Estimating Cartesian Compression via Deep Learning

András Lőrincz, András Sárkány, Zoltán Á. Milacski, and Zoltán Tősér

Faculty of Informatics, Eötvös Loránd University,
Pázmány Péter sétány 1/C, Budapest H-1117, Hungary

Abstract. We introduce a learning architecture that can serve compression while it *also* satisfies the constraints of factored reinforcement learning. Our novel Cartesian factors enable one to decrease the number of variables being relevant for the ongoing task, an exponential gain in the size of the state space. We demonstrate the working, the limitations and the promises of the abstractions: we develop a representation of space in allothetic coordinates from egocentric observations and argue that the lower dimensional allothetic representation can be used for path planning. Our results on the learning of Cartesian factors indicate that (a) shallow autoencoders perform well in our numerical example and (b) if deeper networks are needed, e.g., for classification or regression, then sparsity should also be enforced at (some of) the intermediate layers.

1 Introduction

An intriguing fact about intelligence is the following. Our scientific discoveries have a history of about $15{,}000$ years. The same holds for the technological developments and this progress was made by billions of people. Still, this knowledge, or a large part of it, can be passed to a child in 15 years. It looks that innovations and discoveries take long, whereas explaining and proving them are much faster. One may consider the complexity of optimization and problem solving. They may scale polynomially or exponentially both from the point of view of solving them and verifying them (see, e.g., [4] and the cited references). The polynomial type is called easy and the exponential type is called hard. Out of the four options, the *hard* to solve and *easy* to verify — such as the Traveling Salesman Problem — can be particularly useful for communicating agents. The relevance of the other three classes is less. It thus seems that human knowledge is concerned with hard to solve and easy to verify problems. After all, mathematical theorems are hard to discover, but the proof is 'linear'.

Consider sensory information. It brings about the possibility of information fusion from different sensory modalities. For example, smell may be associated with tasty food and then the smell can be used for searching for food. Thus, information fusion concerns Cartesian factors (e.g., smell and taste) and spatio-temporal patterns (e.g., during hunting the smell predicts the taste). Smell and taste have their respective sensors, unlike many Cartesian factors. The creation (or the derivation) of such factors is of high importance for lowering the number of relevant variables in problem solving.

The concept of numbers is such a Cartesian abstraction; it represents the quantity and separates material properties. '$2+2=4$', no matter if we are concerned with apples or peaches. Cartesian abstractions enable concise formulations for similar tasks.

We turn to the constraints of goal-oriented behavior. Factored reinforcement learning (FRL) is advantageous [15, 32], since without factored description, the state space explodes. Factors of FRL are Cartesian ones [17, 3] and can be latent non-linear *coordinates*. Such abstractions are considered hard, 'although potentially pseudo-polynomial' [7]. We postulate that communication favors such abstractions, provided that the factor, in a given task, enables the agent to neglect other components of the information without high risks, e.g., by applying robust controllers [33, 36].

In our view, creative intelligence finds those factors that suit the essence of the problem and diminish the combinatorial explosion for an approximate FRL framework. In turn, the views that compression is related to intelligence, e.g., the proposal that compression is one of the driving forces of science [28], the thought that compression may lead to AGI[1] together with the warning that intelligence is not simply compression [8] may all gain support through the concept of Cartesian factors.

Contributions: we treat the Cartesian factor problem by means of deep networks. We assume that we are given a factor and we find the complementing one without exploiting temporal information. The paper is constructed as follows. In the next section, we shall review similar efforts. Section 3 provides the details about the deep learning formalism and the problem itself. Our results are presented in Sect. 4 and discussed in Sect. 5. Conclusions are drawn in Sect. 6.

2 Related Works

Factor or component learning appears in a number of contexts, such as factor analysis (FA), nonlinear matrix factorization (NMF), principal/independent component analysis (PCA/ICA) (see, e.g., [16] and the references therein), sparse coding (SC), also in bilinear forms [34]. The literature is enormous and goes beyond the scope of this paper. These methods have the following properties: (i) they assume the structure of the data (NMF, ICA), (ii) they drop components of small variances (FA, PCA), and (iii) they are essentially additive factor models. Generative models with specified joint probability distributions also form a group of models [27]. By contrast, Cartesian factors are more like *modalities*, can have a *metric*, and may be *deterministic*.

'Place field' cells observed in rat hippocampus [24], also known as the 'cognitive map' [25], motivate our thinking, since place fields are independent from the egocentric direction of the animal and thus, it could be a modality, but it is not sensed directly. The explanation of the place field phenomenon – that gave rise to the Nobel Prize in 2014 – motivates our demonstration. We mention two of the many models and refer the interested reader to the literature. One model starts from ICA on directed spatially local but dispersed information, derives direction dependent place fields, develops an autoregressive (AR) predictor from those and produces direction independent place fields from measuring the innovation of the AR process [19]. However, this method does not work for distant cues. The other model uses distant information and takes advantage of Slow Feature Analysis (SFA), see [29] and the references therein. This method needs a relatively large angle of view to derive the place fields. Both methods exploit temporal information, a strong restriction for the general case.

[1] `http://prize.hutter1.net/`

Here, we assume that we have one Cartesian factor and we will develop a potentially deep architecture that can robustly discover the complementing one.

3 Methods

3.1 Theoretical Background

Deep Autoencoders A *Deep Autoencoder* [12, 38] is the unsupervised version of a modern Multilayer Perceptron (MLP). Consider a series of non-linear mappings (layers) of the form:

$$\boldsymbol{H} = f_N\Big(\cdots f_2\big(f_1(\boldsymbol{X}\boldsymbol{W}_1)\boldsymbol{W}_2\big)\cdots \boldsymbol{W}_N\Big), \tag{1}$$

where $\boldsymbol{X} \in \mathbb{R}^{I\times J}$ is the matrix of I inputs of size J, $\boldsymbol{W}_n \in \mathbb{R}^{Q_{n-1},Q_n}$ are parameters with $Q_0 = J$, and f_n are non-linear almost everywhere differentiable element-wise functions ($n = 1, \dots, N$; $N \in \mathbb{N}$). Then $\boldsymbol{H} \in \mathbb{R}^{I\times Q}$ is called the feature map ($Q_N = Q$). Typically, one takes two such mappings with reversed sizes — an encoder and a decoder — and composes them. Thereupon one can define a so-called reconstruction error between the encoder input $\boldsymbol{X}$ and the decoder output $\widehat{\boldsymbol{X}} \in \mathbb{R}^{I\times J}$ (normally the ℓ_2 or Frobenius norm of the difference, i.e., $\frac{1}{2}\|\boldsymbol{X} - \widehat{\boldsymbol{X}}\|_F^2 = \frac{1}{2}\sum_{i=1,\dots,I}\sum_{j=1,\dots,J}(\boldsymbol{X}_{i,j} - \widehat{\boldsymbol{X}}_{i,j})^2$) and try to find a local minima of it in terms of parameters $\boldsymbol{W}_n$ after random initialization, by taking advantage of a step-size adaptive mini-batch subgradient descent method [9, 39, 18]. The non-linearity can be chosen to be the rectified linear function $f_n(x) = x \cdot \mathbb{I}(x > 0)$ for $x \in \mathbb{R}$ [22, 5] to avoid the vanishing gradient problem [13, 14], where $\mathbb{I}$ designates the indicator function.

Spatial and Lifetime Sparsity Deep Autoencoders are often used as a pretraining scheme [10] or as a part of supervised algorithms [26], but they lack the ability to learn a meaningful or simple data representation without prior knowledge [30]. To obtain such a description, one might add regularizers or constraints to the objective function [11, 1], or employ a greedy procedure [37, 6]. It is well known that minimizing the sum of ℓ_2 norms of parameters $\boldsymbol{W}_n$ can reduce model complexity by yielding a dense feature map, and similarly, the ℓ_1 variant may result in a sparse version [35, 23]. An alternative possibility is to introduce constraints in the non-linear function f_n. For example, one may utilize a k-sparse representation by keeping solely the top k activation values in feature map $\boldsymbol{H}$, and letting the rest of the components zero [20]. This case, when features, i.e., the components of the representation, compete with each other is referred to as *spatial sparsity*. Input indices of the representation may also go up against each other and this case is called *lifetime sparsity* [21].

3.2 Problem Formulation

We assume that a latent random variable Z and an observed random variable Y are continuous and together they fully explain away another observed binary random variable X. The ranges of Z and Y are supposed to be grid discretized finite r- and one-dimensional intervals, respectively, that fits FRL. We denote the resulting grid points by $(z^{(m)}, y^{(l)}) \in \mathbb{R}^r \times \mathbb{R}$; $l = 0, \ldots, L$, $m = 1, \ldots, (M+1)^r$, $L, M, r \in \mathbb{N}$. The indices $m = 1, \ldots, (M+1)^r$ are supposed to be scrambled throughout training (i.e., we assume no topology between $z^{(m)}$). Then observation $\boldsymbol{x}^{(m,l)} \in \{0,1\}^d$ is generated by a highly non-linear function $g\colon \mathbb{R}^r \times \{1, \ldots, L\} \to \{0,1\}^d$ from grid point $z^{(m)}$ and grid interval $[y^{(l-1)}, y^{(l)})$ as

$$\boldsymbol{x}^{(m,l)} = g(\boldsymbol{z}^{(m)}, l) \tag{2}$$

for $m = 1, \ldots, (M+1)^r$; $l = 1, \ldots, L$. For each fixed m, one is given masks $\boldsymbol{V}_{i,\cdot} \in \{0,1\}^L$; $\sum_{l=1}^{L} \boldsymbol{V}_{i,l} = v \in \mathbb{N}$ indexing pairs of the form $(l, \boldsymbol{x}^{(m,l)})$, where $i = 1, \ldots, I$ is a global index. Provided such a sample from Y and X, we aim to approximate the discretized version of Z.

The variables may correspond to discretized latent points in space (Z); direction intervals in an allothetic system, e.g., a compass with limited resolution (Y); and egocentric views of distant cues within a viewing angle (X), see Fig. 1 for an illustration.

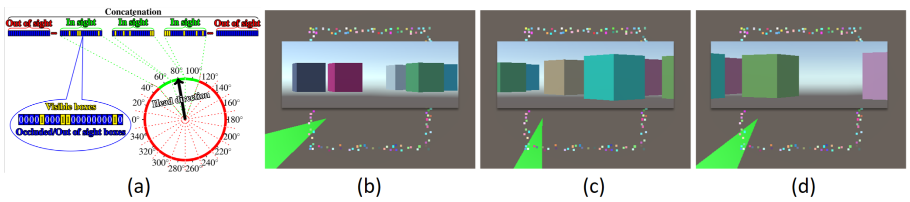

Fig. 1: Numerical experiment. (a): input is concatenated from sub-vectors, which belong to different allothetic directions. A given index corresponds to the same box, the ‘remote visible cue’, in all sub-vectors. The value of the a component of a sub-vector is 1 (0) if the box is visible (non-visible) in the corresponding direction. Three directions are visible (green). Some boxes may be present in neighboring sub-vectors, since they are large. (b)-(d): the ‘arena’ from above with the different boxes around it. Shaded green areas in (b), (c), and (d), show the visible portions within the field of view at a given position with a given head direction. Insets show the visual information for each portion to be transformed to 1s and 0s in the respective components of the sub-vectors. Components of out-of-view sub-vectors are set to zero.

We formulated the above problem as a multilayer feedforward *lifetime sparse autoencoding* [21] procedure with input matrix $\boldsymbol{X} \in \{0,1\}^{I \times J}$ utilizing two novelties: concatenated input vectors and a masked loss function are motivated by the input structure. In order to construct the inputs $\boldsymbol{X}_{i,\cdot}$; $i = 1, \ldots, I$ of size $J = L \cdot d$, we coupled each v-tuple of $\boldsymbol{x}^{(m,l)}$ vectors for fixed m into a single block-vector using the $\boldsymbol{V}_{i,\cdot}$ values as follows:

$$\boldsymbol{X}_{i,\cdot} = \left[\boldsymbol{V}_{i,1} \cdot \boldsymbol{x}^{(m,1)}, \ldots, \boldsymbol{V}_{i,l} \cdot \boldsymbol{x}^{(m,l)}, \ldots, \boldsymbol{V}_{i,L} \cdot \boldsymbol{x}^{(m,L)}\right]. \tag{3}$$

Then, we used the ℓ_2 reconstruction error as the loss, but on a restricted set of elements, namely, on the v non-zero blocks for each input:

$$l(\boldsymbol{X}, \widehat{\boldsymbol{X}}, \boldsymbol{V}) := \frac{1}{I} \sum_{\substack{i=1,\ldots,I \\ j=1,\ldots,J}} \boldsymbol{V}_{i,\lfloor \frac{j-1}{d}+1 \rfloor} \cdot (\boldsymbol{X}_{i,j} - \widehat{\boldsymbol{X}}_{i,j})^2 \tag{4}$$

where $\widehat{\boldsymbol{X}}$ denotes the output of the decoder network. Finally, a sparse non-linearity was imposed on top of each encoder layer, which selected the k percent topmost activations across one component. We applied both lifetime [21] and spatial sparsification [20].

We expected the output of the encoder $\boldsymbol{H} \in \mathbb{R}^{I \times Q}$ to resemble $\boldsymbol{z}^{(m)}$, i.e., not depending on $y^{(l)}$ and discretizing the latent space $\mathbb{R}^r$. The procedure is summarized in Fig. 2.

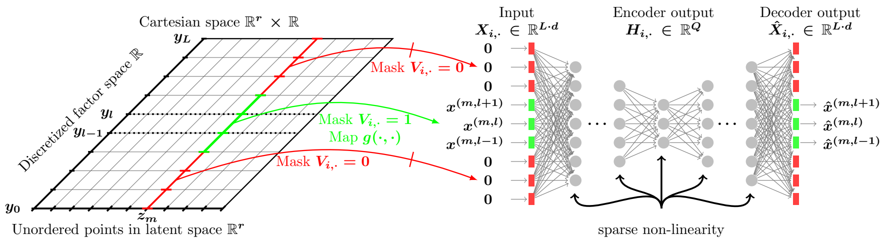

Fig. 2: General architecture. In the numerical experiments the notations correspond to the following quantities: Z latent positions, Y discretized 'compass' values. Input to the network: red: not visible, green: visible at neighboring viewing angle ranges. Each viewing angle range provides inputs about all boxes. The full input equals to the 'No. of boxes $\times$ No. of viewing angle ranges'. See text for details of the autoencoder.

We implemented our method in the Python library Theano [2] based upon the SciPy2015 GitHub repository[2]. We used multilayer autoencoders with rectified linear units, $k = 1$ spatial sparsity and $p\%$-sparse lifetime sparsity. The decoder output layer was linear. We trained parameters using full-batch Adam gradient descent [18] with early stopping (maximum 10,000 epochs with 500 epochs patience).

3.3 Numerical Experiment

For our study, we generated a squared 'arena' surrounded by $d = 150$ boxes in Unity[3] 3D game engine (Fig. 1). The 'arena' had no obstacles. Boxes were placed pseudo-randomly: they did not overlap. The 'arena' was discretized by an $M \times M = 36 \times 36$ grid. From each grid point and for every 20°, a 28° field of view was created (i.e.,

[2] `https://github.com/kastnerkyle/SciPy2015`
[3] `https://unity3d.com`

$L = \frac{360°}{20°} = 18$, overlap: 4° between regions), and the visibility — a binary value (0 for occlusion or out of the angle of view) — for each box was recorded, according to Eq. (2); we constructed a total of $I = 37 \cdot 37 \cdot 18 =$ 24,642 binary ($\boldsymbol{x}^{(m,l)}$) vectors.

These vectors were processed further. Beyond the actual direction of the center of the viewing angle, we introduced some degree of closeness about the input regarding the direction, but not the position: we varied the viewing angle between 28° and 360°. Formally, for various experiments, we defined masks $\boldsymbol{V}_{i,\cdot}$ summing to $v = 1, 3, \ldots, 17, 18$, for which we carried out the concatenation method from Eq. (3): for each visible 28° sector, the corresponding $\boldsymbol{x}^{(m,l)}$ vector; while for all non-visible sectors, an all-zero vector were appended. This manipulation did not change the size of the database.

In some experiments we normalized the inputs to unit ℓ_2 norm for each $d = 150$ dimensional components, provided that at least one of the components differed from zero. This is called normalized experiment. We used spatial sparsification with $k = 1$ and lifetime sparsification with $p = 3.33\%$ and $p = 6.66\%$. In the error of the autoencoder we considered two options: (a) error of the full output and (b) error only on the visible components that belonged to the viewing angle as in Eq. (4). This latter is called masked experiment. We used them in combination. We also tried 3 and 5 layer autoencoders, with the middle layer representing the latent variables.

The size of the middle layer was always $Q = 30$. This means that spatial (i.e., latent component-wise) sparsity gave rise to 3.33% lifetime sparsity. On the other hand, $p = 3.33\%$ lifetime sparsity was effectively larger than 3.33% since it was possible that none of the latent unit was selected for a given input (and thus all of them were set to zero), when backpropagation became ineffective. The same holds for $p = 6.66\%$ lifetime sparsity, which, on the average, would give rise to 1, 2, 3, or more non-zero latent units with average above 2. The sizes of the hidden layers were spaced linearly between 2,700 and 30.

4 Results

The dependencies of the responses in the hidden representation vs. space and direction are shown in Fig. 3 and Fig. 4. Linear responses of randomly selected latent units for different algorithms are depicted in Fig. 3, illustrating the extent that the responses were localized. Figure 4 shows the direction (in)dependence of the responses. For each input, we chose the highest activity latent component and in each position we computed the number of directions (out of the 18 possible) that a neuron was the winner in the dataset. We computed these numbers for all units, selected the largest values at each position and combined them in a single figure that we color coded (Fig. 4): Black color at one position means that unit won in all angles at that position. Lighter colors mean less winning directions.

One should ask (i) if the linear responses are local and activities far from the position of the peak activity are close to zero; (ii) if the number of dead latent units is small, (iii) if responses are direction independent, that is, if we could derive the discretization of space in allothetic coordinates, the complementary component of the egocentric direction. We found that spatial sparsity with the 3 layer network and the 5 layer network with dense 2^{nd} and 4^{th} layers rendered the output of some or sometimes all hidden units

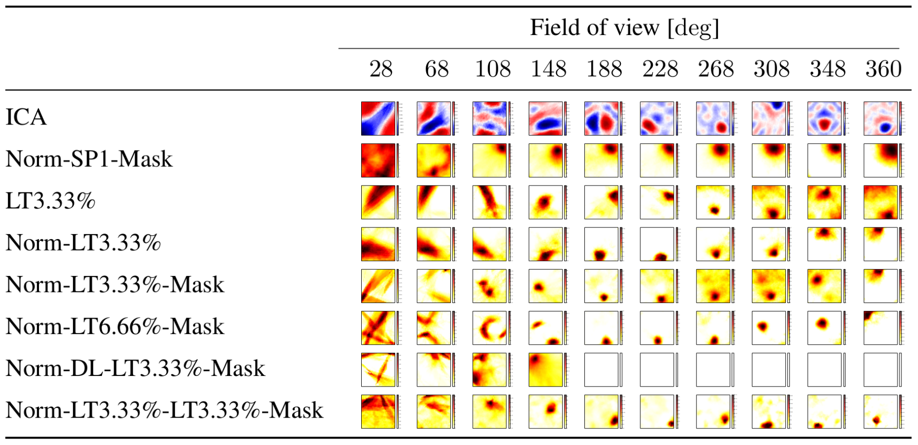

Fig. 3: Linear responses of individual latent units selected randomly: we chose neuron with index 2 from the latent layer. ICA: values may take positive and negative values. Other experiments: all units are ReLUs, except the output, which is linear. Color coding represents the sum of responses for all directions at a given point. SP1: spatial sparsity with $k = 1$, LT3.3%: lifetime sparsity = 3.3%, Norm: for each 150 components, the ℓ_2 norm of input is 1 if any of the components is non-zero, Mask: autoencoding error concerns only the visible part of the scene, DL: dense layer. 'Norm-LT3.3%-LT3.3%-Mask' means normed input, masked error, 5 layers: input layer, 3 layers with LT sparsity equals 3.3%, output layer.

Table 1: Dead neuron count: number of non-responsive computational units.

	Field of view [deg]									
	28	68	108	148	188	228	268	308	348	360
Norm-SP1-Mask	2	0	5	5	10	12	16	18	15	18
LT3.33%	0	0	0	0	0	2	2	6	8	9
Norm-LT3.33%	0	0	0	1	1	3	2	4	9	11
Norm-LT3.33%-Mask	0	0	0	0	0	0	1	2	7	11
Norm-LT6.66%-Mask	0	0	0	0	0	0	1	4	13	13
Norm-DL-LT3.33%-Mask	0	3	1	29	30	30	30	30	30	30
Norm-LT3.33%-LT3.33%-Mask	0	0	0	0	0	0	0	0	0	0

to zero (Table 1). On the othe hand, lifetime sparsity with the 5 layer network produced excellent results. Lifetime sparsity $p = 6.66\%$ can still produce place fields. Note that local responses appear without the mask, but only for very large viewing angles. For the sake of completeness, we also provide the ICA responses in Fig. 3. We discuss the relevance, the limitations and the promises of our results in the next section.

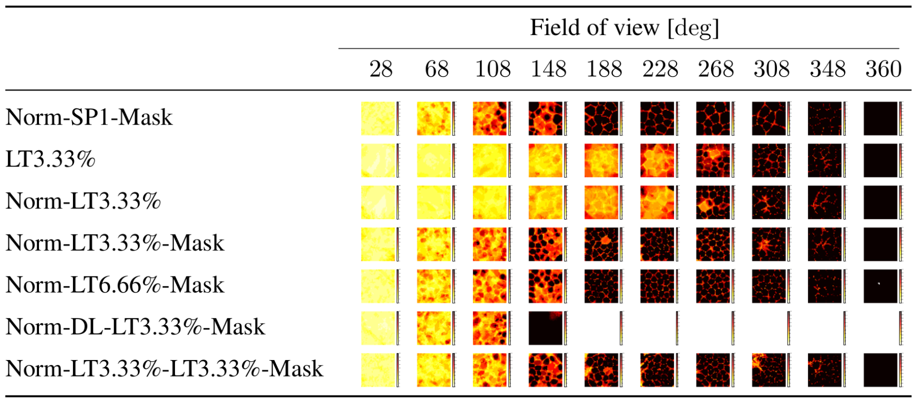

Fig. 4: Angle independence. Notations are the same as in Fig. 3. The highest activity (winning) unit was selected for each input at each position in each direction. We counted the number of wins at each position for each unit and selected the largest number. Results are color coded. Black (18): there is a single winner for all angles at that position. White (0): no response at that point from any neuron in any direction. Values between 1 and 17: the darker the color the larger the direction independence for the best winner at that position.

5 Discussion

Our goal was to find a discretization of hidden Cartesian factors (coordinates) provided that we already have one. The coordinate that we know must have a metric by definition. Such metric may show up in the temporal domain or, alternatively, it may manifest itself implicitly, via discretized spatial information, such as neighboring viewing ranges in our case. Temporal information has been exploited via hierarchical SFA (see, [29] and the cited references). Their goal was to closely model the rat's sensory system and found that indeed, a large viewing angle of 320° gave rise to direction independent place fields.

Here, we could reach about 108°, or so, and according to our results, further improvements can be expected for deeper networks, provided that sparsity is kept at some layers. Our algorithms can work if temporal information is *not* available. Deeper networks may be unavoidable, e.g., if the input structure we used can not be prewired and should be learned. We highlight that (i) high quality reconstruction is corrupted without the mask, but our method shows robustness for this case, too and (ii) if temporal information *is* available then our results may be improved further.

Cartesian factors are needed for FRL. Consider, e.g., that place fields uncover neighboring graph that can be used both for path planning. An integrated path planning and control architecture on place fields has been put forth some years ago [31]. For path planning, the exponent of the state space is lowered, being a critical issue for FRL.

Lifetime sparsity seems important in the development of the Cartesian factor, here, for the place fields. However, real time operation requires spatial sparsity, or possibly some thresholding, or even the linear mode, since responses are fairly local for the linear mode, e.g., $p = 6.66\%$ and for the 5 layer network. The linear mode or even some spatial sparsification uncovers neighbor relations useful for path planning and may support the development of metrics. In turn, two types of operations, namely, learning off-line when lifetime sparsity can be enforced, and working real time with some thresholding for learning neighboring relations seem favorable.

6 Conclusions

We have derived a complementary Cartesian component in discretized form to an existing one by means of deep learning. This novel type of compression supports factored reinforcement learning. Our experimental studies point towards two phase – off-line and a real-time – operation and also to the necessity of sparsification. The novel compression method may have relevance for Big Data and may support the understanding of intelligence, since compression is thought to be closely related to it.

Acknowledgments

This work was supported by the EIT Digital grant (Grant No. 16257). Helpful comments from Gábor Szirtes are gratefully acknowledged.

References

1. Becker, S.R., Candès, E.J., Grant, M.C.: Templates for convex cone problems with applications to sparse signal recovery. Math. Prog. Comp. 3(3), 165–218 (2011)
2. Bergstra, J., Breuleux, O., Bastien, F., Lamblin, P., Pascanu, R., Desjardins, G., Turian, J., Warde-Farley, D., Bengio, Y.: Theano. In: Python Sci. Comp. vol. 4, p. 3. Austin, TX (2010)
3. Boutilier, C., Dearden, R., Goldszmidt, M.: Stochastic dynamic programming with factored representations. Artif. Intel. 121(1), 49–107 (2000)
4. Culberson, J.C.: On the futility of blind search: An algorithmic view of no free lunch. Evol. Comp. 6(2), 109–127 (1998)
5. Dahl, G.E., Sainath, T.N., Hinton, G.E.: Improving deep neural networks for LVCSR using rectified linear units and dropout. In: Acoust., Speech Sign. Proc. (ICASSP), 2013. pp. 8609–8613. IEEE (2013)
6. Dai, W., Milenkovic, O.: Subspace pursuit for compressive sensing signal reconstruction. Info. Theo. 55(5), 2230–2249 (2009)
7. Daswani, M., Sunehag, P., Hutter, M.: Feature reinforcement learning: state of the art. In: Sequential decision-making with big data: AAAI-14. Assoc. Adv. Artif. Intel. (2014)
8. Dowe, D.L., Hernández-Orallo, J., Das, P.K.: Compression and intelligence: social environments and communication. In: Artif. Gen. Intell., pp. 204–211. Springer (2011)
9. Duchi, J., Hazan, E., Singer, Y.: Adaptive subgradient methods for online learning and stochastic optimization. J. Mach. Learn. Res. 12, 2121–2159 (2011)
10. Erhan, D., Bengio, Y., Courville, A., Manzagol, P.A., Vincent, P., Bengio, S.: Why does unsupervised pre-training help deep learning? J. Mach. Learn. Res. 11, 625–660 (2010)
11. Grant, M., Boyd, S.: CVX: Matlab software for disciplined convex programming, version 2.1. `http://cvxr.com/cvx` (Mar 2014)
12. Hinton, G.E., Salakhutdinov, R.R.: Reducing the dimensionality of data with neural networks. Science 313(5786), 504–507 (2006)

13. Hochreiter, S.: Untersuchungen zu dynamischen neuronalen netzen. Master's thesis, Institut für Informatik, Technische Universität, München (1991)
14. Hochreiter, S., Bengio, Y., Frasconi, P., Schmidhuber, J.: Gradient flow in recurrent nets: the difficulty of learning long-term dependencies (2001)
15. Hutter, M.: Feature reinforcement learning: Part I. unstructured MDPs. J. Artif. Gen. Intel. 1, 3–24 (2009)
16. Hyvärinen, A., Karhunen, J., Oja, E.: Independent component analysis, vol. 46. John Wiley & Sons (2004)
17. Kearns, M., Koller, D.: Efficient reinforcement learning in factored MDPs. In: IJCAI. vol. 16, pp. 740–747 (1999)
18. Kingma, D., Ba, J.: Adam: A method for stochastic optimization. arXiv:1412.6980 (2014)
19. Lőrincz, A., Szirtes, G.: Here and now: How time segments may become events in the hippocampus. Neural Netw. 22(5), 738–747 (2009)
20. Makhzani, A., Frey, B.: k-sparse autoencoders. arXiv:1312.5663 (2013)
21. Makhzani, A., Frey, B.J.: Winner-take-all autoencoders. In: Adv. Neural Info. Proc. Sys. pp. 2773–2781 (2015)
22. Nair, V., Hinton, G.E.: Rectified linear units improve restricted Boltzmann machines. In: Proc. 27th Int. Conf. Mach. Learn. pp. 807–814 (2010)
23. Ng, A.Y.: Feature selection, l1 vs. l2 regularization, and rotational invariance. In: Proc. 21st Int. Conf. Mach. Learn. p. 78. ACM (2004)
24. O'Keefe, J., Dostrovsky, J.: The hippocampus as a spatial map. preliminary evidence from unit activity in the freely-moving rat. Brain Res. 34(1), 171–175 (1971)
25. O'Keefe, J., Nadel, L.: The hippocampus as a cognitive map. Clarendon Press Oxford (1978)
26. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.: Semi-supervised learning with ladder networks. In: Adv. Neural Info. Proc. Sys. pp. 3532–3540 (2015)
27. Salakhutdinov, R.: Learning deep generative models. Annual Review of Statistics and Its Application 2, 361–385 (2015)
28. Schmidhuber, J.: Driven by compression progress. In: Anticip. Behav. in Adapt. Learn. Syst., pp. 48–76. Springer (2009)
29. Schönfeld, F., Wiskott, L.: Modeling place field activity with hierarchical slow feature analysis. Frontiers in Comp. Neurosci. 9 (2015)
30. Sun, Y., Mao, H., Sang, Y., Yi, Z.: Explicit guiding auto-encoders for learning meaningful representation. Neural Comp. Appl. pp. 1–8 (2015)
31. Szepesvári, C., Lőrincz, A.: An integrated architecture for motion-control and path-planning. J. Robot. Syst. 15(1), 1–15 (1998)
32. Szita, I., Lőrincz, A.: Optimistic initialization and greediness lead to polynomial time learning in factored MDPs. In: Proc. 26th Int. Conf. Mach. Learn. pp. 1001–1008. ACM (2009)
33. Szita, I., Takács, B., Lőrincz, A.: ε-MDPs: Learning in varying environments. J. Mach. Learn. Res. 3, 145–174 (2003)
34. Tenenbaum, J.B., Freeman, W.T.: Separating style and content with bilinear models. Neural Comp. 12, 1247–1283 (2000)
35. Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Royal Stat. Soc. Ser. B (Meth.) pp. 267–288 (1996)
36. Tősér, Z., Lőrincz, A.: The cyber-physical system approach towards artificial general intelligence: The problem of verification. In: Artif. Gen. Intel., pp. 373–383. Springer (2015)
37. Tropp, J.A., Gilbert, A.C.: Signal recovery from random measurements via orthogonal matching pursuit. Info. Theo. 53(12), 4655–4666 (2007)
38. Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., Manzagol, P.A.: Stacked denoising autoencoders. J. Mach. Learn. Res. 11, 3371–3408 (2010)
39. Zeiler, M.D.: Adadelta: an adaptive learning rate method. arXiv:1212.5701 (2012)