Social Networks 79 (2024) 25–33

Contents lists available at ScienceDirect

Social Networks

journal homepage: www.elsevier.com/locate/socnet

Modeling non-linear effects with neural networks in Relational Event Models

Edoardo Filippi-Mazzola *, Ernst C. Wit

Università della Svizzera italiana, Faculty of Informatics/Institute of Computing, Switzerland

ARTICLE INFO

Keywords:
Relational event model
Large networks
Non-linear modeling
Social network analysis

ABSTRACT

Dynamic networks offer an insight of how relational systems evolve. However, modeling these networks efficiently remains a challenge, primarily due to computational constraints, especially as the number of observed events grows. This paper addresses this issue by introducing the Deep Relational Event Additive Model (DREAM) as a solution to the computational challenges presented by modeling non-linear effects in Relational Event Models (REMs). DREAM relies on Neural Additive Models to model non-linear effects, allowing each effect to be captured by an independent neural network. By strategically trading computational complexity for improved memory management and leveraging the computational capabilities of graphic processor units (GPUs), DREAM efficiently captures complex non-linear relationships within data. This approach demonstrates the capability of DREAM in modeling dynamic networks and scaling to larger networks. Comparisons with traditional REM approaches showcase DREAM superior computational efficiency. The model potential is further demonstrated by an examination of the patent citation network, which contains nearly 8 million nodes and 100 million events.

1. Introduction

Dynamic network modeling have emerged as an essential tool in social network studies, providing a nuanced perspective on the evolving nature of interactions and relationships. These networks capture the dynamism inherent in dynamic interacting structures by shedding light on how connections form, dissolve, or transform over time. Although the inclusion of the temporal dimension increases the complexity of the models, it provides richer insights, revealing patterns that static networks might miss.

Relational Event Models (REMs) (Butts, 2008; Perry and Wolfe, 2013; Bianchi et al., 2024) are an efficient and flexible framework for modeling dynamic networks, particularly in settings where events, or interactions between actors, occur sequentially over time. Unlike traditional network models, which focus on aggregated states or snapshots, REM focuses on micro-dynamics, tracking the chronological order of ties as they form or dissolve (Fritz et al., 2020). The great versatility of REMs is underscored by the diverse fields to which they have been applied. In finance, Lomi and Bianchi (2021) and Zappa and Vu (2021) highlight that resource exchanges vary significantly across different trading conditions, temporal contexts, and the values involved. Similarly, in healthcare, Vu et al. (2017) observed that patient transfers tend to form around tightly-knit hospital clusters that frequently reciprocate transfers. Amati et al. (2019) analyzed hospital collaborations in Southern Italy, revealing that interaction dynamics within these networks fluctuate considerably throughout the week. In ecology, Tranmer et al. (2015) investigated social behavioral dynamics, such as food sharing, among jackdaws, whereas (Patison et al., 2015) used REMs to analyze adaptive behavior among cows when introduced to new herd members. In another ecological application, Juozaitienė et al. (2023) utilized REMs to explore the dynamics of ecological niche invasions by modeling interactions between invasive species and their new territories. Despite its easy adaptability, REM's practical applicability is limited by its computational complexity (Welles et al., 2014).

Vu et al. (2015) first tackled the problem by proposing various sampling strategies on the risk-set connected to the partial-likelihood denominator. Lerner and Lomi (2020) demonstrated the robustness of REMs when the risk-set is sub-sampled via a nested-case-control approach, demonstrating that when REMs are used to model large dynamic networks, only one control per case is sufficient to obtain reliable estimates. This sub-sampling strategy was used by Filippi-Mazzola and Wit (2023) to approximate the REM partial likelihood by a logistic regression, which reduces computational complexity and allows for efficient modeling of non-linear effects.

Non-linear approaches to REMs where first tackled by Bauer et al. (2022) using B-splines (De Boor, 1972). By design, these spline-based models require to storing multiple high-dimensional model matrices. When these factors are combined with large networks with millions of dyadic interactions, many REM computing frameworks suffer from both convergence and memory management issues.

* Corresponding author.
E-mail addresses: edoardo.flippi-mazzola@usi.ch (E. Filippi-Mazzola), ernst.jan.camiel.wit@usi.ch (E.C. Wit).

https://doi.org/10.1016/j.socnet.2024.05.004
Received 24 January 2024; Received in revised form 24 April 2024; Accepted 26 May 2024
Available online 7 June 2024

Inspired by Neural Additive Models (Agarwal et al., 2021), we propose to model non-linear effects through a Deep Relational Event Additive Model (DREAM). DREAM strategically trades computational complexity with memory management by letting each effect be modeled by an independent neural network. By leveraging the higher computational power of graphic processor units (GPUs), DREAM is able to capture complex non-linear effects among variables. Each of these independent neural networks is trained at the same time using a Stochastic Gradient Descent (SGD) approach. SGD is especially renowned for its ability to handle large datasets and high-dimensional spaces as it iteratively refines model parameters to ensure optimal convergence. The simultaneous estimation of these neural networks not only increases computational efficiency, but also ensures that interdependencies and mutual information among different effects are captured effectively.

In this paper, we start by describing the methodological background on which REMs are built on in Section 2. After defining how DREAM is structured in Section 3, we provide a comprehensive simulation study to test its robustness and efficiency in Section 4. To conclude, we show an application on the analysis of the US patent citation network in Section 5.

2. Background and methods

REMs are a class of statistical models for sequences of social interaction events occurring over time. The primary focus of these models for is to model the pattern and structure of relationships that emerge as a series of observed social interactions or events.

2.1. Relational event model

In REMs, the primary statistical units are a series of recorded dyadic interaction defined as events. These are denoted as e_i $(i = 1, \dots, n)$ and are typically represented by the triplet $e_i = (s_i, r_i, t_i)$, which denotes that an action was initiated by a sender s_i, targeted towards a receiver r_i, and occurred at a specific time t_i.

Following Perry and Wolfe (2013), it is possible to define a multivariate counting process $N_{sr}(t)$ that records the number of directed interactions between s and r, up to time t,

$$N_{sr}(t) = \sum_{i \geqslant 1} \mathbf{1}_{\{t_i \leqslant t;\ s_i = s;\ r_i = r\}}.$$

$N_{sr}(t)$ is then a local submartingale, where, through Doob–Meyer decomposition $N_{sr}(t) = \Lambda_{sr}(t) + M_{sr}(t)$, we can define its predictable increasing process $\Lambda_{sr}(t)$. REMs describe this predictable increasing process by assuming that the counting process is an inhomogeneous Poisson process, i.e.,

$$\Lambda_{sr}(t) = \int_0^t \lambda_{sr}(\tau) d\tau,$$

where λ_{sr} is the hazard function of the relational event (s, r). Considering the history of prior events $\mathscr{H}_t$ up to time t, a common method for modeling this intensity function relies on the log-linear model (Cox, 1972). Consequently, the intensity function is expressed as the product of a baseline hazard $\lambda_0(t)$ and an exponential function of q $\mathscr{H}_t$-measurable covariates x,

$$\lambda_{sr}(t \mid \mathscr{H}_t) = \lambda_0(t) e^{f(x_{sr})}, \tag{1}$$

where $f(x_{sr}) = \sum_{k=1}^{q} f_k(x_{srk})$ is some additive model. In the original formulation of the REM, $f_k(x_{srk})$ is modeled as a linear function weighted by a coefficient β_k, such that $f_k(x_{srk}) = x_{srk}\beta_k$.

Given the network prior history, the model definition assumes conditional independence of events. Incorporating covariates into this structural design allows for an in-depth investigation into a multitude of individual influences or factors that contribute to the occurrence of the event. These influential components are typically classified as exogenous or endogenous. Exogenous factors generally pertain to attributes or effects related to the sender or receiver, offering insights into external dynamics. These could include individual characteristics, roles within the network, or external circumstances that influence their actions. On the other hand, endogenous factors relate to the intrinsic micro-mechanisms within the network itself. These are patterns or tendencies that arise from the inherent structure and dynamics of the network, becoming visible as the series of events progressively unfold over time. By recognizing and studying these factors, we can gain an understanding of how the network internal mechanisms shape the unfolding of events.

A further characteristic that lends significant appeal to the proportional hazard model is its absence of distributional assumptions concerning activity rates. This flexibility is a notable advantage over fully parametric models and enables to treat the baseline hazard λ_0 as a nuisance parameter (Cox, 1975). This strategic consideration helps simplify the computational complexities that emerge when trying to estimate the weights in the full-likelihood derived from (1). The resulting partial likelihood is expresses as follows,

$$L_P(\beta) = \prod_{i=1}^{n} \left(\frac{\exp\left\{ f(x_{s_i r_i}) \right\}}{\sum_{(s_i^*, r_i^*) \in \mathscr{R}(t_i)} \exp\left\{ f(x^*_{s_i^* r_i}) \right\}} \right), \tag{2}$$

where $\mathscr{R}(t)$ is the risk-set, i.e., the set of all possible relational events that could have happened at time t.

2.2. Nested case control sampling

While the application of partial likelihood in (2) introduces significant simplifications to REM estimation, its practical application is constrained by the dimensionality of its denominator. The risk set $\mathscr{R}(t)$ tends to increase quadratically with the number of nodes in traditional longitudinal networks, although it may vary depending on the specific context. For instance, in citation networks, the risk set tends to expand linearly as new nodes cite existing documents within the network (Vu et al., 2011; Filippi-Mazzola and Wit, 2023). However, irrespective of the individual scenarios under analysis, scalability remains a limiting factor for these models. Large networks, comprising millions of nodes, will inevitably pose computational challenges and potentially limit the model efficiency in such contexts.

A solution to this issue has been proposed by Vu et al. (2015), who introduced the idea of nested case-control sub-sampling the risk set (Borgan et al., 1995). The central idea revolves around analyzing all the observed events, or "cases", while only scrutinizing a smaller subset of non-events, termed "controls". (Borgan et al., 1995) demonstrated that using a nested case-control sampled risk set yields a consistent estimator. Building on this concept, Boschi et al. (2023) and Filippi-Mazzola and Wit (2023) extended the empirical findings of Lerner and Lomi (2020) to show that in scenarios with a large number of nodes, one control per case is sufficient to obtain reliable parameter estimates.

When only a single control per case is considered, the partial likelihood in (2) results in the likelihood of a logistic regression model where only successful outcomes are observed as responses. This key insight further enhances the practicality of REMs, reducing the computational complexity of estimating large and complex risk sets. With this transformation in place, the sub-sampled case-control version of the partial likelihood in (2) is given as,

$$\widetilde{L_P}(\beta) = \prod_{i=1}^{n} \left[\frac{\exp\left\{ f(x_{s_i r_i}) - f(x_{s_i^* r_i^*}) \right\}}{1 + \exp\left\{ f(x_{s_i r_i}) - f(x_{s_i^* r_i^*}) \right\}} \right], \tag{3}$$

where $x_{s_i^* r_i^*}$ is a randomly sampled non-event for ith event sender s_i^* and receiver r_i^* from $\mathscr{R}(t_i)$.

3. Deep relational event additive model

Although standard REM formulations assume that the rate of interaction between s and r exhibits a linear dependence on the covariates, the relationship between predictors and event rates could be non-linear and exhibit a greater degree of complexity. If this is the case, deploying linear effects could inadvertently result in model oversimplification and the production of biased estimates. This highlights the necessity of exploring alternative modeling techniques beyond the traditional linear framework. In this section, we propose the Deep Relational Event Additive Model (DREAM) to estimate non-linear effects in large networks, that leverages machine-learning methods and graphical processors units for efficient computation.

3.1. Non-linear modeling with neural networks

Modeling non-linear effects in REMs was first tackled by Bauer et al. (2022) and Filippi-Mazzola and Wit (2023). Both heavily rely on the use of B-splines (De Boor, 1972), a computational technique that represents curves as a series of interconnected piecewise polynomial functions. While splines are a standard tool in non-linear, additive modeling, their implementation comes with challenges in big data settings, especially with respect to memory: the fitting procedure necessitates the creation of a potentially huge model matrix.

DREAM strategically trades off memory usage with computational complexity. Following the recent developments of Neural Additive Models (Agarwal et al., 2021), DREAM leverages multi-layered neural networks to model non-linear effects, where each effect is modeled by an independent neural network. Let then f_k be a feed-forward Artificial Neural Network (ANN) (Ripley, 1996) with a single input and a single output, separated by L layers, for $l = 1, \dots, L$. Each of these layers contains $m_1, \dots, m_L$ neurons. The output of each f_k is the result of a series of sequential operations, such as

$$
\begin{aligned}
a_k^{(1)} &= \phi\left(\beta_k^{(1)} x_{srk} + \beta_{0k}^{(1)}\right) \\
a_k^{(2)} &= \phi\left(\beta_k^{(2)} a_k^{(1)} + \beta_{0k}^{(2)}\right) \\
&\vdots \\
a_k^{(L-1)} &= \phi\left(\beta_k^{(L-1)} a_k^{(L-2)} + \beta_{0k}^{(L-1)}\right) \\
a_k^{(L)} &= \beta_k^{(L)} a_k^{(L-1)} + \beta_{0k}^{(L)},
\end{aligned}
$$

where $a_k^{(l)}$ represents the output of the lth layer for the kth covariate, $\beta_k^{(l)}$ is the weight matrix of size $m_l \times m_{l-1}$, and $\beta_{0k}^{(l)}$ is the intercept or *bias* of size $m_l \times 1$. f_k is then an ANN with $M = \sum_{l=1}^{L}(m_l \times m_{l-1} + m_l)$ number of parameters. ϕ is a non-linear function, commonly referred as activation function. ANN models use a non-linear function to be able to model complex relationships in data, in a way that a linear function cannot. Noel et al. (2023) recently proposed the Growing Cosine Unit (GCU) activation function, as a way to deal with some of the drawbacks of standard activation functions. In our empirical evaluation, this function seems to perform well. A more detailed discussion on the activation function can be found in Appendix A. f_k can then be expressed as

$$
f_k(x_{srk}) = \beta_k^{(L)}\left(\dots \phi\left(\beta_k^{(2)}\left(\phi\left(\beta_k^{(1)} x_{srk} + \beta_{0k}^{(1)}\right)\right) + \beta_{0k}^{(2)}\right)\dots\right) + \beta_{0k}^{(L)}. \tag{4}
$$

Let then $f(x_{sr})$ represent the collective sum of q independent ANN output effects for $x_{srk}(t)$, with $k = 1, \dots, q$, i.e., $f(x_{sr}) = \sum_{k=1}^{q} f_k(x_{srk})$. Each of these ANNs is then trained simultaneously to maximize (3). Fig. 1 describes the structure of how the information is passing through $f(x_{sr})$, offering a clear understanding of the DREAM framework. The most significant asset of this modeling technique lies in its interpretability. Through a visual examination of the individual functions f_k, one can develop a comprehensive understanding of the dynamic behavior of each effect x_{srk}, mimicking the interpretability of splines. Although this technique increases the computational complexity for evaluating the likelihood in (3) as the passage from one layer of the network to another requires multiple matrix multiplications, it eliminates the heavy memory usage associated with basis transformations. While modern frameworks allows efficient matrix operations, the efficiency of this approach is mainly guaranteed by recent technological advancements in the application of vectorized computations on GPUs.

Like most common machine-learning techniques, DREAM scalability in the estimation process is attained thanks to the adoption of a Stochastic Gradient Descent (SGD) approach. SGD is particularly effective for large datasets and complex models, as it updates model parameters iteratively based on a subsets (batches) of data, rather than the entire dataset.

Among the available SGD methodologies, we use the ADAM optimizer (Kingma and Ba, 2017). ADAM is easily scalable and has reliable convergence (Reddi et al., 2018). Its computational merits are due to the way this approach updates the weights of the model, by calculating individual adaptive learning rates based on estimates of the first and second moments of the gradients. Details on ADAM can be found in Appendix B.

While DREAM's flexibility allows for different regularization techniques to be used, in our modeling approach we used dropout (Srivastava et al., 2014) to prevent overfitting during the estimation process. By randomly omitting subsets of features or neurons during the training phase, dropout helps improve the robustness and generalizability of the neural network. Together with the number of layers, and the number of neurons per layer, dropout constitutes one of the main hyperparameters of the model to infer.

3.2. Uncertainty estimation with Gaussian process regression

Due to its over-parametrization, estimation of ANN hyper-parameters $\{\beta_{0k}^{(l)}, \beta_k^{(l)}\}_{k,l}$ via SGD does not imply achieving convergence to a global optimum. As discussed in Goodfellow et al. (2016), neural network optimization often focuses on finding satisfactory parameters that perform well, rather than continuing the estimation process until a theoretical optimum is reached. This approach recognizes the complex, high-dimensional landscapes in which these models operate, where multiple parameter sets can yield comparably effective outcomes. Consequently, the specific parameters of a neural network should not be interpreted in isolation since their values can vary across different runs of the model without necessarily affecting performance. Instead, our attention focuses on the behavior of the estimated functions $\hat{f}_k$, which provide meaningful insights into the data relationships modeled by the ANN. Consequently, our focus on uncertainty assessment pertains to the estimated functions, rather than the hyper-parameters.

A common solution for non-parametric models is to employ non-parametric bootstrap (Efron, 1979). Such procedure estimates point-wise uncertainty intervals by generating a sufficient number of sampled copies from original dataset. For each repetition, the model is re-trained on a new "bootstrapped" version of the original dataset. Consequently, each re-train yields distinct estimates of the non-linear effects. Once a sufficient number of repetitions are completed, confidence bands can be directly computed from the estimated curves using the desired percentiles.

While bootstrapping offers a flexible way to evaluate estimation uncertainty, it is computationally demanding, particularly when considering the training requirements of neural networks. To address this, we propose to post-process a limited number of bootstrap refits via Gaussian Processes Regression (GPR) (Rasmussen and Williams, 2006) to obtain robust confidence bands.

We assume that the estimated function can be represented by a Gaussian process $\hat{f}_k \sim \mathcal{GP}(f_k, K_k)$ where f_k represents the true mean function of the Gaussian Process, while K_k is a Radial Basis Function (RBF) kernel with the form

$$
K_k = \exp\left(-\frac{\| x - x' \|}{2l^2}\right), \tag{5}
$$

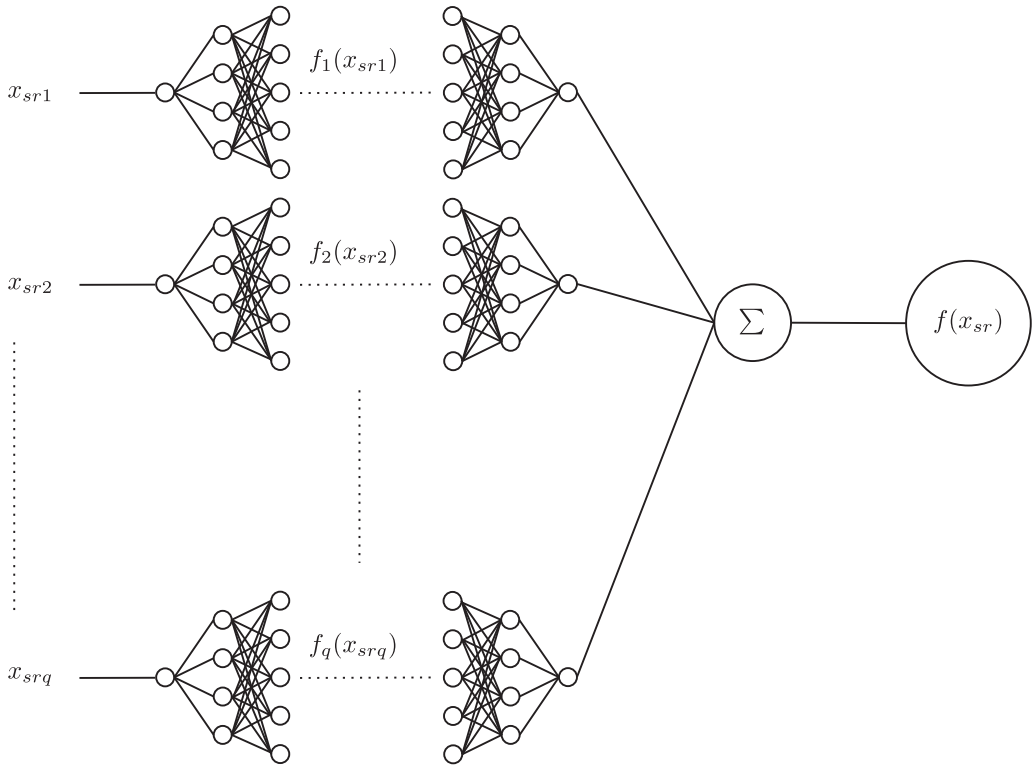


Fig. 1. DREAM framework. Each effect is modeled via an independent ANN structure that captures its non-linearity. It is possible to extend this modeling technique with interaction effects simply by letting f_k have a multivariate input, such as $f_k(x_{srk_1}, x_{srk_2})$, and a univariate output.

where l is a scale parameter, while x' is a subsequent value of x. Computing the posterior mean and covariance matrix, we obtain an estimated mean function whose uncertainty is represented by the standard deviations computed by square rooting the diagonal of the posterior covariance matrix. The $O(n^3)$ computational complexity of GPR models presents a limitation for large-scale applications. This higher computational cost is due to the inversion of the kernel matrix in the posterior. However, to obtain pointwise estimates of the curves, it is sufficient to evaluate the kernel matrix on a reduced sample of equidistant points on the support range of x. Knowing the upper and the lower bound of each covariate, we can generate a vector $\tilde{x}$ of equidistant points. We can then define $\tilde{x} = (\tilde{x}_1, \tilde{x}_2 \dots, \tilde{x}_N)$, where $\tilde{x}_i = x_{\min} + (i-1)\Delta$ for $i = 1, \dots, N$, with $\Delta = \frac{x_{\max} - x_{\min}}{N-1}$ and $N \ll n$. With $\tilde{x}$, we can compute the posterior estimates of the Gaussian Process,

$$\hat{\mu}(\tilde{x}) = K_k^\top [K_k + I]^{-1} \hat{f}(\tilde{x}), \tag{6}$$

$$\hat{\Sigma}(\tilde{x}) = K_k - K_k^\top [K_k + I]^{-1} K_k, \tag{7}$$

where I is the identity matrix that improves numerical stability during inversion. While in many scenarios it is possible to scale this identity matrix by multiplying I to a constant, we have noticed in our applications that the results remain roughly invariant to such scaling. Overall, this alternative approach is designed to offer a more computationally efficient alternative to a full bootstrap approach.

4. Simulation study

In this section, we present a series of simulation studies that highlight DREAM's ability in accurately identifying non-linear effects in dynamic networks. We focus on three specific aspects. The initial simulation study illustrates DREAM's ability in reconstructing the true generating functions behind observed effects. Secondly, given their close similarity, it is natural to compare additive Neural Network models with spline-based additive models such as Generalized Additive Models (GAMs) (Hastie and Tibshirani, 1986). In our comparison, we use the `gam` function from the `mgcv` package in R. Finally, we present a study on the time-complexity of our method. The full code has been written in python within the pytorch suite (Paszke et al., 2019) and it is publicly available in a GitHub repository (https://github.com/efm95/DREAM) together with all the simulations and application.

4.1. True function recovery

We simulated relational event data under the assumption that each node possesses both a sender and a receiver covariate simulated as uniform variables $\mathcal{U}(0, 1)$. The effect of each covariate is given in red in Fig. 2. A network with 5000 nodes and 500,000 edges is sampled. The fitted curve was estimated via five bootstrap refits, followed by the application of a GPR model using the scikit-learn Python library (Pedregosa et al., 2011). The ANN architecture was determined via CV (details can be found in the supplementary materials in S1, while formulas of true generating functions can be found in the supplementary materials S2).

The results show how the estimated curves follow the true generating functions behavior, while the estimated confidence intervals obtained through this approach encompass the true functions over the variables support.

4.2. Accuracy comparison with GAM

As previously noted, non-linear effects in a REM can be estimated by using a logistic regression additive model approach. A computationally efficient choice is the `gam` function within the `mgcv` package in R, where the smooth terms are estimated via penalized b-splines. The great advantage of `mgcv` is that the degrees of freedom of the splines are automatically selected, thus reducing the number of hyperparameters that are required to be set.

Comparing models using traditional information criteria such as AIC or BIC may not provide a fair assessment in the context of ANNs.

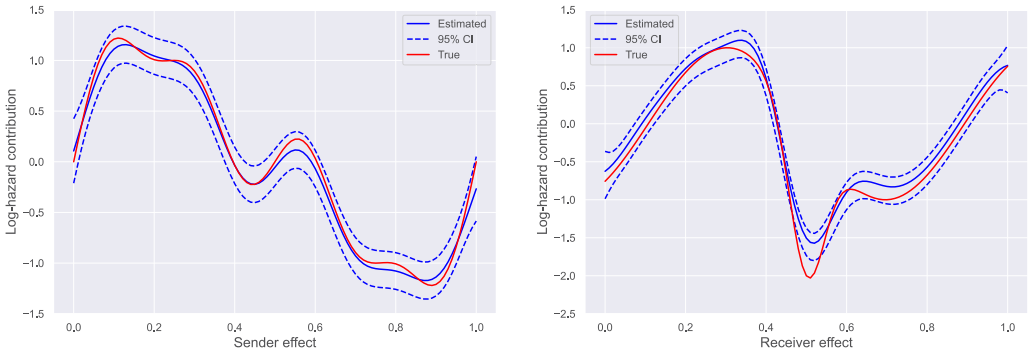


Fig. 2. True and estimated effects along with their confidence intervals. The red lines denote the actual effects, whereas the blue lines are the estimated effects, and the dashed-blue lines represent the confidence intervals. Confidence intervals are calculated by adding and subtracting twice the local standard error from the estimated functions. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 1
Log-partial-likelihood values for each estimated compared with the one computed from the sampled population using the true generating model. Subsequently, the KL-divergence values are assessed in relation to this same sampled population.

	GAM	DREAM	Population
$\log L_P$	−232'991.59	−232'102.28	−231'634.90
KL(Pop.‖Model)	1794.71	1647.98	–

This is because ANNs incorporate a substantially larger number of parameters compared to GAMs or other classical statistical models. To address this challenge, we adopt an alternative metric for model comparison. Specifically, in Table 1, we report the maximized log-partial-likelihood values obtained by DREAM and a GAM, alongside the log-partial-likelihood computed from the sampled population with the true generating functions. By using the Kullback–Leibler (KL) divergence assessed with the sampled population, we provide a comparison that considers model fit and the ability to accurately reconstruct the true generating functions.

From the results in Table 1, it is possible to notice DREAM attains a log-partial-likelihood score that more closely attains the one of the sampled population. As a consequence, this is also reflected in its KL-divergence scores. For this scenario, GAM with smooth terms is slightly outperformed. However, the proximity of the performances between DREAM and GAM suggests that both models offer a similar level of accuracy in approximating the true model.

4.3. Time efficiency comparison with GAM

Estimation in the `mgcv` package use highly optimized Newtonian solvers, written in C routines to achieve rapid convergence. However, the computational efficiency of these solvers is frequently undermined by R less-than-optimal memory management system (Kotthaus et al., 2015). This results in computational bottlenecks, prolonging the time required for the algorithm to converge. In some instances, the inefficiency in memory management can even lead to computational overflow, further complicating the estimation process.

In contrast, DREAM relies on the PyTorch suite (Paszke et al., 2019), a deep learning framework that excels in handling vectorized operations. PyTorch is specifically designed to leverage the computational capabilities of Graphics Processing Units (GPUs). Using Google Colab free GPUs (Nvidia Tesla T4 with 15 GB of memory), we run two sets of simulations to compare `mgcv` convergence times with DREAM with the implementation of the GPR approach to estimate the curves.

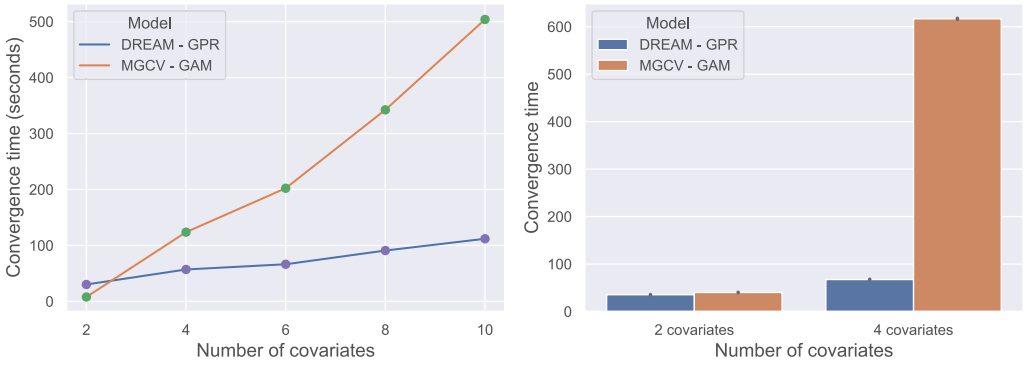
It is important to consider that the convergence time of DREAM should not be evaluated in isolation, as it depends on various hyperparameters such as the learning rate and number of epochs. To address this, we always fitted DREAM with the default learning rate of 0.001, and we employed an early stopping technique to stop the training process. The convergence timings presented in this section include not only the duration required to achieve a convergence but also the multiple iterations needed to fit the Gaussian process for uncertainty estimations. Fig. 3(a) compares the convergence times of `mgcv` and DREAM using generated REM data that comprised 1000 nodes and 100,000 events. We gradually augmented the complexity by sequentially covariates, thus increasing the number of non-linear effects that each model needed to estimate. We carried out the fitting procedure ten times. While `mgcv` convergence time initially appeared faster with only two covariates, its performance rapidly degraded as the complexity grows, revealing the computational bottleneck within R. Conversely, DREAM exhibited only a modest uptick in convergence time as the complexity increased. Fig. 3(b) presents for a larger dataset comprising 5000 nodes and 500,000 events. While the C routines lend `mgcv` stability in its convergence times, it becomes noticeably strained with the inclusion of 4 covariates, taking considerably longer. It is to note that with this size of data, we were not able to fit a model with more covariates as the algorithm failed to converge.

5. US patent citation network

To demonstrate DREAM practical applicability on large networks, we model non-linear effects in the US patent citation network that contains nearly 100 million citations and almost 8 million patents from 1976 to 2022. We chose this specific application not only because of its size and complexity, but also because the data preprocessing procedures and the computation of the statistics are well-defined (Filippi-Mazzola and Wit, 2023), making the study more accessible and simple to replicate. The preprocessing of the patent citation network can be found at https://github.com/efm95/STREAM. A detailed model selection for the application to the US patent citation network is extensively covered in the supplementary material, under section S3.

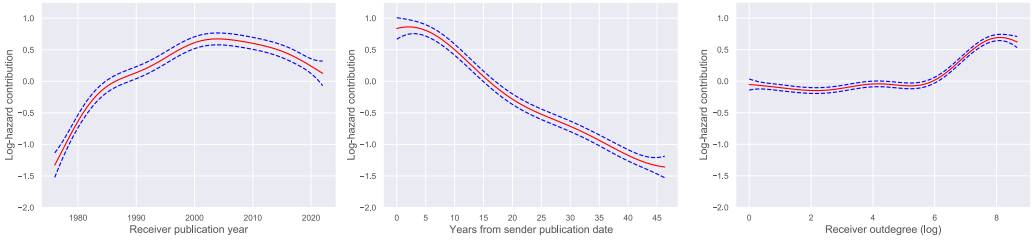
In order for a patent to be formally issued, the applicant must disclose all relevant prior art. As a result, the US patent citation network consists of patents that cite earlier works in relation to their issuance date. This results in a dynamic network that is constantly growing and expanding. Within this network, nodes are represented by patents, and as they are published, they establish connections to pre-existing nodes in the network via citations. The aim of this modeling exercise is to identify what drives a patent s to cite a patent r at time t.

Filippi-Mazzola and Wit (2023) proposed to model the network via three different set of statistics: patent effects, patent similarity effects and endogenous temporal effects. The first set of effects are portrayed in Fig. 4 and consists of the receiver publication year, the time-difference between the sender issue date and the receiver publication date, and the receiver outdegree. Fig. 4(a) shows a maximum around the year 2000. Potentially, this indicates that increased technological innovation happened during that time. The time-difference effect identifies a period of approximately 5 years following the patent publication date



(a) 100'000 events with 1'000 actors. (b) 500'000 events with 5'000 actors.

Fig. 3. Comparison of convergence times between MGCV and DREAM across two simulated REM data scenarios.



(a) (b) (c)

Fig. 4. Nodal effects consisting of a receiver publication year effect, a time-difference effect and a receiver outdegree effect.

when citations are most likely. Finally, the receiver outdegree confirms that patents with a higher number of citations at the time of publication tend to play a more central role in the network, consequently increasing their chances of accumulating more citations over time.

The second set of effects, shown in Fig. 5, delves into the patent similarity characteristics that contribute to a citation. The first statistic is textual similarity. We embed patent abstracts in an euclidean space using a pre-trained SBERT model (Reimers and Gurevych, 2019), and calculate pairwise cosine similarities. The resulting non-linear effects conclusively demonstrate that patents are more likely to cite each other when their abstracts share significant textual similarities. Specifically, the hazard of a citation occurring is 60 times higher for patients who share a textual similarity larger than 0.5 when compared to those who share a similarity of 0.2. Secondly, we consider the technological relationship between the two patents, as indicated by their shared International Patent Classification (IPC) classes. We capture the proportion of shared classes to the total classes observed across both patents by computing the Jaccard similarity among these IPC classes. According to the results Fig. 5(b), the rate of one patent citing another increases as shared the number of technology classes increases. Indeed, when comparing citations with a Jaccard index of 0.4 to those with 0, the hazard rate increases by about 7 times.

Fig. 6 captures time-varying factors that influence the rate of a patent being cited. The first of these is the cumulative citations a patent has received, illustrating that as a patent accumulates more citations, its probability of receiving additional ones increases, until it reaches a plateau around $e^{5.4} \approx 221$. The second effect evaluates the time elapsed since a patent most recent citation. This indicates that the longer the duration since the last citation, the less probable it becomes for the patent to be cited again.

For an alternative interpretability of the model's outputs, the supplementary materials section S4 include figures that present the effects of our the fitted effects expressed in terms of Hazard contributions rather than Log-hazard contributions.

Following the classification system proposed by Oliver et al. (2017), the contributions of effects to the hazard can be categorized into *large*, *medium*, or *small* based on their hazard ratio sizes. Observations from our analysis reveal significant variations in hazard contributions across the support of these effects. Most notably, the majority of these hazard ratio sizes fall within the *medium* or *large* categories. This indicates not only the substantial impact of these effects on the model's outcomes but also underscores their relevance in describing the overall dynamics of the patent citation network.

6. Conclusions

Relational Event Models offer a versatile framework for modeling dynamic networks. Yet, their real-time application often faces challenges due to the computational complexity in their fitting procedures. Such challenges tend to amplify as the volume of observed events increases. In this study, we present a solution to these computational issues by introducing the Deep Relational Event Additive Model (DREAM). In DREAM, the non-linear behavior of each covariate is captured by an independent Artificial Neural Network, providing both precision and efficiency in capturing network dynamics.

We proposed two distinct methods in DREAM for estimating areas of uncertainty. The first method entails non-parametrically bootstrapping the observed dataset and then refitting the model multiple times. The second method, employs Gaussian Process Regression based on a small subset of non-parametric bootstrap refits, offering a more efficient way to handle uncertainty while maintaining robustness in our estimations.

Throughout a series of simulation studies, we introduced and tested the capabilities of DREAM, emphasizing its ability in capturing non-linear effects within large dynamic networks. The robustness and efficiency of DREAM became clear when compared to existing methods such as GAMs from the MGCV package in R. DREAM strength lies not only in its ability to accurately model nonlinear effects, but also in its fast convergence, which is accomplished by leveraging the computational advantages of Pytorch and GPUs. We further demonstrated the

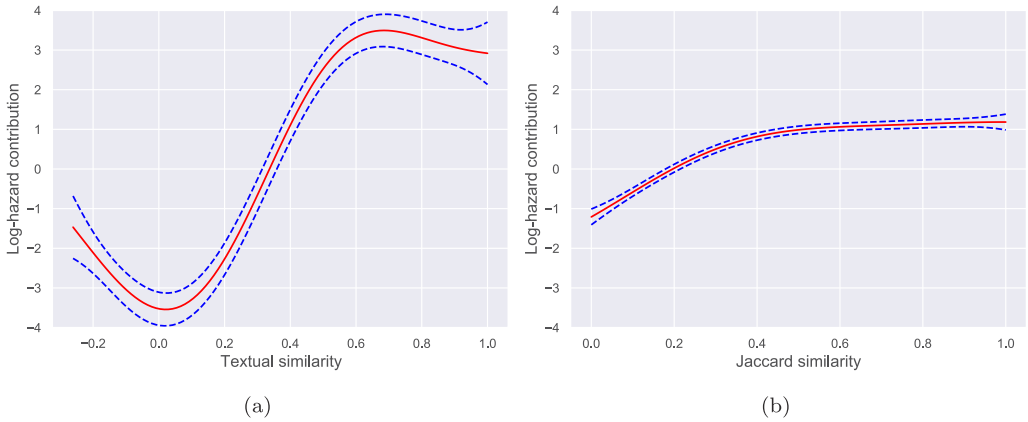


Fig. 5. Similarity effect consisting of the textual similarity effect and the technological relatedness effect.

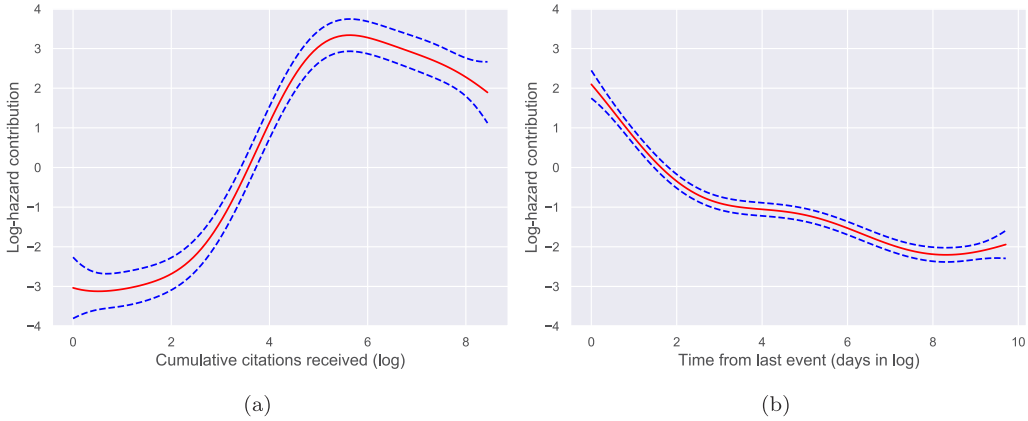


Fig. 6. Time-varying effects consisting of the cumulative citations received and the time from last event.

practical significance of DREAM by modeling a patent citation network, which encompasses nearly 100 million events and about 8 million actors.

In our study, we have not addressed the complex challenge of assessing model fit for REMs, particularly when applied to large-scale and complex datasets. The metrics employed, like KL divergence, while effective under controlled simulation conditions, fall short in empirical scenarios where the true underlying function is unknown. Consequently, we used a held-out set and cross-validation techniques in both our simulation study and our empirical setting, to evaluate the fit of our the chosen model. However, this approach falls short when discerning which non-linear relationships are most accurately represented. This underscore the importance of methodological innovation in the area of model evaluation in REMs to keep pace with the evolving complexity of data structures and analysis techniques in social network research.

DREAM not only offers an efficient and scalable approach to analyzing longitudinal networks and capturing complex non-linear effects, but it also offers remarkable flexibility to customize model complexity. This adaptability includes compatibility with traditional regularization methods like dropout, ridge, and lasso. As speculated in Section 3.1, DREAM has the potential to be expanded to capture complex non-linear interactions among covariates. Currently, our architecture processes each covariate separately. However, future iterations of this model could explore the implementation of a single, wider neural network that processes the entire q-dimensional covariate vector $(x_{sr1}, \dots, x_{srq})$. This approach would directly transform the vector into $f(x_{sr})$, potentially enhancing the model's ability to capture and interpret higher-order interactions without the combinatorial increase in complexity seen with models that estimate functions on pairs of covariates. Such a configuration would not only simplify the architecture but could also offer more profound insights by varying covariate combinations systematically, fixing others at their means to isolate effects. The inherent capability of neural networks to manage high-dimensional inputs suggests that this could be a feasible and valuable direction for future research, particularly as it might overcome the limitations associated with more traditional methods like B-splines in modeling complex interactions within large datasets. Moreover, DREAM flexibility allows to be easily adapted to address multi-cast interactions (Perry and Wolfe, 2013) and further extended to model hyperedges (Lerner and Lomi, 2023).

CRediT authorship contribution statement

Edoardo Filippi-Mazzola: Writing – original draft, Visualization, Software, Resources, Methodology, Formal analysis, Data curation, Conceptualization. **Ernst C. Wit:** Writing – review & editing, Validation, Supervision, Resources, Project administration, Funding acquisition.

Acknowledgment

This work was supported by funding from the Swiss National Science Foundation (SNSF grant 192549).

Appendix A. Oscillatory activation functions

Among the prevalent choices for activation functions lies the family of Rectified Linear Units (ReLU) (Agarap, 2019), known for their

simplicity and computational efficiency. Despite the attractive features, ReLU functions are often affected by the issue known as the "dying ReLU" (Lu, 2020). This emerges in many empirical applications when certain neurons within the network become perpetually inactive, i.e., they continuously output zeros for specific regions of the input support space. This behavior makes the affected neurons essentially irrelevant during the training phase as once a neuron enters in this state, the gradient at that point becomes zero. Consequently, during the backpropagation phase, no updates are made to the weights connected to that neuron. The absence of any weight adjustment leads to a state of inertia where the neuron remains inactive, never contributing to the model again.

Within the family of non-linear activation functions, two noteworthy alternatives to the ReLU are the Sigmoid (Narayan, 1997) and Hyperbolic Tangent (tanh) (Namin et al., 2009). While both functions share similar sigmoidal curves, they exhibit distinct behaviors concerning their derivatives. Specifically, the sigmoid function derivative quickly approaches zero on both right and left sides. This behavior translates to smaller gradients, which in turn leads to protracted training periods. Furthermore, this rapid decline in the derivative magnitude opens to the vanishing gradient problem during backpropagation, which poses a significant challenge in achieving swift and stable convergence in the neural network.

In contrast, the tanh function mitigates some of these difficulties. Its derivative is characteristically sharper and maintains non-zero values over a more extended range on both ends. This design helps in alleviating the vanishing gradient problem to some extent. However, tanh is not without its limitations. Its adaptability is limited by its rigidity in defining the non-linear transformation shape. As a result, the tanh function sometimes struggles to model more intricate and nuanced patterns present in complex datasets.

The challenges associated with previously discussed activation functions prompted a rigorous exploration of alternative units. This led to the adoption of the Growing Cosine Unit (GCU) (Noel et al., 2023). Initially conceptualized to mitigate the "dying ReLU" problem in convolutional neural networks, GCUs have emerged as a promising contender among oscillatory activation functions. Defined as

$$\phi\left(a_k^{(l-1)}\right) = \left(\beta_k^{(l)} a_k^{(l-1)} + \beta_{0k}^{(l)}\right) \cos\left(\beta_k^{(l)} a_k^{(l-1)} + \beta_{0k}^{(l)}\right), \tag{A.1}$$

where $a_k^{(l-1)}$ represents the output from the previous layer, GCUs exhibit a unique property. Unlike ReLU units, which typically yield a singular decision boundary, GCU neurons decision boundary comprises infinitely many parallel hyperplanes. This is attributable to the GCU activation function infinite zeros. Additionally, GCUs offer consistent and favorable derivatives, acting as a countermeasure against the vanishing gradient issue. Furthermore, this trait produces a more efficient training process, marked by reduced duration and improved convergence rates.

Appendix B. ADAM

Consider $\nabla\widetilde{L_P}(\beta)_b$ as the gradient of the partial likelihood for batch b. In the ADAM optimization process, the first and second moment estimations are updated as follows:

$$m_b \leftarrow \xi_1 m_{b-1} + (1-\xi_1)\nabla\widetilde{L_P}(\theta)_b,$$
$$v_b \leftarrow \xi_2 v_{b-1} + (1-\xi_2)\nabla\widetilde{L_P}(\theta)_b^2,$$

where m and v represent the first and second moment gradients, respectively. The hyperparameters ξ_1 and ξ_2 are instrumental in determining the extent to which past gradients influence the current moment updates.

ADAM incorporates bias correction to adjust for the initial bias in the first and second moments of the gradients. This correction is crucial because the moving averages of these gradients start from zero, leading to an initial bias toward zero, particularly noticeable at the early stages of training. To counteract this, ADAM modifies the moving averages with a correction factor that is directly related to the learning rate and inversely related to the iteration count. Denoting the current training step as s, the first and second moments undergo bias correction as follows:

$$\hat{m}_{b,s} = \frac{m_b}{1-\xi_1^s},$$
$$\hat{v}_{b,s} = \frac{v_b}{1-\xi_2^s},$$

with ξ_1^s and ξ_2^s approaching zero as s increases. Consequently, the model parameters are updated by:

$$\beta_b \leftarrow \beta_{b-1} - \psi \frac{\hat{m}_{b,s}}{\sqrt{\hat{v}_{b,s}} + \epsilon},$$

where ψ denotes the learning rate, determining the step size of each parameter update, and ϵ is a small constant (typically $1e-8$) to avoid any division by zero. It is important to note that adjusting the learning rate during training can further refine the estimation of the weights. However, due to the complexity involved in determining an optimal learning rate decay, we chose to maintain a constant learning rate, denoted as ψ, throughout the training process in our simulation studies and application.

Appendix C. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.socnet.2024.05.004.

References

Agarap, A.F., 2019. Deep learning using rectified linear units (relu). arXiv preprint arXiv:1803.08375.

Agarwal, R., Melnick, L., Frosst, N., Zhang, X., Lengerich, B., Caruana, R., Hinton, G., 2021. Neural additive models: Interpretable machine learning with neural nets. arXiv preprint arXiv:2004.13912.

Amati, V., Lomi, A., Mascia, D., 2019. Some days are better than others: Examining time-specific variation in the structuring of interorganizational relations. Social Networks 57, 18–33.

Bauer, V., Harhoff, D., Kauermann, G., 2022. A smooth dynamic network model for patent collaboration data. AStA Adv. Stat. Anal. 106 (1), 97–116.

Bianchi, F., Filippi-Mazzola, E., Lomi, A., Wit, E.C., 2024. Relational event modelling. Annu. Rev. Stat. Appl. 11 (1).

Borgan, O., Goldstein, L., Langholz, B., 1995. Methods for the analysis of sampled cohort data in the Cox proportional hazards model. Ann. Statist. 23 (5).

Boschi, M., Juozaitienė, R., Wit, E.-J.C., 2023. Smooth alien species invasion model with random and time-varying effects. arXiv preprint arXiv:2304.00654.

Butts, C.T., 2008. 4. a relational event framework for social action. Sociol. Methodol. 38 (1), 155–200.

Cox, D.R., 1972. Regression models and life-tables. J. R. Stat. Soc. Ser. B Stat. Methodol. 34 (2), 187–220.

Cox, D.R., 1975. Partial likelihood. Biometrika 62 (2), 269–276.

De Boor, C., 1972. On calculating with b-splines. J. Approx. Theory 6 (1), 50–62.

Efron, B., 1979. Bootstrap methods: Another look at the jackknife. Ann. Statist. 7 (1), 1–26.

Filippi-Mazzola, E., Wit, E.C., 2023. A stochastic gradient relational event additive model for modelling US patent citations from 1976 until 2022. arXiv preprint arXiv:2303.07961.

Fritz, C., Lebacher, M., Kauermann, G., 2020. Tempus volat, hora fugit: A survey of tie-oriented dynamic network models in discrete and continuous time. Stat. Neerl. 74 (3), 275–299.

Goodfellow, I., Bengio, Y., Courville, A., 2016. Deep Learning. MIT Press, Cambridge; Massachusetts.

Hastie, T., Tibshirani, R., 1986. Generalized additive models. Statist. Sci. 1 (3), 297–310.

Juozaitienė, R., Seebens, H., Latombe, G., Essl, F., Wit, E.C., 2023. Analysing ecological dynamics with relational event models: The case of biological invasions. Divers. Distrib. 29 (10), 1208–1225.

Kingma, D.P., Ba, J., 2017. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Kotthaus, H., Korb, I., Lang, M., Bischl, B., Rahnenführer, J., Marwedel, P., 2015. Runtime and memory consumption analyses for machine learning R programs. J. Stat. Comput. Simul. 85 (1), 14–29.

Lerner, J., Lomi, A., 2020. Reliability of relational event model estimates under sampling: How to fit a relational event model to 360 million dyadic events. Netw. Sci. 8 (1), 97–135.

Lerner, J., Lomi, A., 2023. Relational hyperevent models for polyadic interaction networks. J. Roy. Statist. Soc. Ser. A 186 (3), 577–600.

Lomi, A., Bianchi, F., 2021. A time to give and a time to receive: Role switching and generalized exchange in a financial market. Social Networks.

Lu, L., 2020. Dying ReLU and initialization: Theory and numerical examples. Commun. Comput. Phys. 28 (5), 1671–1706.

Namin, A.H., Leboeuf, K., Muscedere, R., Wu, H., Ahmadi, M., 2009. Efficient hardware implementation of the hyperbolic tangent sigmoid function. In: 2009 IEEE International Symposium on Circuits and Systems. pp. 2117–2120.

Narayan, S., 1997. The generalized sigmoid activation function: Competitive supervised learning. Inform. Sci. 99 (1), 69–82.

Noel, M.M., L, A., Trivedi, A., Dutta, P., 2023. Growing cosine unit: A novel oscillatory activation function that can speedup training and reduce parameters in convolutional neural networks. arXiv preprint arXiv:2108.12943.

Oliver, J., May, W.L., Bell, M.L., 2017. Relative effect sizes for measures of risk. Comm. Statist. Theory Methods 46 (14), 6774–6781.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S., 2019. Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32. Curran Associates, Inc, pp. 8024–8035.

Patison, K.P., Quintane, E., Swain, D.L., Robins, G., Pattison, P., 2015. Time is of the essence: an application of a relational event model for animal social networks. Behav. Ecol. Sociobiol. 69 (5), 841–855.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E., 2011. Scikit-learn: Machine learning in Python. J. Mach. Learn. Res. 12, 2825–2830.

Perry, P.O., Wolfe, P.J., 2013. Point process modelling for directed interaction networks. J. R. Stat. Soc. Ser. B Stat. Methodol. 75 (5), 821–849.

Rasmussen, C.E., Williams, C.K.I., 2006. Gaussian Processes for Machine Learning. In: Adaptive Computation and Machine Learning, MIT Press, Cambridge, Mass., OCLC: ocm61285753.

Reddi, S.J., Kale, S., Kumar, S., 2018. On the convergence of adam and beyond. arXiv preprint arXiv:1904.09237.

Reimers, N., Gurevych, I., 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics.

Ripley, B.D., 1996. Pattern Recognition and Neural Networks. Cambridge University Press, Cambridge ; New York.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R., 2014. Dropout: A simple way to prevent neural networks from overfitting. J. Mach. Learn. Res. 15, 1929–1958.

Tranmer, M., Marcum, C.S., Morton, F.B., Croft, D.P., de Kort, S.R., 2015. Using the relational event model (REM) to investigate the temporal dynamics of animal social networks. Anim. Behav. 101, 99–105.

Vu, D.Q., Asuncion, A.U., Hunter, D.R., Smyth, P., 2011. Dynamic egocentric models for citation networks. In: Proceedings of the 28th International Conference on Machine Learning. ICML-11, pp. 857–864.

Vu, D., Lomi, A., Mascia, D., Pallotti, F., 2017. Relational event models for longitudinal network data with an application to interhospital patient transfers. Stat. Med..

Vu, D., Pattison, P., Robins, G., 2015. Relational event models for social learning in MOOCs. Social Networks 43, 121–135.

Welles, B.F., Vashevko, A., Bennett, N., Contractor, N., 2014. Dynamic models of communication in an online friendship network. Commun. Methods Meas. 8 (4), 223–243.

Zappa, P., Vu, D.Q., 2021. Markets as networks evolving step by step: Relational event models for the interbank market. Phys. A 565, 125557.