Learning Conditional Mappings between Population-Coded Modalities

Fabian Schrodt and Martin V. Butz

Cognitive Modeling, Department of Computer Science,
University of Tübingen, Germany
{tobias-fabian.schrodt,martin.butz}@uni-tuebingen.de

Abstract. It is still an open question how the brain manages to map various modalities onto each other. We introduce a tripartite neural network architecture that is able to learn non-linear mappings between topological, population-encoded modalities. The neural network gains this capability by creating sparse modal correlation maps. By applying factorization, the correlation maps serve as mutually conditional transformations onto a third modality. We show that such a combination is able to solve the locally linear task of learning forward velocity kinematics of a simple arm. In comparison to other approaches, the architecture is robust and predictable in terms of learning performance and efficient in terms of model complexity. The model is neurally plausible, mimicking coordinate transformations known to be computed in parietal cortex, and may serve as a basic building block to model non-linear mappings between population-encoded modalities, which are typically grounded in different frames of reference.

Keywords: Gain-Field Encodings, Population Coding, Factorization, Sparse Coding, Non-Linear Mappings, Neural Networks

1 Introduction

The brain is able to combine information of various modalities and frames of reference to learn predictive models about its body and its interaction with the environment. Typically, sensory input can be assumed to be encoded by populations of locally receptive cells with tunings to specific stimulus characteristics [1]. To eventually establish a body model, conditionals of multiple population-coded stimuli have to be learned to resolve non-linearities and minimize surprise about action outcomes. In parietal cortex, mappings between population-encoded modalities have been suggested to be established by means of gain-fields [2, 3].

One example of a tripartite mapping problem is learning a combined model of kinematic and dynamic bodily control: Given the proprioception of a limb, what effect does a change in its posture (or a motor command) have in terms of visually perceived motion? In this question, two modal sources of information have to be mapped onto a third one, resulting in a locally linear, conditional mapping

problem. A neurocognitively plausible model may learn such a forward model by means of supervised training because the outcomes of actions are directly observable [4]. Once a forward model has been learned, it is possible to derive an inverse mapping, which is typically ambiguous due to redundancies, that is, extra degrees of freedom, in the system. Population codes have the potential to resolve resulting uncertainties [5]. For many degrees of freedom, however, classical gain-field models are not applicable due to the resulting huge network sizes, diminishing their cognitive plausibility when no modularization is applied [6]. We asked the question how such mappings can be learned considering the conditional, tripartite nature of this problem (and others) while keeping the computational effort minimal.

In the following, we describe the Conditional Mapping Architecture (CMA) and show that it is able to learn a combined model of forward dynamics and kinematic control. The architecture relies on pair-wise multiplications in modal modules instead of tensor products and is thus easily differentiable and computationally more efficient than related approaches inspired by gain-field encodings. In subsequent experiments, we show that the model features fast and predictable learning performance that is adjustable via a single topological parameter.

2 Conditional Mapping Architecture

CMA is a tripartite architecture sketched in Fig. 1. Two of the parts serve as input to the network, each fed by data of a different modality. The third part is defined as the output modality. We assume that there is a non-linear, but distinct dependency of the output modality on the inputs. That said, the output is determined by a many-to-one mapping from the first input modality to the output modality, under the condition of the second input modality – or vice versa.

2.1 Population Coding

Each modal information is assumed to be limited to a hyperrectangular range in $\mathbb{R}^{D^m}$, $m \in \{1,2,3\}$[1]. Each component or dimension $d^m \in \{1..D^m\}$ in a modal space is represented separately by a population of N neurons[2], providing regularly distributed tuning prototypes in the range of this component. Then, given a modal input sensation $s^{m,d^m} \in [s^{m,d^m}_{\max}, s^{m,d^m}_{\min}]$, $m \in \{1,2\}$, each neuron $i \in \{1..N\}$ in an input population responds in the form of a unit Gaussian radial basis function defined by:

$$a_i^{m,d^m} = \exp\left(-1/2 \left(\frac{p_i^{m,d^m} - s^{m,d^m}}{b^{m,d^m}}\right)^2\right), \; m \in \{1,2\}\;, \tag{1}$$

[1] Superscripts of scalars or vectors denote descriptors, while subscripts denote indices (with the exception of 'min' and 'max').

[2] We assume that each modal component is represented by the same number of neurons for reasons of simplicity.

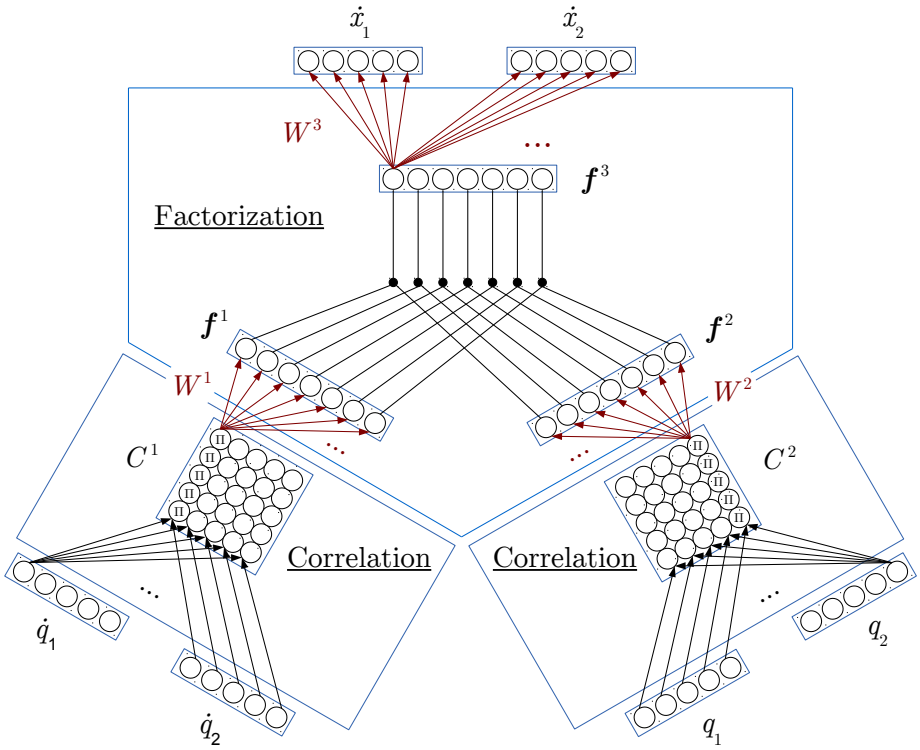

Fig. 1. Connection scheme of the tripartite mapping architecture applied to learning a forward model of endeffector motion $\dot{\boldsymbol{x}}$, given angles $\boldsymbol{q}$ and angular velocities $\dot{\boldsymbol{q}}$. Free parameters are shown in red.

where p_i^{m,d^m} is the prototype in the range of the modal component and b^{m,d^m} is the breadth of the tuning of neurons in the population representing information in dimension d^m of modality m. The breadth depends on an a-priori range of the modal component:

$$b^{m,d^m} = \frac{s_{\max}^{m,d^m} - s_{\min}^{m,d^m}}{2N}, \; m \in \{1,2,3\} \; . \tag{2}$$

For output populations ($m = 3$ in the following), each neuron $i \in \{1..N\}$ responds linearly to upstreamed signals, resulting in activities a_i^{m,d^m} (see Eq. 7). Given a target sensation or observation $s^{m,d^m} \in [s_{\max}^{m,d^m}, s_{\min}^{m,d^m}]$, an error term δ_i^{m,d^m} can be calculated suitable for backpropagation:

$$\delta_i^{m,d^m} = t_i^{m,d^m} - a_i^{m,d^m}, \; m = 3 \tag{3}$$

where target activities t_i^{m,d^m} are determined analogously to Equation 1.

2.2 Correlating Modalities

To build a processing architecture that is able to learn conditional, tripartite mappings, we at first employ a *correlation map* between all dimensions of each input modality. This correlation map is realized by all *pair-wise products* of single activations of cells in unequal populations, resulting in $N^2 \cdot D^m \cdot (D^m - 1)/2$ multiplications. Thus, the activation $c_{i,j}^{m,d^m,e^m}$ of a multiplicative unit correlating activations i and j of dimension d^m and e^m of modality m is given by

$$c_{i,j}^{m,d^m,e^m} = a_i^{m,d^m} \cdot a_j^{m,e^m} \ , \tag{4}$$

where $d^m \neq e^m \in \{1..D^m\}$, $i, j \in \{1..N\}$. This is similar but not equivalent to tensor products often used in gain-field models, because our approach does not require D^m-fold multiplications, which reduces the number of parameters drastically. For $D^m = 2$, this approach is equivalent to the outer product of population activity vectors. Note that only the two input modalities have to be correlated independently.

2.3 Factorization

The resulting correlation maps in our architecture can each be considered a 2D image C^m with width $N \cdot D^m \cdot (D^m - 1)/2$ and height N, reflecting all possible, pair-wise logical AND interactions. By nature, image processing methods are applicable – the aim is to find a suitable and yet sparse connectivity that is able to realize image transformations systematically, whereas one modality can be considered the input image, and the other can be considered a *warp*. Allowing all possible mappings from the two correlation maps to the output modality would result in a three-way interaction tensor with cubic number of parameters. However, it has been shown that *factorization* can reduce the number of parameters drastically, given that local regularities in the mapping exist [7].

Factorization is realized by learning a triplet of linear transformations W^m: Two of them factorize given input images each onto a modal factor vector $\boldsymbol{f}^m \in \mathbb{R}^F$, and the third transforms the *elementwise multiplication* (denoted $\odot$) of the modal factors to the output image. The factors are represented by linear hidden units. This yields the factor cell activations

$$\boldsymbol{f}^m = W^m \cdot C^m, m \in \{1, 2\} \tag{5}$$

$$\boldsymbol{f}^m = \boldsymbol{f}^1 \odot \boldsymbol{f}^2, \ m = 3 \ . \tag{6}$$

The transformation matrices W^m are the only free parameters of the architecture, resulting in $F \cdot N \cdot \big(N \cdot D^1 \cdot (D^1 - 1)/2 + N \cdot D^2 \cdot (D^2 - 1)/2 + D^3\big)$ parameters overall. Finally, $\boldsymbol{f}^3$ is transformed to the concatenated output activations, effectively including a projection of the combined correlation maps:

$$(\boldsymbol{a}^{m,1}, \dots, \boldsymbol{a}^{m,D^m}) = W^m \cdot \boldsymbol{f}^m, \ m = 3 \ . \tag{7}$$

Training the architecture is possible by means of error gradient descent by backpropagation, since all parameters are differentiable. In the next section, we evaluate the architecture.

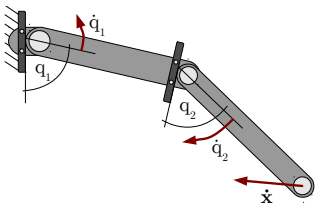

Fig. 2. Simulation of a 2-DOF arm with angular constraints.

3 Evaluation Setup

We evaluate the architecture on the task of approximating the forward velocity kinematics of a simple 2 DOF arm with two limbs with unit length and an endeffector in a 2D positional space, resulting in $D^m = 2\ \forall m$. A sketch of the arm is shown in Fig. 2. The learning objective is the prediction of endeffector velocities $\dot{\boldsymbol{x}} \stackrel{!}{=} \boldsymbol{s}^3$, given angular velocities $\dot{\boldsymbol{q}} = \boldsymbol{s}^1$ under the condition of angular postures $\boldsymbol{q} = \boldsymbol{s}^2$. Given the condition, this mapping is onto and locally linear.

All modal components were represented each by $N = 10$ population neurons. For evaluation, three Conditional Mapping Architectures with different number of parameters were compared to three Multilayer Perceptrons (MLP) with three hidden layers (HL) consisting of neurons with hyperbolic tangent activation functions and biases, resulting in approximately the same number of free parameters (FP):

- **CMA-16**: $F = 16$ factor units $\Rightarrow$ 3520 FP. Learning rate 0.001.
- **MLP-30**: 30 neurons per HL $\Rightarrow$ 3690 FP. Learning rate 0.001.
- **CMA-49**: $F = 49$ factor units $\Rightarrow$ 10780 FP. Learning rate 0.004.
- **MLP-60**: 60 neurons per HL $\Rightarrow$ 10980 FP. Learning rate 0.002.
- **CMA-99**: $F = 99$ factor units $\Rightarrow$ 21780 FP. Learning rate 0.008.
- **MLP-90**: 90 neurons per HL $\Rightarrow$ 21870 FP. Learning rate 0.003.

Learning rates were optimized heuristically and separately for each network, all momenta were set to 0.8. Small learning rates are required for learning this prediction online to avoid catastrophic inference and forgetting of already learned, locally linear mappings [8]. Fig. 1 shows a CMA architecture example for $N = 4$ and $F = 7$ applied to the above learning task. A classical computational gain-field as originally described by Zipser and Andersen [9] would have 200k free parameters in this configuration. It would however not be possible to train it for higher-dimensional problems.

For sampling the training data, we set goal postures $\boldsymbol{q}^g$ randomly distributed in posture space and moved the arm according to the normalized postural distance

$$\dot{\boldsymbol{q}}(t) = \frac{\boldsymbol{q}^g(t) - \boldsymbol{q}(t)}{||\boldsymbol{q}^g(t) - \boldsymbol{q}(t)||} \cdot \mathbb{U}^2(0.1) \tag{8}$$

where $\mathbb{U}^2(0.1)$ is a 2-dimensional, uniformly distributed random variable in the interval $[0, 0.1]$ rad. By adding noise to each angular movement, we could enforce

local variance both in the direction and velocity of the movement of the endeffector. By targeting goal postures randomly, we could ensure that the whole postural space was uniformly sampled throughout the training procedure. As soon as the Euclidean distance $||\boldsymbol{q}^g(t) - \boldsymbol{q}(t)||$ between the current posture vector and the goal posture vector fell below 0.2, a new goal posture was set. Goal postures were restricted by the angular constraint $q_i \in [0, \pi]$. The resulting modal ranges for angular and endeffector velocities were set accordingly.

We trained each network for 10^7 time steps in 6 independent trials, including normally distributed parameter initialization with mean 0 and variance 0.1 with different random seeds. After training, we averaged the performance in a 50k time steps evaluation phase without parameter adaptation to test for local overadaptation to the target mapping. The results are shown and interpreted in the following.

4 Results

The output components $\boldsymbol{a}^{3,d^3}$ provided by a network were used to decode the predicted directional endeffector velocity $\dot{\boldsymbol{x}}$ by polling their prototypes and weighting them according to their activation for each modal output component d^m, yielding:

$$\dot{x}^{d^m} = \frac{\sum_{i=1}^{N} x_i^{m,d^m} \cdot a_i^{m,d^m}}{\sum_{i=1}^{N} a_i^{m,d^m}},\ d^m \in \{1..D^m\},\ m = 3 \tag{9}$$

Then, the prediction $\dot{\boldsymbol{x}}$ was compared to the actual observation $\boldsymbol{s}^3$. To evaluate the networks' performances, we derived two measuring units from this comparison: First, we compared the *angular error* in the direction of the predicted endeffector motion, defined by

$$\Delta\alpha = \operatorname{acos}\left(\frac{\dot{\boldsymbol{x}} \bullet \boldsymbol{s}^3}{||\dot{\boldsymbol{x}}|| \cdot ||\boldsymbol{s}^3||}\right) \cdot \frac{180}{\pi} \tag{10}$$

where $\bullet$ is the scalar product. Secondly, we defined an exponential *velocity error*

$$\Delta v = \left|\log\left(\frac{||\dot{\boldsymbol{x}}||}{||\boldsymbol{s}^3||}\right)\right| \tag{11}$$

The development of these two error measures over time in comparison of the different network types is shown in Fig. 3(a) and Fig. 3(b).

At first, it has to be noted that finding a working MLP configuration for the task at hand was not trivial. Since it can be considered a highly non-linear classification problem, at least three hidden layers (resulting in 5 consecutive linear transformations) were required for decent performance. We achieved our best results using hidden layers of equal size. All MLPs had a rather slow learning progress early on: Training began to have a noticeable effect only after about 80k time steps. At this time, CMAs already reached an acceptable performance

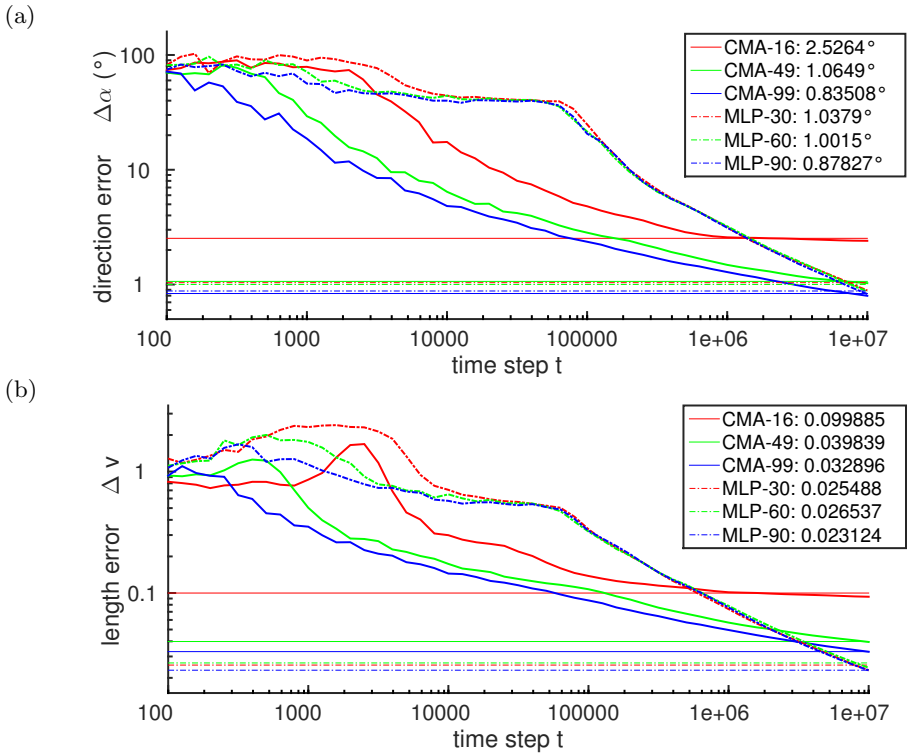

Fig. 3. Log-log plots of the deviation of (a) the predicted endeffector movement direction from the actual movement direction in degree and (b) the predicted endeffector movement speed from the actual movement direction in exponential relation. Horizontal lines represent the average error levels while testing for local overfitting, which is also indicated in the legends, respectively.

e.g. with angular errors between 2.5 and 5. However, upon reaching a specific leverage point, the errors decreased about dual-logarithmically. When increasing the learning rate, MLPs tended to strong overadaptation to the locally linear forward dynamics, resulting in worse performance after all. Interestingly, as the results show, increasing the number of hidden neurons did not improve the convergence rate nor final performance significantly. MLPs with larger hidden layers however seemed slightly less prone to overfitting.

In comparison, CMAs can especially be characterized by superior rate of learning progress during the early training period. This suggests that the solution spaces of CMAs are more convex than for MLPs, such that they were able to outperform MLPs at least for a finite time span. Also, CMAs did not tend to overfit that much when increasing the learning rate. Instead of using a hierarchy of non-linear classifiers, CMA relies on pair-wise multiplications of (linear) factors only, making them easy and robust to train. In contrast to MLP, CMA's performance increases with F as the only topological parameter of the architecture. Thus, using CMAs seems suitable to quickly find an approximate solution

without challenging topological parameters. Considering the results, though, we cannot draw the conclusion that CMAs were able to outperform MLPs in terms of the attainable error limit.

5 Conclusions

In the course of this ongoing investigation, we introduced an architecture that is able to learn online an approximation of population-coded forward dynamics in about 80k time steps with accuracy. Transforming pair-wise correlation maps of two modal input populations is sufficient to solve this problem. In contrast to conservative gain-fields, our approach is potentially able to handle higher-dimensional problems (e.g. 7 DOF forward dynamics) with a reasonable number of parameters. The applied factorization can also be considered a fusion of a pair of two-dimensional gain-field mappings onto a third modality. Just as hierarchical gain-fields can be used to reduce the number of parameters [6], applying hierarchical factorizations of correlation maps could thus result in further reduction of the number of parameters, particularly when applied to higher-dimensional problems. Specifically relating to the task of learning forward dynamics, adding recurrences from angular changes to angles might provide the network with further capabilities. In future work, also opportunities to resolve the inverse dynamics could be investigated based on this architecture. Moreover, the factorization structure should be analyzed further, in which case we expect to uncover dimensional properties of the outside environment.

References

1. Pouget, A., Dayan, P., Zemel, R.: Information processing with population codes. Nature Reviews Neuroscience **1** (2000) 125–132
2. Andersen, R.A., Essick, G.K., Siegel, R.M.: Encoding of spatial location by posterior parietal neurons. Science **230** (1985) 456–458
3. Salinas, E., Sejnowski, T.J.: Correlated neuronal activity and the flow of neural information. Nature reviews neuroscience **2** (2001) 539–550
4. Jordan, M.I., Rumelhart, D.E.: Forward models: Supervised learning with a distal teacher. Cognitive science **16** (1992) 307–354
5. Doya, K.: Bayesian brain: Probabilistic approaches to neural coding. MIT press (2007)
6. Kneissler, J., Butz, M.V.: Learning spatial transformations using structured gain-field networks. In: Artificial Neural Networks and Machine Learning–ICANN 2014. Springer (2014) 683–690
7. Memisevic, R., Hinton, G.E.: Learning to represent spatial transformations with factored higher-order boltzmann machines. Neural Computation **22** (2010) 1473–1492
8. McCloskey, M., Cohen, N.J.: Catastrophic interference in connectionist networks: The sequential learning problem. Psychology of learning and motivation **24** (1989) 109–165
9. Zipser, D., Andersen, R.A.: A back-propagation programmed network that simulates response properties of a subset of posterior parietal neurons. Nature **331** (1988) 679–684