Robust classification using balanced networks

Julian Büchel jubueche@ethz.ch
aiCTX AG
Thurgauerstrasse 40
Zürich, ZH, Switzerland

Contents

Abstract

In balanced networks, excitatory and inhibitory currents are matched on either a long or short timescale and lead to sparse and irregular firing patterns, often observed in cortical neurons. In Bourdoukan et al. (2012), a pre-defined network architecture is derived from the condition that a neuron only spikes if it contributes to reducing the representation error of some input signal. This precise network architecture leads to a robust, energy efficient and precise signal representation that can be augmented with additional recurrent weights to perform more complex tasks.

While it has been shown that these networks can be used to implement linear (Boerlin et al. (2013)) or even non-linear dynamical systems (Alemi et al. (2017)), no paper (to our best knowledge) has showed a way to train a classifier using efficient balanced networks (EBN) as a foundation.

In this report, we demonstrate how a tightly balanced network can learn to do arbitrary classification tasks of varying complexity, while still keeping the benefits of the balanced network presented in Bourdoukan et al. (2012).

1. Learning in adaptive non-linear control theory

Let us assume an arbitrary dynamical system of the form

$$\dot{x} = f(x) + c(t) \tag{1}$$

where x is a vector of state variables x_j, $f(.)$ is a non-linear function (e.g. $tanh(.)$), and c is a time-dependent input of the same dimensionality of x. Furthermore, let us assume a "student" dynamical system of the form

$$\dot{\hat{x}} = -\lambda\hat{x} + \mathbf{W}^T\psi(\hat{x}) + c(t) + ke \tag{2}$$

where $\hat{x}$ is a vector of state variables $\hat{x}_j$, λ is a leak term, and c is the same time-dependent input as in Equation 1. Over time, the signed error $e = x - \hat{x}$ between the teacher dynamics (Eq. 1) and the student dynamics (Eq. 2) is computed and fed into the student dynamics, causing the student state variables $\hat{x}$ to follow the target variables x more closely. This close tracking enables us to update the weights $\mathbf{W}$ using Eq. 3, so that over the course of learning, the factor k can be reduced to zero and the network follows the teacher dynamics autonomously, using only a weighted sum of basis functions, given by $\psi(x) = \phi(\mathbf{M}\hat{x} + \theta)$, for some non-linear function ϕ, and some random $\mathbf{M}$ and θ. The learning rule used to adapt the weights $\mathbf{W}$ is given by

$$\dot{\mathbf{W}} = \eta\phi(\hat{x})e^T \tag{3}$$

and can be shown to let the weights $\mathbf{W}$ converge towards the optimal weights, denoted $\mathbf{W}^{\text{true}}$, assuming that the input $c(t)$ does not lie on a low-dimensional manifold and that the student system has enough high-dimensional basis functions. For more information and a proof of the above statement, see Alemi et al. (2017). It should be noted that the relation between x and c should be well-defined by an autonomous (non-)linear dynamical system of the form $\dot{x} = f(x)+c(t)$ and that one cannot simply assume a "black box" dynamical system implementing *any* relation of the form $\dot{x} = \mathcal{B}(x, c)$. Concretely, in the light of classification, one cannot simply assume there exists an autonomous dynamical system relating the input c to some target response variable, and that this relation can be learned by the learning rule described above. This observation is important, as it makes it harder to build a classifier given the tools described above. In the next section, we will review how a network of spiking neurons can implement the above learning rule in order to learn the dynamics of a teacher dynamical system.

2. Learning arbitrary dynamical systems in EBN's

In this section, we will briefly recapitulate how an efficient balanced network of spiking neurons can learn to implement any non-linear dynamical system of the form $\dot{x} = f(x)+c(t)$. We will assume that, given a network of N neurons, one can use a decoder $\mathbf{D}$ to reconstruct the target variable from the filtered spike trains of the population using $\hat{x} = \mathbf{D}r$, so that $x \approx \hat{x}$.

Derived from the fact that in an efficient balanced network (Bourdoukan et al. (2012)), a neuron only fires a spike if it contributes to reducing the loss L, given by

$$L = \frac{1}{T}\sum_{t=0}^{T} \|x(t) - \hat{x}(t)\|_2^2 + \mu\|r(t)\|_2^2 + \nu\|r(t)\|_1$$

the membrane potentials in the network are defined by

$$V(t) = \mathbf{D}^T x(t) - D^T D r(t) - \mu r(t) \tag{4}$$

Following Alemi et al. (2017), we differentiate Eq. 4 and substitute $\dot{r}(t) = -\lambda r(t) + o(t)$, $\dot{x}(t) = f(x(t)) + c(t)$ and $\hat{x} = \mathbf{D}r$ to get

$$\begin{aligned}\dot{V}(t) &= -\lambda V(t) + \mathbf{D}^T(f(x(t) + c(t))) - \mathbf{D}^T\mathbf{D}(-\lambda r(t) + o(t)) - \mu(-\lambda r(t) + o(t))\\ &= -\lambda V(t) + \mathbf{D}^T c(t) - (\mathbf{D}^T\mathbf{D} + \mu I)o(t) + \mathbf{D}^T(\lambda\hat{x} + f(x))\end{aligned}$$

We then approximate the term $\lambda\hat{x} + f(x)$ [Not sure here. In the paper it is $\lambda x + f(x)$, but I can't derive it] by a weighted set of basis functions, given by $\mathbf{\Omega^s}\Psi(r)$, where $\Psi(r) = \phi(\mathbf{M}r + \theta)$. We note that in our simulations, we omitted the term $\Psi(r)$ and simply replaced it with the population spike trains $o(t)$.
By feeding the error term $e = x - \hat{x}$ back into the network using the optimal feed-forward weights $\mathbf{D}^T$, we obtain the final network dynamics

$$\dot{V}(t) = -\lambda V(t) + \mathbf{F}c(t) - \mathbf{\Omega^f}o(t) + \mathbf{\Omega^s}o(t) + k\mathbf{D}^T e(t)$$

where the optimal feed-forward weights $\mathbf{F}$ are given by $\mathbf{D}^T$ and the optimal fast recurrent weights by $\mathbf{\Omega^f} = -(\mathbf{D}^T\mathbf{D} + \mu\mathbf{I})$.
Similar to the case in control-theory, the learning rule for the slow recurrent weights is given by

$$\dot{\mathbf{\Omega}}^{\mathbf{s}} = \eta\Psi(r)(\mathbf{D}^T e)^T$$

where we simply replaced $\Psi(r)$ with r.
To speed up the simulations, we assumed that the accumulated updates to $\mathbf{\Omega^s}$, given by $\sum_{t=0}^{T}\eta\Psi(r)(\mathbf{D}^T e)^T$ are approximately the same as accumulating the rates and errors into large arrays of size $N \times T$ and performing the update in a batched fashion, after a whole signal was evaluated rather than after every timestep:

$$\dot{\mathbf{\Omega}}^{\mathbf{s}} = \eta\mathbf{R}(\mathbf{D}^T\mathbf{E})^T$$

3. Training a classifier based on EBN's

The coding properties of efficient balanced networks make them attractive for many applications, especially in the neuromorphic community. However, so far it has only been shown how to implement linear and non-linear dynamical systems of a specific form. In this section we provide a method to train an efficient balanced network to perform classification tasks of varying complexity.
Let us define our time-varying, real-valued input as c_t, where $c \in \mathcal{R}^{N_c \times 1}$ and $t \in \{0..T\}$. The goal of training a classifier is to find a mapping $f(\Theta)$ that maps any input $\mathbf{c}$ to the

desired target variable $\mathbf{y}$, where $\mathbf{y}$ is an indicator variable over time. Concretely, $\mathbf{y}$ is a Gaussian kernel with mean t_{target}, where t_{target} is the time when a classification response of the network is expected. For simplicity, we will consider the case of binary classification. We however note that our method can be easily extended to a multi-class classification task. Considering the ability of the learning rule presented in section 2, one might be be inclined to assume a "black box" dynamical system $\mathcal{B}$ of the form $\dot{y} = f(y) + c(t)$ that, given input $\mathbf{c}$, autonomously produces the desired target $\mathbf{y}$. Two problems come with this approach: 1) The dynamical system is not well-defined, as it is no-longer explicitly defined by a teacher dynamical system. Furthermore, the system is autonomous, as the function $f(.)$ does not depend on the input, but only on past values of the target variable, making it impossible for the system to find a complex relationship between input and target. 2) This approach assumes that the input and target variable have the same number of dimensions, which is almost never the case.
How can we use the above learning rule to train an efficient balanced network to perform arbitrary classification tasks at low metabolic cost, high robustness and good classification performance?
To answer this question, we consider a simple rate network consisting of $\hat{N}$ units, following

$$\tau_j \dot{x}_j = -x_j + \hat{\mathbf{F}}c(t)_j + \hat{\mathbf{\Omega}} f(x(t))_j + b_j + \epsilon_j \tag{5}$$

where τ_j is the time constant of the *j-th* unit, $\hat{\mathbf{F}}$ are feed-forward weights of shape $\hat{N} \times N_c$, $c(t)$ is the N_c dimensional input at time t, $f(.)$ is a non-linear function like $tanh(.)$, $\hat{\mathbf{\Omega}}$ are recurrent weights of shape $\hat{N} \times \hat{N}$ and b_j and ϵ_j are bias and noise terms, respectively. We observe that a rate network can be written in the general form $\dot{x} = \tilde{f}(x) + \tilde{c}(t)$, with $\tilde{f}(x) = 1/\tau(-x + \hat{\mathbf{\Omega}} f x(t)) + \epsilon)$ and $\tilde{c}(t) = 1/\tau(\hat{\mathbf{F}}c(t) + b)$, meaning that an efficient balanced network can be trained to implement the dynamics of *any* given rate network obeying the dynamics of Eq. 5.
Let us now restate the dynamics of the spiking network with adapted notation for ease of understanding:

$$\dot{V}(t) = -\lambda V(t) + \mathbf{F}\tilde{c}(t) - \mathbf{\Omega^f} o(t) + \mathbf{\Omega^s} o(t) + k\mathbf{D}^T e(t)$$

And let us assume that we have trained a rate nework receiving inputs $\mathbf{c}$ to successfully produce a good approximation $\hat{\mathbf{y}} = \hat{\mathbf{D}}\mathbf{x}$ of the target variable $\mathbf{y}$, so that $\hat{\mathbf{y}} \approx \mathbf{y}$.
We now see that we can train a network of spiking neurons to encode the *dynamics* $\mathbf{x}$ of the rate network, by giving the spiking network input $\tilde{c}(t) = 1/\tau(\hat{\mathbf{F}}c(t) + b)$. This makes the *dynamics* of the rate network the new target of the spiking network: $\tilde{\mathbf{x}} = \mathbf{D}r$, so that $\tilde{\mathbf{x}} \approx \mathbf{x}$. After the recurrent weights $\mathbf{\Omega^s}$ have been learned to implement a network that encodes the rate-network dynamics, classification can be performed by the simple computation $y(t) = \hat{\mathbf{D}}\mathbf{D}r(t)$, where $\hat{\mathbf{D}}$ are the rate-network read-out weights, $\mathbf{D}$ are the spiking read-out weights and $r(t)$ are the filtered spike trains at time t of the spiking network.

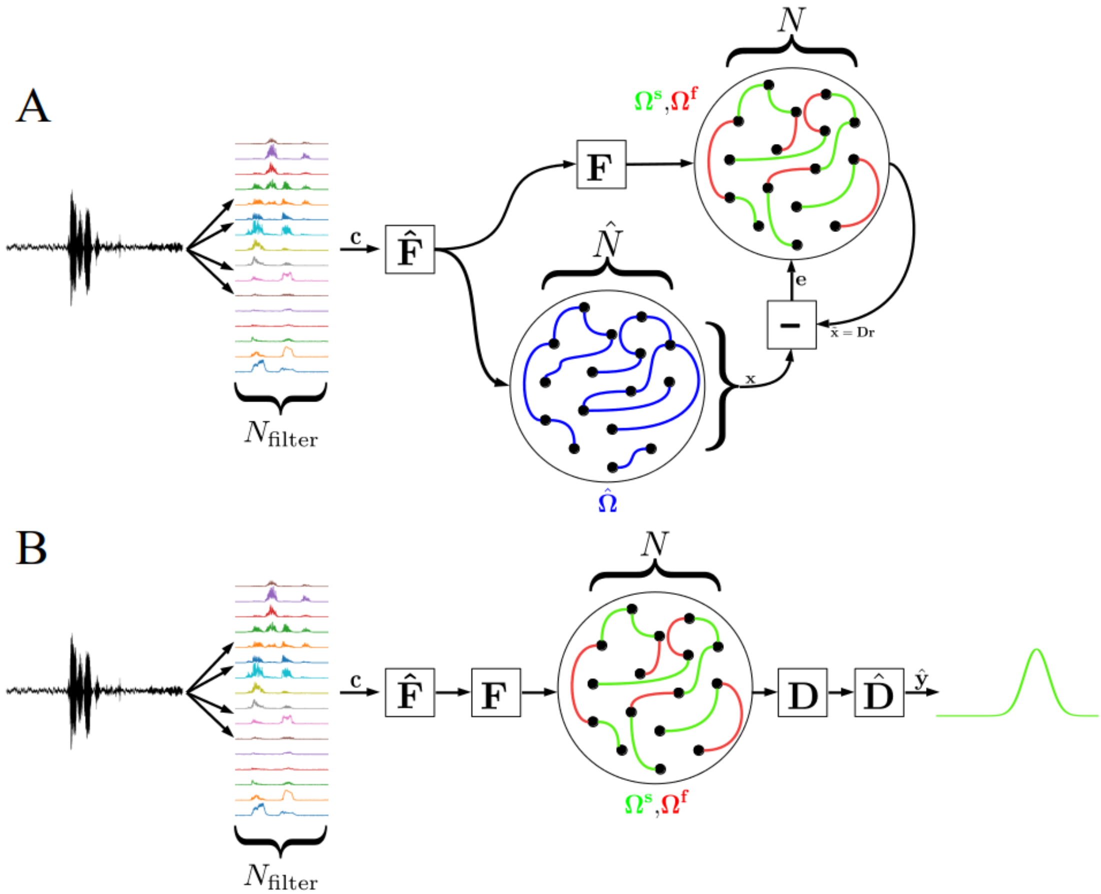

Figure 1: A high-level illustration of the training (**A**) and inference (**B**) process using the presented framework for an auditory classification task. **A:** During training, auditory samples are filtered using a set of band-pass filters with varying cut-off frequencies to produce a 16 channel input, denoted $\mathbf{c}$. This input is then fed into the pre-trained rate network, which does inference on the presented signal. The dynamics $\mathbf{x}$ produced by the rate network are then used to compute the error $\mathbf{e} = \mathbf{x} - \hat{\mathbf{x}}$, which is then fed back into the spiking network using the feed-forward weights $\mathbf{D}^T$. Simultaneously, the spiking network receives the same input as the rate network, simply projected by the feed-forward weights $\mathbf{F} = \mathbf{D}^T$. Using the read-out weights $\mathbf{D}$, it then produces an estimation of the rate network dynamics $\hat{\mathbf{x}}$, which is then used to compute error.
The inference mode, depicted in **B**, is straightforward: The now trained spiking network receives the same input the rate network would receive, but transformed using the feed-forward weights $\mathbf{F}$. It then produces the network dynamics $\tilde{\mathbf{x}} = \mathbf{D}r$ that closely match the ones of the rate network (if it would have received the signal) and produces the output $\tilde{\mathbf{y}} = \hat{\mathbf{D}}\tilde{\mathbf{x}}$. The value $\tilde{\mathbf{y}}$ is then thresholded to obtain the final prediction. encoding the signal with high precision while m.

4. Results

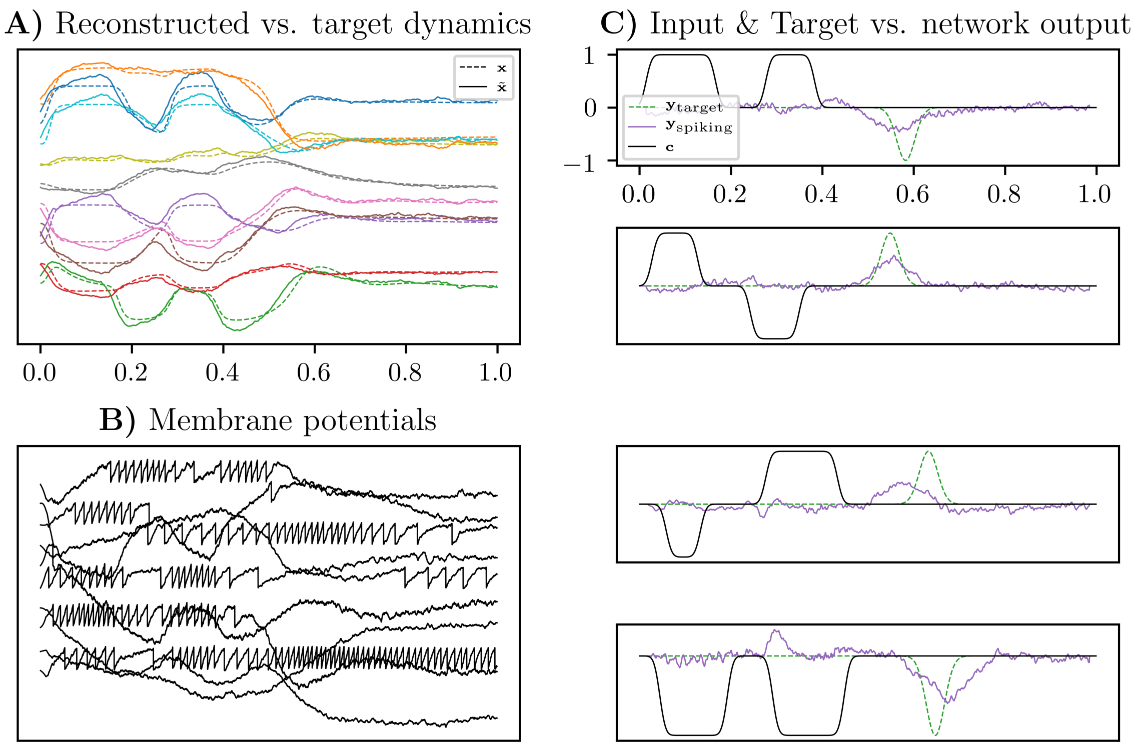

Figure 2: In this experiment, a rate network of 25 neurons was trained to perform temporal XOR on a one-dimensional input. The network was presented with two bumps of magnitude ± 1, indicating True or False. The bumps had a variable length between 100 and 230ms and were separated by a 100ms pause, requiring the network to memorize the first bump for some time. **A:** The spiking network consisting of 300 neurons successfully reconstructs the dynamics of the rate network. **B:** Due to the underlying balanced network, the membrane potentials show irregular and distributed spiking activity. [This is not the case yet, but should follow from the theory] **C:** Our network predicts 98% of 300 unseen samples correctly. Note that although the reconstruction of the dynamics is quite precise, the later obatined response matches the target response quite poorly. This is simply due to the fact that a 25-dimensional rate network was the smallest possible rate network we could train to solve the task, meaning that every dimension of the dynamics is necessary to compute the final output to a high precision. To see that this is actually the reason, consider the second experiment: The reconstruction of the 128-dimensional dynamics is much worse compared to the temporal XOR task, but the final output still matches the actual target quite well.

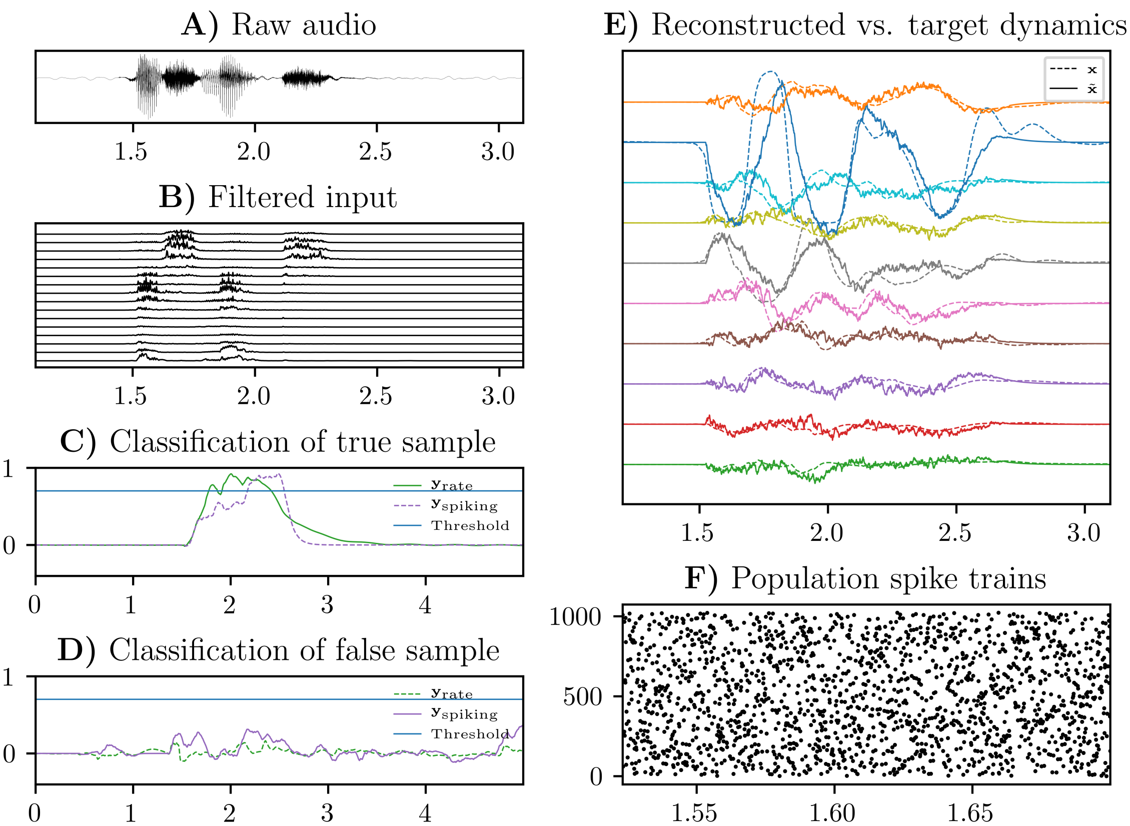

Figure 3: **A:** Audio samples either containing a key phrase, like "Wake up!", or plain talking were presented to the network, which then had to decide whether the input was the target phrase or not. **B:** To make classification easier, 16 butterworth filters with different cutoff frequencies were applied to create a 16-dimensional input signal. **C:** For a sample containing the key phrase, our network output closely follows the output of the rate network and thus makes the correct decision. In this experiment, a threshold at 0.6 determined whether the prediction was positive or negative. **D:** Negative samples on the other hand did not cause a rise in the output variable and were therefore correctly classified as negative samples. For this particular experiment, the rate network achieved 91% on a held out test set containing 300 samples. The spiking network achieved 89.03% test accuracy, while imitating the dynamics of the rate network (**E**). **F:** As expected, the spiking activity of the network is distributed and does not exhibit any output dependent patterns [This is not the case yet, but should follow from the theory]. For this experiment, 2000 neurons were used, but a network of 1000 neurons achieves similar performance.

References

Alireza Alemi, Christian Machens, Sophie Denève, and Jean-Jacques Slotine. Learning arbitrary dynamics in efficient, balanced spiking networks using local plasticity rules, 2017.

M. Boerlin, C. K. Machens, and S. Denève. Predictive coding of dynamical variables in balanced spiking networks. *PLoS Comput. Biol.*, 9(11):e1003258, 2013.

Ralph Bourdoukan, David G. T. Barrett, Christian K. Machens, and Sophie Denève. Learning optimal spike-based representations. In *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'12, pages 2285–2293, USA, 2012. Curran Associates Inc. URL `http://dl.acm.org/citation.cfm?id=2999325.2999390`.