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

Elmar Rueckert[1], David Kappel[2], Dejan Pecevski[2] and Jan Peters[1,3]

[1]Intelligent Autonomous Systems Lab, Technische Universität Darmstadt
64289 Darmstadt, Germany
rueckert@ias.tu-darmstadt.de

[2]Institute for Theoretical Computer Science, Technische Universität Graz
8020 Graz, Austria
kappel@igi.tugraz.at
pecevski@igi.tugraz.at

[3]Robot Learning Group, Max-Planck Institute for Intelligent Systems
72076 Tuebingen, Germany
mail@jan-peters.net

Abstract

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

1 Formulating planning as inference as optimization problem

In the considered planning as inference process a sequence of discrete states $\boldsymbol{\nu}_{1:T}$ 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 $\mathrm{p}\left(\boldsymbol{\nu}_{1:T} \mid r=1\right)$. 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 $\underline{\boldsymbol{\nu}}$ as shorthand for the sequence of T states $\boldsymbol{\nu}_{1:T}$.

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

$$\begin{aligned} D_{KL}(p(\underline{\boldsymbol{\nu}}|r=1)||q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right)) &= \sum\nolimits_{\underline{\boldsymbol{\nu}}} p(\underline{\boldsymbol{\nu}}|r=1)\log\frac{p(\underline{\boldsymbol{\nu}}|r=1)}{q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right)} \\ &= \sum\nolimits_{\underline{\boldsymbol{\nu}}} p(\underline{\boldsymbol{\nu}}|r=1)\log p(\underline{\boldsymbol{\nu}}|r=1) - \sum\nolimits_{\underline{\boldsymbol{\nu}}} p(\underline{\boldsymbol{\nu}}|r=1)\log q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right) \\ &= -H_{p(\underline{\boldsymbol{\nu}}|r=1)} - \left\langle \log q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right)\right\rangle_{p(\underline{\boldsymbol{\nu}}|r=1)} \quad . \end{aligned}$$

The entropy of the true data distribution (for planning) is denoted by $H_{p(\underline{\boldsymbol{\nu}}|r=1)}$ 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 $\boldsymbol{\theta}$ through maximum likelihood the entropy can be ignored as it is independent of $\boldsymbol{\theta}$. The optimal parameters minimizing the KL divergence are given by $\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} \left\langle \log q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right)\right\rangle_{p(\underline{\boldsymbol{\nu}}|r=1)}$. In planning, the distribution $p(\underline{\boldsymbol{\nu}}|r=1)$ is unknown and we cannot draw samples from it. However, we can draw samples from $\xi = \mathrm{p}\left(r \mid \underline{\boldsymbol{\nu}}\right)\, q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right)$ and update the parameters such that the probability of receiving a reward is maximized. The parameter update is applied iteratively, where η denotes a small learning rate

$$\Delta\theta \;=\; \eta\left\langle r\,\frac{\partial}{\partial\theta}\,\log q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right)\right\rangle_{\xi} \;=\eta\left\langle r\sum_{t=1}^{T}\frac{\partial}{\partial\theta}\,\log \phi_t(\boldsymbol{\nu}_t\,;\,\boldsymbol{\theta})\right\rangle_{\xi} \quad , \tag{1}$$

where we chose functions ϕ_t that factorize i.e., $\phi_t(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}) = \prod_{t=1}^{T}\,\phi_t(\boldsymbol{\nu}_t;\boldsymbol{\theta})$, and we exploited that only the function ϕ_t depends on the parameters $\boldsymbol{\theta}$ in $q\left(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta}\right) = \mathrm{p}\left(\boldsymbol{\nu}_0\right)\prod_{t=1}^{T}\mathcal{T}\left(\boldsymbol{\nu}_t \mid \boldsymbol{\nu}_{t-1}\right)\,\phi_t(\boldsymbol{\nu}_t;\boldsymbol{\theta})$. 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 $\boldsymbol{\nu}_t$ and $\boldsymbol{y}_t$, and by assuming linear dendritic dynamics we can define the membrane potential of state neuron k in discrete time

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

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

$$\rho_{t,k} = \mathrm{p}\left(\nu_{t,k} = 1 \mid \boldsymbol{\nu}_{t-1}, \boldsymbol{y}_t; \theta\right) = \frac{\exp\left(u_{t,k}\right)}{\sum_{l=1}^{K} \exp\left(u_{t,l}\right)} \quad . \tag{3}$$

These network dynamics realize a distribution over network state trajectories $\underline{\boldsymbol{\nu}} = \boldsymbol{\nu}_{1:T}$ given by

$$\begin{aligned} q\left(\underline{\boldsymbol{\nu}}\,;\boldsymbol{\theta}\right) \quad &= \quad p(\boldsymbol{\nu}_0) \prod_{t=1}^{T} \prod_{k=1}^{K} {\rho_{t,k}}^{\nu_{t,k}} \;=\; p(\boldsymbol{\nu}_0) \prod_{t=1}^{T} \prod_{k=1}^{K} \left(\frac{\exp\left(u_{t,k}\right)}{\sum_{l=1}^{K} \exp\left(u_{t,l}\right)} \right)^{\nu_{t,k}} \\ &= \quad p(\boldsymbol{\nu}_0) \prod_{t=1}^{T} \mathcal{T}\left(\boldsymbol{\nu}_t \mid \boldsymbol{\nu}_{t-1}\right) \phi_t\left(\boldsymbol{\nu}_t\,;\boldsymbol{\theta}\right) \quad , \\ \text{with} \qquad & \mathcal{T}\left(\boldsymbol{\nu}_t \mid \boldsymbol{\nu}_{t-1}\right) = \prod_{k=1}^{K} \exp\left(\sum_{i=1}^{K} w_{ki}\, \nu_{t-1,i} \right)^{\nu_{t,k}} \quad , \\ \text{and} \qquad & \phi_t\left(\boldsymbol{\nu}_t\,;\boldsymbol{\theta}\right) = \prod_{k=1}^{K} \left(\frac{\exp\left(\sum_{j=1}^{N} \theta_{kj}\, y_{t-1,j} \right)}{\sum_{l=1}^{K} \exp\left(u_{t,l}\right)} \right)^{\nu_{t,k}} \quad . \end{aligned} \tag{4}$$

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 ϕ_t parametrized by the feedforward weights θ_{kj}.

3 Derivation of a reward-modulated Hebbian learning rule

In (4) we have established the link between the parametrized distribution $q\left(\underline{\boldsymbol{\nu}}\,;\boldsymbol{\theta}\right)$ 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 θ_{kj}, where

$$
\begin{aligned}
\Delta\theta_{kj} &= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{k,j}} \log \phi_t(\boldsymbol{\nu}_t \,;\, \boldsymbol{\theta}) \right\rangle_{\xi} \\
&= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{k,j}} \log \prod_{k=1}^{K} \left(\frac{\exp\left(\sum_{j=1}^{N} \theta_{kj}\, y_{t-1,j}\right)}{\sum_{l=1}^{K} \exp\left(u_{t,l}\right)} \right)^{\nu_{t,k}} \right\rangle_{\xi} \\
&= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{k,j}} \sum_{k=1}^{K} \log \left[\exp\left(\sum_{j=1}^{N} \theta_{kj}\, y_{t-1,j} \right)^{\nu_{t,k}} \right] - \sum_{k=1}^{K} \log \left(\sum_{l=1}^{K} \exp(u_{t,l}) \right)^{\nu_{t,k}} \right\rangle_{\xi} \\
&= \eta \left\langle r \sum_{t=1}^{T} \frac{\partial}{\partial \theta_{k,j}} \left(\sum_{k=1}^{K} \sum_{j=1}^{N} \theta_{kj}\, y_{t-1,j}\, \nu_{t,k} \; - \; \log \sum_{l=1}^{K} \exp\left(u_{t,l}\right) \right) \right\rangle_{\xi} \qquad (5) \\
&= \eta \left\langle r \sum_{t=1}^{T} \left[y_{t-1,j}\, \nu_{t,k} - \frac{\exp(u_{t,k})}{\sum_{l=1}^{K} \exp\left(u_{t,l}\right)} \frac{\partial}{\partial \theta_{k,j}} u_{t,k} \right] \right\rangle_{\xi} \qquad (6) \\
&= \eta \left\langle r \sum_{t=1}^{T} y_{t-1,j} \; (\nu_{t,k} - \rho_{t,k}) \right\rangle_{\mathrm{p}(r \,|\, \underline{\boldsymbol{\nu}})\, q(\underline{\boldsymbol{\nu}}\,;\,\boldsymbol{\theta})} . \qquad (7)
\end{aligned}
$$

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 $\boldsymbol{\nu}_t$ 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

A. Solway and M. M. Botvinick. Goal-directed decision making as probabilistic inference: A computational framework and potential neural correlates. *Psychological Review*, 119(1):120–154, 2012. ISSN 1939-1471(Electronic);0033-295X(Print). doi: 10.1037/a0026435.