Derivations of the update rules for the paper: Spiking networks can solve planning tasks!

Elmar Rueckert\textsuperscript{1}, David Kappel\textsuperscript{2}, Dejan Pecevski\textsuperscript{2} and Jan Peters\textsuperscript{1,3}

\textsuperscript{1}Intelligent Autonomous Systems Lab, Technische Universität Darmstadt 64289 Darmstadt, Germany
\textsuperscript{2}Institute for Theoretical Computer Science, Technische Universität Graz 8020 Graz, Austria
\textsuperscript{3}Robot Learning Group, Max-Planck Institute for Intelligent Systems 72076 Tübingen, Germany
rueckert@ias.tu-darmstadt.de kappel@igi.tugraz.at pecevski@igi.tugraz.at

Abstract

In the paper \textit{Spiking networks can solve planning tasks!} the posterior distribution \( p(\nu_{1:T} \mid r = 1) \) is approximated by the model distribution \( q(\nu_{1:T} ; \theta) \). In this document update rules are derived that minimize the Kullback-Leibler (KL) divergence between these two distributions \( D_{KL}(p(\nu_{1:T} | r = 1) || q(\nu_{1:T} ; \theta)) \).

1 Formulating planning as inference as optimization problem

In the considered planning as inference process a sequence of discrete states \( \nu_{1:T} \) is computed through conditioning on receiving a reward in each time step. The magnitude of the rewards is encoded as binary event, which is denoted by \( r = 1 \) in \( p(\nu_{1:T} | r = 1) \). Note that this shift to binary events from reward magnitude or utility is a crucial step in formulating decision making as inference problem (Solway and Botvinick, 2012). To keep the notation uncluttered we use the symbol \( \nu \) as shorthand for the sequence of \( T \) states \( \nu_{1:T} \).

The goal of the neural network learning is to minimize the KL divergence

\[
D_{KL}(p(\nu \mid r = 1) || q(\nu ; \theta)) = \sum_\nu p(\nu \mid r = 1) \log \frac{p(\nu \mid r = 1)}{q(\nu ; \theta)}
= \sum_\nu p(\nu \mid r = 1) \log p(\nu \mid r = 1) - \sum_\nu p(\nu \mid r = 1) \log q(\nu ; \theta)
= -H_{p(\nu \mid r = 1)} - \langle \log q(\nu ; \theta) \rangle_{p(\nu \mid r = 1)}.
\]

The entropy of the true data distribution (for planning) is denoted by \( H_{p(\nu \mid r = 1)} \) and the second term denotes the expectation of the log-likelihood of the model distribution w.r.t the true posterior.

To derive update rules for the parameters \( \theta \) through maximum likelihood the entropy can be ignored as it is independent of \( \theta \). The optimal parameters minimizing the KL divergence are given by \( \theta^* = \arg\max_{\theta} \langle \log q(\nu ; \theta) \rangle_{p(\nu \mid r = 1)} \).

In planning, the distribution \( p(\nu \mid r = 1) \) is unknown and we cannot draw samples from it. However, we can draw samples from \( \xi = p(\nu \mid \nu) \) \( q(\nu ; \theta) \) and update the parameters such that the probability of receiving a reward is maximized. The parameter update is applied iteratively, where \( \eta \) denotes a small learning rate

\[
\Delta \theta = \eta \left( \frac{r}{\partial \theta} \log q(\nu ; \theta) \right)_\xi = \eta \left( r \sum_{t=1}^{T} \frac{\partial}{\partial \theta} \log \phi_t(\nu_t ; \theta) \right)_\xi,
\]

where we chose functions \( \phi_t \) that factorize i.e., \( \phi_t(\nu ; \theta) = \prod_{t=1}^{T} \phi_t(\nu_t ; \theta) \), and we exploited that only the function \( \phi_t \) depends on the parameters \( \theta \) in \( q(\nu ; \theta) = p(\nu_0) \prod_{t=1}^{T} T(\nu_t | \nu_{t-1}) \phi_t(\nu_t ; \theta) \). This distribution as well as the parameter update can be implemented in recurrent spiking neural networks.
2 Solving a planning problem with recurrent neural networks

We denote the activity of the two populations at time $t$ by $\nu_t$ and $y_t$, and by assuming linear dendritic dynamics we can define the membrane potential of state neuron $k$ in discrete time

$$u_{t,k} = \sum_{i=1}^{K} w_{ki} \nu_{t-1,i} + \sum_{j=1}^{N} \theta_{kj} y_{t-1,j}.$$  \hspace{1cm} (2)

The activity of the state neurons are constrained to winner-take-all (WTA) dynamics, which assures that exactly one neuron is active in each time step, i.e. $\sum_{k=1}^{K} \nu_{t,k} = 1 \forall t$. Therefore the probability $\rho_{t,k}$ of neuron $k$ to spike at time $t$ is given by

$$\rho_{t,k} = p(\nu_{t,k} = 1 | \nu_{t-1}, y_{t}; \theta) = \frac{\exp(u_{t,k})}{\sum_{l=1}^{K} \exp(u_{t,l})}.$$ \hspace{1cm} (3)

These network dynamics realize a distribution over network state trajectories $\nu = \nu_1: T$ given by

$$q(\nu; \theta) = p(\nu_0) \prod_{t=1}^{T} \prod_{k=1}^{K} \rho_{t,k}^{\nu_{t,k}} = p(\nu_0) \prod_{t=1}^{T} \prod_{k=1}^{K} \left( \frac{\exp(u_{t,k})}{\sum_{l=1}^{K} \exp(u_{t,l})} \right)^{\nu_{t,k}},$$ \hspace{1cm} (4)

with

$$\mathcal{T}(\nu_t | \nu_{t-1}) = \prod_{k=1}^{K} \exp\left( \sum_{i=1}^{K} w_{ki} \nu_{t-1,i} \right)^{\nu_{t,k}},$$

and

$$\phi_t(\nu_t; \theta) = \prod_{k=1}^{K} \left( \frac{\exp(\sum_{j=1}^{N} \theta_{kj} y_{t-1,j})}{\sum_{l=1}^{K} \exp(u_{t,l})} \right)^{\nu_{t,k}}.$$  \hspace{1cm} (5)

Thus the first term of (2) determines the transition operator $\mathcal{T}$ implemented through the lateral weights $w_{ki}$, and the second term realizes the function $\phi_t$ parametrized by the feedforward weights $\theta_{kj}$.

3 Derivation of a reward-modulated Hebbian learning rule

In (4) we have established the link between the parametrized distribution $q(\nu; \theta)$ and the neural implementation. This result is now used to derive a Hebbian learning rule that implements the iterative updates in (1). We solve (1)
for partial derivatives in $\theta_{kj}$, where

\[
\Delta \theta_{kj} = \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{kj}} \log \phi_t(\nu_t; \theta) \right\rangle_{\xi} 
\]

\[
= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{kj}} \log \prod_{k=1}^{K} \left( \frac{\exp \left( \sum_{j=1}^{N} \theta_{kj} y_{t-1,j} \right)}{\sum_{l=1}^{K} \exp (u_{t,l})} \right)^{\nu_{t,k}} \right\rangle_{\xi} 
\]

\[
= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{kj}} \sum_{k=1}^{K} \log \left[ \exp \left( \sum_{j=1}^{N} \theta_{kj} y_{t-1,j} \right)^{\nu_{t,k}} \right] - \sum_{k=1}^{K} \log \left( \sum_{l=1}^{K} \exp (u_{t,l}) \right)^{\nu_{t,k}} \right\rangle_{\xi} 
\]

\[
= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{kj}} \left[ \sum_{k=1}^{K} \sum_{j=1}^{N} \theta_{kj} y_{t-1,j} \nu_{t,k} - \log \sum_{k=1}^{K} \exp (u_{t,l}) \right] \right\rangle_{\xi} 
\]

\[
= \eta \left\langle r \sum_{t=1}^{T} \left[ y_{t-1,j} \nu_{t,k} - \frac{\exp (u_{t,k})}{\sum_{l=1}^{K} \exp (u_{t,l})} \frac{\partial}{\partial \theta_{kj}} u_{t,k} \right] \right\rangle_{\xi} 
\]

\[
= \eta \left\langle r \sum_{t=1}^{T} (\nu_{t,k} - \rho_{t,k}) \right\rangle_{\xi} 
\]

(5)

In Equation (5) we used $\sum_{k=1}^{K} \nu_{t,k} = 1 \forall t$. In (6) the definition of the $k$'s neuron firing probability $\rho_{t,k}$ in (3) was plugged in. In addition we exploited the fact that the partial derivative of the membrane potential in (2) is $y_{t-1,j}$.

The resulting iterative update in (7) is a reward-modulated Hebbian learning rule that gives a positive update only if a reward ($r = 1$) is delivered at the end of a trial.

### 4 Assumptions

We made two assumptions. First, in (2) we assumed linear dendritic dynamics without synaptic delays. Second, the state neurons $\nu_t$ are constrained to winner-take-all (WTA) dynamics, which assures that exactly one neuron is active in each time step, i.e. $\sum_{k=1}^{K} \nu_{t,k} = 1 \forall t$.

### References