Disentangling the independently controllable factors of variation by interacting with the world

Valentin Thomas[*1,2] Emmanuel Bengio[*3] William Fedus[*1] Jules Pondard[4]
Philippe Beaudoin[2] Hugo Larochelle[1,5] Joelle Pineau[3,6] Doina Precup[3,7]
Yoshua Bengio[1,8]

Abstract

It has been postulated that a good representation is one that disentangles the underlying explanatory factors of variation. However, it remains an open question what kind of training framework could potentially achieve that. Whereas most previous work focuses on the static setting (e.g., with images), we postulate that some of the causal factors could be discovered if the learner is allowed to interact with its environment. The agent can experiment with different actions and observe their effects. More specifically, we hypothesize that some of these factors correspond to aspects of the environment which are independently controllable, i.e., that there exists a policy and a learnable feature for each such aspect of the environment, such that this policy can yield changes in that feature with minimal changes to other features that explain the statistical variations in the observed data. We propose a specific objective function to find such factors, and verify experimentally that it can indeed disentangle independently controllable aspects of the environment without any extrinsic reward signal.

1 Introduction

When solving Reinforcement Learning problems, what separates great results from random policies is often having the right feature representation. Even with function approximation, learning the right features can lead to faster convergence than blindly attempting to solve given problems [Jaderberg et al., 2016].

The idea that learning good representations is vital for solving most kinds of real-world problems is not new, both in the supervised learning literature [Bengio, 2009, Goodfellow et al., 2016], and in the RL literature [Dayan, 1993, Precup, 2000]. An alternate idea is that these representations do not need to be learned explicitly, and that learning can be guided through internal mechanisms of reward, usually called intrinsic motivation [Barto et al., Oudeyer and Kaplan, 2009, Salge et al., 2013].

We build on a previously studied [Thomas et al., 2017] mechanism for representation learning that has close ties to intrinsic motivation mechanisms and causality. This mechanism explicitly links the agent's control over its environment to the representation of the environment that is learned by the agent. More specifically, this mechanism's hypothesis is that most of the underlying factors of variation in the environment can be controlled by the agent independently of one another.

We propose a general and easily computable objective for this mechanism, that can be used in any RL algorithm that uses function approximation to learn a latent space. We show that our mechanism can push a model to learn to disentangle its input in a meaningful way, and learn to represent factors

[*]Equal contribution, random order, [1]MILA, Université de Montréal, [2]Element AI, [3]McGill University, [4]ENS Paris, [5]Google Brain, [6]Facebook AI Research, [7]Google Deepmind, [8]CIFAR Senior Fellow

Learning Disentangled Representations, NIPS 2017 Workshop. Long Beach, CA, USA.

which take multiple actions to change and show that these representations make it possible to perform model-based predictions in the learned latent space, rather than in a low-level input space (e.g. pixels).

2 Learning disentangled representations

The canonical deep learning framework to learn representations is the autoencoder framework [Hinton and Salakhutdinov, 2006]. There, an encoder $f : S \to H$ and a decoder $g : H \to S$ are trained to minimize the *reconstruction error*, $\|s - g(f(s))\|_2^2$. H is called the latent (or representation) space, and is usually constrained in order to push the autoencoder towards more desirable solutions. For example, imposing that $H \in \mathbb{R}^K, S \in \mathbb{R}^N,\ K \ll N$ pushes f to learn to compress the input; there the bottleneck often forces f to extract the principal factors of variation from S. However, this does not necessarily imply that the learned latent space disentangles the different factors of variations. Such a problem motivates the approach presented in this work.

Other authors have proposed mechanisms to disentangle underlying factors of variation. Many deep generative models, including variational autoencoders [Kingma and Welling, 2014] , generative adversarial networks [Goodfellow et al., 2014] or non-linear versions of ICA [Dinh et al., 2014, Hyvarinen and Morioka, 2016] attempt to disentangle the underlying factors of variation by assuming that their joint distribution (marginalizing out the observed s) factorizes, i.e., that they are marginally independent.

Here we explore another direction, trying to exploit the ability of a learning agent to act in the world in order to impose a further constraint on the representation. We hypothesize that interactions can be the key to learning how to disentangle the various causal factors of the stream of observations that an agent is faced with, and that such learning can be done in an unsupervised way.

3 The selectivity objective

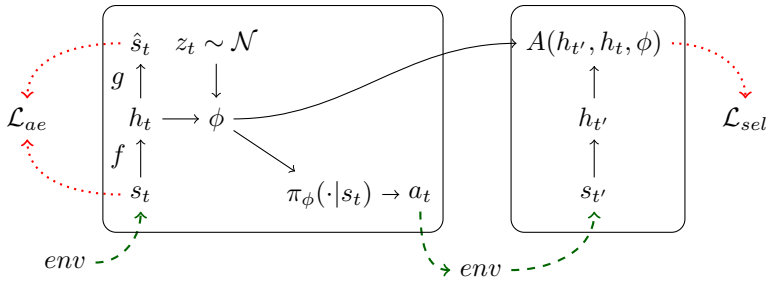
We consider the classical reinforcement learning setting but in the case where extrinsic rewards are not available. We introduce the notion of **controllable factors of variation** $\phi \in \mathbb{R}^K$ which are generated from a neural network $\Phi(h, z), z \sim \mathcal{N}(0, 1)^m$ where $h = f(s)$ is the current latent state. The factor ϕ represents an embedding of a policy π_ϕ whose goal is to realize the variation ϕ in the environment.

To discover meaningful factors of variation ϕ and their associated policies π_ϕ, we consider the following general quantity $\mathcal{S}$ which we refer to as selectivity and that is used as a reward signal for π_ϕ:

$$\mathcal{S}(h, \phi) = \mathbb{E}\left[\log \frac{A(h', h, \phi)}{\mathbb{E}_{p(\varphi|h)}[A(h', h, \varphi)]} \;\middle|\; s' \sim \mathcal{P}_{ss'}^{\pi_\phi}\right] \tag{1}$$

Here $h = f(s)$ is the encoded initial state before executing π_ϕ and $h' = f(s')$ is the encoded terminal state. ϕ and φ represent factors of variation a *factor*. $A(h', h, \phi)$ should be understood as a score describing how close ϕ is to the variation it caused in (h', h). For example in the experiments of section 4.1, we choose A to be a gaussian kernel between $h' - h$ and ϕ, while in the experiments of section 4.2, we choose $A(h', h, \phi) = \max\{0, \langle h' - h, \phi\rangle\}$. The intuition behind these objectives is that in expectation, a factor ϕ should be close to the variation it caused (h', h) when following π_ϕ compared to other factors φ that could have been sampled and followed thus encouraging **independence** within the factors.

Conditioned on a scene representation h, a distribution of policies are feasible. Samples from this distribution represent ways to modify the scene and thus may trigger an internal selectivity reward signal. For instance, h might represent a room with objects such as a light switch. $\phi = \phi(h, z)$ can be thought of as the distributed representation for the "name" of an underlying factor, to which is associated a policy and a value. In this setting, the light in a room could be a factor that could be either on or off. It could be associated with a policy to turn it on, and a binary value referring to its state, called an attribute or a feature value.We wish to jointly learn the policy $\pi_\phi(\cdot|s)$ that modifies the scene, so as to control the corresponding value of the attribute in the scene, whose variation is computed by a scoring function $A(h', h, \phi) \in \mathbb{R}$. In order to get a distribution of such embeddings, we compute $\phi(h, z)$ as a function of h and some random noise z.



(a)

Figure 1: The computational model of our architecture. s_t is the first state, from its encoding h_t and a noise distribution z, ϕ is generated. ϕ is used to compute the policy π_ϕ, which is used to act in the world. The sequence $h_t, h_{t'}$ is used to update our model through the selectivity loss, as well as an optional autoencoder loss on h_t.

The goal of a selectivity-maximizing model is to find the density of factors $p(\phi|h)$, the latent representation h, as well as the policies π_ϕ that maximize $\mathbb{E}_{p(\phi|h)}[\mathcal{S}(h, \phi)]$.

3.1 Link with mutual information and causality

The selectivity objective, while intuitive, can also be related to information theoretical quantities defined in the latent space. From [Donsker and Varadhan, 1975, Ruderman et al., 2012] we have $\mathcal{D}_{\text{KL}}\big(p||q\big) = \sup_{A \in \mathcal{L}^\infty(q)} \mathbb{E}_p[\log A] - \log \mathbb{E}_q[A]$. Applying this equality to the mutual information $\mathcal{I}_p(\phi, h'|h) = \mathbb{E}_{p(h'|h)}\big[\mathcal{D}_{\text{KL}}\big(p(\phi|h', h)||p(\phi|h)\big)\big]$ gives

$$\mathcal{I}_p(\phi, h'|h) \geq \sup_\theta \mathbb{E}_{p(\phi|h)}\big[\mathcal{S}(h, \phi)\big]$$

where θ is the set of weights shared by the factor generator, the policy network and the encoder.

Thus, our total objective along entire trajectories is a lower bound on the causal [Ziebart, 2010] or directed [Massey, 1990] information $\mathcal{I}_p(\phi \mapsto h) = \sum_t \mathcal{I}_p(\phi_t, h_t|h_{t-1})$ which is a measure of the **causality** the process ϕ exercises on the process h. See Appendix C for details.

4 Experiments

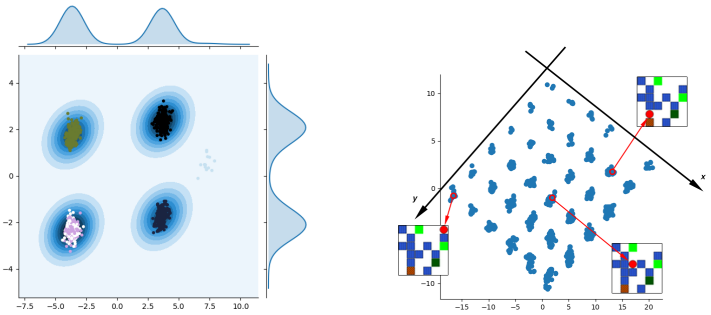
We use MazeBase [Sukhbaatar et al., 2015] to assess the performance of our approach. We do not aim to solve the game. In this setting, the agent (a red circle) can move in a small environment (64×64 pixels) and perform the actions `down, left, right, up`. The agent can go anywhere except on the orange blocks.

4.1 Learned representations

After jointly training the reconstruction and selectivity losses, our algorithm disentangles four directed factors of variations as seen in Figure 2: $\pm x$-position and $\pm y$-position of the agent. For visualization purposes we chose the bottleneck of the autoencoder to be of size $K = 2$. To complicate the disentanglement task, we added the redundant action `up` as well as the action `down+left` in this experiment.

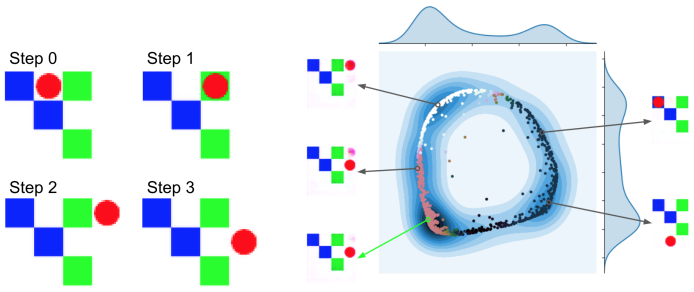
The disentanglement appears clearly as the latent features corresponding to the x and y position are orthogonal in the latent space. Moreover, we notice that our algorithm assigns both actions `up` (white and pink dots in Figure 2.a) to the same feature. It also does not create a significant mode for the feature corresponding to the action `down+left` (light blue dots in Figure 2.a) as this feature is already explained by features `down` and `left`.

[1] pink and white for `up`, light blue for `down+left`, green for `right`, purple black `down` and night blue for `left`.



(a) (b)

Figure 2: (a) Sampling of 1000 variations $h' - h$ and its kernel density estimation encountered when sampling random controllable factors ϕ. We observe that our algorithm disentangles these representations on 4 main modes, each corresponding to the action that was actually taken by the agent.[1] (b) The disentangled structure in the latent space. The x and y axis are disentangled such that we can recover the x and y position of the agent in any observation s simply by looking at its latent encoding $h = f(s)$. The missing point on this grid is the only position the agent cannot reach as it lies on an orange block.



(a) (b)

Figure 3: (a) The actual 3-step trajectory done by the agent. (b) PCA view of the space $\phi(h_0, z), z \sim \mathcal{N}(0, 1)$. Each arrow points to the reconstruction of the prediction $T_\theta(h_0, \phi)$ made by different ϕ. The ϕ at the start of the green arrow is the one used by the policy in (a). Notice how its prediction accurately predicts the actual final state.

4.2 Multistep embedding of policies

In this experiment, ϕ are embeddings of 3-steps policies π_ϕ. We add a model-based loss $\mathcal{L}_{MB} = ||h_{t+3} - T_\theta(h_t, \phi)||^2$ defined only in the latent space, and jointly train a decoder alongside with the encoder. Notice that we never train our model-based cost at pixel level. While we currently suffer from mode collapsing of some factors of variations, we show that we are successfully able to do predictions in latent space, reconstruct the latent prediction with the decoder, and that our factor space disentangles several types of variations.

5 Conclusion, success and limitations

Pushing representations to model independently controllable features currently yields some encouraging success. Visualizing our features clearly shows the different controllable aspects of simple environments, yet, our learning algorithm is unstable. What seems to be the strength of our approach could also be its weakness, as the independence prior forces a very strict separation of concerns in the learned representation, and should maybe be relaxed.

Some sources of instability also seem to slow our progress: learning a conditional distribution on controllable aspects that often collapses to fewer modes than desired, learning stochastic policies that

often optimistically converge to a single action, tuning many hyperparameters due to the multiple parts of our model. Nonetheless, we are hopeful in the steps that we are now taking. Disentangling happens, but understanding our optimization process as well as our current objective function will be key to further progress.

References

Andrew G Barto, Satinder Singh, and Nuttapong Chentanez. Intrinsically motivated learning of hierarchical collections of skills.

Yoshua Bengio. *Learning deep architectures for AI*. Now Publishers, 2009.

Peter Dayan. Improving generalization for temporal difference learning: The successor representation. *Neural Computation*, 5(4):613–624, 1993.

Laurent Dinh, David Krueger, and Yoshua Bengio. NICE: Non-linear Independent Components Estimation. arXiv:1410.8516, ICLR 2015 workshop, 2014.

Monroe D Donsker and SR Srinivasa Varadhan. Asymptotic evaluation of certain markov process expectations for large time, i. *Communications on Pure and Applied Mathematics*, 28(1):1–47, 1975.

Carlos Florensa, Yan Duan, and Pieter Abbeel. Stochastic neural networks for hierarchical reinforcement learning. *arXiv preprint arXiv:1704.03012*, 2017.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *NIPS'2014*, 2014.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

Aapo Hyvarinen and Hiroshi Morioka. Unsupervised Feature Extraction by Time-Contrastive Learning and Nonlinear ICA. In *NIPS*, 2016.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456, 2015.

Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. *arXiv preprint arXiv:1611.05397*, 2016.

Durk P. Kingma and Max Welling. Auto-encoding variational Bayes. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2014.

Yann LeCun. The MNIST database of handwritten digits. *http://yann.lecun.com/exdb/mnist/*, 1998.

James Massey. Causality, feedback and directed information. In *Proc. Int. Symp. Inf. Theory Applic.(ISITA-90)*, pages 303–305, 1990.

Pierre-Yves Oudeyer and Frederic Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 1:6, 2009.

Doina Precup. Temporal abstraction in reinforcement learning. 2000.

Avraham Ruderman, Mark Reid, Darío García-García, and James Petterson. Tighter variational representations of f-divergences via restriction to probability measures. *arXiv preprint arXiv:1206.4664*, 2012.

Christoph Salge, Cornelius Glackin, and Daniel Polani. Empowerment - an introduction. *CoRR*, abs/1310.1863, 2013. URL `http://arxiv.org/abs/1310.1863`.

Sainbayar Sukhbaatar, Arthur Szlam, Gabriel Synnaeve, Soumith Chintala, and Rob Fergus. MazeBase: A sandbox for learning from games. *arXiv preprint arXiv:1511.07401*, 2015.

Valentin Thomas, Jules Pondard, Emmanuel Bengio, Marc Sarfati, Philippe Beaudoin, Marie-Jean Meurs, Joelle Pineau, Doina Precup, and Yoshua Bengio. Independently controllable factors. *CoRR*, abs/1708.01289, 2017. URL `http://arxiv.org/abs/1708.01289`.

Brian D Ziebart. *Modeling purposeful adaptive behavior with the principle of maximum causal entropy*. Carnegie Mellon University, 2010.

A Additional details

A.1 Architecture

Our architecture is as follows: the encoder, mapping the raw pixel state to a latent representation, is a 4-layer convolutional neural network with batch normalization [Ioffe and Szegedy, 2015] and leaky ReLU activations. The decoder uses the transposed architecture with ReLU activations. The noise z is sampled from a 2-dimensional gaussian distribution and both the generator $\Phi(h, z)$ and the policy $\pi(h, \phi)$ are neural networks consisting of 2 fully-connected layers. In practice, a minibatch of $n = 256$ or 1024 vectors $\phi_1, \ldots, \phi_n$ is sampled at each step. The agent randomly choses one $\phi = \phi_{behavior}$ and samples actions from its policy $a \sim \pi(h, \phi_{behavior})$. Our model parameters are then updated using policy gradient with the REINFORCE estimator and a state-dependent baseline and importance sampling. For each selectivity reward, the term $\mathbb{E}_{\phi'}[A(h', h, \phi')]$ is estimated as $\frac{1}{n}\sum_{i=1}^{n} A(h', h, \phi_i)$.

In practice, we don't use concatenation of vectors when feeding two vectors as input for a network (like (h, z) for the factor generator or (h, ϕ) for the policy). For vectors $a, b \in \mathbb{R}^{n_a \times n_b}$. We use a bilinear operation $bil(a, b) = (a_i * b_j)_{i \in [[n_a]], j \in [[n_b]]}$ as in Florensa et al. [2017]. We observe the bilinear integrated input to more strongly enforce dependence on both vectors; in contrast, our models often ignored one input when using a simple concatenation.

Through our research, we experiment with different outputs for our generator $\Phi(h, z)$. We explored embedding the ϕ-vectors into a hypercube, a hypersphere, a simplex and also a simplex multiplied by the output of a $tanh(\cdot)$ operation on a scalar.

A.2 First experiment

In the first experiment, figure 2, we used a gaussian similarity kernel i.e $A(h', h, \phi) = \exp(-\frac{||h'-(h+\phi)||^2}{2\sigma^2})$ with $\sigma = \sqrt{dim(h)}$. In this experiment only, for clarity of the figure, we only allowed permissible actions in the environment (no no-op action).

B Additional Figures

B.1 Discrete simple case

Here we consider the case where we learn a latent space H of size K, with K factors corresponding to the coordinates of h ($h_i,\ i \in [k]$), and learn K separately parameterized policies $\pi_i(a|h),\ i \in [k]$. We train our model with the selectivity objective, but no autoencoder loss, and find that we correctly recover independently controllable features on a simple environment. Albeit slower than when jointly training an autoencoder, this shows that the objective we propose is strong enough to provide a learning signal for discovering a disentangled latent representation.

We train such a model on a gridworld MNIST environment, where there are two MNIST digits . The two digits can be moved on the grid via 4 directional actions (so there are 8 actions total), the first digit is always odd and the second digit always even, so they are distiguishable. In Figure 4 we plot each latent feature h_k as a curve, as a function of each ground truth. For example we see that the black feature recovers $+x_1$, the horizontal position of the first digit, or that the purple feature recovers $-y_2$, the vertical position of the second digit.

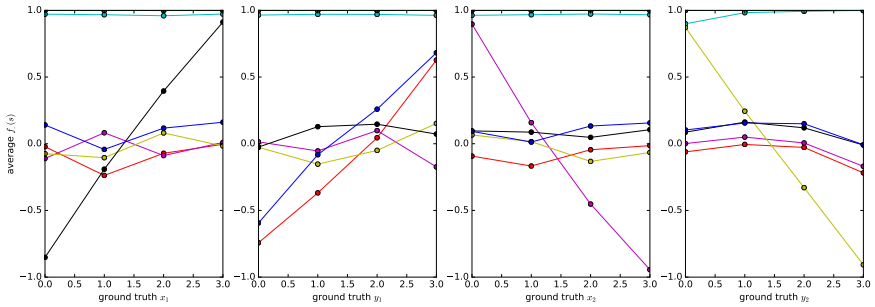


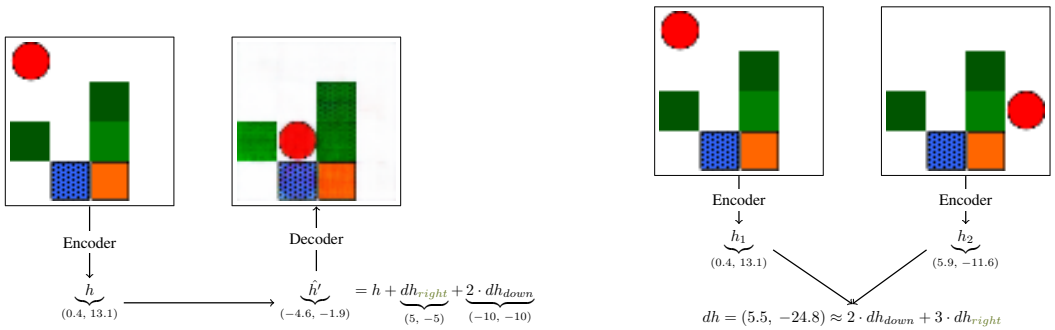
Figure 4: In a gridworld environment with 2 objects (in this case 2 MNIST digits), we know there are 4 underlying features, the (x_i, y_i) position of each digit i. Here each of the four plots represents the evolution of the f_k's as a function of their underlying feature, from left to right x_1, y_1, x_2, y_2. We see that for each of them, at least one f_k recovers it almost linearly, from the raw pixels only.

B.2 Planning and policy inference example in 1-step

This disentangled structure could be used to address many challenging issues in reinforcement learning. We give two examples in figure 5:

- Model-based predictions: Given an initial state, s_0, and an action sequence $a_{\{0:T-1\}}$, we want to predict the resulting state s_T.
- A simplified deterministic policy inference problem: Given an initial state s_{start} and a terminal state s_{goal}, we aim to find a suitable action sequence $a_{\{0:T-1\}}$ such that s_{goal} can be reached from s_{start} by following it.

Because of the $tanh$ activation on the last layer of $\Phi(h, z)$, the different factors of variation $dh = h' - h$ are placed on the vertices of a hypercube of dimension K, and we can think of the the policy inference problem as finding a path in that simpler space, where the starting point is h_{start} and the goal is h_{goal}. We believe this could prove to be a much easier problem to solve.

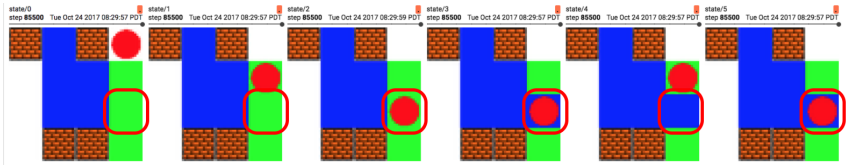


(a) (b)

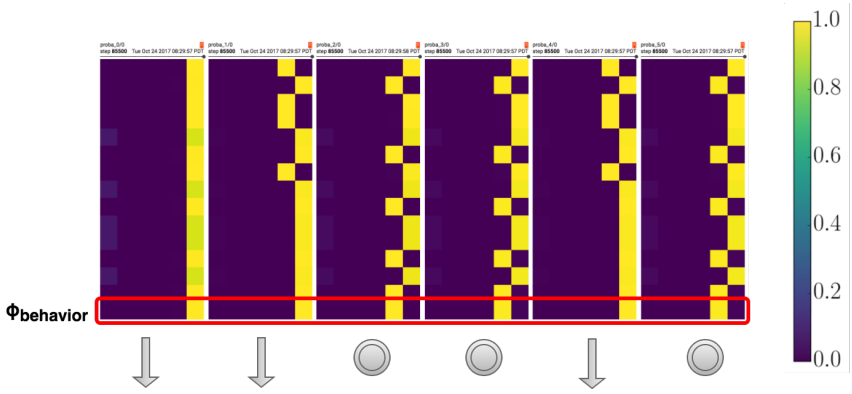
Figure 5: (a) Predicting the effect of a cause on Mazebase. The leftmost image is the visual input of the environment, where the agent is the round circle, and the switch states are represented by shades of green. After the training, we are able to distinguish one cluster per dh (Figure 2), that is to say per variation obtained after performing an action, independently from the position h. Therefore, we are able to move the agent just by adding the corresponding dh to our latent representation h. The second image is just the reconstruction obtained by feeding the resulting h' into the decoder. (b) Given a starting state and a goal state, we are able to decompose the difference of the two representations dh into a (non-directed) sequence of movements.

B.3 Multistep Example

We demonstrate an instance of ICF operating in a 4×4 Mazebase enviroment over five time steps in Figure 6. We consistently witness a failure of mode collapse in our generator Φ and therefore the generator only produces a subset of all possible ϕ-variations. In Figure 6, we observe the ϕ governing the agent's policy π_ϕ appears to correspond to moving two positions down and then to repeatedly toggle the switch. A random action due to ϵ-greedy led to the agent moving up and off the switch at time step-4. This perturbation is corrected by the policy π_ϕ by moving down in order to return to toggling the relevant switch.



(a)



(b)

Figure 6: (a) Mazebase environment over five time-steps. Here the red dot denotes the position of the agent. The $\phi_{behavior}$ governing the agent's policy appears to control toggling the switch indicated by the red rounded box. (b) Visualization of the policies instantiated by different ϕs. Each box represents the probability distribution of the policies at that time step. Each row is generated by a different ϕ and each column corresponds to an action (up, left, pass, right, toggle, down) in order. The boxed column shows the $\phi_{behavior}$. The symbols below each box represent the most-probable action for the behavioral policy, where the grey circle indicates toggling the switch.

C Variational bound and the selectivity

Let us call $p(h_{t+1}|\phi_{t+1}, h_t) = \mathcal{P}^{\phi}_{h',h}$ the probability distribution over final hidden states starting from h and using the policy parametrized by the embedding ϕ.

$p(h_{t+1}|\phi_{t+1}, h_t) = \Pi_{k=1}^{K} \pi_{\phi_{t+1}}(a_{t+\frac{k-1}{K}}|h_{t+\frac{k-1}{K}}) p_{env}(s_{t+\frac{k}{K}}|a_{t+\frac{k-1}{K}}, s_{t+\frac{k-1}{K}})$. $p(\bullet|\phi, h)$

For simplicity, let's refer to h_t as h, h_{t+1} as h' and ϕ_{t+1} as ϕ

C.1 Lower bound on the mutual information

In the case where $A(h', h, \phi)$ is actually is probability density $q(h'|h, \phi)$ (or any unnormalized density such that the normalization factor does not depend on ϕ) we have:

$$\begin{aligned}
\mathcal{S}(\phi, h) &= \mathbb{E}_{h' \sim p(h'|\phi,h)} \log \frac{q(h'|\phi, h)}{\mathbb{E}_{\varphi|h} q(h'|\varphi, h)} \\
&= \mathbb{E}_{\phi|h}\big[\mathcal{D}_{\mathrm{KL}}\big(p(h'|\phi, h)||q(h'|h)\big) - \mathcal{D}_{\mathrm{KL}}\big(p(h'|\phi, h)||q(h'|\phi, h)\big)\big] \\
&= \mathbb{E}_{\phi|h}\big[\mathcal{D}_{\mathrm{KL}}\big(p(h'|\phi, h)||q(h'|h)\big)\big] - \mathcal{D}_{\mathrm{KL}}\big(p(h'|h)||q(h'|h)\big) - \mathbb{E}_{p(h'|h)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(\phi|h', h)||q(\phi|h', h)\big)\big] \\
&= \mathbb{E}_{\phi|h}\big[\mathcal{D}_{\mathrm{KL}}\big(p(h'|\phi, h)||p(h'|h)\big)\big] - \mathbb{E}_{p(h'|h)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(\phi|h', h)||q(\phi|h', h)\big)\big] \\
&= \mathcal{I}^p(\phi, h'|h) - \mathbb{E}_{p(h'|h)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(\phi|h', h)||q(\phi|h', h)\big)\big]
\end{aligned}$$

where we used that fact that $p(\phi|h) = q(\phi|h)$ and by design we only sample ϕs from one generator.

Thus, the gap between the selectivity and the mutual information is the KL divergence of the two posterior distributions.

As we sample the factors ϕ uniformly, our total objective is then

$$\mathcal{J}(\theta) = \mathcal{I}(\phi^T \mapsto h^T) - \sum_t \mathbb{E}_{p(h_{t+1}|h_t)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(\phi_{t+1}|h_{t+1}, h_t)||q(\phi_{t+1}|h_{t+1}, h_t)\big)\big]$$

where $\mathcal{I}^p(\phi \mapsto h)$ is often refered as the *directed information* [Massey, 1990] Ziebart [2010]

The bound

$$\mathcal{I}_p(\phi, h'|h) \geq \sup_\theta \mathbb{E}_{p(\phi|h)}\big[\mathcal{S}(h, \phi)\big]$$

is true even when A is not a probability density and can be proven directly by using the equality from [Donsker and Varadhan, 1975, Ruderman et al., 2012]

$$\mathcal{D}_{\mathrm{KL}}\big(p||q\big) = \sup_{T \in \mathcal{L}^\infty(q)} \mathbb{E}_p\big[T\big] - \log \mathbb{E}_q\big[e^T\big]$$

for $T = \log A$ and using the identity $\mathcal{I}_p(X, Y) = \mathbb{E}_{p(y)}\big[\mathcal{D}_{\mathrm{KL}}\big(p(x|y)||p(x)\big)\big]$

D Additional information on the training

In our experiments, we use the selectivity objective, an autoencoding loss and an entropy regularization loss $\mathcal{H}(\pi_\phi)$ for each of the policies π_ϕ. Furthermore, in experiment 4.2 we added the model-based cost $||h' - T(h, \phi)||^2$ with T a learned two layer fully connected neural network.

The selectivity is used to update the parameters of the encoder, factor generator and policy networks. We use the following equation for computing the gradients

$$\nabla_\theta \mathbb{E}_{\pi_\theta}\big[f_\theta\big] = \mathbb{E}_{\pi_\theta}\big[\nabla_\theta f_\theta + f_\theta \nabla_\theta \log \pi_\theta\big]$$

We also use a state dependent baseline V as a control variate to reduce the variance of the REINFORCE estimator.

Furthermore, to be able to train the factor generator efficiently, we train all ϕ sampled in a mini-batch (of size 1024) by importance sampling on the probability ratio of the trajectory under each ϕ