An algorithm for fast pattern recognition with random spikes

U.A. Ernst, D. Rotermund and K.R. Pawelzik

Institute for Theoretical Neurophysics, University of Bremen, D-28359 Bremen, Germany
E-Mail: {udo,davrot,pawelzik}@neuro.uni-bremen.de

Abstract. The human brain classifies natural scenes and recognizes objects in complex visual patterns with a high precision in a minimum amount of processing time. Only few action potentials (spikes) per neuron per processing stage are sufficient to achieve this astonishingly high performance, despite the random nature of the incoming spike trains. In this contribution, we will present a novel algorithm which updates the internal representation of patterns in a generative model with each incoming spike. We first demonstrate that our algorithm is capable of learning a suitable representation of pattern ensembles from stochastically generated spike trains. This representation is then used for classifying test patterns, requiring less than one spike per input node to achieve a performance comparable to standard algorithms in pattern recognition.

1 Introduction

Recently, experimental work has shown that humans can categorize natural scenes within 150 ms after onset of the presentation [9]. At least ten different, hierarchically ordered processing stages (brain areas) are involved in this task. With typical firing frequencies of about 50 Hz, this leaves only time for less than one spike per neuron for a successful processing of the stimuli. Making things even worse, spikes are elicited randomly, their statistics resembling a Poissonian process.

These observations pose a challenge for pattern recognition algorithms which are required to achieve a high performance under restrictive boundary conditions. In our case we have three main restrictions: the recognition process should rely on single spikes, it should be robust against a high degree of noise, and it should require only about one spike per input node until the scene or pattern is recognized. To explain the brain's performance, one has to propose a suitable neuronal algorithm which fulfills all three of these requirements.

Previous work has shown that analog values can be transmitted faithfully with single spikes in a population code [3, 1]. However, this approach requires the allocation of several channels for only one analog value. In a different paradigm devised by Thorpe et al. [10], this excessive usage of resources is overcome by employing a rank order code for spike emission. Images to be represented are decomposed into their principal components, and spikes are transmitted in an

order determined by the strength of the component's coefficients. However, this rank order code can be very sensitive to noise, and the decomposition process is an extensive pre-processing of the images which can take a lot of extra time in the brain.

In this article, we take a different approach by modifying a generative model [5] to work with single, random spikes. The model has originally been used in the context of non-negative matrix factorization. A generative model can be described as a network consisting of input nodes connected to hidden nodes (Fig.1(a)). The connections are interpreted as conditional probabilities to observe an activation in one of the input nodes, given an activation in one of the hidden nodes. The dynamics in a generative model updates the internal representation in the hidden nodes (reconstruction) and/or the conditional probabilities (learning). The goal of this update is to accurately predict the input from the hidden representation. In terms of neuronal information processing, a successful prediction can be interpreted as a correctly perceived stimulus. While in the probabilistic framework of a generative model, connections are *interpreted* in a feedback manner, update algorithms often *use* these connections like synaptic weights in a feed-forward neural network.

In the next sections, we will first present algorithms for batch learning, on-line learning and reconstruction. Second, the algorithms will be applied to a standard benchmark of handwritten digit recognition (USPS). Finally, the results will be shown and discussed in the contexts of machine learning and neuronal networks.

2 Generative Model

The generative model (Fig.1(a)) consists of $s = 1, \ldots, M$ input nodes, $i = 1, \ldots, H$ hidden nodes, and conditional probabilities $p(s|i)$. The K input patterns $v_k(u) \in [-\infty, +\infty]$ are converted to firing rates $r_k(s) \geq 0$, from which spike trains are drawn for all input nodes s. Accordingly, the probabilities for observing a spike in node s are given by $p_k(s) = r_k(s) / \sum_{m=1}^{M} r_k(m)$. The sequence of active input nodes is generated from a Bernoulli process with respect to the probabilities $p_k(s)$. For long spike trains and for each input node, its Binomial distribution converges to the Poisson distribution. Each time t a spike is observed at node s^t, the hidden representation $h(i)$ and/or the $p(s|i)$ are updated. One presentation of a pattern k extends over T input spikes. The input is then fully specified by the sequence vector of temporally ordered indices $\mathbf{s}^T = \{s^1, \ldots, s^t, \ldots, s^T\}$ of the nodes at which those spikes were observed. During update, the goal is to maximize the likelihood $P\left(\mathbf{s}^T | h(i), p(s|i)\right)$ of observing $\mathbf{s}^T$ over the model parameter space $h(i)$ and $p(s|i)$. With $\hat{p}(s) = 1/T \sum_{t=1}^{T} \delta_{s,s^t}$ counting the relative number of spikes at node s in the observation sequence over the time window T, the likelihood is given by

$$P\left(\mathbf{s}^T | \left\{h(i), p(s|i)\right\}_{i=(1,\ldots,H), s=(1,\ldots,M)}\right) = D \Pi_{s=1}^{M} p(s)^{T\hat{p}(s)} \qquad (1)$$

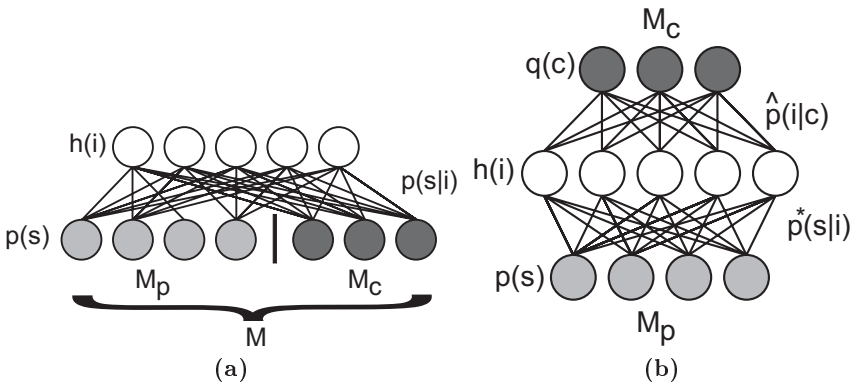

Fig. 1. (a) Spike-by-Spike network comprised of M input and N hidden nodes connected by the conditional probabilities or weights $p(s|i)$. During training of the network, the pattern and its correct classification are presented together to the first M_p, and the remaining M_c input nodes, respectively. (b) Modified Spike-by-Spike network for reconstruction and classification. The M_c weight vectors used for training have been transposed and normalized to form the new weight vectors $\hat{p}(i|c)$ which are used to classify the test patterns.

with $p(s) = \sum_{j=1}^{H} p(s|j)h(j)$ and the combinatorial factor D. For practical reasons, we minimize the negative logarithm of the likelihood,

$$-\log(P/T) = -\log(D/T) - \sum_{s=1}^{M} \hat{p}(s) \log p(s) \; . \tag{2}$$

This minimization problem is penalized by the non-negativity constraints $p(s|i) \geq 0$ and $h(i) \geq 0$ and the normalization constraints $\sum_{s=1}^{M} p(s|i) = 1$ and $\sum_{i=1}^{H} h(i) = 1$, respectively.

To avoid situations in which an input pattern of negative values leads to problems in the spike generation process in the input nodes, the original patterns $v_k(u)$ are pre-processed yielding the input rates $r_k(s)$: in a first step, patterns $v_k(u)$ (with $L = M/2$ components) are corrected by subtracting the individual mean for each pattern $< v_k(u) >^L = 2/M \sum_{u=1}^{L} v_k(u)$,

$$\tilde{v}_k(u) := v_k(u) - < v_k(u) >^L \; . \tag{3}$$

Each of the L components is duplicated and distributed over the even and uneven input node pairs, yielding M non-negative rate components $r_k(s)$ according to the expressions

$$r_k(2u-1) = \begin{cases} +\tilde{v}_k(u) & \text{for } \tilde{v}_k(u) > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

$$r_k(2u) = \begin{cases} 0 & \text{for } \tilde{v}_k(u) > 0 \\ -\tilde{v}_k(u) & \text{otherwise} \end{cases} \; . \tag{5}$$

This pre-processing is motivated by the nature of our brain: the splitting into negative and positive values closely resembles the analysis of visual stimuli by on- and off-cells in the lateral geniculate nucleus (LGN).

3 Algorithms

The likelihood P can be minimized by updating $h(i)$ (reconstruction), or by updating $p(s|i)$ and $h(i)$ together (learning). First, we will consider the $p(s|i)$ as fixed. Within this section, superscripts denote time indices.

Reconstruction. For the reconstruction, we adopt an existing algorithm from Lee and Seung [5] starting from the update equation

$$h^{t+1}(i) = h^t(i) \sum_{s=1}^{M} p(s|i) \frac{\hat{p}(s)}{p^t(s)} \; . \tag{6}$$

In our case, we observe only one spike per time step in input node s^t, and not the whole pattern $\hat{p}(s)$. Thus, we require the algorithm to predict the next spike by first substituting $\hat{p}(s)$ by δ_{s,s^t} in Eq.(6). Because the next spike is almost always not representative for the whole input pattern $\hat{p}(s)$, we apply an additional low-pass filter with time constant ϵ leading to the reconstruction algorithm

$$h^{t+1}(i) = h^t(i) \left[(1-\epsilon) + \epsilon \frac{p(s^t|i)}{p^t(s^t)} \right] \; . \tag{7}$$

Batch Learning. In general, our brain acquires its knowledge and experience over long time scales ranging from hours to years. In contrast, the fast spiking dynamics takes place on a time scale of milliseconds. Therefore, it is reasonable to separate the update time scales of $h(i)$ and $p(s|i)$. While $h(i)$ is changed every time a spike occurs, $p(s|i)$ will be changed only after the presentation of K patterns with T spikes each. For such a batch learning rule, we can apply the corresponding formula of Lee and Seung [5]

$$\tilde{p}^z(s|i) = p^z(s|i) \sum_{k=1}^{K} < h_k(i) >^{\Delta} \hat{p}_k(s) \Big/ \sum_{j=1}^{H} p^z(s|j) < h_k(j) >^{\Delta} \tag{8}$$

$$p^{z+1}(s|i) = \tilde{p}^z(s|i) \Big/ \sum_{u=1}^{M} \tilde{p}^z(u|i) \; , \tag{9}$$

with $< h_k(i) >^{\Delta} = 1/\Delta \sum_{t=T-\Delta}^{T} h_k^t(i)$. Each time a new pattern is presented, the hidden nodes are initialized with $h_k^0(i) = 1/H$.

Online Learning. Eq.(8) has a slight disadvantage, because it requires to remember the final average mean internal states $< h_k(i) >^{\Delta}$ of K pattern presentations for one update step. While a computer has no problems in fulfilling this requirement, the brain could lack the possibility to temporarily store all

$< h_k(i) >^{\Delta}$'s. This limitation can be overcome by deriving an on-line learning rule from scratch, which uses only one pattern at once and takes the form

$$p^{t+1}(s|i) = p^t(s|i) \left\{ 1 + \frac{\gamma h(i)}{1 + \gamma h(i) \frac{p^t(s^t|i)}{p^t(s^t)}} \left[\frac{\delta_{s,s^t}}{p^t(s^t)} - \frac{p^t(s^t|i)}{p^t(s^t)} \right] \right\} . \qquad (10)$$

γ is an update constant which is in general much smaller than ϵ. The derivation of Eq.(10) is based on optimization with Karush-Kuhn-Tucker conditions with positivity constrains [4] and will be explained in detail in [2].

4 Simulations

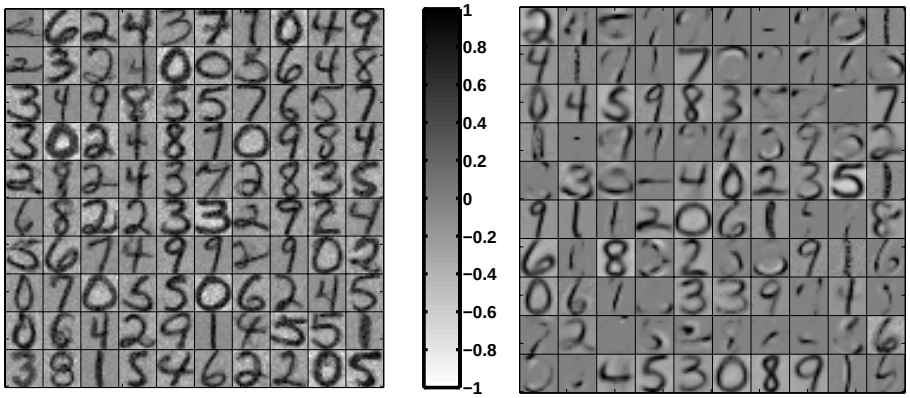

Fig. 2. Conditional probabilities or weight vectors $p^*(s|i)$ for (a) the online learning, and (b) the batch learning algorithm for $H = 100$ hidden nodes. Each vector is displayed in a quadratic raster of 16×16 pixels. The individual vectors i for even and odd input nodes are combined and normalized to a grey value g_u between 1 and -1 by means of the transformation $\tilde{g}_u = p^*(2u-1|i) - p^*(2u|i)$, $g_u = \tilde{g}_u / \max\{|\tilde{g}_u|\}$. Parameters for on-line learning were $w = 0.9$, $\epsilon = 0.9375$, $\gamma = 0.0005$, and $T = 2000$. Parameters for batch learning were $w = 0.5$, $\epsilon = 0.1$, $\Delta = 500$ and $T = 5620$. During on-line learning, all training patterns were presented only once. In contrast, training patterns were presented repeatedly during 20 learning steps in the batch procedure.

Learning and classification. For the learning, M input nodes are divided into M_p pattern nodes and M_c classification nodes. K_{tr} training patterns v_k^{tr} together with their correct classification c_k^{tr} (coded in the firing probabilities in the M_c input nodes; see Fig.1(a)) are presented successively to the network, while $p(s|i)$ and $h(i)$ are updated according to Eqs.(7), (8), (9), and (10).

For the classification run, the network uses only the first M_p input nodes for pattern input. The first part of the weight vectors $p(s|i)$ are re-normalized yield-

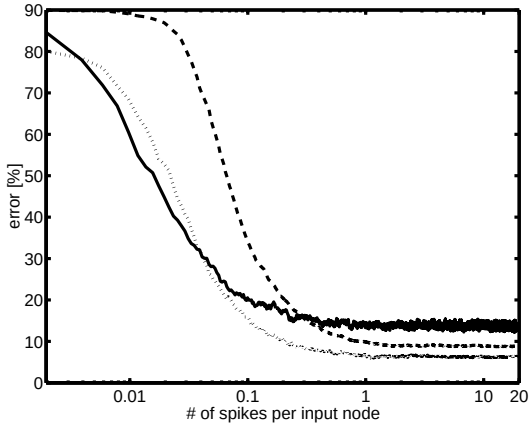

Fig. 3. Mean classification error e for the USPS database shown for the batch learning rule (dashed line) and for the on-line learning rule (solid line), in dependence of the number of spikes per input node. Chance level is at 90 percent. For a comparison, the dotted line shows the classification performance with a standard nearest neighbor algorithm. Parameters for on-line and batch learning were choosen as in Fig.2.

ing the new weights $p^*(s|i) = p(s|i)/\sum_{u=1}^{M_p} p(u|i)$. The remaining M_c weight vectors are transposed and normalized yielding the classification weights $\hat{p}(i|c) := p(c+M_p|i)/\sum_{l=1}^{M_c} p(l+M_p|i)$ for $c = 1, \ldots, M_c$. From $h(i)$ and $\hat{p}(i|c)$, the probabilities $q_k(c) = \sum_{i=1}^{H} \hat{p}(i|c) h_k(i)$ for each of the K_{ts} test patterns v_k^{ts} to belong to the class c are computed, leading to the predicted classification

$$\hat{c}_k = \mathrm{argmax}_c \; q_k(c) \; . \tag{11}$$

The mean classifcation error e over all patterns is then computed by $e = 1/K_{ts} \sum_{k=1}^{K_{ts}} \delta_{c_k^{ts}, \hat{c}_k}$.

Data Base. We subjected our algorithms to the problem of recognizing handwritten digits. The data base from the United States Postal Service (USPS) consists of $K_{tr} = 7291$ training patterns v_k^{tr} and $K_{ts} = 2007$ test patterns v_k^{ts}. Each pattern comprises $M_p/2 = 16 \times 16$ grey scale values (pixels) ranging from -1 (white) to 1 (black). During the test run, the patterns v_k^{ts} were applied according to Eq.(5), leading to input rates r_k^{ts}. During the training run, however, the patterns v_k^{tr} are first normalized and duplicated according to Eq.(5). Together with the correct assignment $c_k^{tr} \in \{0, \ldots, 9\}$ to one digit class, the input rates to all $M = M_p + 10$ nodes s are defined as

$$r_k(s) = w \, r_k^{tr}(s) \Big/ \sum_{u=1}^{M_p} r_k^{tr}(u) \quad \text{for } s \in [1, M_p] \tag{12}$$

$$\text{and} \quad r_k(s) = (1-w) \delta_{s+M_p, c_k^{tr}} \quad \text{otherwise.} \tag{13}$$

The weighting parameter w controls the balance between the pattern and classification inputs.

5 Results

We applied the learning and reconstruction algorithms to the USPS database, varying the parameters ϵ, γ, and w to achieve the best possible performance. In Fig.2, the comparison between the weights shows that the on-line learning rule leads to the formation of digit templates, whereas the batch learning rule in addition extracts typical features common to more than one digit. Consequently, batch learning is slower during the first 0.3 spikes per input node, but achieves a lower classification error in the long run (Fig.3). We also subjected our algorithms to different types of noises in order to test for robustness of learning and classification: first, a varying number of rows or columns in the digit image has been occluded by setting the corresponding pixels to a value of 0. Up to a number of about 6 rows or columns, the classification error nevertheless remains below 20 percent (Fig.4(a)). Second, we superimposed each digit pattern $v_k(u)$ with an image entirely consisting of random pixel values $v_k^{rnd}(u)$ uniformly distributed between -1 and 1. The noise level was varied by means of a parameter $\eta \in [0,1]$ by combining the original pattern and noise as $(1-\eta)v_k(u) + \eta v_k^{rnd}(u)$. Fig.4(b) shows that it requires a fair amount of noise of $\eta \approx 0.35$ to increase the error rate to values above 20 percent.

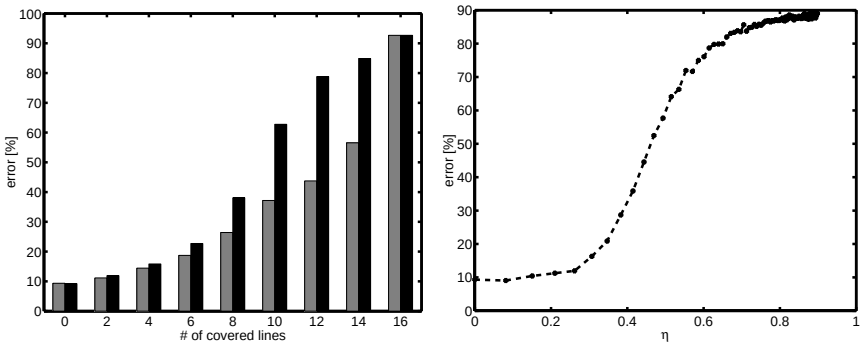

Fig. 4. Minimum classification error in dependence on (a) the number of horizontally (light bars) or vertically occluded (dark bars) lines in the digit images, and (b) in dependence on the amount of noise η on the digit pixels. Parameters for the learning were chosen as in Fig.2.

6 Summary and Discussion

In this contribution, we have developed a framework which can explain the tremendous speed of our brain in processing and categorizing natural stimuli. Our 'Spike-by-Spike'-network is able to classify patterns with less than one spike from each input node, despite the randomness in the information transmission. In addition, we presented two algorithms for on-line and batch learning being capable of finding suitable representations for arbitrary pattern ensembles.

Further development of our algorithms will focus on classifying mixtures of patterns and on non-stationary pattern presentations. Preliminary studies have shown that the Spike-by-Spike algorithm can, under special circumstances, extract the different sources which were superimposed on one input pattern (blind source separation [8]). As an example for non-stationary stimuli, it is possible to learn and to estimate the intended arm movement of a neural prosthesis from the spike data recorded in the motor system of primates [6].

A similar approach for classifying *temporal* patterns from one input channel has been investigated by Wiener and Richmond [11]. They use an iterative Bayesian scheme to successfully re-estimate the presence of a specific time-varying stimulus with each incoming spike.

While the brain is highly modular, our Spike-by-Spike network is only a two-layered system with no hierarchy. Therefore, it remains to show that these networks can be used like logical modules in a computer, grouping arbitrary functional units together in order to realize more complex computations. First results with hand-coded weights are very promising [2], but still a suitable learning algorithm for layered networks has to be found.

References

1. Bethge, M., Rotermund, D. and Pawelzik, K. Optimal short-term population coding: When Fisher information fails. Neural Computation **14(10)** (2002) 2317–2351; and: A second order phase transition in neural rate coding: Binary encoding is optimal for rapid signal transmission. Phys. Rev. Lett. **90** (2003) 088104.
2. Ernst, U.A., Rotermund, D., and Pawelzik, K.R. Fast reconstruction and learning on random spike trains. submitted to Neural Computation.
3. Gerstner, W. Population Dynamics of Spiking Neurons: Fast Transients, Asynchronous States, and Locking. Neural Computation **12** (2000) 43–89.
4. Lanteri, H., Roche, M., and Aime, C.: Penalized maximum likelihood image restoration with positivity constraints: multiplicative algorithms. Inverse Problems **18** (2002) 1397–1419.
5. Lee, D.D., Seung, S.H.: Learning the parts of objects by non-negative matrix factorization. Nature **401** (1999) 788–791.
6. Pawelzik, K.R., Rotermund, D., and Ernst, U.A.: Building representations spike by spike. In: Elsner, N., Zimmermann, H. (eds.): Proceedings of the 29th Göttingen Neurobiology Conference, Georg Thieme Verlag, Stuttgart, (2003) 1041.
7. Pawelzik, K.R., Ernst, U.A., Trenner, D., and Rotermund, D. Building representations spike by spike. In: Proceedings of the Society of Neuroscience Conference 2002, Orlando, (2002) 557.12.
8. Laskov, P., Ziehe, A., Müller, K.R, Pawelzik, K. Non-Negativity and Sparseness in Neural Computation. in preparation.
9. Thorpe, S., Fize, D., and Marlot, C. Speed of processing in the human visual system. Nature **381** (1996) 520–522.
10. Thorpe, S., Delorme, A., van Rullen, R. Spike-based strategies for rapid processing. Neural Networks **14** (2001) 521–525.
11. Wiener, M., and Richmond, B.J.: Decoding spike trains instant by instant using order statistics and the mixture-of-Poissons model. J. Neurosci. **23** (2003) 2394–2406.