Published in **KI - Künstliche Intelligez 29(4):1–10 (2015)**.
The final publication is available at Springer via http://dx.doi.org/10.1007/s13218-015-0356-1 .

Autonomous Learning of State Representations for Control

An emerging field aims to autonomously learn state representations for reinforcement learning agents from their real-world sensor observations.

Wendelin Böhmer · Jost Tobias Springenberg · Joschka Boedecker · Martin Riedmiller · Klaus Obermayer

Received: 03.02.2015 / Accepted: 05.03.2015 / Published online: 19.03.2015

Abstract This article reviews an emerging field that aims for autonomous reinforcement learning (RL) *directly* on sensor-observations. Straightforward *end-to-end* RL has recently shown remarkable success, but relies on large amounts of samples. As this is not feasible in robotics, we review two approaches to *learn* intermediate *state representations* from previous experiences: *deep auto-encoders* and *slow-feature analysis*. We analyze theoretical properties of the representations and point to potential improvements.

Keywords end-to-end reinforcement learning · representation learning · deep auto-encoder networks · slow feature analysis · autonomous robotics

1 Introduction

Feature engineering is an important part in solving machine learning problems. In the case of reinforcement learning agents for real-world control tasks, this task can be especially challenging as it involves finding representations from raw sensor measurements (such as images or laser scan data) which are rich enough to describe the full *state* of the agent (and its environment), while still being compact enough to enable fast convergence of the learning algorithm.

Taking learning in robotics applications as an example, the agent is faced with sensory data in form of a stream of relatively high dimensionality, e.g. from $4 - 7$ values of joint angles plus their velocities (and possibly accelerations) for robotic arms, up to 50+ joint angle values on very complex robots, hundreds of sensor values measured by pressure-sensitive robotic skins, or on the order of thousands to millions of pixels from camera images, nowadays often augmented with depth information.

As mentioned, the challenge is to extract meaningful bits of information, i.e. a low-dimensional representation relevant to the learning task from this high-dimensional load of data. Traditionally, an approach to achieve this is to use the insights of the system designer into the task at hand, and specify the most important task variables explicitly. Algorithms for extracting their values from the data stream are then usually hand-coded, and parameters are tuned manually. A concrete example from a robotic soccer application is the extraction of task variables for learning to dribble a soccer ball. In [44], the authors specified the velocities in x and y direction, rotation speed, and heading angle of the robot (with respect to some target) as the relevant pieces of information, which could be extracted from the raw data of the camera (using hand coded computer vision approaches) and odometry sensors of a robot in the RoboCup MidSize Soccer League. This representation contained sufficient information in order for the robot to learn quick, space-efficient turns with the ball without loosing it in the process. Similar hand-coded feature extraction pipelines underpin most successful applications of RL to robotics (we refer the interested reader to [21] for a recent review).

However, it is often a tedious, time-consuming process to find these useful features for a task at hand since the design choices underlying this step typically do not generalize over different tasks or control domains. Another limiting factor can be the experience of the system designer who needs to have enough insight into the problem in order to decide what constitutes useful features. Methods that learn representations from data automatically have emerged as a promising alternative; in

This work was partially funded by the German science foundation (DFG) within the priority program SPP 1527.

W. Böhmer · K. Obermayer
Neural Information Processing Group,
Technische Universität Berlin, Sekr. MAR 5-6
Marchstrasse 23, 10587 Berlin, Germay
E-mail: wendelin@ni.tu-berlin.de

J. T. Springenberg · J. Boedecker · M. Riedmiller
Machine Learning Lab, Universtät Freiburg

particular, automatic feature learning with deep convolutional neural networks is now the dominating method when it comes to image classification problems. In the field of robotics, these methods have found application as well, as we describe below. Note that despite the focus on one sensory modality in our examples, i.e. image data, similar problems arise for other modalities, such as tactile information used for solving robot control problems with contacts and even simple readings from joint encoders.

In the following, we will first present *representation-free* approaches, followed by *deep auto-encoder networks* and *slow feature analysis* as emerging techniques to learn state representations. A comparison of approximation properties and practical concerns in learning representations is completed by a discussion of open questions and promising directions of future research.

2 End-to-end reinforcement learning

The most direct approach to avoiding tedious hand crafting of representations is to learn a non-linear control policy directly operating on the raw sensory inputs (often referred to as *end-to-end* learning). The goal of end-to-end learning in RL is to directly train a non-linear function approximator that represents the target function (e.g. a Q-function or a policy π) we care about.

One prominent recent example of this line of work is the Deep Q-Networks (DQN) approach by [37]. Here the authors learn a policy for playing several ATARI games with human level performance directly from pixel images captured from an ATARI simulator. DQN trains a convolutional neural network (CNN) to approximate the highly non-linear Q-function with an online gradient descent approach from a large amount of interactions with the system. Formally this minimizes the squared *Bellman error* [2,51,19], of function $\tilde{Q}_\theta$ with parameters θ, for all training samples $\{z^t, a^t, r^t\}_{t=1}^n$:

$$\min_\theta \sum_{t=1}^{n-1} \Big(r^t + \gamma \max_a \tilde{Q}_\theta(z^{t+1}, a) - \tilde{Q}_\theta(z^t, a^t) \Big)^2, \qquad (1)$$

where $z^t \in \mathcal{Z} \subset \mathbb{R}^d$ denotes the sensor observation at time t a^t, r^t the chosen action and collected reward and γ is the discounting factor.

It should be noted, that the idea of using neural networks as non-linear (Q)-value function approximators has a long history in RL. Successes have, however, mainly been restricted to complicated control problems with low-dimensional inputs [43] and simple toy examples[1] [45,28]. There are several factors allowing DQN to succeed where previous attempts failed: (i) the advent of modern GPU computing allows for training extremely large neural networks on huge datasets (several million example images were used to train DQN). (ii) DQN makes use of a large deep CNN as compared to traditional shallow neural networks, thus having a large representational power while constraining representable functions with insights from image processing. (iii) DQN uses experience replay to circumvent sampling problems[2] that plague online RL.

In the last years several researchers have also considered end-to-end learning of behavioral policies π, represented by general function approximators. First attempts towards this include an actor-critic formulation termed NFQ-CA [14], in which both the policy π and the Q-function are represented using neural networks and learning proceeds by back-propagating the error from the Q-network directly into the policy network – which was, however, only applied to low dimensional control problems. Closely connected to NFQ-CA, recent work on policy gradient algorithms revealed a principled formulation for end-to-end learning of deep neural network policies using a deterministic policy gradient formulation [48]. Notably, this approach was successfully applied to control the high-dimensional problem of controlling a 30-DOF robotic arm. Other recent attempts towards learning neural network policies include applications of joint trajectory optimization and neural network policy learning for robotics problems (see e.g. [27,38]) as well as playing Go from raw visual input [32] and learning attention policies for object recognition [36].

The main advantage of end-to-end learning for RL is that it results in policies, without the need for intermediate representation learning. The main drawback is that end-to-end learning of deep neural network policies from raw visual input often requires thousands or even millions of samples, making these approaches extremely data hungry. This is often not feasible in robotics.

3 Deep representation of states

Keeping the number of necessary training samples low is also the key motivation for heuristic representations. Faced with some (usually continuous) set of possible *observations* $\mathcal{Z} \subset \mathbb{R}^d$ from d sensors, an expert uses his considerable knowledge of the task to specify a mapping $\phi : \mathcal{Z} \to \mathcal{X}$ into a p dimensional *representation* $\mathcal{X} \subset \mathbb{R}^p$. This "knowledge", however, has been inferred from previous experiences.

[1] Perhaps with the exception of TD-Gammon [53], which relied heavily on a well chosen representation as input.

[2] Sampling from trajectories with changing policies leads to non-stationary training distributions and prevents convergence in online gradient descent algorithms.

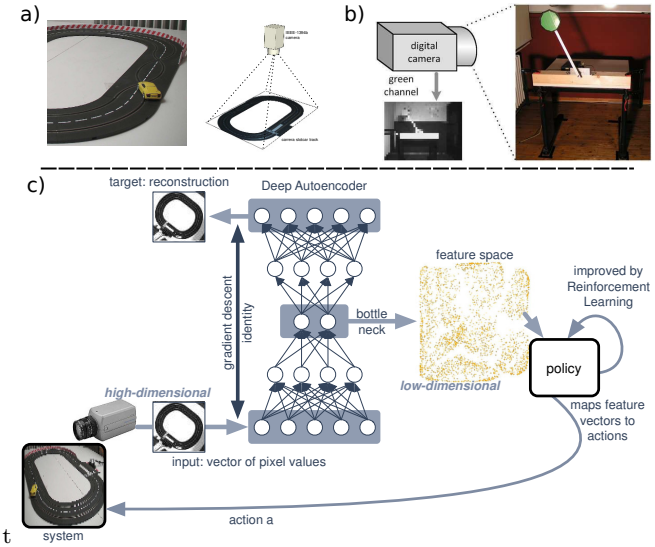

Fig. 1 Depiction of the experimental setup for learning with DFQ for (a) the slot-car scenario [25] and (b) the visual pole task [35]. (c) detailed visualization of representation learning and Q-learning as performed by DFQ.

Analogously, an autonomous robot could infer such a mapping from experiences in previous tasks or passive observations. While numerous unsupervised learning algorithms exist in literature, which could be employed to learn such a representation, the high-dimensionality of the sensory input in the problems we consider makes effective representation learning difficult. This is why only few learning algorithms have been successful in this setting [26,25,16,5].

To exemplify this, let us take a look at one such approach, that was successfully applied to learn a sensory representation for control of a *slot-car racer* on a track [25], as well as an inverted pendulum [35], using pixel information extracted from a high-resolution camera only: the *deep fitted Q* (DFQ) algorithm. The general setup of DFQ for both problems is depicted in Figure 1. It consists of an unsupervised learning component, that first learns to extract the necessary information from the images, as well as a reinforcement learning component, that carries out the task (steering the slot-car around the track or swinging up the pendulum) based on the learned representation. The figure already illustrates several key aspects necessary for successful learning:

1. Since control requires interaction with a real system, learning has to be data efficient. Only a few hundred experiments (resulting in few thousand observations) can be carried out without causing excessive wear of the system.
2. In order to enable efficient reinforcement learning, the learned representation $\phi(z) \in \mathcal{X} \subset \mathbb{R}^p$ has to be of low intrinsic dimensionality p.
3. We assume that each observation $\mathbf{z} \in \mathcal{Z}$ captures all information necessary to describe the state of the system we aim to control.
4. How the state is represented in $\mathcal{X}$ depends on an *optimization problem*, in the case of DFQ an *auto-encoder* [15,3] of the observations $\mathcal{Z}$.

The DFQ algorithm employs *deep neural networks* to represent both the *encoder* $\phi : \mathcal{Z} \to \mathcal{X}$ and the inverse *decoder* $\psi : \mathcal{X} \to \mathcal{Z}$. Pre-training with the auto-encoder minimizes the *least-squares reconstruction error* of all training samples $\{\boldsymbol{z}^t\}_{t=1}^n \subset \mathcal{Z}$ for the first layer:

$$\min_{\phi,\psi} \sum_{t=1}^{n} \left\| \psi\big(\phi(\boldsymbol{z}^t)\big) - \boldsymbol{z}^t \right\|_2^2 \qquad \text{s.t.} \quad p \ll d\,. \tag{2}$$

After convergence, the parameters of the trained layer are frozen and its output is used as reconstruction targets for the next layer. Reducing the number of artificial neurons on each successive layer yields a low-dimensional representation $\mathcal{X}$, that is able to reconstruct the observation and thus must contain the *state*. After this layer-wise training all weights of the complete, stacked, auto-encoder are jointly fine-tuned to improve the reconstruction (post-training).

After learning the encoder network ϕ, a *fitted Q algorithm* is applied to learn an approximate Q function $\tilde{Q} : \mathcal{X} \to \mathbb{R}$ of the control problem[3]. A plethora of function approximators can be utilized to represent $\tilde{Q}$. In the case of DFQ, a clustering based algorithm was used. When applied to the slot-car task and the visual pole swing-up task, this algorithm successfully learns a controller solving the task from raw sensory input. In the case of the slot-car racer the policy is competitive to an experienced human on this task (see Table 1 for a performance comparison).

Similar to other successful applications of unsupervised representation learning to RL, the DFQ approach has several potential weaknesses which we will further discuss in Section 7: (i) in contrast to end-to-end learning, the state representation learned by DFQ is not reward-based and we hence cannot expect the resulting representation to be "goal directed"; (ii) learning auto-encoders for inputs with high variability (i.e. many objects of relevance) can be hard. Both problems could potentially be addressed by adding explicit regularization terms to the formulation from Eq. (2). An interesting recent attempt in that direction is described in [17]. Additionally recent research in the machine learning

[3] The back-propagated Bellman-error could potentially also be used to fine-tune the representation, but both [25] and [35] chose not to adapt the representation to the task.

Performance on Slot-Car

Controller	Time per round	Crash-free
Random	-	No
Constant velocity	$6.408s$	Yes
Experienced Human	$\approx 3s$	Yes
DFQ	$\mathbf{1.869s}$	Yes

Performance on visual swing-up

Controller	Average reward	Success
Random	-1	No
Fitted-Q (True State)	$\mathbf{-0.15 \pm 0.01}$	Yes
DFQ	-0.205 ± 0.075	Yes

Table 1 Performance of the DFQ Algorithm on the slot-car benchmark (top) and on the visual swing-up task (bottom) over 20 trials. The best performance is marked bold. On the swing-up task DFQ performs about as good as a policy learned using fitted-Q iteration using the true state information (pole angular position and velocity). On the slot-car task DFQ completes a successful lap faster than an experienced human. Tables adapted from [25,35].

community has resulted in several auto-encoder variants and deep probabilistic models that might be easier to train (thus addressing problem (ii)) [54,20,42].

Despite these drawbacks DFQ also comes with advantages over end-to-end learning: since the auto-encoder merely learns to reconstruct sampled observations and it can be fed with samples generated by any sampling policy for any task, is thus less susceptible to non-stationarity of the training data. And more importantly, since the learned non-linear embedding into the representation space $\mathcal{X}$ is low-dimensional, an RL algorithm based on $\mathcal{X}$ can succeed using only few samples.

4 Slow feature analysis as state representation

Less restricted by the pitfalls of non-linear optimization, the field of discrete RL has developed their own methodology to learn representations. These fall roughly in two categories: *reward-based* and *subspace-invariant* features [39]. The first type aims to represent the propagated reward (Krylov-bases [41], BEBF [40] and BARB [33]). This allows context-dependent representations, but prohibits the (re-)use of samples from other sources. The second type is reward independent and can *transfer* knowledge from previous tasks [52]. In the following we will introduce two prime examples of such representations to compare them in theory and practice.

Proto-value functions (PVF, [34]) use *Laplacian eigenmaps* (LEM, [1]), a technique from *spectral clustering* [47], to provide a state representation $\mathcal{X}$. The learned features are the smallest eigenvectors of a *connectivity graph*, that is generated from a random-walk through the discrete state space. As this graph is identical for all tasks with the same transition model, PVF can use training data from previous tasks to reduce training time [11,12], similar to *shaping* [49]. Motivated by spectral clustering, a continuous extension of PVF extracts the eigenvectors of a *k-nearest-neighbor* graph [34], to apply PVF to high-dimensional *observation spaces*.

Recent work demonstrated that the unsupervised technique *slow feature analysis* (SFA, [57,56]) approximates LEM in a similar way as PVF [50]. In difference to PVF, however, the spectral encoding of SFA representations is based on the *transition probability* rather than the *connectivity* of states [5]. Instead of extracting eigenvectors explicitly, SFA minimizes the *slowness* of an observed sequence of observations $\{\boldsymbol{z}_t\}_{t=1}^n$ in $\mathcal{X}$:

$$\min_{\{\phi_i\}} \sum_{i=1}^{p} \underbrace{\sum_{t=1}^{n-1} \Big(\phi_i(\boldsymbol{z}^{t+1}) - \phi_i(\boldsymbol{z}^t)\Big)^2}_{\text{slowness of } \phi_i}, \tag{3}$$

under some constraints to avoid trivial or correlated solutions [57]. This objective can be implemented with non-linear function classes, for example, deep convolution neural networks [57,13] or sparse kernel methods [4]. Both methods need to be implicitly or explicitly regularized, though, as SFA is prone to over-fitting.

4.1 Theoretical analysis

Analysis of unrestricted SFA solutions and experiments in simulated environments have demonstrated that non-linear SFA is able to extract the underlying three dimensional state space of a wheeled robot in a static environment [13]. Furthermore, the learned feature space approximates a *Fourier-basis* in this space, which is known to be a universal basis for continuous functions. In this light it is not surprising that there have been many successful attempts using non-linear SFA to learn RL representations from observations, ranging from simple top-down perspective pixel-environments [26,30] to simulated and real-world first-person perspective robot experiments [5].

From a theoretical point of view, SFA and PVF both approximate *subspace-invariant features*. This classification has its origin in the the analysis of approximation errors in linear RL [39]. Here subspace-invariant features induce no errors when the future reward is *propagated* back in time. It can be shown that under these conditions the *least-squares temporal difference* algorithm (LSTD, [9]) is equivalent to supervised *least-squares regression* of the true value function [5]. However, this is only possible for the class of RL-tasks with a *self-adjoint transition model*. As this class is very rare, both SFA and PVF substitute a self-adjoint ap-

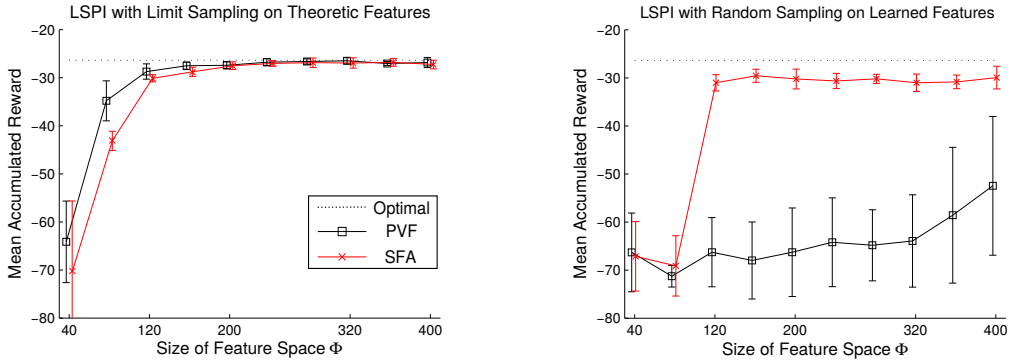

Fig. 2 Performance of LSPI [23] (y-axis) with learned SFA and PVF representations of varying size (x-axis), in the discrete *puddle-world* task. In the left plot representations and policy iteration are based on all state-action pairs, whereas the right plot uses a randomly drawn Markov chain as training set. Mean and standard deviation are w.r.t. state space sizes $\{20 \times 20, 25 \times 25, \ldots, 50 \times 50\}$. The dotted line marks the performance of the optimal policy. Figure modified from [5].

proximation of the transition model to compute *almost* subspace-invariant representations[4].

An analysis of the optimal solution[5] shows that SFA approximates eigenfunctions of the symmetrized transition operator [50]. Moreover, with a Gaussian prior for the reward, one can show that SFA representations minimize a bound on the expected LSTD error of *all tasks in the same environment* [5]. However, as the solution depends on the sampling distribution, straight forward application for transfer learning is less obvious than in the case of PVF. Future works may rectify this with some sensible importance sampling, though.

[4] See [5] for a comparison of SFA/PVF subspace-invariance.

[5] In the limit of infinite training samples, the optimization problem can be analyzed by function analysis in $L^2(\mathcal{Z}, \xi)$.

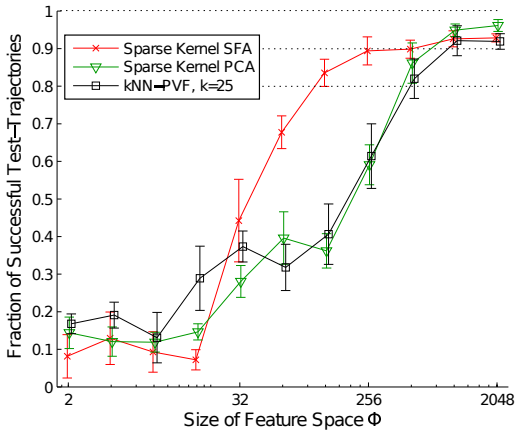

Fig. 3 Mean and standard deviation of the navigation performance in 10 independent training sets (y-axis) in a simulated *robot navigation* task based on first-person perspective images. The LSPI representations of varying size (logarithmic x-axis) have been learned by continuous sparse kernel SFA, PCA and PVF. Dotted lines represent the 80%, 90% and 100% performance levels. Figure modified from [5].

4.2 Empirical comparison

All theoretical arguments presented in this section hold strictly for LSTD only, that is, become invalid when for example *least squares policy iteration* (LSPI, [23]) changes the policy. How do SFA/PVF perform here?

The left side of Figure 2 compares the LSPI performance (mean accumulated reward) of SFA and PVF representations in discrete *puddle-world* tasks [8] of various sizes. Details can be found in [5]. The training set contained all state-action pairs and shows therefore the best possible performance. As number of features increases (x-axis), both representations similarly approach the optimal performance (dotted line). Under ideal conditions both representations are thus coequal. A different picture emerges when the training set is drawn by a random walk, as shown on the right side of Figure 2. Here the SFA representation caves in only slightly, whereas PVF essentially fails[6].

In the more realistic scenario of first-person perspective images from a wheeled robot, rendered in a virtual environment, the difference is less obvious but still visible. Figure 3 shows the LSPI performance (fraction of successful test-trajectories) based on sparse kernel SFA [4], sparse kernel PCA [46] and PVF of a k-nearest-neighbor graph [34]. All algorithms used the same set of 4000 support vectors with the same Gaussian kernel. For further details see [5]. Note that SFA representations reach both the 80% and 90% performance levels with only a quarter of the representation size required by both PVF and PCA, which behave very similar.

In summary, SFA and PVF approximate subspace-invariant features, which are especially suited for linear algorithms like LSTD. In particular SFA seems to be one of the prime contestants for good representations.

[6] It is not entirely clear *why* empirical PVF fail here. One can observe that ideal PVF features have higher frequencies than SFA's, which may be harder to estimate empirically.

5 Properties of good state representations

This section will compare the *properties* of a representation, with no concern how they are *learned*. In general, approximate RL methods based on a value function have common demands on the representation $\mathcal{X}$:

(a) $\mathcal{X}$ must be *Markov* (no partial observability),
(b) $\mathcal{X}$ must be able to *represent the true value* of the current policy well enough for policy improvement,
(c) $\mathcal{X}$ must *generalize* the learned value-function to unseen states with *similar futures* and
(d) $\mathcal{X}$ must be *low dimensional* for efficient estimation.

All discussed methods aim to construct such a representation $\mathcal{X}$ from a high-dimensional observation space $\mathcal{Z}$ with a non-linear mapping $\phi : \mathcal{Z} \to \mathcal{X}$. If $\mathcal{Z}$ is only partially observable, however, $\mathcal{X}$ cannot have the Markov property (a). There are several possibilities to make $\mathcal{Z}$ Markov, for example temporal embedding [37], *liquid state machines* [31], *belief states* [18] or *predictive state representations* [29,55], which each come with their own disadvantages. Temporal embedding, for example, implicitly increases the state space uneccessarily by the history of actions. An adequate discussion of these techniques is beyond the scope of this article and we will in the following assume that $\mathcal{Z}$ has the Markov property.

5.1 Isomorphic representations

We intend to train the representation with data from previous tasks or passive observations, without knowledge of the reward function we will face. As each state may be rewarding, each state must therefore be distinguishable in $\mathcal{X}$ to represent the value function of the current policy. In this case, any representation $\mathcal{X}$ must be an *isomorphism* of $\mathcal{Z}$. The only conceptual difference between isomorphic $\mathcal{X}$ is the *metric* that measures how similar two states are in training and generalization. For example, take the set of all images $\mathcal{Z}$ a camera can record in a specific environment. If this environment is static and diverse enough, each image $z \in \mathcal{Z}$ will correspond to exactly one camera position $x \in \mathcal{X}$ and vice versa. $\mathcal{X}$ is therefore an isomorphism of $\mathcal{Z}$. Note the conceptual equivalence to auto-encoders in Section 3.

Representation $\mathcal{X}$ does not have to be a *vector-space*, though. The mapping $\phi : \mathcal{Z} \to \mathcal{X}$ can also define a *manifold* of representations $\mathcal{X}$, embedded in some p dimensional *feature space* $\mathbb{R}^p$. For linear RL algorithms (like LSTD) such a representation is necessary to fulfill demand (b). The embedding must provide a *functional basis* of the underlying state, able to approximate the value function sufficiently. An example would be a Fourier expansion of the above camera positions [22]. As mentioned in Section 4, non-linear SFA will approximate this feature space [13], which explains the good performance with linear RL algorithms [5].

5.2 Representations encode metrics

Non-linear algorithms like *neural fitted Q-iteration* (NFQ [43]) or *deep Q-networks* (DQN [37]), on the other hand, can in principle work on *any* isomorphic representation $\mathcal{X}$. These algorithms will nonetheless benefit in training and generalization from some embeddings. The reason is the aforementioned *metric*. For example, when a robot navigates between multiple rooms, the underlying space of positions is two-dimensional. Positions on both sides of a wall would *appear* to be very similar, but yet have *dissimilar* values. Any approximate RL algorithm will benefit from a representation that maps these two points far away from each other, but keeps positions the robot can immediately travel to similar.

The Euclidean metric in feature space $\mathcal{X} \subset \mathbb{R}^p$ should therefore be proportional to the travel-distances between states. In stochastic environments one can only compare probability distributions over future states based on a random policy, called *diffusion distances*. It can be shown that SFA approximates eigenfunctions of the symmetrized transition operator, which encode diffusion distances [5]. SFA features are therefore a good representation for non-linear RL algorithms as well.

In summary, SFA representations $\mathcal{X}$ seem *in principle* the better choice for both linear and non-linear RL: non-linear SFA extracts eigenfunctions of the transition model P^π, which are the *same* for every isomorphic observation space $\mathcal{Z}$, encode a diffusion metric that *generalizes* to states with similar futures and approximates a *Fourier basis* of the (unknown) underlying state space.

6 How to learn good state representations

Section 5 argued that SFA representations have outstanding properties for RL. However, in this section we will discuss conceptual problems that limit SFAs applicability and how deep networks can overcome them.

6.1 Slowness and representation size

Learning SFA representations provides challenges that reduce the practical benefits considerably. *Slowness* (Equation 3) is in the limit of an infinite training set

$$\min_{\{\phi_i\}} \quad \sum_{i=1}^{p} \mathbb{E}\left[\left(\phi_i(z') - \phi_i(z) \right)^2 \,\middle|\, \begin{array}{l} z \sim \xi(\cdot) \\ z' \sim P^\pi(\cdot|z) \end{array} \right]. \qquad (4)$$

SFA features depend therefore on the *distribution* ξ of the training samples $z \in \mathcal{Z}$ and on the *sampling policy* π. Empirical studies suggest that both should be as close to uniform distributions as possible [5], which is not feasible if one reuses data from previous tasks. The effect could in principle be balanced out by *importance sampling*, and optimal importance weights are a promising field of future research.

Moreover, a Fourier basis as approximated by SFA grows exponential in the underlying state dimensionality. Linear algorithms, which depend on this basis to approximate the value function, are therefore restricted to low dimensional problems with few or no variables unrelated to the task. Non-linear RL algorithms, on the other hand, could work in principle well with only the first few SFA features of each state-dimension/variable. The *order* in which these variables are encoded as SFA features, however, depends on the *slowness of that variable*. This can in practice lead to absurd effects. Take our example of a wheeled robot, living in a naturally lit room. The underlying state space that the robot can control is three-dimensional, but the image will also depend on illumination, that is, the position of the sun. As the sun is by far the slowest variable, the majority of SFA features will encode its position and (non-existing) interactions with other state variables. This effect makes most of the representation $\mathcal{X}$ unrelated to the controllable state in the presence of *slow distractors*.

6.2 Slowness and deep networks

In contrast, *deep auto-encoder networks*, as introduced in Section 3, do not suffer the above problem. They encode the state-variables according to their influence in reconstructing observations $z \in \mathcal{Z}$, not the slowness of their representation $x \in \mathcal{X}$. On the negative side, deep representations encode metrics that can be arbitrarily bad for function approximation. Also, auto-encoders minimize the squared error over *all* input dimensions of $\mathcal{Z}$ equally. This can produce incomplete representations if a robot, for example, combines observations from a camera with measurements from multiple joints. Due to the large number of pixels, small improvements in the reconstruction of the image can outweigh large improvements in the reconstruction of the joint positions.

Recently an interesting compromise has been proposed: training a neural network with an objective that combines slowness with *predictability* of the successive state [16]. The learned representations are similarly compact as those from an auto-encoder, but encode a diffusion metric. Figure 4 compares these representations at the example of the slot-car task (see Figure 1). The right plot shows how the above objective captures the

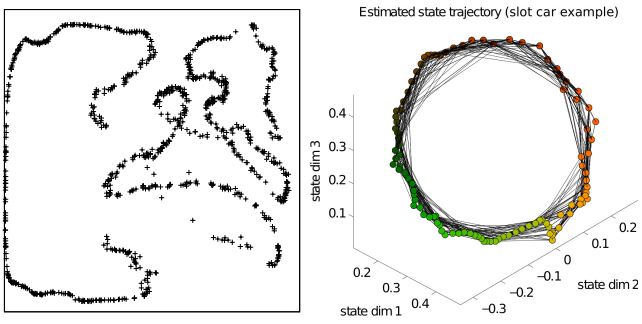

Fig. 4 Representations of the *slot-car* task (Fig. 1) learned by a *deep auto-encoder* from physical experiments (left, from [25]) and by a neural network trained with the objectives *slowness* and *predictability* from a simulated experiment (right, from [16]). Plots reproduced with authors permissions.

circular state metric of the task, which we attribute to its slowness term. The representation of the auto-encoder in the left plot, on the other hand, maps various dissimilar states close-by each other, which complicates the approximation of different values in these states. Furthermore, one can also extend this framework to suppress state-variables *not* related to the task by enforcing the predictability of rewards [17]. This demonstrates one possible way to deal with *slow distractors*.

6.3 Conclusion

We discussed two basic approaches to learn state representations from observations, *slow-feature analysis* and *deep auto-encoder networks*. SFA construct representations with a metric and embedding that is especially suited for linear RL, but also works very well with non-linear RL algorithms. The same properties curse SFA with huge feature spaces, when faced with a high-dimensional underlying state space. Deep networks, on the other hand, produce very compact representations. These do not control the encoded metric, which often complicates value approximation considerably.

Future works may train deep auto-encoder networks similar to [25], that enforce a suitable metric on the representation layer, similar to [16,17]. This could marry the generalization of deep networks with the preferable representation properties of SFA.

7 Outlook

This article presented the emerging field of autonomously learning state representations directly from observations. There are still many unresolved questions; in the following we will present some of the most pressing concerns and most promising research directions.

7.1 Unsolved problems of learning representations

In our opinion, the biggest challenge today is the *curse-of-dimensionality* of observed manifold $\mathcal{Z} \subset \mathbb{R}^d$. Notice that the dimensionality d of observation space $\mathbb{R}^d$ does not pose a problem, if the underlying state space $\mathcal{S}$ is low-dimensional. Adding sensors or pixels does not change the isomorphic state and methods like SFA will approximate the same representation. If the underlying state $\mathcal{S}$ is high dimensional, on the other hand, the discussed learning methods need to sample *all* regions of $\mathcal{S}$. We believe this to be the main reason why unsupervised learning of deep representations does not work for large environments like ATARI games [37], which remain the domain of end-to-end reinforcement learning.

Take the example of uncontrollable distractors like blinking lights or activity outside a window. Each distractor is an independent variable of the isomorphic state $\mathcal{S}$, and to learn an isomorphic representation $\mathcal{X}$ requires thus samples from all possible *combinations* of these variables. The required training samples grow exponentially in the number of distractors. By sacrificing isomorphy, one could suppress some of those variables similar to [17] and thus reduce the training set drastically. However, it is not clear how to *identify* controllable variables without restricting representable tasks.

As discussed for SFA in Section 6, all successful methods rely on *averages* over a training set. If the training distribution ξ or sampling policy π are biased, as one would expect when the observations are generated from previous tasks, large parts of the state space $\mathcal{S}$ would be inadequately encoded. Auto-encoders (see Section 3) are less affected by this, as they do not depend on the sampling policy. A training set uniformly sampled in $\mathcal{S}$ (not in $\mathcal{Z}$) would probably yield the most general representations [5]. If one could estimate such a distribution, its *inverse* would yield optimal importance sampling weights. Alternatively, one could change the objectives from the L^2 to the L^∞ norm. However, optimization in this norm is usually more expensive.

7.2 Factored representations and symbolic RL

If we can overcome the above challenges with some mapping $\phi : \mathcal{Z} \to \mathcal{X}$, we will still face a major underlying problem: it is not feasible to learn tasks in a representation $\mathcal{X}$ of the *full* isomorphic state $\mathcal{S}$ of most environments. $\mathcal{S}$ may simply be too large for sampling. Take the example of a household robot living in a kitchen: each object in the room represents multiple variables of $\mathcal{S}$ that the robot can interact with and that may be necessary for one of the many tasks the robot faces. On the other hand, solving such a task in a representation $\hat{\mathcal{X}}$ of some subset $\hat{\mathcal{S}} \subset \mathcal{S}$ could be feasible, as most of these tasks depend only on few variables.

$$\mathcal{Z} \overset{\text{map}}{\longmapsto} \mathcal{S} \overset{\text{task}}{\longmapsto} \hat{\mathcal{S}} \overset{\text{model}}{\longmapsto} \hat{\mathcal{X}} \overset{\text{rl}}{\longmapsto} \hat{Q}\,.$$

Learning a representation can thus be seen as learning a map ϕ into a set of *almost independent* variables s_i, which compose the isomorphic world state $\boldsymbol{s} \in \mathcal{S}$. Some separate procedure could then choose a small subset of variables $\hat{\mathcal{S}} \subset \mathcal{S}$ that is sufficient to solve the task at hand. A direct approach $\hat{Q} : \hat{\mathcal{S}} \to \mathbb{R}$ to learn the Q-value still requires many samples to estimate the transitions and interactions of variables. Instead, one could learn *independent transition models* for each s_i and *interaction models* between variables. For example, dishes and tables can be manipulated independently, unless one is placed upon the other. Except for those few states, the joint transition model is factorized [7]. For each selection $\hat{\mathcal{S}}$, one can therefore *generate* a representation $\hat{\mathcal{X}}$ that encodes the metric of the joint transition model (with or without interactions) and *learn* $\hat{Q} : \hat{\mathcal{X}} \to \mathbb{R}$, if possible by exploiting factorization [6].

Moreover, a group of variables can also be seen as an instance of a *class of objects*. Similar to clustering, classes could be learned by enforcing that all variables obey a small set of transition models. In a world of dishes and tables, for example, most transitions should be well predictable using only two transition models. Labeling each variable as part of one *object* also allows to use *symbolic RL* algorithms (e.g. relational RL [10, 24]) to select the subset $\hat{\mathcal{S}}$. It may be a long shot, but one could imagine a hierarchical framework that plans far ahead using symbolic RL and solves detailed subproblems in metric subspaces.

Most important, however, is the possibility to *regularize* the optimization of $\phi : \mathcal{Z} \to \mathcal{S}$ by sparse transition models. In the face of large underlying state spaces $\mathcal{S}$, regularization is necessary to keep the demand for samples feasible. If one learns the *state* $\mathcal{S}$ and its *transitions* at the same time, the sparsity of the above transition models would strongly constrain the possible mappings ϕ and therefore require much less training samples. Such a joint optimization will be challenging, but has the potential to *break* the curse-of-dimensionality.

Acknowledgements We would like to thank Sebastian Höfer and Rico Jonschkowski for many fruitful discussions.

References

1. Belkin, M., Niyogi, P.: Laplacian eigenmaps for dimensionality reduction and data representation. Neural Computation **15**(6), 1373–1396 (2003)

2. Bellman, R.E.: Dynamic programming. Princeton University Press (1957)
3. Bengio, Y., Lamblin, P., Popovici, D., Larochelle, H.: Greedy layer-wise training of deep networks. In: Advances in Neural Information Processing Systems (2007)
4. Böhmer, W., Grünewälder, S., Nickisch, H., Obermayer, K.: Generating feature spaces for linear algorithms with regularized sparse kernel slow feature analysis. Machine Learning **89**(1-2), 67–86 (2012)
5. Böhmer, W., Grünewälder, S., Shen, Y., Musial, M., Obermayer, K.: Construction of approximation spaces for reinforcement learning. Journal of Machine Learning Research **14**, 2067–2118 (2013)
6. Böhmer, W., Obermayer, K.: Towards structural generalization: Factored approximate planning. ICRA Workshop on Autonomous Learning (2013). URL `http://autonomous-learning.org/wp-content/uploads/13-ALW/paper_1.pdf`
7. Boutilier, C., Dean, T., Hanks, S.: Decision-theoretic planning: Structural assumptions and computational leverage. Journal of Artificial Intelligence Research **11**, 1–94 (1999)
8. Boyan, J.A., Moore, A.W.: Generalization in reinforcement learning: safely approximating the value function. In: Advances in Neural Information Processing Systems, pp. 369–376 (1995)
9. Bradtke, S.J., Barto, A.G.: Linear least-squares algorithms for temporal difference learning. Machine Learning **22**(1/2/3), 33–57 (1996)
10. Džeroski, S., Raedt, L.D., Drissens, K.: Relational reinforcement learning. Machine Learning **43**, 7–52 (2001)
11. Ferguson, K., Mahadevan, S.: Proto-transfer learning in Markov decision processes using spectral methods. In: ICML Workshop on Transfer Learning (2006)
12. Ferrante, E., Lazaric, A., Restelli, M.: Transfer of task representation in reinforcement learning using policy-based proto-value functions. In: International Joint Conference on Autonomous Agents and Multiagent Systems (2008)
13. Franzius, M., Sprekeler, H., Wiskott, L.: Slowness and sparseness leads to place, head-direction, and spatial-view cells. PLoS Computational Biology **3**(8), e166 (2007)
14. Hafner, R., Riedmiller, M.: Reinforcement learning in feedback control. Machine Learning **27**(1), 55–74 (2011)
15. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science **313**(5786), 504–507 (2006)
16. Jonschkowski, R., Brock, O.: Learning task-specific state representations by maximizing slowness and predictability (2013). URL `http://www.robotics.tu-berlin.de/fileadmin/fg170/Publikationen_pdf/Jonschkowski-13-ERLARS-final.pdf`
17. Jonschkowski, R., Brock, O.: State representation learning in robotics: Using prior knowledge about physical interaction. In: In Proceedings of Robotics: Science and Systems (2014)
18. Kaelbling, L.P., Littman, M.L., Cassandra, A.R.: Planning and acting in partially observable stochastic domains. Artificial Intelligence **101**, 99–134 (1998)
19. Kaelbling, L.P., Littman, M.L., Moore, A.W.: Reinforcement learning: a survey. Journal of Artificial Intelligence Research **4**, 237–285 (1996)
20. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. In: ICLR (2014)
21. Kober, J., Bagnell, D., Peters, J.: Reinforcement learning in robotics: A survey. International Journal of Robotics Research **32**(11), 1238–1274 (2013)
22. Konidaris, G.D., Osentoski, S., Thomas, P.: Value function approximation in reinforcement learning using the Fourier basis. In: Proceedings of the Twenty-Fifth Conference on Artificial Intelligence (2011)
23. Lagoudakis, M.G., Parr, R.: Least-squares policy iteration. Journal of Machine Learning Research **4**, 1107–1149 (2003)
24. Lang, T., Toussaint, M.: Planning with noisy probabilistic relational rules. Journal of Artificial Intelligence Research **39**, 1–49 (2010)
25. Lange, S., Riedmiller, M., Voigtlaender, A.: Autonomous reinforcement learning on raw visual input data in a real world application. In: International Joint Conference on Neural Networks, Brisbane, Australia (2012)
26. Legenstein, R., Wilbert, N., Wiskott, L.: Reinforcement learning on slow features of high-dimensional input streams. PLoS Computational Biology **6**(8), e1000,894 (2010)
27. Levine, S., Abbeel, P.: Learning neural network policies with guided policy search under unknown dynamics. In: Advances in Neural Information Processing Systems (2014)
28. Lin, L.J.: Reinforcement learning for robots using neural networks. Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA (1992)
29. Littman, M.L., Sutton, R.S., Singh, S.: Predictive representations of state. In: In Advances In Neural Information Processing Systems 14 (2001)
30. Luciw, M., Schmidhuber, J.: Low complexity proto-value function learning from sensory observations with incremental slow feature analysis. In: International Conference on Artificial Neural Networks and Machine Learning, vol. III, pp. 279–287. Springer-Verlag (2012)
31. Maass, W., Natschlaeger, T., Markram, H.: Real-time computing without stable states: A new framework for neural computation based on perturbations. Neural Computation **14**(11), 2531–2560 (2002)
32. Maddison, C.J., Huang, A., Sutskever, I., Silver, D.: Move evaluation in go using deep convolutional neural networks. arXiv preprint arXiv:1412.6564 (2014)
33. Mahadevan, S., Liu, B.: Basis construction from power series expansions of value functions. In: Advances in Neutral Information Processing Systems, pp. 1540–1548 (2010)
34. Mahadevan, S., Maggioni, M.: Proto-value functions: a Laplacian framework for learning representations and control in Markov decision processes. Journal of Machine Learning Research **8**, 2169–2231 (2007)
35. Mattner, J., Lange, S., Riedmiller, M.: Learn to swing up and balance a real pole based on raw visual input data. In: Proceedings of the 19th International Conference on Neural Information Processing (5) (ICONIP 2012), pp. 126–133. Dohar, Qatar (2012)
36. Mnih, V., Hees, N., Graves, A., Kavukcuoglu, K.: Recurrent models of visual attention. In: Advances in Neural Information Processing Systems (2014)
37. Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M.: Playing atari with deep reinforcement learning. In: NIPS Deep Learning Workshop (2013)
38. Mordatch, I., Todorov, E.: Combining the benefits of function approximation and trajectory optimization. In: Proceedings of Robotics: Science and Systems (RSS) (2014)
39. Parr, R., Li, L., Taylor, G., Painter-Wakefiled, C., Littman, M.L.: An analysis of linear models, linear value-function approximation, and feature selection for rein-

forcement learning. In: International Conference on Machine Learning (2008)
40. Parr, R., Painter-Wakefield, C., Li, L., Littman, M.: Analyzing feature generation for value-function approximation. In: International Conference on Machine Learning (2007)
41. Petrik, M.: An analysis of Laplacian methods for value function approximation in MDPs. In: International Joint Conference on Artificial Intelligence, pp. 2574–2579 (2007)
42. Rezende, D.J., Mohamed, S., Wierstra, D.: Stochastic backpropagation and approximate inference in deep generative models. In: International Conference on Machine Learning (2014)
43. Riedmiller, M.: Neural fitted q iteration - first experiences with a data efficient neural reinforcement learning method. In: 16th European Conference on Machine Learning, pp. 317–328. Springer (2005)
44. Riedmiller, M., Gabel, T., Hafner, R., Lange, S.: Reinforcement learning for robot soccer. Autonomous Robots **27**(1), 55–74 (2009)
45. Sallans, B., Hinton, G.E.: Reinforcement learning with factored states and actions. Journal of Machine Learning Research **5**, 1063–1088 (2004)
46. Schölkopf, B., Smola, A., Müller, K.R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Computation **10**(5), 1299–1319 (1998)
47. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22**(8), 888–905 (2000)
48. Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., Riedmiller, M.: Deterministic policy gradient algorithms. In: The 31st International Conference on Machine Learning (ICML 2014) (2014)
49. Snel, M., Whiteson, S.: Multi-task reinforcement learning: Shaping and feature selection. In: European Workshop on Reinforcement Learning, pp. 237–248 (2011)
50. Sprekeler, H.: On the relationship of slow feature analysis and Laplacian eigenmaps. Neural Computation **23**(12), 3287–3302 (2011)
51. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. MIT Press (1998)
52. Taylor, M.E., Stone, P.: Transfer learning for reinforcement learning domains: A survey. Journal of Machine Learning Research **10**, 1633–1685 (2009)
53. Tesauro, G.: Temporal difference learning and td-gammon. Commun. ACM **38**(3), 58–68 (1995)
54. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. J. Mach. Learn. Res. (JMLR) **11**, 3371–3408 (2010)
55. Wingate, D., Singh, S.P.: On discovery and learning of models with predictive representations of state for agents with continuous actions and observations. In: International Joint Conference on Autonomous Agents and Multiagent Systems, pp. 1128–1135 (2007)
56. Wiskott, L.: Slow feature analysis: a theoretical analysis of optimal free responses. Neural Computation **15**(9), 2147–2177 (2003)
57. Wiskott, L., Sejnowski, T.: Slow feature analysis: unsupervised learning of invariances. Neural Computation **14**(4), 715–770 (2002)

Wendelin Böhmer received his Diploma in computer science from the Technische Universität Berlin, Germany, in 2009, where he also started as a PhD student in Klaus Obermayers neural information processing group. He received a scholarship of the Human Centric Communication Center and worked later on two DFG funded project within SPP-1527 *autonomous learning*. He wrote 2 journal papers about slow feature analysis as representation and participated in two articles about cognitive modeling.

His research focuses on generalization to unseen situations in autonomous reinforcement learning. His interests are primarily in representations and computational issues, but extend to applications in robotics, games and cognitive modeling as well.

Klaus Obermayer received his Diplom degree in physics in 1987 from the University of Stuttgart, Germany, and the Dr. rer. nat. degree in 1992 from the Department of Physics, Technical University of Munich, Germany. From 1992 and 1993 he was a postdoctoral fellow at the Rockefeller University, New York, and the Salk Institute for Biological Studies, La Jolla, USA. From 1994 to 1995 he was member of the Technische Fakultät, University of Bielefeld, Germany. He became associate professor in 1995 and full professor in 2001 at the Department of Electrical Engineering and Computer Science of the Berlin University of Technology, Germany. He is head of the Neural Information Processing Group and member of the steering committee of the Bernstein Center for Computational Neuroscience Berlin. He was member of the governing board of the International Neural Network Society from 2004 - 2012 and was Vice-President of the Organisation for Computational Neuroscience from 2008-2011. From 1999-2003 he was one of the directors of the European Advanced Course of Computational Neuroscience.

His current areas of research are computational neuroscience, artificial neural networks and machine learning, and the analysis of neural data. He co-authored more than 250 scientific publications.

Tobias Springenberg is a PhD student in the machine learning lab at the University of Freiburg, Germany, supervised by Prof. Martin Riedmiller. Prior to starting his PhD, Tobias studied Cognitive Science at the University of Osnabrück, earning his BSc in 2009. From 2009-2012 he then went to obtain a MSc in Computer Science from the University of Freiburg, focusing on representation learning with deep neural networks for computer vision problems.

His research interests include machine learning, especially representation learning, and learning efficient control strategies for robotics.

Joschka Boedecker studied computer science at the University of Koblenz-Landau, Germany, and artificial intelligence at the University of Georgia, USA. He did his PhD in engineering at the department of adaptive machine systems at Osaka University, Japan, receiving the degree in 2011. From 2011-2012, he continued to work at the same department as a postdoctoral researcher, before starting his second, and ongoing postdoc at the Machine Learning Lab, department of computer science, at University of Freiburg in early 2013.

His research interests include machine learning, especially (recurrent) neural networks and reinforcement learning, as well as learning for robotics.

Martin Riedmiller studied Computer Science at the University of Karlsruhe, Germany, where he received his PhD in 1996. In 2002 he became a professor for Computational Intelligence at the University of Dortmund, from 2003 to 2009 he was heading the Neuroinformatics Group at the University of Osnabrück. Since April 2009 he is a professor for Machine Learning at the Albert-Ludwigs-University Freiburg. He was participating with his teams in the RoboCup competitions from 1998 to 2009, winning 5 world championship titles and several European championships.

His research interests include machine learning, neural networks, reinforcement learning and robotics.