Efficient online bootstrapping of sensory representations

Alexander Gepperth

École Nationale Superieure de Techniques Avancées, 858 Blvd des Maréchaux, 91762 Palaiseau, France
INRIA FLOWERS

Abstract

This is a simulation-based contribution exploring a novel approach to the open-ended formation of multimodal representations in autonomous agents. In particular, we address the issue of transferring ("bootstrapping") feature selectivities between two modalities, from a previously learned or innate *reference representation* to a new *induced representation*. We demonstrate the potential of this algorithm by several experiments with synthetic inputs modeled after a robotics scenario where multimodal object representations are "bootstrapped" from a (reference) representation of object affordances. We focus on typical challenges in autonomous agents: absence of human supervision, changing environment statistics and limited computing power. We propose an autonomous and local neural learning algorithm termed PROPRE (projection-prediction) that updates induced representations based on *predictability*: competitive advantages are given to those feature-sensitive elements that are inferable from activities in the reference representation. PROPRE implements a bi-directional interaction of clustering ("projection") and inference ("prediction"), the key ingredient being an efficient online measure of predictability controlling learning in the projection step. We show that the proposed method is computationally efficient and stable, and that the multimodal transfer of feature selectivity is successful and robust under resource constraints. Furthermore, we successfully demonstrate robustness to noisy reference representations, non-stationary input statistics and uninformative inputs.

Email address: alexander.gepperth@ensta-paristech.fr (Alexander Gepperth)

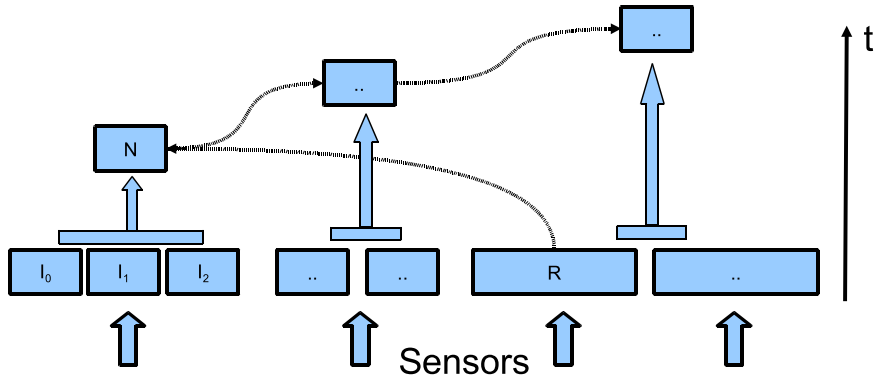

Figure 1: Bootstrapping of new representations in autonomous agents as a long-term goal of our work. New representations are formed based on the combination of sensory signals and may, once fully formed, in turn control the bootstrapping of even more complex representations by an identical learning process. We envisage this to be an open-ended and largely unsupervised process, leading to internal representations of ever increasing specialization and usefulness to the agent. The labeled boxes indicate the work covered in this article: I_1, I_2, I_3 stand for input representations derived from sensors, N indicates the newly formed induced representation, and R stands for the reference representation that is derived from another sensory modality, controlling the bootstrapping process.

1. Introduction

The autonomous formation of representations is a currently very active research topic in developmental robotics[1, 2, 3, 4]. In this contribution, we investigate how simple, task-relevant *reference representations* may induce the formation of more detailed representations by a learning algorithm termed PROPRE (short for "projection-prediction"). Typical reference representations in biology are the innate selectivity for faces[5], dangerous forms[6] and probably odors[7] in newborn primates. Considering autonomous robotic agents, simple "innate" reference representations may be derived from, e.g., proprioception ("have I moved?"), proximity sensors ("collision imminent") or affordances ("does this object allow a certain manipulation/action?"). The corresponding *induced representations* may be fed from different sensors, such as laser-range sensors, 3D sensing or RGB cameras. From such high-dimensional data, an agent may form complementary, more detailed representations which it needs to successfully perform in its environment. We imagine this process to be open-ended, forming new representations of

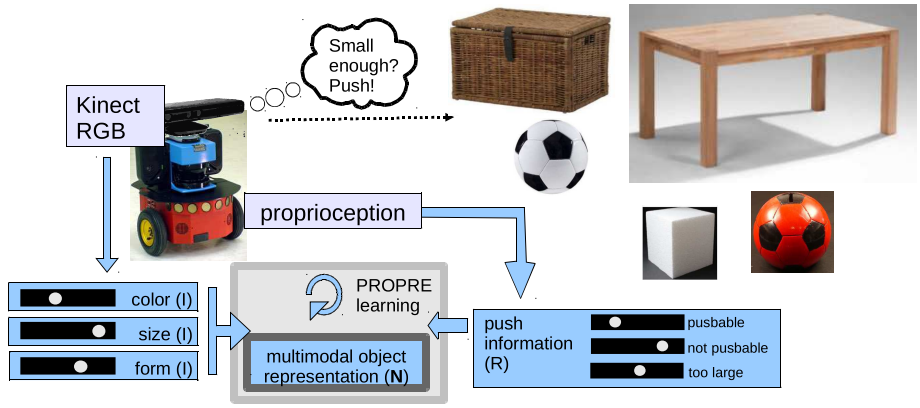

Figure 2: Illustration of the robotic model task, based on the mobile robot currently in use at our lab. Based on a Kinect sensor, we obtain a rough segmentation of objects which is the basis for the subsequent determination of simple object properties. Each detected object segment is approached and an attempt to push it is made. Based on proprioceptive information, it can be judged whether the attempt is successful or not. This information, or the information that no attempt to push is currently being made, is entered into the reference representation R on the right-hand side.

ever increased specialization and complexity, according to the agent's needs, as depicted in Fig. 1. In this contribution, we investigate how a single step in such a bootstrapping process may come about. In particular, we focus on online learning under typical constraints faced by autonomous agents, such as the absence of human supervision, the limited computation and memory capacities of embedded hardware and the variability of environment statistics.

1.1. Robotic model task

Consider a mobile robot in an indoor scenario, equipped with a mechanism to unspecifically detect objects and analyze their appearance along simple visual dimensions: color and form from an RGB camera with suitable processing, size from a 3D sensor[8]. That such simple representations of object properties can indeed facilitate significant discrimination capabilities has been shown many times, e.g., in [9, 10]. We equip our imagined robot with two innate basic behaviors: a drive to approach small objects, trying to push them gently, and a drive to avoid large objects (walls, cupboards, ..) because they cannot be pushed anyway. Furthermore, the robot should be capable of detecting whether the attempt to push an object is successful or

not.

As we know from experience, the "pushable" property is usually correlated with appearance and geometry, i.e., one can push balls or styrofoam cubes, but not bricks or table legs.

In such an environment, a robot endowed with a suitable learning algorithm is expected to *transfer* the simple non-visual information about the success of the current attempt to push ("reference representation") to a corresponding "induced representation" of combined visual object properties that allows to take "approach or avoid"-decisions more reliably, and most importantly, in advance. As resources in a mobile agent are usually severely constrained, it is out of the question to represent all possible combinations of basic visual properties; there needs to be guidance as to which combinations to represent and which to suppress. The whole process should be unsupervised to minimize the amount of human effort. As a last point, the induced representation should be continuously adapted to changing environment statistics, since an unfriendly human might introduce new objects from time to time.

The synthetic input data for the simulation experiments described in this article are modeled after this robotic model task, which we also aim to implement on our mobile robotic platform[8].

1.2. Related work

With the proposed PROPRE algorithm, we focus on predictability as a guiding principle for learning. This was motivated by a conceptual work [11], arguing that symbolic quantities should be diverse on the one hand, and on the other hand be defined by their *power to predict* other quantities. Our work differs from [11] in that we focus on quantities that *can be predicted*, and in that we propose and evaluate a concrete algorithm. We focus on predictability because we find that it can be naturally incorporated into local learning algorithms, whereas using predictive power necessarily involves bi-directional non-local operations.

The successful learning of relations between simple object properties and object identity has been described, e.g., in [9], although object-identity relations were direct links from a single property to one object. The presented work goes beyond [9] in that it could represent higher-order correlations between object identity and object properties because an intermediate representation (the induced representation N) is created that represents specific combinations of simple properties.

Conceptually very close to our PROPRE approach is the *predictive coding* model originally proposed by [12] and elaborated by, e.g., [13, 14]. As in our PROPRE algorithm, predictive coding implements bi-directional learning between a receptive-field-generating process and a prediction process where receptive field generation is influenced by predictability. However there are important differences: Most basically, predictive coding was originally proposed to model observed single-neuron data, whereas PROPRE is intended for online use in robotic agents which implies a certain ease-of-use (no pre-whitening, no stability controls on learning), computational efficiency and robustness to noise. Furthermore, predictions in the predictive coding model always arise from higher hierarchy levels of the *same* processing stream, whereas PROPRE allows multimodality as it imposes no constraints on reference representations. The price for this flexibility is that, in contrast to predictive coding, PROPRE is not rigorously derived from a probabilistic model. Lastly, predictive coding is intrinsically tied to the notion of stackable hierarchies; it was shown to work at least for two-level hierarchies[12] but works (in theory) for arbitrarily deep architectures, whereas the issue of deep network building with PROPRE is not (yet) addressed in this article.

1.3. The PROPRE algorithm

For transferring simple selectivities from *reference representations* to more powerful and discriminative *induced representations* using a limited number of feature-sensitive model neurons as basic representational elements, we propose the PROPRE (projection-prediction) learning algorithm. This algorithm is guided by diversity and *predictability* (see also [11]), resulting in selectivity to distinct patterns whose presence can be inferred from activity in the reference representation[1]. Conversely, patterns that can not be inferred in this way will be under-represented or even disregarded, thus making the best use of limited representational capacity. As can be seen from Fig. 4, PROPRE uses concurrent *projection* and *prediction* algorithms to *project* the possibly high-dimensional input representation I onto the induced representation N, as well as to *predict* N based on the reference representation R. Learning in projection and prediction algorithms is mutually dependent: whereas prediction ($R \rightarrow N$) performs supervised learning based on the current induced representation, the projection algorithm ($I \rightarrow N$) is modulated

[1]The term "prediction" is meant to indicate "meaningful inference", irrespective of time.

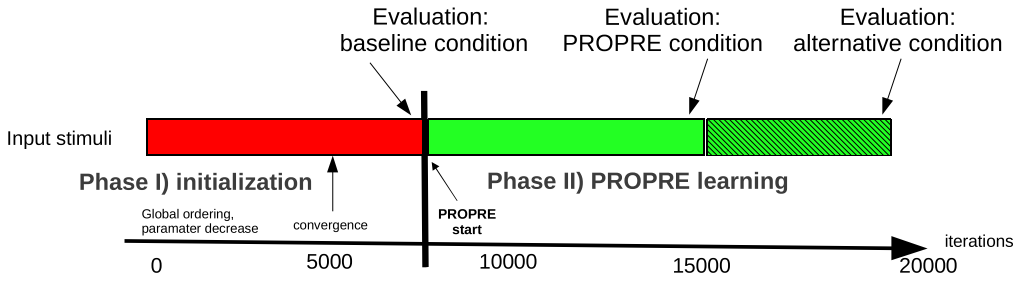

Figure 3: Time course of PROPRE learning. In phase I, PROPRE learning is not fully applied: all input samples contribute to the formation of the induced representation, therefore the induced representation is formed by a pure SOM algorithm. In phase II, PROPRE learning is enabled: the current pattern contributes to learning only if activity in the induced representation is predictable enough. In order to demonstrate PROPRE's robustness, we change input statistics slightly in the middle of phase II, indicated by the crosshatch. Time points of performance evaluations (see text) are given on the upper side of the diagram: at the beginning of PROPRE learning, before and after the changes in input statistics.

by a measure of predictability derived from the prediction algorithm, in a way to favor the "growth" of predictable elements in N.

2. Methods

In the whole article, we will work with two-dimensional distributions on a discrete grid, denoted "representations", and we will express "neural activity" in an arbitrary representation X by $z^X(\vec{x}, t)$. Synaptic connections between positions $\vec{x}$ and $\vec{y}$ in representations X and Y are denoted by $w_{\vec{x}\vec{y}}^{X-Y}$.

2.1. Time course of PROPRE learning

PROPRE learning consists of numerous iterations which are executed sequentially during operation of the embedding system (online learning), be it real or simulated as in this contribution. However, although PROPRE learning is a fully online operation, it requires a suitable initialization from input stimuli, i.e, an induced representation where preliminary selectivities have already been formed (see Sec. 4 for a discussion of this fact). Such an initialization is required to compute a central quantity in the PRORPE algorithm, the *update indicator* $\lambda(t)$, which governs learning for the projection step

$I \to N$. $\lambda(t)$ basically expresses whether the current induced representation N can be statistically predicted from the current reference representation R.

Therefore, training is conducted in two phases (see also Fig. 3): in phase I (initialization), the update indicator is kept at $\lambda(t) \equiv 1$ which causes all inputs to be considered in the updating of the projection algorithm, leading to a purely input-driven representation of stimuli in the induced representation N.

In phase II, the update indicator $\lambda(t)$ is computed from the results of the prediction step $R \to N$, and its value is used to decide whether the projection step should be adapted using the current input I. This leads to a preferred representation of predictable stimuli in N. Performance evaluations are taken at three distinct times during phase II: first, at its very start when the projection is fully initialized but PROPRE learning has not yet started (*baseline condition*). Second, in the approximate middle of phase II (*PROPRE condition*) and third, at the very end of the phase II. This is done in order to compare the effect of PROPRE learning to a purely input-driven approach (baseline condition -vs- PROPRE condition), as well as to demonstrate PROPRE's ability to adapt to changing input statistics (PROPRE condition -vs- alternative condition) since, as will be detailed in Secs. 3.1, the input statistics change in the middle of phase II.

2.2. Workflow of a single PROPRE iteration

The presented algorithm is a priori independent of the embedding into any kind of robotic system; its only assumption is that input representation I and reference representation R periodically receive new values in a synchronous manner. In most robotic agents, such values are provided at regular intervals by sensors updating their perception of the world, or by subsequent computations on new sensory inputs. Every time this happens, one PROPRE *iteration* is executed, consisting of the following steps which are also illustrated in Fig. 4:

1. new data is fed into input representation I and reference representation R
2. **projection:** activity in the induced representation N is formed by projection of the input representation I, see Sec. 2.4
3. **prediction:** based on activity in the reference representation R, the reference-based prediction $\mathrm{pr}_{R \to N}$ is computed as described in Sec. 2.4

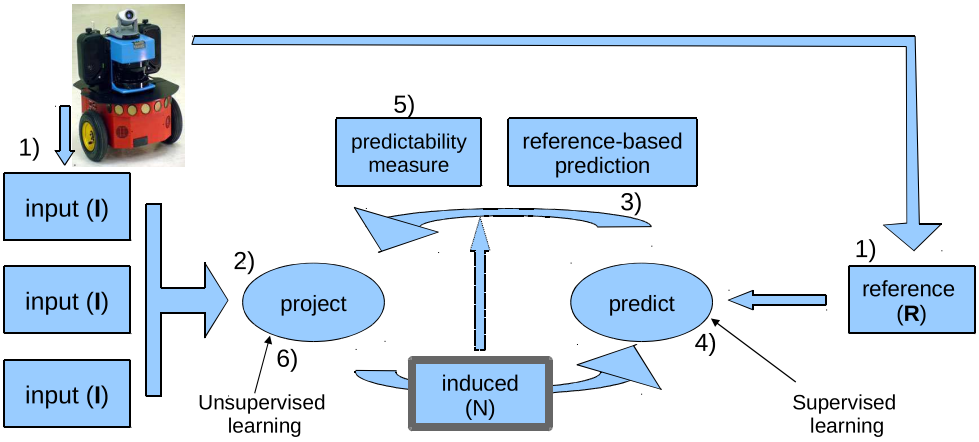

Figure 4: Schematic overview of the PROPRE learning algorithm. Given a reference representation R and sensory inputs I which possibly originate from another modality, the algorithm learns to map the sensory inputs I to a new induced representation N whose activity can be well predicted from R. Please see text for details of the algorithm.

4. **update of prediction:** the mapping between reference representation and induced representation $R \to N$ is updated based on the accuracy of the prediction $\mathrm{pr}_{R\to N}$, see Sec. 2.4
5. **calculation of predictability**: a predictability measure is computed from $\mathrm{pr}_{R\to N}$, resulting in the update indicator $\lambda(t) \equiv \lambda(\mathrm{pr}_{R\to N}, t)$, see Sec.2.3
6. **update of projection:** the projection $I \to N$ is updated (see Sec. 2.4). The learning rate of the adaptation is gated by the update indicator $\lambda(t)$ which effectively determines whether the projection $I \to N$ is adapted with the current pattern or not
7. **update of evaluation measures**, see Sec. 2.5

Details for each step of a single PROPRE iteration are described in the following subsections.

2.3. Quantification of predictability

A key concept is the quantification of predictability, which is used to decide whether the projection step should be updated with the current input or not. We define an easy-to-compute, online *predictability measure* π and

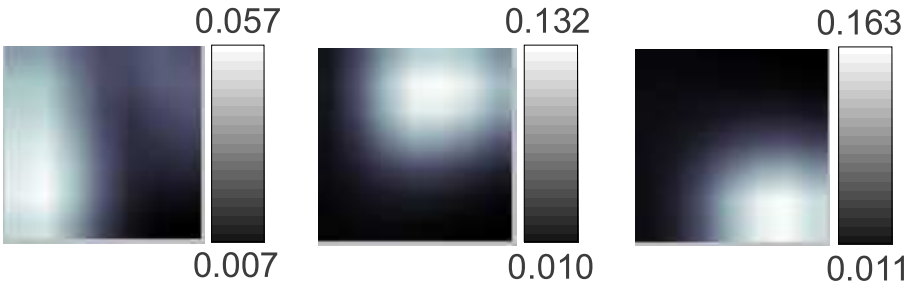

Figure 5: Examples of reference-based predictions $\mathrm{pr}_{R\to N}(\vec{y},t)$ indicating different predictability. Please note the different value ranges indicated by the colorbars. Left: prediction derived from the "too large" signal in the reference representation R: in this case the predictability of the induced representation is low as compared to the case of the "not pushable" (middle) and "pushable" (right) classes.

an *update indicator* λ as follows:

$$\begin{aligned}
\mu_{\min}(0) &= \mu_{\max}(0) = 0 \\
\mu_{\min}(t) &= (1-\alpha)\mu_{\min}(t-1) + \alpha\ \min_{\vec{x}} \mathrm{pr}_{R\to N}(\vec{x},t) \\
\mu_{\max}(t) &= (1-\alpha)\mu_{\max}(t-1) + \alpha\ \max_{\vec{x}} \mathrm{pr}_{R\to N}(\vec{x},t) \\
\pi(\mathrm{pr}_{R\to N},t) &= \max\ \mathrm{pr}_{R\to N}(\vec{x},t) - \min\ \mathrm{pr}_{R\to N}(\vec{x},t) \\
\lambda(\mathrm{pr}_{R\to N},t) &= \begin{cases} 1 & \pi(\mathrm{pr}_{R\to N},t) > \kappa[\mu_{\max}(t) - \mu_{\min}(t)] \\ 0 & \text{otherwise} \end{cases}
\end{aligned} \tag{1}$$

Here $\mu_{\max}, \mu_{\min}$ are running averages (at a time scale α) over the minimal and maximal values of $\mathrm{pr}_{R\to N}$. Put simply, the binary measure λ compares the (spatial) variability of the current reference-based prediction, represented by $\pi(\mathrm{pr}_{R\to N},t)$, to the average variability of previous predictions represented by $[\mu_{\max}(t)-\mu_{\min}(t)]$. This is motivated by the fact that zero variability indicates no predictability since all elements of the induced representation are equally likely to be active, whereas large variability indicates predictability since some elements are much more likely to be active than others. The constant κ determines by how much the average variability has to be exceeded to trigger the case of $\lambda = 1$.

2.4. The projection and prediction steps

We choose to implement the prediction step by logistic regression (LR)[15] operating between the representations R, N by adapting the connection

weights $w^{R-N}_{\vec{x}\vec{y}}$ that ideally achieve an error-free mapping. Using the weights, LR computes an estimate of N given R, $\mathrm{pr}_{R\to N}(\vec{y},t)$, which we term "reference-based prediction". The following training of the connection weights $w^{R-N}_{\vec{x}\vec{y}}$ is governed by a single parameter, the learning rate ϵ^{pred}.

The projection step is realized using the SOM algorithm proposed in [16], thus obtaining a topology-preserving projection from the possibly high-dimensional input representation I to the induced representation N. A SOM step consists of data transmission and weight adaptation. In our notation, the data transmission step reads

$$z^N(\vec{y},t) = \sum_{\vec{x}\in I} w^{I-N}_{\vec{x}\vec{y}} z^I(\vec{x},t). \tag{2}$$

Subsequently, weight adaptation and normalization is performed based on the current learning rate $\epsilon^{\mathrm{proj}}(t) = \lambda(\mathrm{pr}_{R\to N},t)\tilde{\epsilon}^{\mathrm{proj}}(t)$ (gated by the current update indicator λ) and neighbourhood radius $r \equiv r(t)$:

$$w^{I-N}_{\vec{x}\vec{y}}(t+1) = \mathrm{norm}\left(w^{I-N}_{\vec{x}\vec{y}}(t) + \epsilon^{\mathrm{proj}}(t)\; g^r_{\vec{y}^*}(\vec{y})[w^{I-N}_{\vec{x}\vec{y}}(t) - z^I(\vec{x},t)]\right) \tag{3}$$

$$\text{where } g^r_{\vec{y}^*}(\vec{y}) = \mathrm{e}^{-\frac{(\vec{y}-\vec{y}^*)^2}{2r(t)^2}}, \quad \vec{y}^* = \mathrm{argmax}_{\vec{y}} z^N(\vec{y}).$$

As proposed in the original SOM algorithm [16], neighbourhood radius and learning rate are time-dependent. Initially they are kept constant at large values for $t < t_0$ ("global ordering"), whereas decay constants $\rho_{r,\epsilon}$ governs their slow decrease for $t > t_0$, and is always chosen such that learning rate and neighbourhood radius approach their asymptotic stable values $\tilde{\epsilon}^{\mathrm{proj}}_\infty, r_\infty$ at time t_1, after which they are kept constant. The values of t_0, t_1 are chosen such that stable values for radius and learning rate are reached and maintained well before the end of PROPRE phase I, see Sec. 2.1.

$$\tilde{\epsilon}^{\mathrm{proj}}(t) = \begin{cases} \tilde{\epsilon}^{\mathrm{proj}}_0 & t < t_0 \\ \tilde{\epsilon}^{\mathrm{proj}}_0 e^{-\rho_\epsilon t} & t_0 < t < t_1 \\ \tilde{\epsilon}^{\mathrm{proj}}_\infty & t > t_1 \end{cases} \tag{4}$$

$$r(t) = \begin{cases} r_0 & t < t_0 \\ r_0 e^{-\rho_r t} & t_0 < t < t_1 \\ r_\infty & t > t_1 \end{cases} \tag{5}$$

In order to extract a simple interpretation from the output of the projection step z^N, and also to non-linearly suppress noise, we perform non-maxima suppression by simply setting $z^N(\vec{y},t) \to g^r_{\vec{y}^*}(\vec{y})$.

2.5. Evaluation measures

In order to quantify the effect of PROPRE learning on the induced representation N, we assume that patterns in the input representation I can be, for evaluation purposes, grouped into well-defined *classes* $c = 0, 1 \ldots$. We furthermore assume to know the true class of the inputs being presented to the system and denote it by class(t). Based on this, we first define the temporal class average operator

$$E_c^T \left[f(t) \right] = \frac{1}{N_c} \sum_{\substack{t'=t-T \\ \text{class}(t')=c}}^{t} f(t') \tag{6}$$

N_c indicating the number of times class c occurred in the interval $[t - T, t]$. Since PROPRE is an online learning algorithm, averages should be taken over finite time intervals T which must be small enough so the statistics can be considered stationary, but large enough to allow for the smoothing of outliers.

Using this definition, we can write down the *class preference map* for the induced representation N, $p^N(\vec{x}, t)$. It shows, for each neuron in N, to which class of inputs (if any) it preferentially responds to:

$$p^N(\vec{x}, t) = \begin{cases} c+1 & \text{if } \exists c : E_c^T \left[z^N(\vec{x}, t) \right] > 2E_{c'}^T \left[z^N(\vec{x}, t) \right] \forall c' \neq c \\ 0 & \text{else} \end{cases} \tag{7}$$

The class preference map $p^N(\vec{x}, t)$ can be reduced, for a given class c, to the *class preference measure* $\chi_c^N(t)$ indicating the percentage of neurons in N that respond preferentially to class c:

$$\chi_c^N(t) = \frac{1}{s^2} \sum_{\substack{\vec{x} \in p^N(\vec{x},t) \\ p^N(\vec{x},t)=c+1}} 1, \tag{8}$$

where s^2 denotes the number of neurons in the induced representation N with dimensions $s \times s$.

In order to relate PROPRE's performance to the large body of literature on self-organizing maps, it is useful to compute the quantization error of standard SOM approaches which measures how well the current pattern is represented by the weight vectors (sometimes denoted "codebook vectors")

of the SOM:

$$e(t) = \frac{1}{s^2} \sum_{\vec{y} \in N} \left(\sum_{\vec{x} \in I} w^{\text{I-N}}_{\vec{x}\vec{y}}(t) - z^{\text{I}}(\vec{x}, t) \right)^2$$

Given that weights and inputs are normalized, this reduces to

$$\begin{aligned} e(t) &= K - \frac{1}{s^2} \sum_{\vec{y} \in N} \left(\sum_{\vec{x}} \vec{w}^{\text{I-N}}_{\vec{x}\vec{y}}(t) z^{\text{I}}(\vec{y}, t) \right) = \\ &= K - \frac{1}{s^2} \sum_{\vec{y} \in N} z^N(\vec{y}, t) \end{aligned} \tag{9}$$

where we have used eqn.(2) for the last transformation. K is a bounded constant that depends neither on the weights nor the data. Since we are only interested in determining changes of e_q, we will set $K = 0$ with no adverse effects. Averaging $e(t)$ using eqn.(6), we finally obtain the averaged quantization error

$$\mathcal{E}_c(t) = -E_c^T \left[e(t') \right]. \tag{10}$$

which we have made positive for better visualization.

The class preference measure $p^N(\vec{x}, t)$ lends itself particularly to being plotted in two dimensions for a specific time t to visualize the instantaneous distribution of class-sensitive neurons, whereas the derived measure $\chi_c^N(t)$ is better suited for being plotted over time to indicate the overall success of PROPRE learning. The class-specific quantization error $\mathcal{E}_c(t)$ will be plotted over time to indicate changes in the accuracy individual classes are represented by the induced representation N. In order to directly view the preferred stimuli of individual neurons in N, we sometimes also directly plot the projection weights $w^{I-N}_{\vec{x}\vec{y}}(t)$.

3. Experiments

3.1. Synthetic input data

Our choice of synthetic stimuli is inspired by the robotic model task described in Sec. 1.1. We model a world filled with objects that can be in one of three possible classes: "pushable", "non-pushable" and "too large",

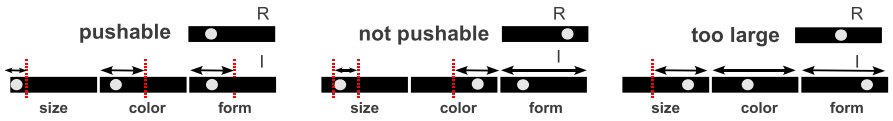

Figure 6: Input statistics used in experiments. Shown are value ranges and realization examples for the artificial input representations I and reference representation R for $t \in [8000, 15000]$. Left: pushable objects are defined by a a certain range of colors and forms, and a low size Middle: non-pushable objects are characterized by a slightly larger size and a certain range of colors, irrespective of form which can take any value. Right: Objects too large for pushing are characterized by a large size, whereas the the other two visual properties can take any value. This is why this class is less predictable: when the reference representation indicates the "too large" class, little can be inferred about the values of "form" and "color", leading to flat predictions and therefore low predictability.

each of which occurs with a certain probability. Based on object class, we fill the reference representation R with one of three unique patterns, indicating whether an object is too large for pushing behavior, whether it can be pushed, or whether it cannot be pushed. We define the artificial object properties of "size", "form" and "color" for the input representation I, which are drawn from a probability distribution that relates to object class. The probabilistic definition of object classes, and the corresponding generation of input and reference representations, is visualized in Fig. 6. The basic idea is to create three classes whose input representations I are predictable, to different degrees, from the reference representation R. From the point of view of the robotic model task, it is easily understood that pushable and non-pushable objects might have quite specific visual properties which are more predictable from the "pushable" or "not pushable" properties. However, the "too large" class contains all objects that are by definition not pushable for a small robot (chairs, tables, walls, humans, etc.); the visual properties of such objects are subject to much more variation, which is why this class is much less predictable.

Since this is a simulation-based work, we have full knowledge of object classes even at training time when generating the artificial input and reference representations I, R. We stress, however, that in the general case, both the number and the definition of classes need to be known only for evaluation purposes. The learning algorithms do not depend on this information in any way.

The definition of classes for simulated input and reference representations

occurs by specifying four random variables ν_k: $\nu_{\text{size}}, \nu_{\text{color}}$ and ν_{form} for the input, ν_{R} for the reference representation. In accordance with the model task and Fig. 6, $\nu_{\text{size}}, \nu_{\text{color}}, \nu_{\text{form}}$ and ν_{R} are chosen such that there are statistical relations between $\nu_{\text{size}}, \nu_{\text{color}}, \nu_{\text{form}}$ and ν_{R}. As can be seen from the following equations, these relations undergo a change at $t = 15000$ (in the middle of phase II, see Fig. 3) to demonstrate PROPRE's ability of adapting to changing input statistics without exhibiting catastrophic forgetting:

$$\begin{aligned}
\eta &\sim \mathcal{U}(0,1) \\
\Delta &= \begin{cases} 0.15 & t < 15000 \\ 0.3 & t >= 15000 \end{cases} \\
\text{pushable: } & \nu_{\text{size}} \sim \mathcal{U}(0, 0.15), \nu_{\substack{\text{color} \\ \text{form}}} \sim \mathcal{U}(0, 0.5), \nu_{\text{R}} = 0.25 \text{ if } \eta \in [0, 0.25] \\
\text{not pushable: } & \nu_{\text{size}} \sim \mathcal{U}(\Delta, \Delta + 0.15), \nu_{\text{color}} \sim \mathcal{U}(0.5, 1), \nu_{\text{form}} \sim \mathcal{U}(0, 1), \\
& \nu_{\text{R}} = 0.75 \text{ if } \eta \in [0.25, 0.5] \\
\text{too large: } & \nu_{\text{size}} \sim \mathcal{U}(0.45, 1), \nu_{\substack{\text{color} \\ \text{form}}} \sim \mathcal{U}(0, 1), \nu_{\text{R}} = 0.5 \text{ if } \eta \in [0.5, 1].
\end{aligned} \tag{11}$$

Here, $\mathcal{U}(a, b)$ denotes the uniform distribution in the interval $[a, b]$. After the four ν_i are generated, they are encoded into four corresponding two-dimensional blocks realizing the representations $I_{\text{size}}, I_{\text{color}}, I_{\text{form}}$ which together form the input representation I, as well R. The size of the blocks is fixed such that a single number $\nu_i \in [0, 1]$ can be comfortably encoded by creating a Gaussian $g_{\vec{y}^*}^{\sigma=1.5}(\vec{y})$ at $\vec{y}^* = (32 * \nu_i(t), 3)^T$. Our choice of block dimensions of 32x7=214 elements is a compromise between considerations of simulation speed and representational accuracy. While larger blocks can represent more different values of the ν_i, the speed of the whole simulation suffers if the blocks get too large. Our choice gives a rather limited value resolution to I and R without adverse effects; larger blocks are always possible but do not affect the outcome of the simulation except for making it slower.

3.2. Implementation and parameter values

In all experiments, we simulate 20000 iterations using induced representations of $s \times s$ elements; phase I of PROPRE training (see Fig.3) is always conducted for 8000 pattern presentations, where radius and learning rate are decreased from initial values of $\bar{\epsilon}^{\text{proj}}(t < t_0 = 800) = 0.8$ and $r(t < t_0 = 800) = s/2$, until values of $\epsilon_\infty^{\text{proj}} = 0.02$ and $r_\infty = s/6$ are reached

approximately at $t = t_1 = 5000$. This parameter decrease, as described in Sec. 2.4, is performed using decay constants of $\rho_\epsilon = 0.0007$ for the learning rate, and $\rho_r = 0.00025$ for the neighbourhood radius.

For $t > t_1 = 5000$, which includes the last part of phase I and all of phase II, we use $\tilde{\epsilon}^{\text{proj}}(t) \equiv \epsilon_\infty^{\text{proj}}$ and $r(t) \equiv r_\infty$. The learning constant of the prediction step is always kept at $\epsilon^{\text{pred}} = 0.01$, and the time constant of the running average calculation (1) is chosen as $\alpha = 0.002$. For the predictability threshold in (1), we use $\kappa = 1.1$, and weight vectors of projection and prediction are randomly initialized to small values in the interval $[-0.001, 0.001]$. The interval T used to measure the class averages $\chi_c^N(t)$ is set to $T = 600$ iterations.

The effects of varying these parameter values are discussed in-depth in Sec. 4.2.

In order to demonstrate adaptation to changing input statistics while avoiding catastrophic forgetting, we change the input statistics at $t = 15000$ (see Fig. 3). The effect of the change, as specified by eqn.(11), is such that the "not pushable" class suddenly consists of slightly larger objects.

All the experiments described here are implemented in Python and C using the OpenCV library to accelerate neural network execution and learning. On a standard off-the-shelf PC with a 2GHz multicore processor, it is possible to simulate roughly 20 iterations per second while running the whole simulation on a single CPU core[2].

3.3. Development of preferences for predictable stimuli

This simulation experiment demonstrates the translation of concepts from a simple reference representation using the concept of predictability. In this section, we will compare the baseline condition to the PROPRE condition (see Fig. 3) for establishing that PROPRE learning indeed achieves successful bootstrapping from reference representation R to induced representation N.

We use an induced representation of 10×10 elements and follow the experimental time course outlined in Fig. 3 and Sec. 2.1. Results are given as the temporal development of the class preference measure χ_c^N and by the visualization of the class preference map p^N in the baseline, PROPRE and alternative conditions. Whereas $p^N(\vec{x}, t)$ directly shows the class preference of

[2]The Python/C code of the simulation will be made available on the site www.gepperth.net

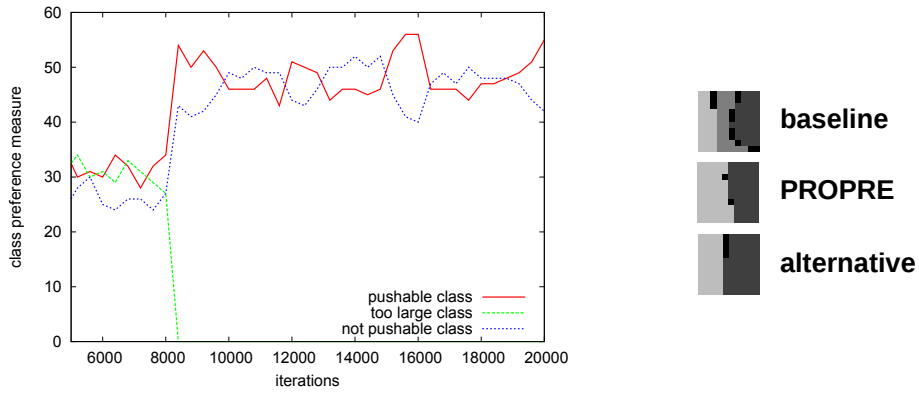

Figure 7: Development of neural selectivities in our robotic model task under the influence of PROPRE learning. Left: temporal evolution of the class preference measure $\chi_c^N(t)$ for the input classes "pushable","not pushable" and "too large". Right: class preference map p_c^N evaluated in the baseline condition (top), the PROPRE condition (middle) and alternative condition (bottom). Bright gray pixels indicate the "not pushable" class, moderately gray pixels indicate the "too large" class, and dark gray pixels the "pushable" class. Black pixels indicate no sufficient class selectivity.

individual neurons in the induced representation N, $\chi_c^N(t)$ gives the fraction of neurons N which, at time t, respond preferentially to stimuli from class c. Please see Sec. 2.5 for precise definitions of these measures.

As can be seen in Fig. 7 (right), the ability of the induced representation N to represent input stimuli of the predictable "pushable" and "not pushable" classes is strongly increased in the PROPRE condition w.r.t. the baseline condition, at the expense of the least predictable "too large" class which is completely suppressed. The most notable observation when considering the class preference measure χ_c^N is the speed at which the reorganization occurs after PROPRE learning starts at $t = 8000$. We can therefore state that predictable stimuli are favored by PROPRE learning, and that there is fast convergence to a stable equilibrium state.

3.4. Avoidance of catastrophic forgetting

Using the experimental setup of the previous section 3.3, we wish to determine how a persistent change in input statistics will affect the induced representation N. Eqn.(11) governs the modification of the "not pushable" class w.r.t the "size" input at $t = 15000$ (see Fig. 3): translated to our robotic model task, it implies that non-pushable objects are suddenly slightly bigger than before. An optimal response of the learning algorithm would be to

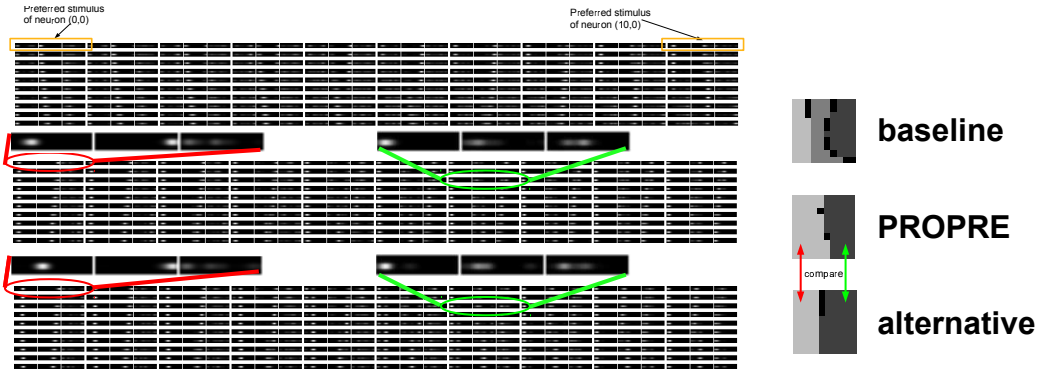

Figure 8: Behavior of PROPRE learning under a change in input statistics. The left diagram shows all projection weights to the induced representation N, $w_{\vec{x}\vec{y}}^{I-N}(t)$ in the baseline ($t = 7800$, top), the PROPRE ($t = 14800$, middle) and the alternative ($t = 17800$, bottom) conditions. As N contains 10x10 neurons in this experiment, diagrams consist of 10x10 weight blocks, each comprising 3 patterns corresponding to the "size", "color" and "form" inputs. The position of each weight block indicates the position of its associated neuron in the 10×10 grid. The ellipses and corresponding magnifications highlight two neurons, with different class preferences, whose selectivity has adapted to the new input statistics at $t = 15000$ (red ellipse, selectivity for "size" input is clearly shifted), or has not changed sigificantly (green ellipse, no relevant change). To facilitate comparison, the class preference maps are given on the right of each weight diagram, showing that the two compared neurons really prefer different classes. It can be observed that all neurons preferring the "non-pushable" class (bright pixels in the class preference maps) have adapted their selectivities.

exclusively modify the selectivities of neurons that already prefer the "non-pushable" class, and to leave the rest untouched. This is in contrast to the behavior of, e.g., multilayer perceptrons[17], where retraining with slightly different input statistics can lead to a complete reorganization of the hidden layer structure, and therefore to a loss of already learned capabilities.

To assess PROPRE's capacity to avoid catastrophic forgetting, we therefore compare the organization of neural selectivities in the induced representation N in the PROPRE and the alternative conditions. As can be seen from Fig. 8, the comparison demonstrates very favorable behavior in the face of changing input statistics. Particularly, two important facts can be observed: first, the overall class preferences of neurons are left virtually unchanged. Second, only the stimulus preferences of "not pushable"-specific neurons are adapted to the new statistics, while the remaining neurons are not affected. The (small) changes that do occur for "pushable"-specific neurons are due

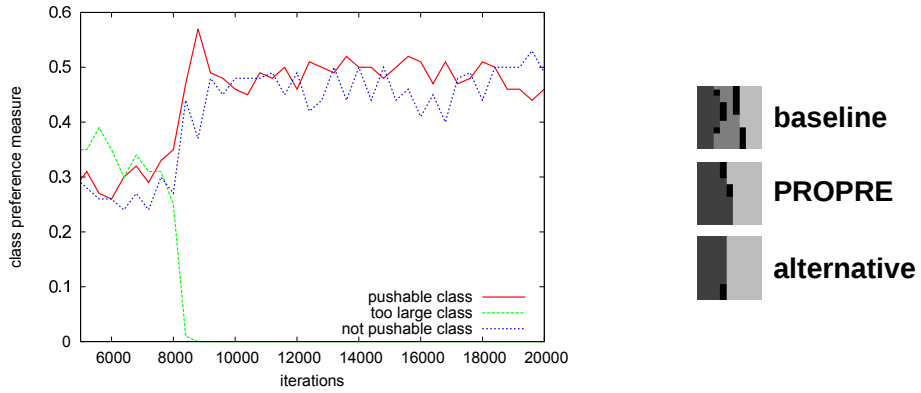

Figure 9: Behavior of PROPRE learning when learning from a noisy reference signal. Left: class preference measure $\chi_c^N(t)$ plotted over time for the three classes "pushable", "not pushable" and "too large". Right: class preference map p_c^N evaluated in the baseline condition (top), the PROPRE condition (middle) and alternative condition (bottom). Bright gray pixels indicate the "not pushable" class, moderately gray pixels indicate the "too large" class, and dark gray pixels the "pushable" class. Black pixels indicate no sufficient class selectivity.

to normal "drift" that would occur even without a change in input statistics. PROPRE has therefore been successfully demonstrated to avoid catastrophic forgetting.

Two observations are of importance when regarding Fig. 8: first of all, the topological organization of neurons in the induced layer N is very obvious. Second, those weights that process less predictable parts of the input (i.e., the "form" input) tend to be correspondingly "smeared out". This will be followed up upon in a later experiment.

3.5. Effects of a noisy reference signal

In real-world robotics applications there is often no way to obtain an absolutely trustworthy reference signal as we assume it in the previous experiments. In this experiment, we wish to determine which effect such a noisy reference signal has on the bootstrapping process using PROPRE learning. To this end, we repeat the experiment of Sec. 3.3 and flip the reference signal randomly to a different class for 10% of inputs. The learning performance under such noisy conditions is given in Fig. 9. Overall, it can be stated that PROPRE learning works even with a noisy reference signal, and that the outcome is comparable to the noise-free experiment of Sec. 3.3. Nevertheless it may be discerned in Fig. 9 (left) that the convergence of learning is

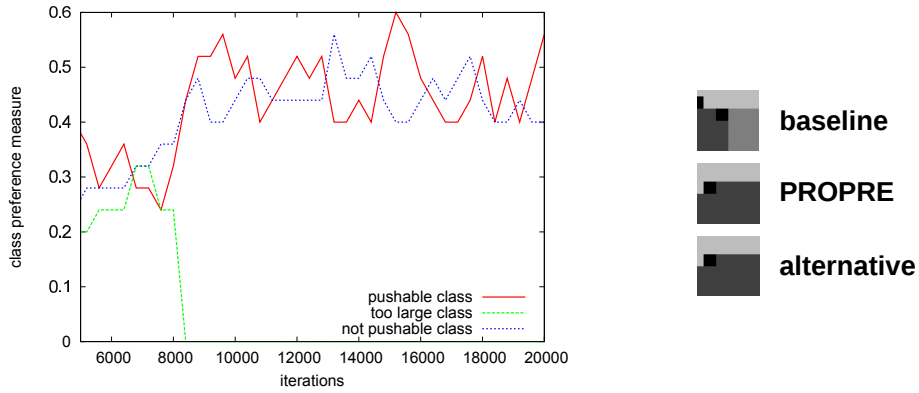

Figure 10: Development of neural selectivities under resource constraints, using an induced representation N of 5×5 neurons. Left: class preference measure $\chi_c^N(t)$ plotted over time for the three classes "pushable", "not pushable" and "too large". Right: class preference map p_c^N evaluated in the baseline condition (top), the PROPRE condition (middle) and alternative condition (bottom). Bright gray pixels indicate the "not pushable" class, moderately gray pixels indicate the "too large" class, and dark gray pixels the "pushable" class. Black pixels indicate no sufficient class selectivity.

slightly delayed. This stands to reason since incorrect class labels in the reference representation will admit unsuitable patterns into the learning of the projection weights, thus delaying convergence.

3.6. Graceful decay under severe capacity limitations

In this experiment, we wish to investigate the robustness of PROPRE under strong resource constraints which we model by a reduced size of the induced representation N. This is motivated from the fact that either memory or simulation capacity in autonomous agents are usually restricted, and some limit on the size of representations always exists. Therefore, it must be ensured that our learning algorithm degrades gracefully if constraints get tighter, and that the benefits of PROPRE learning (i.e., the "focusing" on the most predictable stimuli) still apply. The experimental setup is identical to that of Sec. 3.3 except that the size of the induced representation N is reduced to 5×5 elements and subsequently to 2×2 elements. As can be seen from the plots of Figs. 10 and 11, the behavior of class preference measure χ_c^N remains qualitatively unchanged, demonstrating a transfer of representation capacities from the least predictable class to the two most predictable ones. This is a very desirable behavior, as it shows that PROPRE learning works even under rather extreme conditions as may be encountered on embedded

computers. This result also implies that the size of the induced representation is not a critical parameter, and can therefore be set by a system designer without too much consideration.

3.7. Dismissal of irrelevant input dimensions

Lastly, we are going to follow up on an observation made in Sec. 3.4: that uninformative inputs to the induced representation N cause the corresponding receptive fields to be strongly "smeared out", especially when capacity limits apply. As input weights to N are normalized, this should imply reduced sensitivity to such stimuli. Which, if true, would represent a desirable property: *irrelevant parts of the input are ignored*. In order to test this, we run the experiment of the previous section using an induced representation of 5×5 elements. For the period $t \in [12000, 14000]$ we erase all activity either in the "size" or "form" inputs to the induced representation N (see Fig. 6) and measure the corresponding changes in the quantization error $\mathcal{E}_c(t)$ for the "not pushable" class. The measure $\mathcal{E}_c(t)$ reflects the averaged total neural activity for a certain class (see Sec. 2.5). During this interval, we also disable learning for both the projection and the prediction steps because we want to avoid an adaptation to this change in input statistics. We chose the "size" and "form" inputs for elimination because they differ most strongly in terms

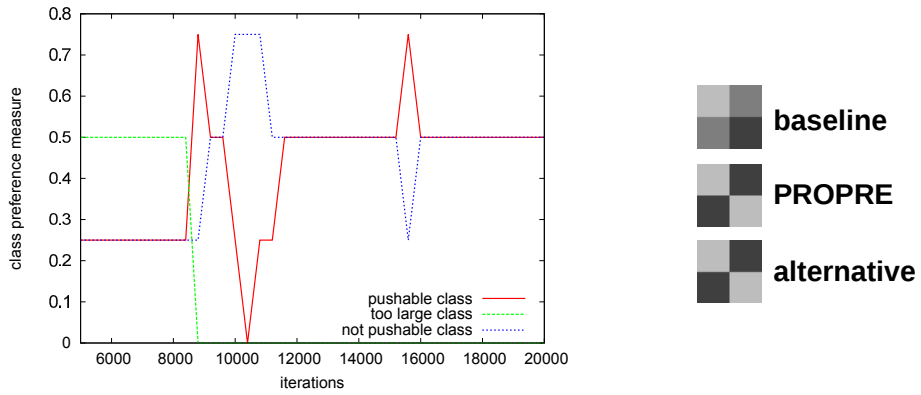

Figure 11: Development of neural selectivities under resource constraints, using an induced representation N of 2×2 neurons. Left: class preference measure $\chi_c^N(t)$ plotted over time for the three classes "pushable", "not pushable" and "too large". Right: class preference map p_c^N evaluated in the baseline condition (top), the PROPRE condition (middle) and alternative condition (bottom). Bright gray pixels indicate the "not pushable" class, moderately gray pixels indicate the "too large" class, and dark gray pixels the "pushable" class. Black pixels indicate no sufficient class selectivity.

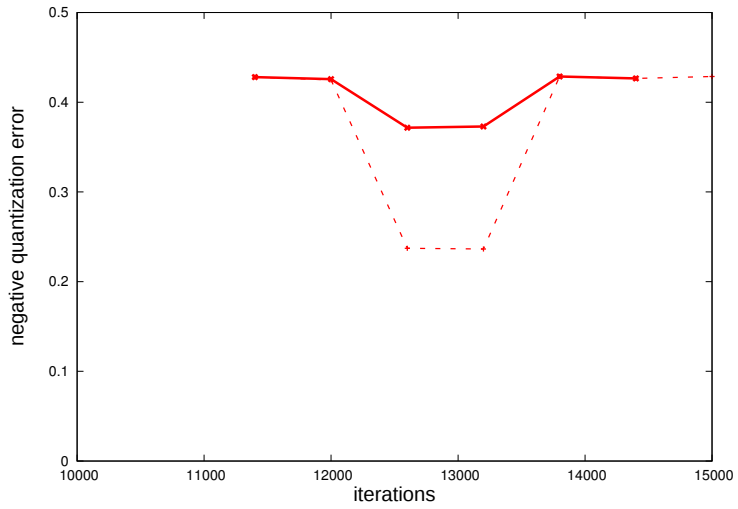

Figure 12: Contribution of different parts of the input to the induced representation N. Shown is the negative quantization error for the "not pushable" class of inputs, when either the "size" (solid line) or the "form" input (dashed line) are fixed to zero activity in the interval $t \in [12000, 14000]$. As the "size" input is in a very narrow range for the "not pushable" class, whereas the "form" input can take any value, the information gain from the "size" input is much higher. It therefore stand to reason that the "size" input should contribute more strongly to activity in N. As can be seen, this is indeed the case: the drop in activity for the "size" input is severe, whereas it is relatively moderate for the "form" input.

of being (ir)relevant for the "not pushable" class: the "size" input is highly localized (i.e., relevant) whereas the "form" input is completely unlocalized, therefore irrelevant. As can be seen from Fig. 12, this translates into slight drops in average activity when the irrelevant input is suppressed, in contrast to a huge drop in case the relevant input is suppressed.

4. Discussion

In this section, we will summarize the key points of the method and draw conclusions from the presented experimental results. Subsequently, we will critically discuss these findings and the presented model. Finally, we will point out weak points of the presented material and, by this, motivate future work.

4.1. Synopsis and conclusions from experiments

Bootstrapping of representations under real-world conditions. We have shown that the presented PROPRE algorithm can achieve a successful bootstrapping of representations from existing simple reference representations to more complex induced representations (see Sec. 3.3). Is has been demonstrated experimentally that the process is efficient and robust to capacity limits (see Sec. 3.6), noisy reference representations (see Sec. 3.5) and changing input statistics (see Sec. 3.4). Only those properties are transferred by the bootstrapping process for which a statistical relation between sensory inputs and reference representation exists. In fact there is strong competition for representation between different classes of inputs (see Sec. 3.6), which leads to the survival only of those classes for which statistical relations are strongest. In the same vein, PROPRE learning is robust to uninformative input subspaces which can always occur in sensory signals: as was shown in Sec. 3.7, such subspaces are approximately ignored.

Local quantification of statistical relations. PROPRE quantifies the notion of "statistical relations" given in the previous paragraph by a simple, instantaneous and online measure of predictability. This measure may be regarded as an approximation to information (entropy) measures: flat predictions are assigned low values, strongly peaked predictions are assigned high values. In this way, an ordering between different classes of inputs is induced which can be used for deciding which ones to represent. The key point is that the concept of the predictability of the induced representation N is very easy to compute locally at N: when thinking in neural terms and remembering Sec. 2, the prediction $\mathrm{pr}_{R\to N}$ "arrives" at the location of N, and the updating of the prediction weights $w_{\vec{x}\vec{y}}^{R-N}$ also happens at N. The computation of decayed averages in N is local as well, therefore all quantities needed to compute λ are available locally. This would not be the case if one would try to form N based on its power to predict R, as it was suggested in [11].

Stability and adaption. As can be seen in Sec. 3.3, the PROPRE algorithm is stable in the face of stationary input distributions. This can most clearly be seen from the graph of $\chi_c^N(t)$ over time in Fig.7, and also directly from the development of receptive field profiles over time (not shown here) which do not change significantly after the first formation period. Stability is achieved without a dedicated learning rate control and is due to the fact that the predictability of an input class decreases under the influence of PROPRE

learning. This occurs because predictions get less "peaked" after the number of class-selective neurons increases. This incurs a smaller difference between minimum and maximum, and therefore a decrease in predictability. Finally, an approximate equilibrium is achieved where all classes have the same predictability.

Autonomous learning capability. As the PROPRE method described here is meant to be used in autonomous agents, it should have a strong degree of autonomy in itself. By this we understand three things: first of all, the independence from explicit external supervision; furthermore, the capability to learn (online) continuously, without making assumptions about the number of data samples; and lastly, the autonomous control of learning, i.e., autonomy in the decision when to learn and when not to learn, based only on the statistical properties of the input data. The first requirement is manifestly fulfilled: no external supervision signal is applied, removing the need to create databases of training examples by hand. The second requirement is equally fulfilled by construction since all components of PROPRE are fully online algorithms that do not require to know the number of data samples, and which can always be fed with new data independently of previous learning sessions. Lastly, in deriving the update indicator $\lambda(t)$ (see Sec. 2.3) from the properties of the data, PROPRE realizes a fully autonomous control of learning that avoids external readjustment or control phases, and which enables the algorithm to run stably for prolonged time spans in the face of potentially changing input distributions (see Sec. 3.4).

4.2. Critical examination and justification of used methods

Choice of projection method. For performing the dimensionality-reducing projection step from input representations I to the induced representation N, numerous methods such as multilayer perceptrons (MLP)[18], PCA[19], sparse coding[20], k-means and indeed, SOM[16] are available. Given our long-term goals for the PROPRE algorithm, the open-ended formation of representations as envisioned in Fig. 1, we demand of the projection method a *transitivity property*, meaning that the induced representation N, as the result of one PROPRE instance, can serve as population-coded input to another PROPRE instance. This implied that projection outputs must be topologically organized on a two-dimensional grid (regardless of the input data), both of which are requirements for population coding. Given this

goal, we can immediately discard k-means because it only estimates cluster memberships. Another very simple possibility is to use a three-layered multilayer perceptron[21], feeding the input representation into the input layer and using the reference representation as target values for the output layer, the hidden layer having an arbitrarily reduced size. While this will certainly generate an induced representation, this solution is unsuitable because the hidden layer, while manifestly two-dimensional irrespectively of input data, would lack any topological organization if the MLP is trained using a back-propagation learning algorithm[15]. Furthermore, issues of catastrophic forgetting[17] would complicate the use of MLP still further. As an alternative, PCA produces, for each input, a set coordinates in the space of principal components which are not in any way topologically organized. Due to the data-compression property of PCA, most of these coordinates can usually be disregarded while still faithfully representing the original data. Therefore, one could theoretically create a two-dimensional population code from the first two coordinates, but the quality of this approximation to the original data would usually be very bad. For sparse coding, there exists no data compression property, so there is no obvious way to create a two-dimensional topologically organized output at all. In contrast, the SOM algorithm fulfills all of our requirements beautifully: the output naturally lives on a two-dimensional grid, it is topologically organized (see Fig .8) and, if using the data transmission and noise suppression steps of Sec. 2.4, it is already in the form of a population code which can be used as input for another PROPRE instance.

Choice of parameters. The PROPRE model presented here contains several parameters whose values influence its performance. These are, mainly, the learning constants $\epsilon_{\infty}^{\text{proj}}$, ϵ^{pred} for the projection and prediction networks (see Sec. 2.4), the size s of the induced representation N (see Sec. 3.2), the time constant α for the running average in the computation of the update indicator $\lambda(t)$ (see Sec. 2.3), and the predictability threshold κ (see Sec. 2.3). Some of these constants are not really independent, so several constraints apply: first of all, the projection and prediction time constants should be in a similar range since both methods interact on equal terms with neither component driving or controlling the other. Secondly, the running average constant α must be at least one order of magnitude smaller than ϵ^{pred} since the running average operates on the output of the prediction step and must be therefore slower if it is to have a stabilizing/smoothing effect. With these

constraints, the initial choice of, e.g., ϵ^{pred} more or less determines the other learning constants. The value of $\epsilon^{\text{pred}} = 0.01$ was chosen such that the broad time scale on which learning can have an effect is $\frac{1}{\epsilon^{\text{pred}}} = 100$ iterations. We verified that, as long as the constraints mentioned here are respected, PROPRE learning is robust to changes in ϵ^{pred} in the range of 0.0001 to 0.1. If a larger value is chosen, learning becomes numerically unstable because the assumption of a small learning rate in gradient-based learning is no longer valid. If the time constants of projection and prediction are too different, undesirable oscillations take place which destroy the learning effects. The size s of the induced representation N is a parameter that is usually severely bounded from above by application constraints such as simulation speed, so the heuristic for tuning this parameter is simply to choose s as big as possible while maintaining a sufficiently high simulation speed, leading us to the used value of $s = 10$. Tests with values of $s = 12, 15, 20$ showed that the basic operation of PROPRE is completely unaffected by such variations. The predictability threshold κ denotes the ratio of measured to average predictability that is to be considered the lower limit for learning. Evidently, we should have $\kappa \geq 1$; the behavior of PROPRE with high values of κ depends on the input data: on the one hand, learning will be slower but focus more exclusively on the most predictable classes. For all input distributions we simulated, learning was robust to choices of $\kappa \in [1.1, 1.5]$ while being notably impeded if κ exceeded 1.5.

Complexity analysis. PROPRE learning is independent of the nature of the input data, and only linear in the total number of neurons used in the simulation. The single most time-consuming process is the projection step described in Sec. 2.4 which is nevertheless linear in the number of neurons in the induced representation N. This number, given in Sec. 3.2 as $s \times s$, has therefore a strong influence on the total simulation speed. As both projection and prediction steps are highly parallelizable on, e.g., present-day GPU hardware, the linear complexity behavior could be improved even further, leading to larger allowed sizes for the induced representation in autonomous agents.

4.3. Shortcomings of the presented model

The presented work has several shortcomings: first of all, all experiments, although modeled after a real robotic task, are done in simulation only. This was done in order to put emphasis on a thorough analysis of the algorithm

itself in a situation where we have a great deal of control over the data it is applied to. It will be one of our next steps to implement the described task on our robotic platform (see [8]) to ensure that the obtained results indeed carry over to the real world domain.

In any case, as the input data are generated from only four dependent random variables (see Sec. 3.1), one might ask whether this is not too simple or too low-dimensional a task for evaluating PROPRE. To counter this, we wish to point out that the task has been modeled after our experiences in the robotic domain[9, 8] where object representations such as those used here are quite common, and also easy to compute. Furthermore, the *actual* input representation is very high-dimensional due to the population encoding step (see Fig. 6), comprising a total of $3x(32x7) = 672$ dimensions. It is therefore reasonable to assume that the robotic model task itself has sufficient realism and difficulty to allow a meaningful evaluation.

Furthermore, the chosen way of controlling learning in the projection step is overly simplistic: the binary "on"/"off" modulation of the learning rate by λ may not be very appropriate for inputs composed of more classes that are similar to each other, and which also have similar predictability scores. In such a case, the binary threshold would probably lead to the near-complete suppression of the least predictable classes even if the differences are not very large. To counter this, we suggest that the learning rate can be modulated by a continuous quantity, such that highly predictable patterns in the induced representation N get preferred, but slightly less predictable ones also get a chance to be represented.

Moreover, although the algorithm evidently works well, what is missing is some kind of deeper understanding of the dynamics of PROPRE. Typically, this is achieved by finding a smooth energy or, more generally, a Lyapunov functional which is minimized during the evolution of a dynamical system. The interpretation of such functionals and their minimization sometimes allow deeper insights into what problem a dynamical system is actually solving. Especially if the energy functionals have probabilistic interpretations, such approaches allow tremendous insights. Contrary to, e.g., the predictive coding model, PROPRE was not *derived* from an energy functional but the learning rules were set up directly. It is therefore hard to give, e.g., proofs of convergence, or to understand what is being optimized. The problem to be overcome is that there exist no well-behaved energy functionals for the SOM algorithm, which forms an essential component of PROPRE. In order to formulate an energy functional for the complete PROPRE algorithm,

a slightly modified versions of the self-organizing map algorithm [22] could be used. This model involves only a very slight change in the determination of the best-matching unit but nevertheless preserves the topological and dimensionality reduction properties of SOMs. At the same time, it has a well-behaved and easy-to-compute energy functional, just as it exists for logistic regression, the other essential component of PROPRE. The remaining issue would be to integrate the predictability measure and the learning control into the desired energy functional, which does not seem to be impossible.

A last, and rather subtle point, is the control of learning. As opposed to the robotic task presented here, there may be scenarios where the imbalance between different classes of inputs is very large. That is, the predictable data samples are statistically totally insignificant w.r.t. the unpredictable ones, which means that convergence will be extremely slow since the learning rate for the prediction step will have to be very low. Classical over/under-sampling are difficult to apply here as class memberships of patterns are unknown at training time. Another manifestation of this problem is the existence of prolonged time periods where only inputs of a single class are observed. This is quite usual in real-world tasks and would, in the presented formulation of PROPRE, inevitably lead to the suppression of certain classes from the induced representation N. Both problems could be resolved if learning were controlled by *novelty* in addition to predictability. Novelty can be defined as a strong difference between reference-based prediction and induced representation. In this way, samples that do not contribute any new information (since the prediction is already good) would be ignored after a while, solving both the problems of imbalance and prolonged exposure to single classes. Moreover, the learning rate could remain high to ensure fast adaptation to changes in input statistics.

5. Conclusion and further work

In this article, we have presented an algorithm for real-world use in robotic agents. We hope that it will ultimately allow such agents to achieve an unsupervised discovery and step-by-step acquisition of new, increasingly powerful and complex internal representation of the external world. This article is an exclusively theoretical work; now that the basic feasibility of the PROPRE algorithm has been demonstrated, we intend to implement it in a real robot as soon as possible, to ultimately show that the developmental approach to robotics can be both feasible and efficient.

6. Acknowledgements

We wish to thank the anonymous reviewers who have contributed hugely to improve the clarity and the correctness of the article, and who gave many useful suggestions for additional interesting experiments.

References

[1] P-Y Oudeyer. Developmental robotics. In NM Seel, editor, *Encyclopedia of the Sciences of Learning*, Springer Reference Series. Springer, 2011.

[2] S Kirstein, H Wersing, and E Körner. Towards autonomous bootstrapping for life-long learning categorization tasks. In *IJCNN*, pages 1–8, 2010.

[3] B Ridge, D Skocaj, and A Leonardis. A system for learning basic object affordances using a self-organizing map. In T Asfour and R Dillmann, editors, *Proceedings of the 1st Int. Conf. on Cognitive Systems*, 2008.

[4] B Ridge, D Skočaj, and A Leonardis. Self-supervised cross-modal online learning of basic object affordances for developmental robotic systems. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Anchorage, USA, May 2010.

[5] E Valenza, F Simion, VM Cassia, and C Umilt. Face preference at birth. *J Exp Psych: Human Perception and Performance*, 22(4), 1996.

[6] M Shibasaki and N Kawai. Rapid detection of snakes by japanese monkeys (macaca fuscata): an evolutionarily predisposed visual system. *Journal of Comparative Psychology*, 123(2), 2009.

[7] R Soussignan, B Schaal, L Marlier, and T Jiang. Facial and autonomic responses to biological and artificial olfactory stimuli in human neonates: Re-examining early hedonic discrimination of odors. *Physiology & Behavior*, 62(4):745 – 758, 1997.

[8] I Jebari, S Bazeille, E Battesti, H Tekaya, M Klein, A Tapus, D Filliat, C Meyer, S Ieng, R Benosman, E Cizeron, J.-C Mamanna, and B Pothier. Multi-sensor semantic mapping and exploration of indoor environments. In *Proceedings of the 3rd International Conference on Technologies for Practical Robot Applications (TePRA)*, 2011.

[9] A Gepperth, S Rebhan, S Hasler, and J Fritsch. Biased competition in visual processing hierarchies: A learning approach using multiple cues. *Cognitive Computation*, 3(1):146–166, 2011.

[10] C Faubel and G Schöner. Learning to recognize objects on the fly: a neurally based dynamic field approach. *Neural Networks Special Issue on Neuroscience and Robotics*, 21(4):Pages 562–576, May 2008.

[11] P König and N Krüger. Symbols as self-emergent entities in an optimization process of feature extraction and predictions. *Biological Cybernetics*, 94, 2006.

[12] RPN Rao and DH Ballard. Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects. *nature neuroscience*, 2(1):79, 1999.

[13] K Friston. Learning and inference in the brain. *Neural Networks*, 16(9):1325–1352, 2003.

[14] MW Spratling. Predictive coding as a model of biased competition in visual attention. *Vision Research*, 48(12):1391 – 1408, 2008.

[15] CM Bishop. *Pattern recognition and machine learning*. Springer-Verlag, New York, 2006.

[16] T Kohonen. Self-organized formation of topologically correct feature maps. *Biol. Cybernet.*, 43:59–69, 1982.

[17] RM French. Catastrophic forgetting in connectionist networks. *Trends in Cognitive Sciences*, 3(4):128–135, 1999.

[18] RJ Reed and RJ Marks II. *Neural smithing: Supervised Learning in Feedforward Artificial Neural Networks*. MIT Press, 1999.

[19] A Hyvarinen. *Independent component analysis*. John Wiley Sons., 2001.

[20] BA Olshausen and DJ Field. Sparse coding with an overcomplete basis set: a strategy employed by v1. *Vision Research*, 37:3311–3325, 1997.

[21] S Haykin. *Neural networks: a comprehensive foundation*. Prentice Hall, 1999.

[22] T Heskes. Energy functions for self-organizing maps. In E Oja and S Kaski, editors, *Kohonen Maps*, pages 303–315. Elsevier, Amsterdam, 1999.