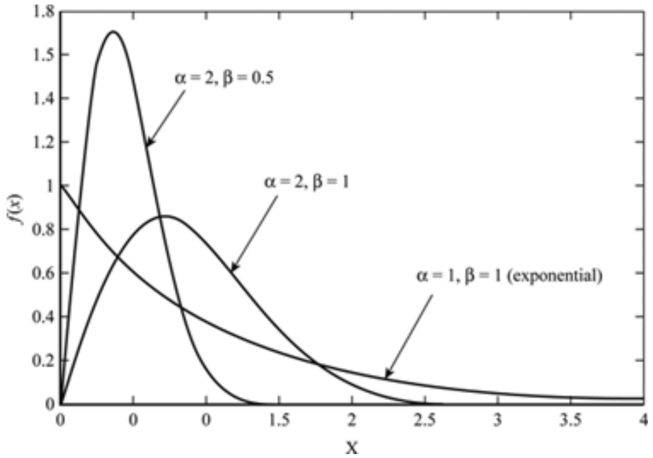
$\lambda^*(t)$ **It is a rate = # of events / unit of time**



Source: Gomez-Rodriguez, Valera: Learning with Temporal Point Processes, ICML2018

# Appropriate distributions for parametric estimation

| Name | $S^*(t)$ | $\lambda^*(t)$ | $f^*(t)$ |
|---|---|---|---|
| Weibull | $exp(-\alpha t^{\beta})$ | $\alpha\beta t^{\beta-1}$ | $\alpha\beta t^{\beta-1}exp(-\alpha t^{\beta})$ |
| Log-Logistic | $\frac{1}{1+(\alpha t)^{\beta}}$ | $\frac{\beta\alpha^{\beta}t^{\beta-1}}{1+(\alpha t)^{\beta}}$ | $\frac{\beta\alpha^{\beta}t^{\beta-1}}{(1+(\alpha t)^{\beta})^2}$ |
| Log-normal | $1-\Phi\left(\frac{\log t+\log\alpha}{\sigma}\right)$ | … | $\frac{1}{t\sqrt{2\pi}\sigma}exp\left(-\frac{1}{2\sigma^2}(\log t+\log\alpha)^2\right)$ |
| Exponential | $\exp(-\alpha t)$ | $\alpha$ | $\alpha exp(-\alpha t)$ |
| Hawkes | … | $\mu+\alpha\sum_{t_i\in\mathcal{H}_{<t}}\beta_i\exp(t-t_i)$ | … |



# Why it doesn't work

- ML is consistent: in principle, it can learn any distribution, provided that it is given infinite data and perfect mode space.
  - Minimizing the ML is equivalent to minimize the Kullback-Leibler (KL) divergence between the true distribution $\mathbb{P}_\mathrm{r}$ and the parametric distribution $\mathbb{P}_\theta$

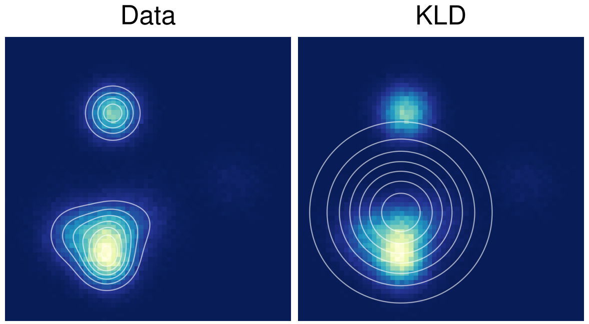
$$KL[\mathbb{P}_\mathrm{r}|\mathbb{P}_\theta] = \int P_r(x) \log \frac{P_r(x)}{P_\theta(x)} dx$$

- However, in real settings (due to model mis-specification and finite data), it tends to produce **overgeneralized models**.

# Why it doesn't work

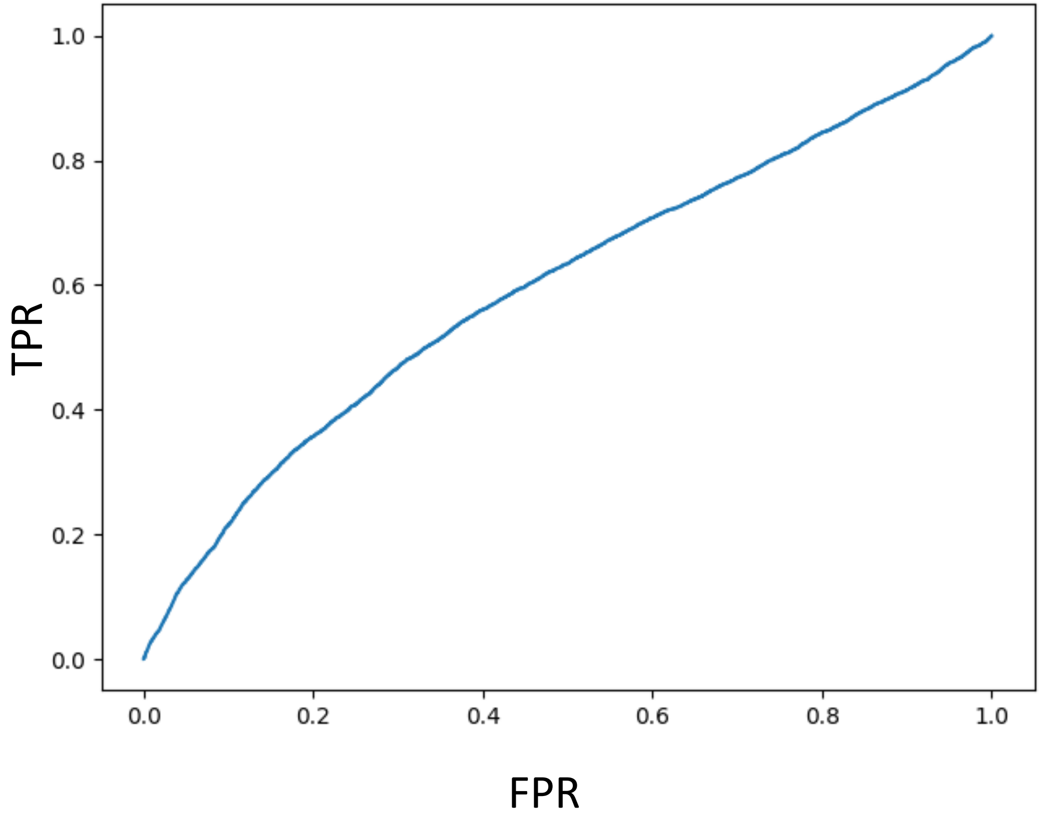
$$KL[\mathbb{P}_\mathrm{r}|\mathbb{P}_\theta] = \int P_r(x) \log \frac{P_r(x)}{P_\theta(x)} dx$$

- When $P_r(x) > P_\theta(x)$, large regions of $\mathbb{P}_\mathrm{r}$ get low values in $\mathbb{P}_\theta$. Their contribution in $KL[\mathbb{P}_\mathrm{r}|\mathbb{P}_\theta]$ tends to infinity.
- However, when $P_r(x) < P_\theta(x)$ then $x$ has a low (true) probability, but high probability of being generated by the model. The contribution to $KL[\mathbb{P}_\mathrm{r}|\mathbb{P}_\theta]$ tends to 0.
  - [Arjovsky 2017]



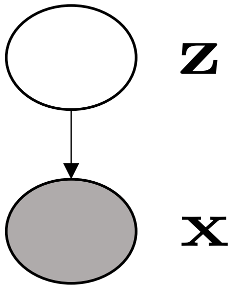

Source: [Theis at al 2016]

# Example: activations on Twitter, Weibull model




- Noisy temporal patterns, data collection bias

# Latent generative models




- Assume a stochastic generative process for observed data, ruled by latent/hidden variables $z$
  - Can capture high-level structure in event sequences and multiple sources of variability
  - E.g.: for business process instances, $z$ can capture variations due to context factors, alternative process configurations/variants, organizational changes

$$\mathbf{z} \sim P_\phi(\cdot)$$

$$\mathbf{x} \sim P_\theta(\cdot|\mathbf{z})$$

- Deep latent generative models:
  - learn $P(\boldsymbol{x}|\boldsymbol{z})$ via a neural network: **no assumption** on distribution's shape
  - (Sequential) Variational Autoencoders

$$P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{z})P(\mathbf{z})\mathrm{d}\mathbf{z}$$


N. Sachdeva, G. Manco, E. Ritacco, V. Pudi, Sequential Variational Autoencoders for Collaborative Filtering. WSDM 2019
G. Manco, F. Pisani, E. Ritacco, N. Sachdeva, Modeling Temporal dependencies in diffusion Networks via Variational Autoencoders, 2020. Submitted.

# Latent Generative Models

- Introduce a proposal distribution $Q_\phi$ parameterized by $\phi$
- Approximate the likelihood with

$$\log P(x) \geq \underbrace{\mathbb{E}_{z \sim Q_\phi}[\log P_\theta(x|z)] - \mathrm{KL}\big[Q_\phi(z)|P(z)\big]}_{\text{Evidence Lower Bound (ELBO)}}$$

- Optimize the ELBO with respect to $\phi$ and $\theta$

# Latent Generative Models

- Pros w.r.t. autoregressive models:
  - More robust to overfitting (regularization effect of latent modeling)
  - Useful latent representations (estimated via inference queries $P(z|x)$)

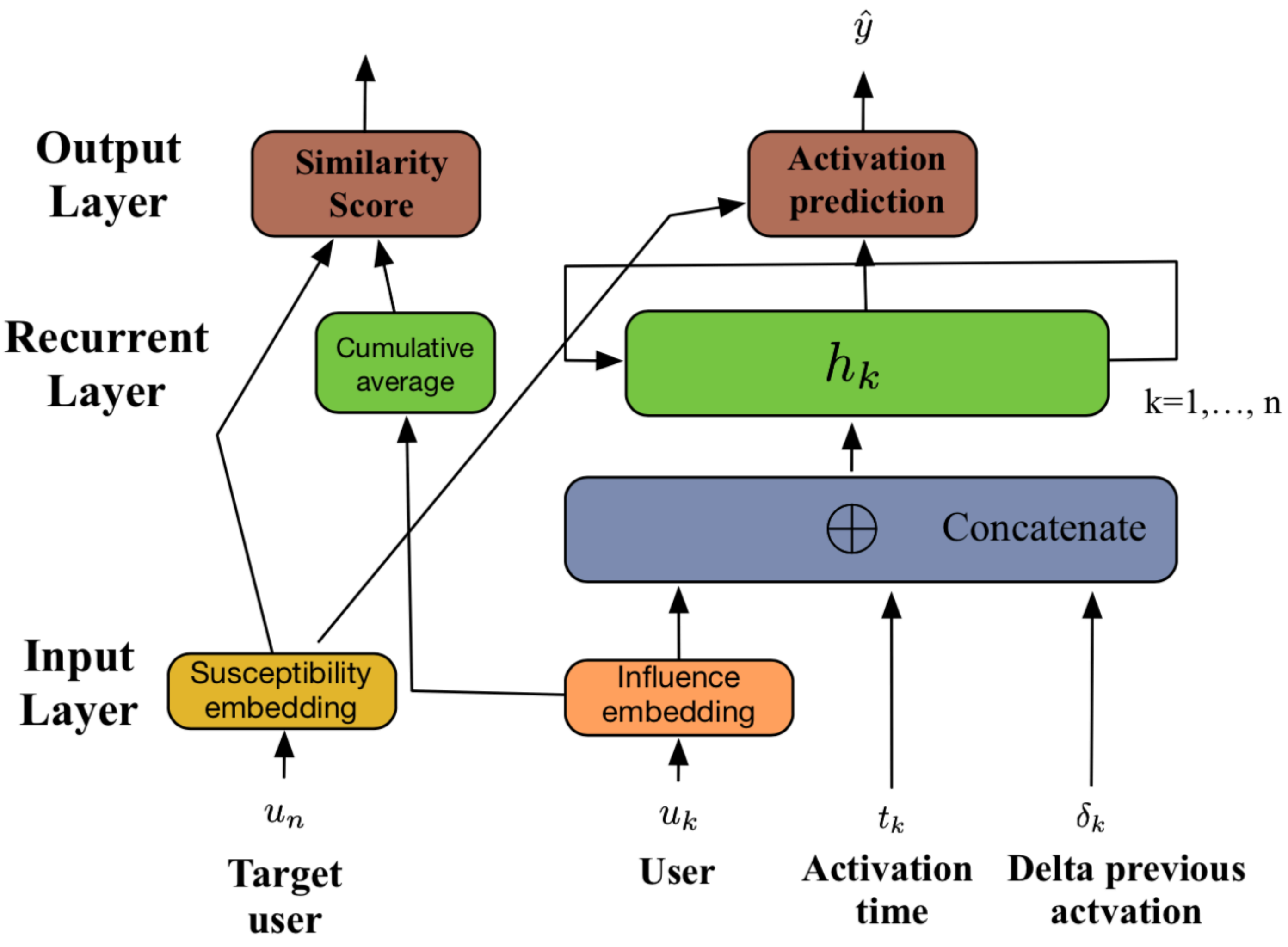
- Con: Imprecise generated samples, mainly due to two sources:
  - Still ML-oriented training
  - combined to approximate (ELBO) optimization, and assumption on z's prior/posterior

# What happens if we use Discriminative Learning instead?

- PATH: Predicting Activation Time Horizon
  - Focuses on predicting the activation of (groups of) users within a time horizon $T_h$
  - Focuses on specific users and predicts their behavior based on the effect of the partial cascading behavior of specific users
  - Captures cumulative history via the embedding
- PATH learns an embedding of the past event history via Recurrent Neural Networks that also cater for the diffusion memory
  - (the representation of) users that frequently become active in different streams within a small time interval are closer

G. Manco, G. Pirrò, E. Ritacco. 2018. Predicting Temporal Activation Patterns via Recurrent Neural Networks. ISMIS 2018.

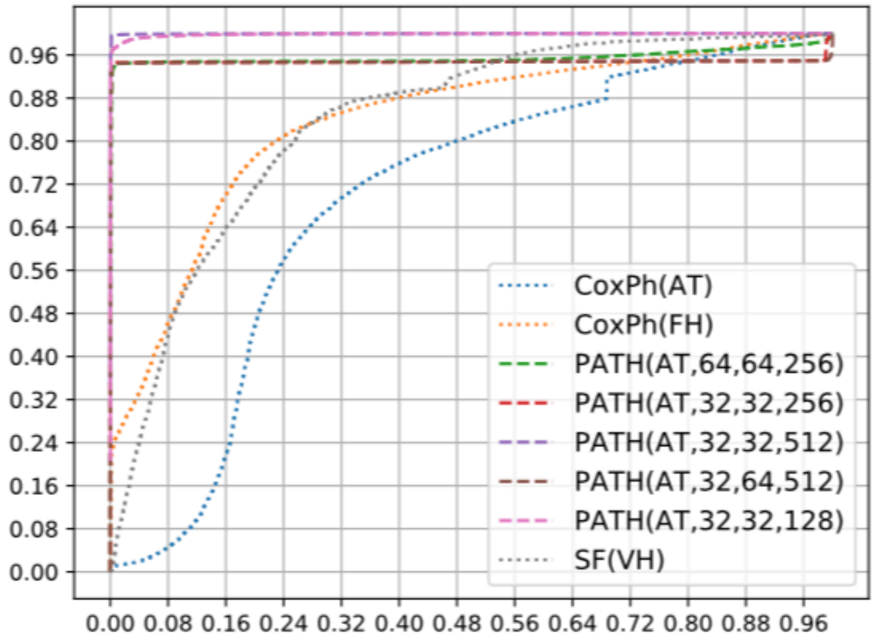
$$
\begin{aligned}
\mathbf{a}_k &= \mathbf{W}_e \mathbf{u}_k \\
\mathbf{s}_k &= \mathbf{V}_e \mathbf{u}_k \\
\mathbf{h}_k &= \mathrm{LSTM}\left([\mathbf{a}_k, t_k, \delta_k], \mathbf{h}_{k-1}\right) \\
\hat{y}_i &= \sigma\left(\mathbf{W}_o[\mathbf{s}_n, \mathbf{h}_n]\right) \\
\tilde{y}_i &= \exp\left\{ - \left\| \mathbf{s}_n - 1/n \sum_{k=1}^{n-1} \mathbf{a}_k \right\|^2 \right\}
\end{aligned}
$$




$$
\mathcal{L} = \sum_{\substack{\langle \mathcal{H}, y \rangle \in \mathcal{T}_{\mathcal{C}} \\ |\mathcal{H}|=n}} \left\{ y\left(\gamma \log(\hat{y}) + \beta \log \tilde{y}\right) + (1-y)\left(\gamma \log(1-\hat{y}) + \beta \log(1-\tilde{y})\right) \right\}
$$

# Why it works

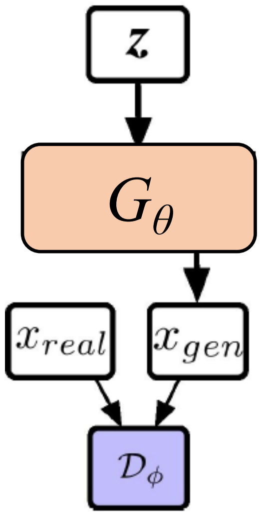
- Checking is easier than generating
  - Autoregressive models try to parameterize $\mathbb{P}_r$
  - By contrast, discriminative models only compute «closeness» to examples within $\mathbb{P}_r$




(b) Twitter

# Generative Adversarial Networks (GANs)

- **Model**: a deterministic function $G_\theta$ is learnt to transform random $z$ into data
  - no assumption on data/latent probability distributions
  - allows for sampling from $P_\theta(x)$ efficiently
  - can produce any distribution $P(x)$ with powerful enough $G_\theta$




- **Learning as a two-player game**
  - *Discriminator* $D_\phi$: Trained to optimally discriminate real data from generated samples
  - *Generator* $G_\theta$ : Trained to generate realistic samples fooling the discriminator

[Goodfellow et al. 2014]

# Is it better than ML?

- Discriminator Loss:

$$L_D(\phi,\theta) = \mathbb{E}_{x\sim\mathbb{P}_\mathrm{r}}\left[\log D_\phi(x)\right] + \mathbb{E}_{x\sim\mathbb{P}_\theta}\left[\log\left(1 - D_\phi\right)\right]$$

- Generator loss

$$L_G(\phi,\theta) = L_D(\phi,\theta)$$

- Adversarial Game

$$\max_{\phi}\min_{\theta}\mathbb{E}_{x\sim\mathbb{P}_\mathrm{r}}\left[\log D_\phi(x)\right] + \mathbb{E}_{x\sim\mathbb{P}_\theta}\left[\log\left(1 - D_\phi\right)\right]$$

# Is it better than ML?

- Optimal discriminator:

$$L_D(\phi, \theta) = \int p_r(x) \log D(x) + p_\theta(x) \log(1 - D(x))\, dx$$

- Maximizing the integrand with respect to $D(x)$ gives

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_\theta(x)}$$

# Is it better than ML?

$$L_D(D^*, \theta) = \int p_r(x) \log D(x) + p_\theta(x) \log(1 - D(x))\, dx$$

$$= \int p_r(x) \log \frac{P_r(x)}{P_r(x)+P_\theta(x)} + p_\theta(x) \log \frac{P_\theta(x)}{P_r(x)+P_\theta(x)}\, dx$$

$$= KL\left[\mathbb{P}_\mathrm{r} | \frac{\mathbb{P}_\theta + \mathbb{P}_\mathrm{r}}{2}\right] + KL\left[\mathbb{P}_\theta | \frac{\mathbb{P}_\theta + \mathbb{P}_\mathrm{r}}{2}\right] - 2\log 2$$

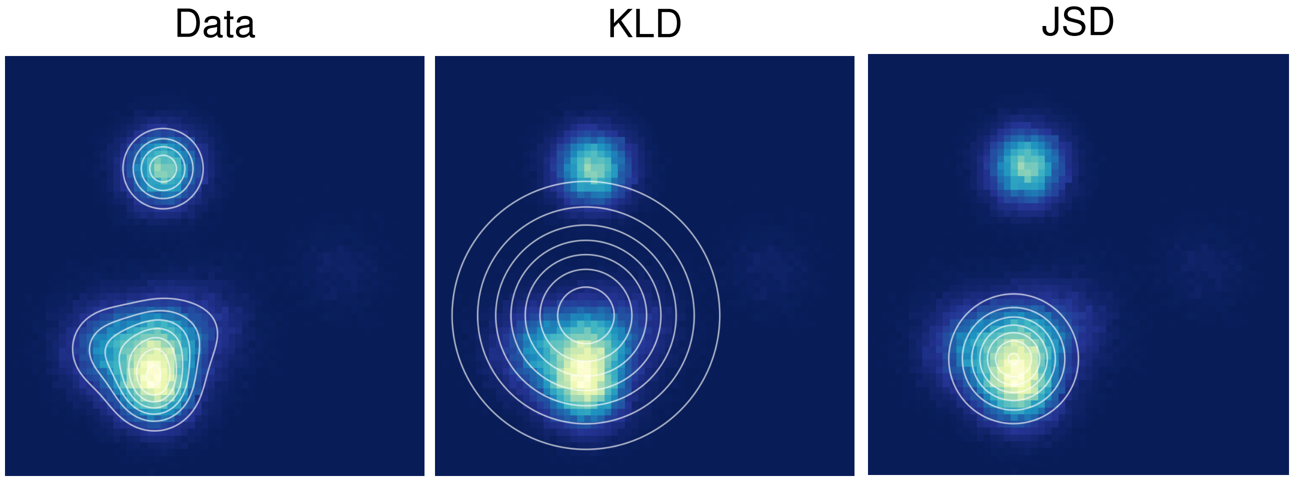
$$= 2JS[\mathbb{P}_\mathrm{r} | \mathbb{P}_\theta] - 2\log 2$$

# Is it better than ML?

$$L_G(D^*, \theta) \approx JS[\mathbb{P}_{\mathrm{r}} | \mathbb{P}_\theta]$$

- Minimizing with regards to $\theta$ is equivalent to minimize the Jensen-Shannon Divergence

$$JS[\mathbb{P}_{\mathrm{r}} | \mathbb{P}_\theta] = \frac{1}{2} KL[\mathbb{P}_{\mathrm{r}} | \mathbb{P}_\theta] + \frac{1}{2} KL[\mathbb{P}_\theta | \mathbb{P}_{\mathrm{r}}]$$




Source: [Theis at al 2016]

# GAN learning scheme

**Algorithm 1** Inference algorithm.

Initialize $\phi$ and $\theta$
**for** number of epochs **do**
    **for** $k$ steps **do**
        Sample $\left\{\tilde{x}_\theta^{(1)}, \ldots, \tilde{x}_\theta^{(m)}\right\}$ from $\mathbb{P}_\theta$;
        Sample $\left\{x^{(1)}, \ldots, x^{(1)}\right\}$ from $\mathbb{P}_r$;
        Update $\phi$ by ascending its stochastic gradient:

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(D_\phi\left(x^{(i)}\right)\right) + \log\left(1 - D_\phi\left(\tilde{x}_\theta^{(1)}\right)\right)\right]$$

    **end for**
    Sample $\left\{\tilde{x}_\theta^{(1)}, \ldots, \tilde{x}_\theta^{(m)}\right\}$ from $\mathbb{P}_\theta$;
    Update $\theta$ by descending its stochastic gradient:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D_\phi\left(\tilde{x}_\theta^{(i)}\right)\right)$$

**end for**
Return $\phi$ and $\theta$.

# GAN learning scheme

- Critical: Backpropagation from samples

**Algorithm 1** Inference algorithm.

Initialize $\phi$ and $\theta$
**for** number of epochs **do**
  **for** $k$ steps **do**
    Sample $\left\{\tilde{x}_\theta^{(1)}, \ldots, \tilde{x}_\theta^{(m)}\right\}$ from $\mathbb{P}_\theta$;
    Sample $\left\{x^{(1)}, \ldots, x^{(1)}\right\}$ from $\mathbb{P}_r$;
    Update $\phi$ by ascending its stochastic gradient:

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(D_\phi\left(x^{(i)}\right)\right) + \log\left(1 - D_\phi\left(\tilde{x}_\theta^{(1)}\right)\right)\right]$$

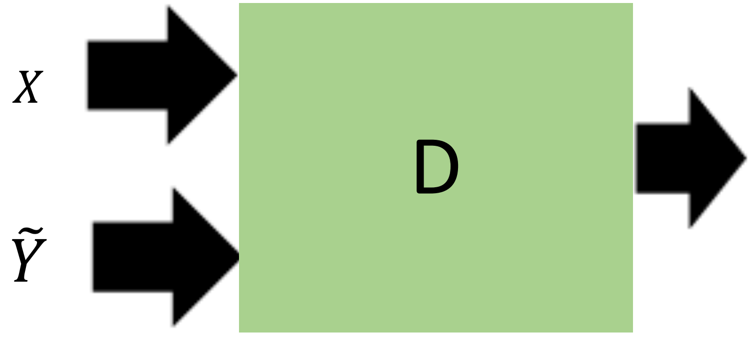
  **end for**
  Sample $\left\{\tilde{x}_\theta^{(1)}, \ldots, \tilde{x}_\theta^{(m)}\right\}$ from $\mathbb{P}_\theta$;
  Update $\theta$ by descending its stochastic gradient:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D_\phi\left(\tilde{x}_\theta^{(i)}\right)\right)$$

**end for**
Return $\phi$ and $\theta$.

# Context: Conditional GAN

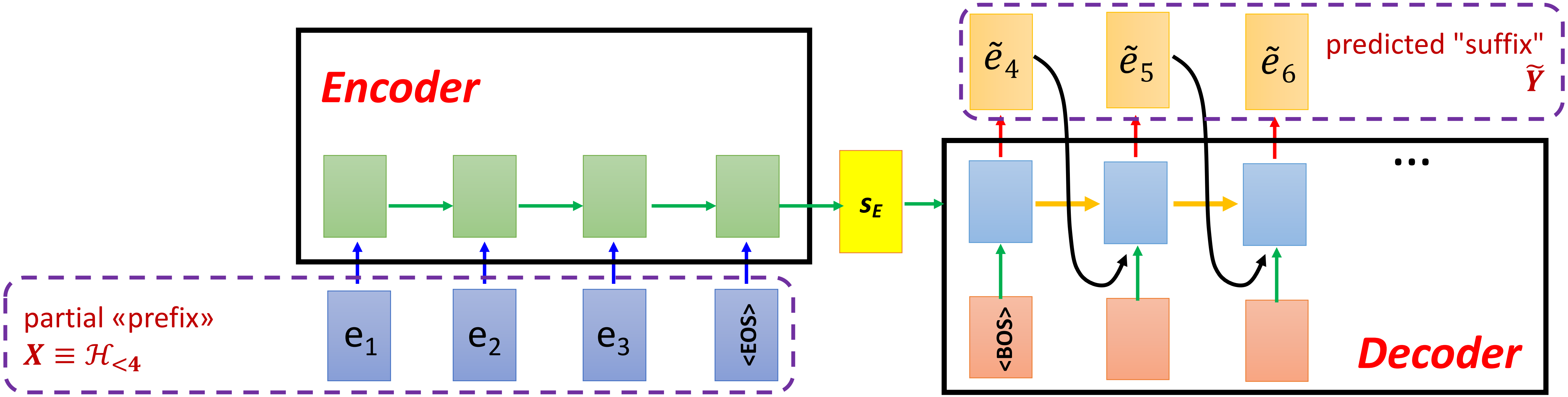
*Real future events (to be predicted)*
$Y = \boldsymbol{e}_h, \boldsymbol{e}_{h+1}, \ldots$


Scalar score, quantifying confidence in both

- $\tilde{Y}$ being realistic
- $\tilde{Y}$ matching context $X$

# Basic seq2seq conditional Generator




- Two components:
  - RNN Encoder: maps a partial trace $\mathcal{H}_{<h}$ into a condensed representation $s_E$
  - RNN Decoder: generates a suffix step-by-step, using $\boldsymbol{s_E}$ as its initial state (or as an additional input for all steps)
- Advantages w.r.t. a single RNN
  - Prefix as a whole has a semantics
  - Correlations within the prefix can influence the prediction of the suffix

# GAN learning scheme

- Critical: Backpropagation from samples
  - How do we sample events while preserving backpropagation?

**Algorithm 1** Inference algorithm.

Initialize $\phi$ and $\theta$

**for** number of epochs **do**

    **for** $k$ steps **do**

        Sample $\left\{\tilde{x}_\theta^{(1)}, \ldots, \tilde{x}_\theta^{(m)}\right\}$ from $\mathbb{P}_\theta$;

        Sample $\left\{x^{(1)}, \ldots, x^{(1)}\right\}$ from $\mathbb{P}_r$;

        Update $\phi$ by ascending its stochastic gradient:

$$\nabla_\phi \frac{1}{m} \sum_{i=1}^{m} \left[\log\left(D_\phi\left(x^{(i)}\right)\right) + \log\left(1 - D_\phi\left(\tilde{x}_\theta^{(1)}\right)\right)\right]$$

    **end for**

    Sample $\left\{\tilde{x}_\theta^{(1)}, \ldots, \tilde{x}_\theta^{(m)}\right\}$ from $\mathbb{P}_\theta$;

    Update $\theta$ by descending its stochastic gradient:

$$\nabla_\theta \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D_\phi\left(\tilde{x}_\theta^{(i)}\right)\right)$$

**end for**

Return $\phi$ and $\theta$.

# GAN for Sequence Data: Challenges (1)

- Sample t $\sim P_\theta(.|\mathcal{H})$
  - Need to make explicit the relationship with $\theta$
  - E.g., Weibull distribution can be reparameterized

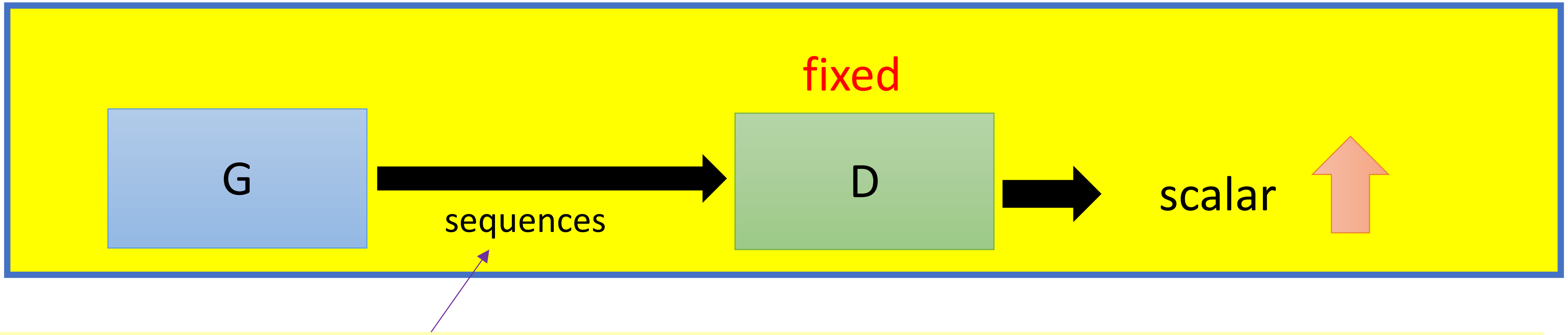
If $\mathrm{u} \sim U(0,1)$

Then $\mathrm{t} = \alpha(-\log u)^{1/\beta} \sim Weibull(\alpha, \beta)$

  - Explicit dependency of t both $\alpha$ and $\beta$

  - Issue with generalized (neural-based) intensity function $\lambda^*(t)$

# GAN for Sequence Data: Challenges (2)

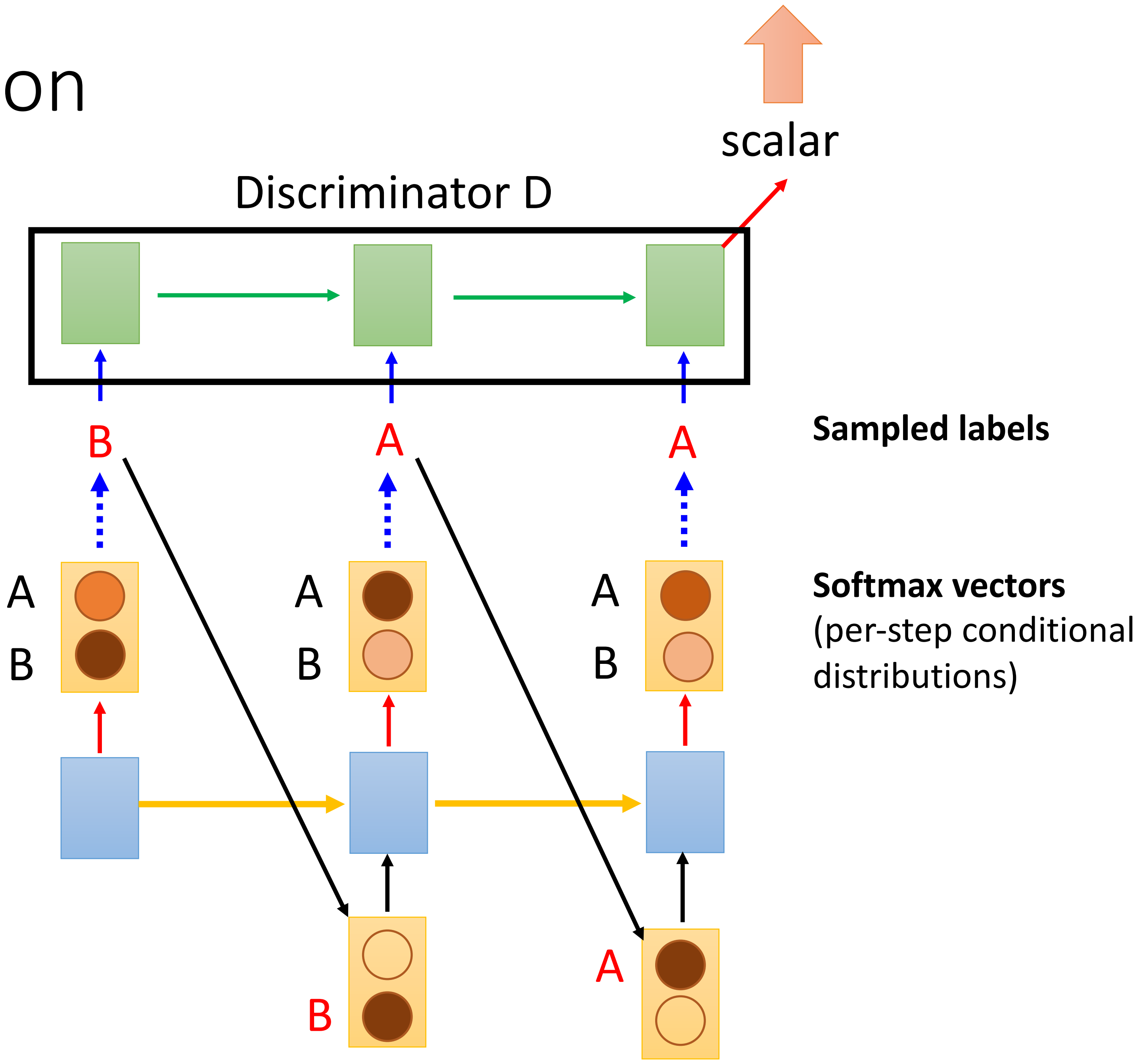



Sequence generation with an RNN decoder entails sampling from a discrete distribution: at every time step the most probable event type is chosen from a softmax output.
**loss gradients cannot backpropagate to the Generator!**

# Discrete Event Generation

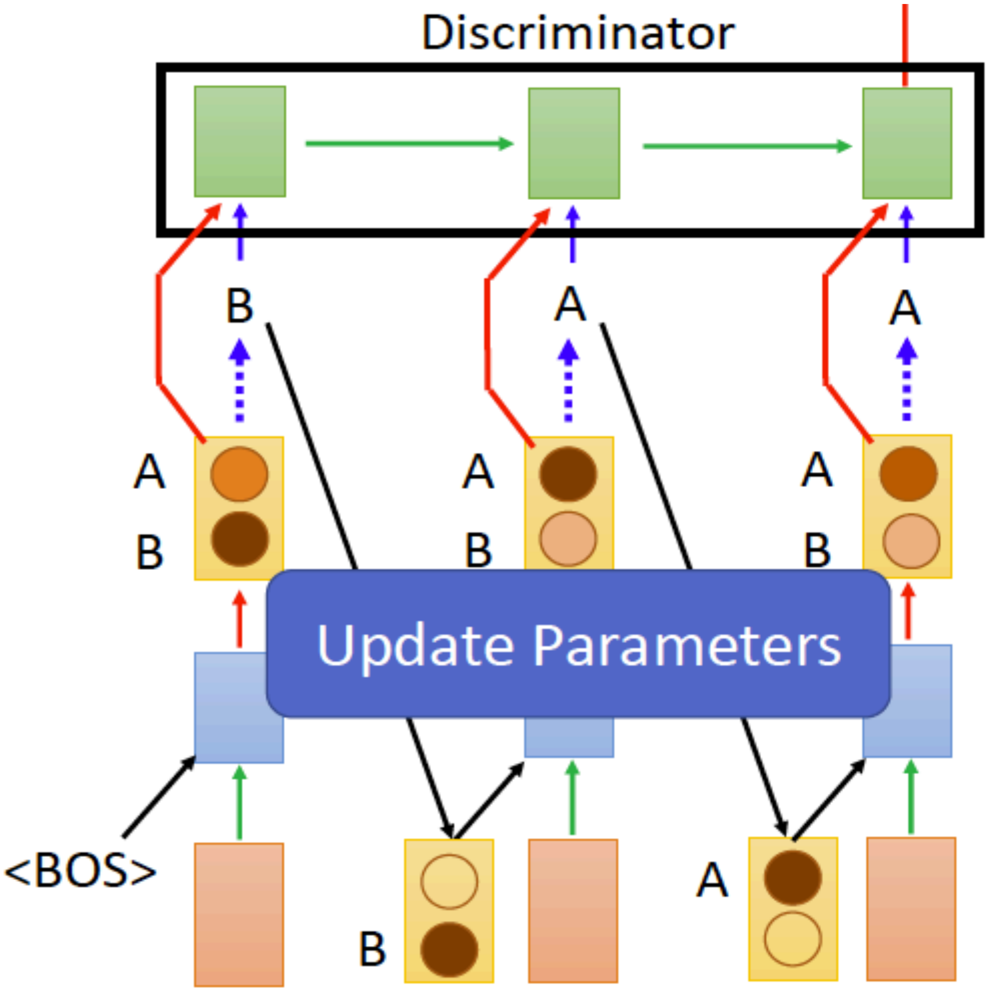
**Discrete** event attributes $\boldsymbol{a}_j$ predicted for future events

Assume possible event types are represented by labels **A, B, …**

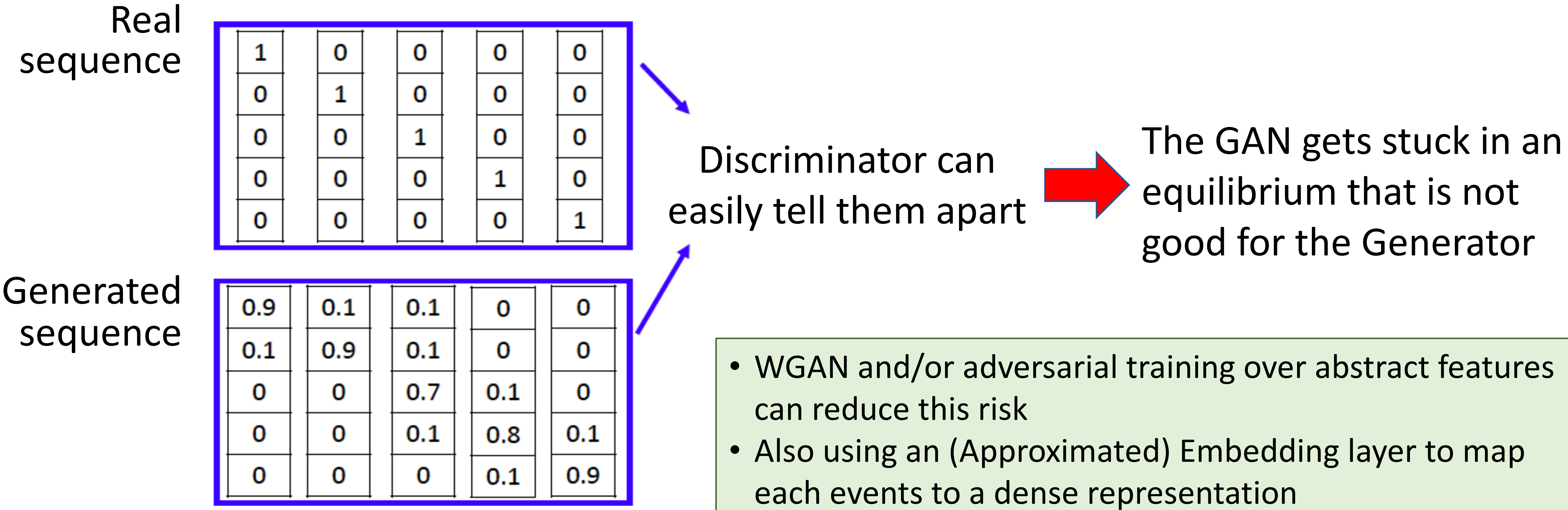


# Dealing with continuous inputs

- Use a softmax distribution as input to the Discriminator
- No direct sampling channel

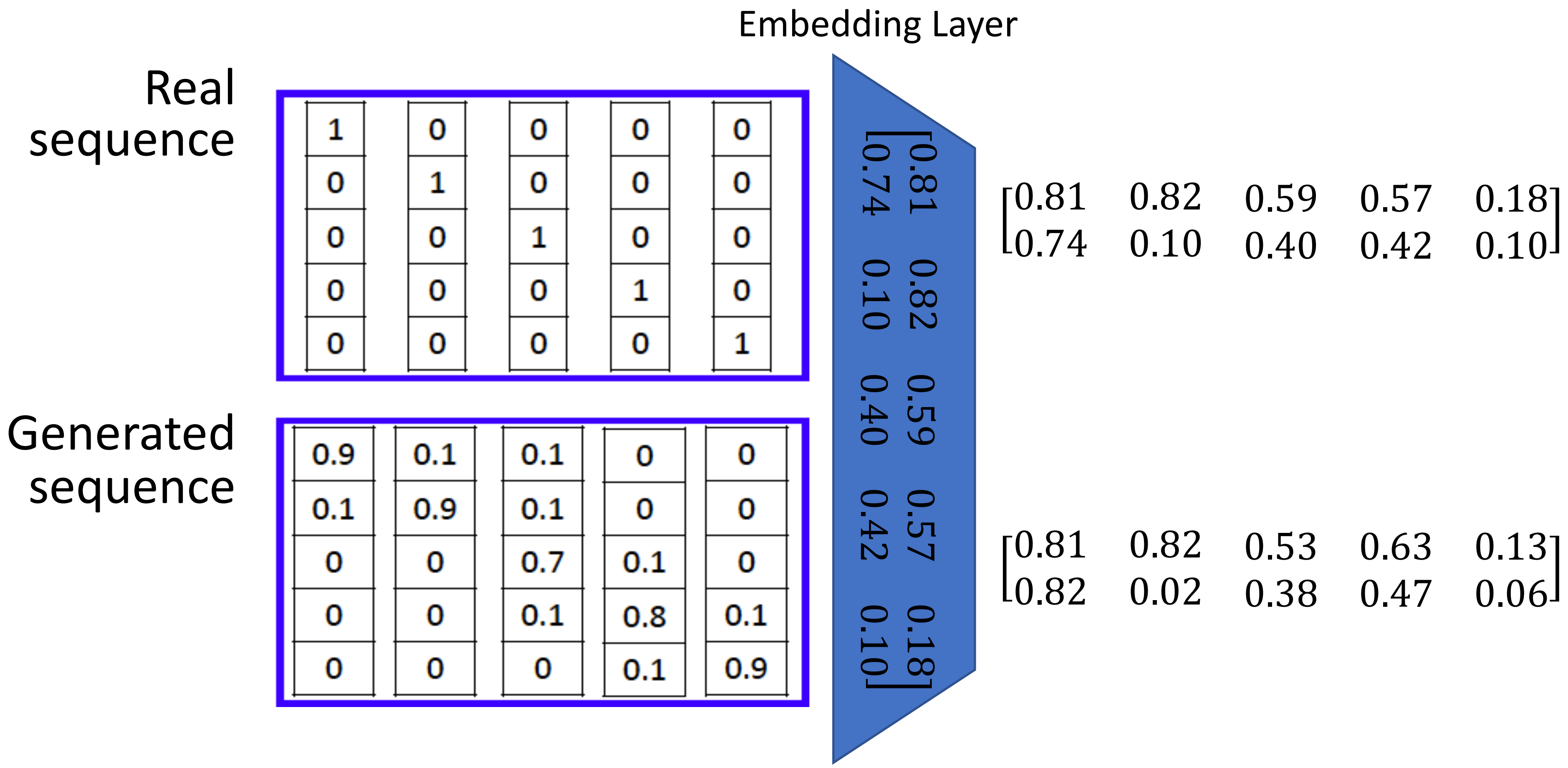


# … but sequences of softmax vectors are easily recognized



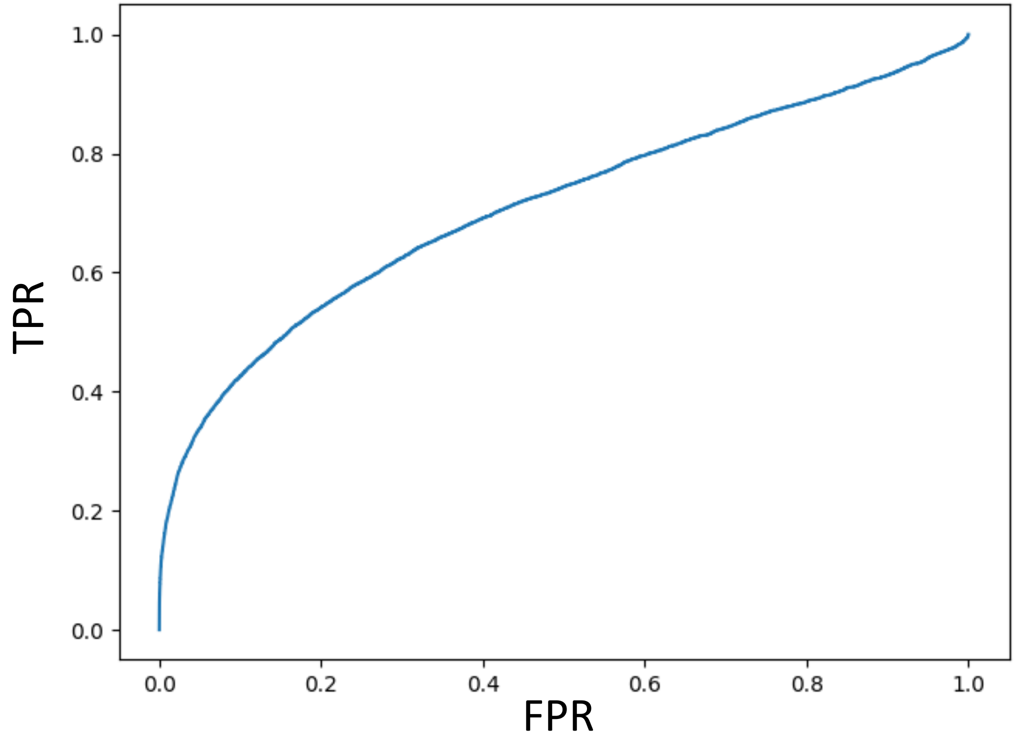

- WGAN and/or adversarial training over abstract features can reduce this risk
- Also using an (Approximated) Embedding layer to map each events to a dense representation
  - softmax components are used as weights for combining the embedding vector of the corresponding event labels [Xu et al., ENMLP’17]

# Approximated Embedding Layer (AEL)




Harsh time for the discriminator

# Activations on Twitter, Weibull model



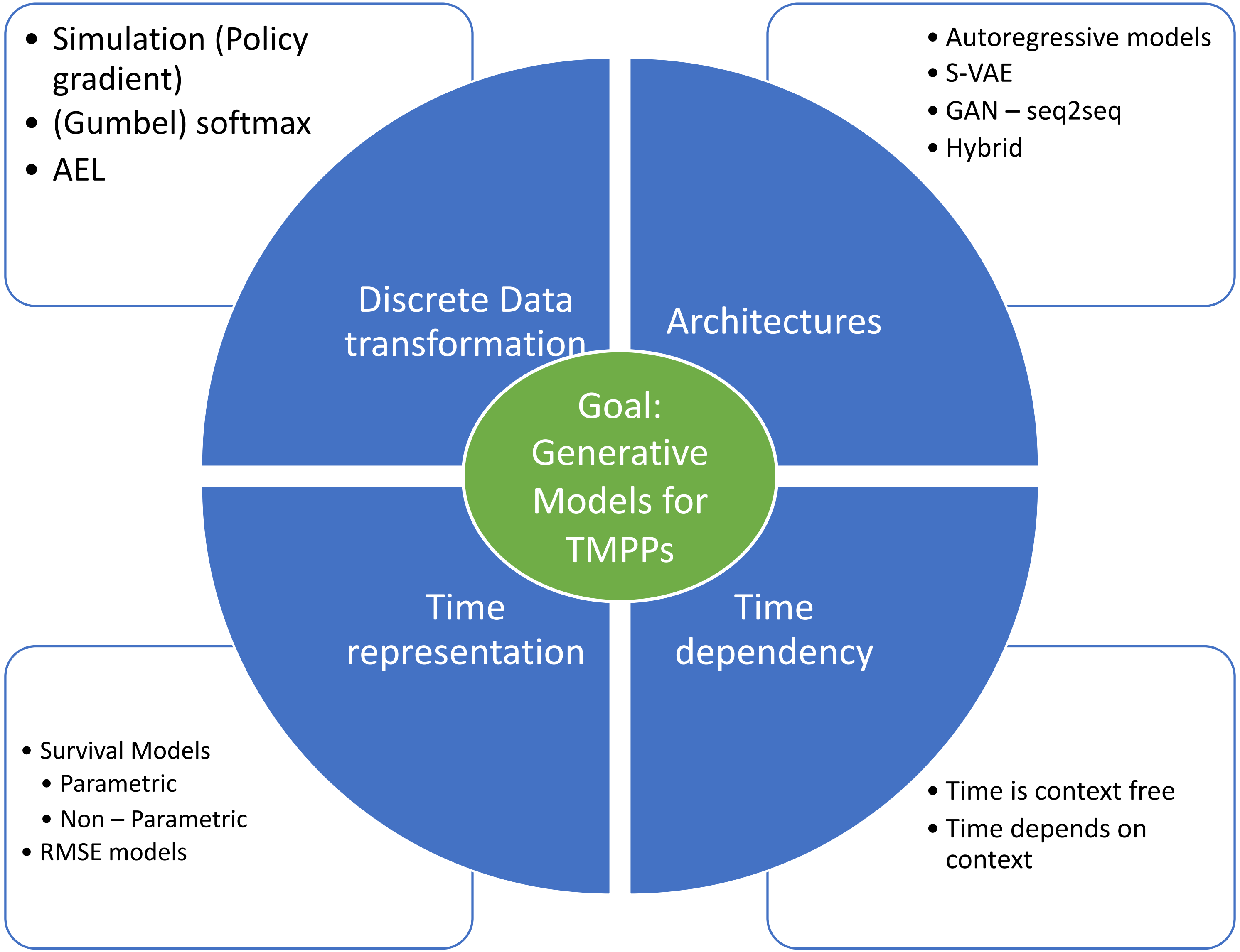

- Noisy temporal patterns, data collection bias

G. Manco, L. Pontieri, F. Folino, E. Ritacco, Generative Adversarial Networks for temporal Point Processes, 2020. Submitted.

# Conclusion and open issues

- Research goal: learn generative models for complex processes (regarded as TMPPs)
  - Two tasks: generate new traces, and predict future events for unfinished traces
- Limitations of ML-based approaches (autoregressive models, VAEs)
- GANs for MMPs: basic scheme
  - Generator **G**: RNN Generator, Seq2Seq Generator
  - Discriminator **D** as a ranker/critic
  - **G** passes to **D** differentiable approximations for the generated event sequences
- Open challenges (and other design choices)…

# Research line: Summary and Open Issues



# References


- H. Mei, J. Eisner. The Neural Hawkes Process: A Neurally Self-Modulating Multivariate Point Process. NeurIPS 2017.
- N.  Du, H.  Dai,R. Trivedi, U. UpadhyayM. Gomez-Rodriguez, L. Song. Recurrent Marked Temporal Point Processes: Embedding Event History to Vector. KD 2016
- Z. Xu, B. Liu, B. Wang, C. Sun, X. Wang, Z. Wang, C. Qi. Neural Response Generation via GAN with an Approximate Embedding Layer. EMNLP 2017
- L. Theis, A. van der Oord, M. Bethge. A note on the evaluation of generative models. ICLR 2016
- M. Arjovsky, L. Bottom. Towards Principled Methods for Training Generative Adversarial Networks. ICLR 2017
- F. Huszar. How (not to) Train your generative model: Scheduled sampling, Likelihood, adversary?. arXiv:1511.05101v1 [stat.ML] 16 Nov 2015
- I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio. Generative Adversarial Networks. NeurIPS 2014.

# Thank you

Questions?

giuseppe.manco@icar.cnr.it

@beman