A Recurrent Neural Cascade-based Model for Continuous-Time Diffusion

Supplementary Material

Sylvain Lamprier [1]

Abstract

This is the supplementary material that serves as an appendix to the paper "A Recurrent Neural Cascade-based Model for Continuous-Time Diffusion".

1. Joint Probability

In this section, we detail the derivation of the equation 10 from the paper, which states that, using the bayesian chain rule, the joint probability $p(D, I)$ can be written as:

$$p(D,I) = \prod_{i=1}^{|D|-1} p(D_i|D_{<i}, I_{<i})\, p(I_i|D_{\leq i}, I_{<i}) \times \prod_{v \notin U^D} p(v \notin U^D | D_{\leq |D|-1}, I) \quad (1)$$

where $D_{<i} = (D_j)_{j \in \{0,\dots,i-1\}}$ corresponds to the sequence of the i first components of D (the i first components in U^D with their associated time-stamps) and $I_{<i} = (I_j)_{j \in \{0,\dots,i-1\}}$ stands for the vector containing the i first components of I. This comes from the fact that, for each infected node at any position i, we need to compute:

- $p(D_i|D_{<i}, z^D_{<i})$: the probability for U^D_i for being infected at its time of infection given the nodes $D_{<i}$ previously infected in D and the states associated to these nodes;
- $p(I_i|D_{\leq i}, z^D_{<i})$: the probability of the ancestor index I_i given the i-th infection D_i, and the previous infections $D_{<i}$ associated to their states $z^D_{<i}$;
- the probability that not infected nodes are actually not infected by the i-th infected node given its state.

This gives:

$$\begin{aligned} p(D,I) &= p(D_0|D_{<0}, z^D_{<0}) p(I_0|D_{\leq 0}, z^D_{<0}) \prod_{v \notin U^D} p(v \notin U^D | D_0, z^D_0) \\ &\times p(D_1|D_{<1}, z^D_{<1}) p(I_1|D_{\leq 1}, z^D_{<1}) \prod_{v \notin U^D} p(v \notin U^D | D_2, z^D_2) \\ &\dots \\ &\times p(D_{|D|-1}|D_{<|D|-1}, z^D_{<|D|-1}) \\ &\quad p(I_{|D|-1}|D_{\leq |D|-1}, z^D_{<|D|-1}) \prod_{v \notin U^D} p(v \notin U^D | D_{|D|-1}, z^D_{|D|-1}) \\ &= \prod_{i=1}^{|D|-1} p(D_i|D_{<i}, I_{<i})\, p(I_i|D_{\leq i}, I_{<i}) \prod_{v \notin U^D} p(v \notin U^D | D_{\leq |D|-1}, I) \end{aligned}$$

2. Generation Process

The generation process of our model is given in algorithm 1. The process iterates while there remains some nodes in a set of infectious nodes (initialized with u_0). $\oplus$ denotes the concatenation between two lists. At each iteration, the process selects the infectious node u with minimal time-stamp of infection (all time-stamps but $t^D_{u_0}$ are initialized to ∞), removes it from the infectious set and records its infector and infection time-stamp in the cascade. Then, for each node v with time-stamp greater than the one of u, u attempts to infects v according to the probability $k_{u,v}(z^D_u)$ (computed with eq 3 from the paper). If it succeeds, v is inserted in the set of infectious nodes and a time t is sampled for v from an exponential law with parameter $r^D_{u,v}$. If the new time t for v is lower than its stored time t^D_v, this new time is stored in t^D_v, u is stored as the infector of v in the $from$ table (used to build I^D) and the new state z^D_v is computed according to its new infector u. The generation process outputs a cascade structure (as described in section 2.1 of the paper). From the classical `CTIC`, the only changes are at lines 1, 1 and 1, respectively for the computation of $k_{u,v}$, $r_{u,v}$ and z^D_v.

[1]Sorbonne Université, LIP6, F-75005, Paris, France. Correspondence to: Sylvain Lamprier <Sylvain.Lamprier@lip6.fr>.

Proceedings of the 36th International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

Algorithm 1 Cascade Generation Process

Input: $\Theta, \mathcal{U}$
for $u \in \mathcal{U}$ **do**
  $t_u^D = \infty$
end for
$U^D = (); I^D = (); t_{u_0}^D = 0; from_{u_0} = 0;$
$Infectious = \{u_0\};$
while $Infectious \neq \emptyset$ **do**
  $u \leftarrow \underset{x \in Infectious}{\arg\min}\ t_x^D;$
  $Infectious \leftarrow Infectious \setminus \{u\};$
  $U^D \leftarrow U^D \oplus u\ ;$
  $I^D \leftarrow I^D \oplus from_u\ ;$
  for $v \in \mathcal{U} : t_v^D > t_u^D$ **do**
    $x \sim Bernouilli\,(k_{u,v}(z_u));$
    if $x == 1$ **then**
      $x \sim Exp\,(r_{u,v});$
      $t \leftarrow t_u^D + t;$
      if $t < t_v^D$ **then**
        $t_v^D \leftarrow t;$
        $from_v \leftarrow |U^D| - 1;$
        $z_v = f_\phi(z_u, \omega_v^{(f)});$
        $Infectious \leftarrow Infectious \cup \{v\};$
      end if
    end if
  end for
end while
Output: $C^D = (U^D, (t_u^D)_{u \in \mathcal{U}}, I^D);$

3. Learning Process

The learning process of our model is depicted in algorithm 2. In this algorithm, the function $makeBins$ first creates minibatches by ordering $\mathcal{D}$ in decreasing length and cutting this ordered list in bins of $batchSize$ episodes each. Each bin contains 3 matrices with $batchSize$ rows (except in the last bin which contains matrices for the remaining $|\mathcal{D}|\%batchSize$ episodes):

- Inf: a matrix where the cell (i, j) contains the j-infected node in the i-th episode of the bin, or -1 if the corresponding episode contains less than j infected nodes. The width of the matrix is equal to the number of infected nodes in the longest episode in the bin (the episode in the first row of the matrix);

- $Times$: a matrix where the cell (i, j) contains the infection time-stamp of the j-infected node in the i-th episode of the bin, or -1 if the corresponding episode contains less than j infected nodes. The width of the matrix is equal to the number of infected nodes in the longest episode in the bin (the episode in the first row of the matrix);

- $NotInf$: a binary matrix with $|\mathcal{U}|$ columns where the cell (i, j) equals 0 if the node j is infected in the i-the episode of the bin, 1 otherwise;

At each epoch, the algorithm iterates on every bin. For each bin, it first initializes the states of the infected nodes using a function $initStates$ which produces a tensor $\boldsymbol{z}$ of $nbRows(Inf)$ matrices $nbCols(Inf) \times d$ whose each row is filled by z_0 (with $nbRows(X)$ and $nbCols(X)$ respectively the number of rows and columns in matrix X). For every step t of infection in the bin, the process first determines in $mask$ the rows of the matrices which correspond to not ended episodes ($Times[:, t]$ refers to the column t of $Times$). Then, if the step is not the initial step $t = 0$, it uses functions $computeLogA$ and $computeLogB$ with nodes previously infected for each episode $Inf[mask, : t]$ associated to their corresponding states $\boldsymbol{z}[mask, : t]$. While the function $computeLogA$ returns a $nbRow(Inf[mask]) \times t$ matrix where the cell (i, j) contains the log-probability for the j-th node in the i-th episode to infect $Inf[i, t]$ at its infection time-stamp (using a matrix version of equation 5 from the paper), the function $computeLogB$ returns a same shape matrix where the cell (i, j) contains the log-probability that the j-th node in the i-th episode does not infect $Inf[i, t]$ before its infection time-stamp (using a matrix version of equation 6 from the paper).

Then, ancestors at step t are sampled from categorial distributions parameterized by $P(I_t | D_{\leq t}, I_{<t})$ (deduced from logits $A - B$). From them, we compute the log-probability for each infected at step t to be actually infected at their time-stamp of infection by its corresponding sampled infector. (line 2, where $sum(X, 1)$ is a function which returns the vector of the sums of each row from X). This quantity is added to the accumulator ll.

Line 2 then computes the states for the nodes infected at step t according to the states of the sampled ancestors in $\boldsymbol{u}$ (via the function $computeStates$ which is a matrix version of equation 2 from the paper).

At the end of each iteration t, the log-likelihood that not infected nodes in $NotInf[mask]$ are actually not infected by infected nodes at step t is computed via $computeLogG$, which is a matrix version of equation 8 from the paper. This quantity is added to the accumulator ll.

At the end of the bin (when $t == nbCols$), a control variate baseline is computed by maintaining a list bh of the quantity vectors considered in $\nabla_\Theta \mathcal{L}(\mathcal{D})$. The baseline b considered in the stochastic gradient for any episode D is thus equal to the average of $(logp(D) - 1)$ for this specific episode taken over the b_length previous epochs.

Finally, the gradients are computed and the optimizer ADAM is used to update the parameters of the model. Note that this algorithm does not use the gradient up-

Algorithm 2 Learning Process

Input: $\mathcal{D}, \mathcal{U}, batchSize, nbEpochs, \Theta, b\_length$
$bins \leftarrow makeBins(\mathcal{D}, \mathcal{U}, batchSize)$;
for $epoch \in \{1, ..., nbEpochs\}$ **do**
  $ibin \leftarrow 0$;
  for (Inf, ,Times, NotInf) in bins **do**
    $ll \leftarrow (0)_{nbRows(Inf)}$;
    $logq \leftarrow (0)_{nbRows(Inf)}$;
    $\boldsymbol{z} \leftarrow initStates(z_0, nbRows(Inf), nbCols(Inf))$;
    for $t \in \{0, ..., nbCols(Inf)\}$ **do**
      $mask \leftarrow (Times[:, t] >= 0)$;
      if $t > 0$ **then**
        $A \leftarrow computeLogA(\boldsymbol{z}[mask, : t], Inf[mask, : t], Inf[mask, t])$;
        $B \leftarrow computeLogB(\boldsymbol{z}[mask, : t], Inf[mask, : t], Inf[mask, t])$;

        # Sample from $P(I_t | D_{\leq t}, I_{<t})$
        $\boldsymbol{u} \sim Categorical(logits = (A - B))$;
        $logq[mask] \leftarrow logq[mask] + logPi[mask, \boldsymbol{u}]$;

        # Compute $P(D_t, I_t | D_{<t}, I_{<t})$
        $H \leftarrow A[mask, \boldsymbol{u}] - B[mask, \boldsymbol{u}] + sum(B, 1)$;
        $ll[mask] \leftarrow ll[mask] + H$

        $\boldsymbol{z}[mask, t] \leftarrow computeStates(\boldsymbol{u}, \boldsymbol{z}[mask, : t], Inf[mask, : t], Inf[mask, t])$;
      end if
      $ll[mask] \leftarrow ll[mask] + computeLogG(\boldsymbol{z}[mask, t], Inf[mask, t], NotInf[mask])$;
    end for
    $bh[ibin] \leftarrow h[ibin] \oplus (ll - logq - 1)$;
    if $epoch \geq b\_length$ **then**
      $bh[ibin].pop(0)$;
    end if
    $b \leftarrow sum(bh[ibin], 1) / min(epoch + 1, b\_length)$;
    $\nabla_\Theta \mathcal{L}(\mathcal{D}; \Theta) \leftarrow \frac{1}{nbRows(Inf)} [sum((ll - logq - 1 - b) \nabla_\Theta logq + \nabla_\Theta \log ll)]$;
    $\Theta \leftarrow ADAM(\nabla_\Theta \mathcal{L}(\mathcal{D}; \Theta))$;

    $ibin \leftarrow ibin + 1$;
  end for
end for

date given in eq. 13 from the paper. It is based on $P(I_t | D_{\leq t}, I_{<t})$ and $P(D_t, I_t | D_{<t}, I_{<t})$ for every $t \in \{0, ..., nbCols(Inf)\}$ (rather than based on the simplification $P(D_t | D_{<t}, I_{<t})$ as given in eq. 12 from the paper). This is equivalent but greatly more efficient since in both cases $P(I_t | D_{\leq t}, I_{<t})$ needs to be estimated for sampling and $P(D_t, I_t | D_{<t}, I_{<t})$ is much easier to compute than $P(D_t | D_{<t}, I_{<t})$ ($P(D_t, I_t | D_{<t}, I_{<t})$ involves a simple product while $P(D_t | D_{<t}, I_{<t})$ involves a sum of products).

4. Conditioned Models

Our experiments include results obtained with known starts of episodes (columns 1, 2 and 3, for which $\tau > 1$). For these cases, one need to be able to condition the models according to right-censored episodes, which we note D^τ in the following (every infected node in $u \in U^D$ with $t_u^D \geq \tau$ is reported as not infected in D^τ). For the NLL measure, one need to be able to compute the negative log-likelihood of the end of any episode D given the beginning D^τ (which means estimating $p(D|D^\tau)$). For the CE measure, one need to estimate the probabilities of final infections in any episode D given its beginning D^τ (i.e., estimating $p(u \in U^D | D^\tau)$

via Monte-Carlo simulations).

For models RNN, CYAN and DAN which have deterministic hidden representations, the conditioning on D^τ is direct: it suffices to traverse the episode from the start to the last infected node in D^τ to obtain the full representation of the input. Then, the model can be normally used from it on the remaining of the test episode for modeling or generation purposes (except for the time-stamp of the first next event for which one has to take into account that it cannot happen before τ).

For CTIC and EmbCTIC, the conditioning is also easy since future infections from τ only depend on time-stamps of infections before τ without any need of trajectory modelling. For simulation purposes and the CE measure, the algorithm 1 can be applied with the infection time-stamps and the $Infectious$ set initialized according to D^τ. Then, the only difference is that infection delays are sampled from a truncated exponential in order to ensure that new infections can not occur before τ. For the NLL measure, one need to reconsider the computation of the infection probabilities from nodes infected before τ, which must take into account that infections did not happened before τ. For each node v which is not infected in D^τ, its probability $a^D_{u,v}$ of being infected by any infected node u in D^τ is divided by the probability that u did not succeed in infecting v before τ:

$$a^{D|D^\tau}_{u,v} = \begin{cases} \dfrac{k_{u,v} r_{u,v} \exp^{-r_{u,v}(t^D_v - t^D_u)}}{k_{u,v} \exp^{-r_{u,v}(\tau - t^D_u)} + 1 - k_{u,v}} & \text{if } t^D_u < \tau, \\ k_{u,v} r_{u,v} \exp^{-r_{u,v}(t^D_v - t^D_u)} & \text{otherwise.} \end{cases} \tag{2}$$

Also, for every v with $t^D_v \geq \tau$ and every u with $t^D_u < t^D_v$, we rewrite the probability $b^D_{u,v}$ that u does not succeed in infecting v at t^D_v as:

$$b^{D|D^\tau}_{u,v} = \begin{cases} \text{If } t^D_u < \tau \text{ and } t^D_v < \infty : \\ \quad \dfrac{k_{u,v} \exp^{-r_{u,v}(t^D_v - t^D_u)} + 1 - k_{u,v}}{k_{u,v} \exp^{-r_{u,v}(\tau - t^D_u)} + 1 - k_{u,v}} \\ \text{If } t^D_u < \tau \text{ and } t^D_v = \infty : \\ \quad \dfrac{1 - k_{u,v}}{k_{u,v} \exp^{-r_{u,v}(\tau - t^D_u)} + 1 - k_{u,v}} \\ \text{Otherwise:} \\ \quad k_{u,v} \exp^{-r_{u,v}(t^D_v - t^D_u)} + 1 - k_{u,v} \end{cases} \tag{3}$$

These quantities are used to compute the conditionnal likelihood $p(D|D^\tau)$:

$$p(D|D^\tau) = \prod_{v \in U^D, t^D_v \geq \tau} h^{D|D^\tau}_v \prod_{v \notin U^D} g^{D|D^\tau}_v$$

where $g^{D|D^\tau}_v = \prod_{u \in U^D} b^{D|D^\tau}_{u,v}$ and, similarly to eq.7 from the paper without the dependency in states $\boldsymbol{z}$ for CTIC, the conditionnal $h^{D|D^\tau}_v$ is given as:

$$h^{D|D^\tau}_v = \prod_{x \in \mathcal{U}, t^D_x < t^D_v} b^{D|D^\tau}_{x,v} \sum_{u \in \mathcal{U}, t^D_u < t^D_v} a^{D|D^\tau}_{u,v} / b^{D|D^\tau}_{u,v} \tag{4}$$

At last, for the RINF measure, the only modification is that the statistic is computed solely on nodes $i \in U^D$ such that $t^D_i \geq \tau$.

For our model, beyond the use of conditionnal probabilities (with similar definitions than those from eq.2, 3 and 4 but with k a function depending on the state z of the emitter such as in the section 3.2 from the paper), we must consider the conditionnal distribution of the ancestors vector of nodes infected before τ, hereafter noted I^τ, given the input D^τ. As done for learning, I^τ is sampled from our propositional distribution $q^{D^\tau}(I^\tau) = \prod_{i=1}^{|D^\tau|-1} p(I^\tau_i | D^\tau_{\leq i}, I^\tau_{<i})$ and the measures are simple averages on a set of sampled I^τ. Given S samples of I^τ, the NLL is thus computed as:

$$\begin{aligned} NLL &= -\frac{1}{|\mathcal{D}_{test}|} \sum_{D \in \mathcal{D}_{test}} \log \frac{1}{S} \sum_{s=1}^{S} p(D|D^\tau, I^{\tau,(s)}) \qquad (5) \\ &= -\frac{1}{|\mathcal{D}_{test}|} \sum_{D \in \mathcal{D}_{test}} \log \frac{1}{S} \sum_{s=1}^{S} \frac{p(D, I^{(s)}|D^\tau, I^{\tau,(s)})}{q^D(I^{(s)}|I^{\tau,(s)})} \end{aligned}$$

with $q^D(I^{(s)}|I^{\tau,(s)}) = q^D(I^{(s)}) / q^{D^\tau}(I^{\tau,(s)})$ and where $p(D, I^{(s)}|D^\tau, I^{\tau,(s)})$ is computed same manner as in section 3.2 using conditionnal versions of probability formulations (as given above for CTIC). The last derivation is obtained according to samples of full ancestor vectors I, where $I^{(s)}$ is an ancestor vector starting with $I^{\tau,(s)}$ for the infections before τ. Similarly, the CE measure considers marginal probabilities $p(u \in U^D | D^\tau)$ for every node $u \in \mathcal{U}$ defined as averages on simulations performed given D^τ and sampled $I^{\tau,(s)}$ as input of the generation algorithm 1. Lastly, the INF measure is evaluated by considering:

$$INF = \sum_{D \in \mathcal{D}_{test}} \sum_{\substack{i \in \{1,\dots,|D|-1\}, \\ t_{U^D_i} \geq \tau}} \frac{p(I_i = fr(i,D) | D_{\leq i}, I^{(s)}_{<i})}{\sum_{D \in \mathcal{D}_{test}} (|D| - |D^\tau|)} \tag{6}$$

where $I^{(s)}$ is an ancestor vector sampled from q^D.