Concept Learning via Meta-Optimization with Energy Models

Igor Mordatch
OpenAI

1 Introduction

Many hallmarks of human intelligence, such as generalizing from limited experience, abstract reasoning and planning, analogical reasoning, creative problem solving, and capacity for language and explanation are still lacking in the artificial intelligent agents. We, as others [14, 12, 11] believe what enables these abilities is the capacity to consolidate experience into *concepts*, which act as basic building blocks of understanding and reasoning.

Examples of concepts include visual (*"red"* or *"square"*), spatial (*"inside"*, *"on top of"*), temporal (*"slow"*, *"after"*), social (*"aggressive"*, *"helpful"*) among many others [12]. These concepts can be either identified or generated - one can not only find a square in the scene, but also create a square, either physical or imaginary. Importantly, humans also have a largely unique ability to combine concepts compositionally (*"red square"*) and recursively (*"move inside moving square"*) - abilities reflected in the human language. This allows expressing an exponentially large number of concepts, and acquisition of new concepts in terms of others. We believe the operations of identification, generation, composition over concepts are the tools with which intelligent agents can understand and communicate existing experiences and reason about new ones.

Crucially, these operations must be performed on the fly throughout the agent's execution, rather than merely being a static product of an offline training process. We believe execution-time optimization, similar to recent work on meta-learning [4] plays a key role in this. We pose the problem of parsing experiences into a compositional, recursive arrangement of concepts as well as the problems of identifying and generating concepts as optimizations performed during execution lifetime of the agent. The meta-level training is performed by taking into account such processes in the inner level.

Specifically, a concept in our work is defined by an energy function taking as input an event configuration (represented as trajectories of entities in the current work), as well as an attention mask over entities in the event. Zero-energy event and attention configurations imply that event entities selected by the attention mask satisfy the concept. Compositions of concepts can then be created by simply summing energies of constituent concepts. Given a particular event, optimization can be used to identify entities belonging to a concept by solving for attention mask that leads to zero-energy configuration. Similarly, an example of a concept can be generated by optimizing for a zero-energy event configuration. See Figure 1 for examples of these two processes.

The energy function defines a family of concepts, from which a particular concept is selected with a specific concept code. Encoding of event and attention configurations can be achieved by execution-time optimization over concept codes. Once an event is encoded, the resulting concept code structure can be used to re-enact the event under different initial configurations (task of imitation learning), recognize similar events, or concisely communicate the nature of the event. We believe there is a strong link between concept codes and language, but leave it unexplored in this work.

At the meta level, the energy function is the only entity that needs to be learned. This is different from generative model or inverse reinforcement learning approaches, which typically also learn an explicit generator/policy function, whereas we define it implicitly via optimization. Our advantage as that the learned energy function can be transferred to other domains, for example using a robot to generate concepts in the physical world. Such transfer is not possible with an explicit generation/policy function, as it is domain-specific.

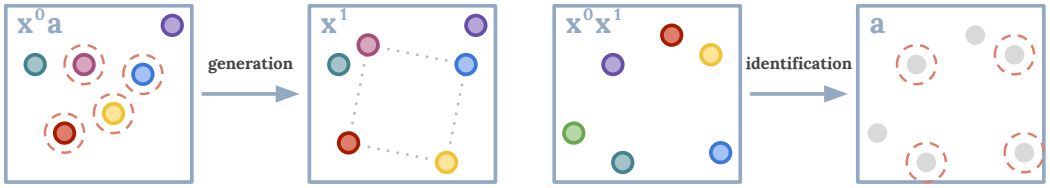

Figure 1: Examples of generation and identification processes for a *"square"* concept. a) Given initial state $\mathbf{x}^0$ and attention mask $\mathbf{a}$, square consisting of entities in $\mathbf{a}$ is formed via optimization over $\mathbf{x}^1$. b) Given states $\mathbf{x}$, entities comprising a square are found by optimization over attention mask $\mathbf{a}$.

2 Energy-Based Concept Models

Concepts operate over events, which in this work is a trajectory of T states $\mathbf{x} = \left[\mathbf{x}^0, ..., \mathbf{x}^T\right]$. Each state contains a collection of N entities $\mathbf{x}^t = [\mathbf{x}_0, ..., \mathbf{x}_N]$ and each entity $\mathbf{x}_i^t$ can contain information such as position and color of the entity. Considering entire trajectories and entities allows us to model temporal or relational concepts, unlike work that focuses on visual concepts [7]. Attention over entities in the event is specified by a mask $\mathbf{a} \in \mathbb{R}^N$ over each of the entities.

Existence of a particular concept is given by energy function $E(\mathbf{x}, \mathbf{a}, \mathbf{w}) \in \mathbb{R}^+$, where parameter vector $\mathbf{w}$ specifies a particular concept from a family. The interpretation of $\mathbf{w}$ is similar to that of a code in an autoencoder. $E(\mathbf{x}, \mathbf{a}, \mathbf{w}) = 0$ when state trajectory $\mathbf{x}$ under attention mask $\mathbf{a}$ over entities satisfies the concept $\mathbf{w}$. Otherwise, $E(\mathbf{x}, \mathbf{a}, \mathbf{w}) > 0$. The conditional probabilities of a particular event configuration belonging to a concept and a particular attention mask identifying a concept are given by the Boltzmann distributions:

$$p(\mathbf{x}|\mathbf{a}, \mathbf{w}) \propto \exp\left\{-E(\mathbf{x}, \mathbf{a}, \mathbf{w})\right\} \qquad p(\mathbf{a}|\mathbf{x}, \mathbf{w}) \propto \exp\left\{-E(\mathbf{x}, \mathbf{a}, \mathbf{w})\right\} \tag{1}$$

Given concept code $\mathbf{w}$, the energy function can be used for both generation and identification of a concept implicitly via optimization (see Figure 1):

$$\mathbf{x}(\mathbf{a}) = \underset{\mathbf{x}}{\operatorname{argmin}}\, E(\mathbf{x}, \mathbf{a}, \mathbf{w}) \qquad \mathbf{a}(\mathbf{x}) = \underset{\mathbf{a}}{\operatorname{argmin}}\, E(\mathbf{x}, \mathbf{a}, \mathbf{w}) \tag{2}$$

Samples from distributions in (1) can be generated via stochastic gradient Langevin dynamics [17], effectively performing stochastic minimization in (2):

$$\begin{aligned} \tilde{\mathbf{x}} \sim \pi_x\left(\,\cdot \mid \mathbf{a}, \mathbf{w}\right) = \mathbf{x}^K, \quad \mathbf{x}^k = \mathbf{x}^{k-1} + \frac{\alpha}{2}\nabla_{\mathbf{x}} E(\mathbf{x}, \mathbf{a}, \mathbf{w}) + \omega^k \\ \tilde{\mathbf{a}} \sim \pi_a\left(\,\cdot \mid \mathbf{x}, \mathbf{w}\right) = \mathbf{a}^K, \quad \mathbf{a}^k = \mathbf{a}^{k-1} + \frac{\alpha}{2}\nabla_{\mathbf{a}} E(\mathbf{x}, \mathbf{a}, \mathbf{w}) + \omega^k, \quad \omega^k \sim \mathcal{N}(0, \alpha) \end{aligned} \tag{3}$$

This stochastic optimization procedure is performed during execution time of the algorithm and is reminiscent of the Monte Carlo sampling procedures in prior work on energy-based models [8, 15, 6], which are not differentiable and present tractability issues for learning. However, in our case the sampling procedure is differentiable, which makes the meta-level learning problem tractable via end to end back-propagation. The above procedure also differs from approaches that use explicit generator functions [2, 10, 9] or explicit attention mechanisms, such as dot product attention [13]. While explicitly learned functions may be faster to evaluate, they offer limited generalization and cannot be transferred across domains.

There are many possible choices for the energy function as long as it is non-negative. The specific form we use in this work is

$$E_\theta(\mathbf{x}, \mathbf{a}, \mathbf{w}) = f_\theta(\sum_{t,i} \sigma(\mathbf{a}_i) \cdot g_\theta(\mathbf{x}_i^t,\ \mathbf{w}^g),\ \mathbf{w}^f)^2, \quad \mathbf{w} = [\mathbf{w}^f, \mathbf{w}^g]$$

Where f and g are multi-layer neural networks that each take concept code as part of their input. σ is the sigmoid function and is used to gate the entities by the attention mask.

3 Learning Concepts from Events

To learn concepts from experience grounded in events, we pose a few-shot prediction task. Given a few demonstration examples X^{demo} containing tuples $(\mathbf{x}, \mathbf{a})$ and initial state $\mathbf{x}^0$ for a new event in X^{train}, the task is to predict attention $\mathbf{a}$ and the future state trajectory $\mathbf{x}^{1:T}$ of the new event. The new event may contain a different configuration or number of entities, so it is not possible to directly transfer attention mask, for instance. To simplify notation, we consider prediction of only one future state $\mathbf{x}^1$, although predicting more states is straightforward. The procedure is depicted in Figure 2.

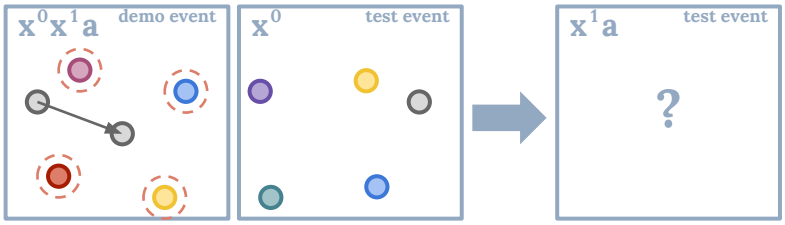

Figure 2: Example of a few-shot prediction task we use to learn concept energy functions.

We follow the maximum entropy inverse reinforcement learning formulation [18] and assume demonstrations are samples from the distributions given by the energy function E. Given a concept code $\mathbf{w}$, finding energy function parameters θ is posed as as maximum likelihood estimation problem over future state and attention given initial state. The resulting loss for a particular dataset X is

$$\mathcal{L}_\theta^{\text{ML}}(X, \mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim X}\left[-\log p_\theta\left(\mathbf{x}^1, \mathbf{a} \mid \mathbf{x}^0, \mathbf{w}\right)\right] \tag{4}$$

Where the joint probability can be decomposed in terms of probabilities in (1) as

$$p_\theta\left(\mathbf{x}^1, \mathbf{a} \mid \mathbf{x}^0, \mathbf{w}\right) = p_\theta\left(\mathbf{x}^1 \mid \mathbf{a}, \mathbf{w}_x\right) p_\theta\left(\mathbf{a} \mid \mathbf{x}^0, \mathbf{w}_a\right), \quad \mathbf{w} = [\mathbf{w}_x, \mathbf{w}_a] \tag{5}$$

We use two concept codes, $\mathbf{w}_x$ and $\mathbf{w}_a$ to specify the joint probability. The interpretation is that $\mathbf{w}_x$ specifies the concept of the action that happens in the event (i.e. *"be in center of"*) while $\mathbf{w}_a$ specifies the argument the action happens over (i.e. *"square"*). This is a concept structure or syntax that describes the event. The same concept can be used either as action or as an argument (because the energy function defining the concept can either be used for generation or identification). This importantly allows concepts to be understood from their usage under multiple contexts.

Conditioned on the two codes concatenated as $\mathbf{w}$, the negative logarithm of joint likelihood in (5) can be approximated as (see Appendix for the derivation)

$$-\log p_\theta\left(\mathbf{x}^1, \mathbf{a} \mid \mathbf{x}^0, \mathbf{w}\right) \approx \left[E_\theta(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_x) - E_\theta(\tilde{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x)\right]_+ + \left[E_\theta(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_a) - E_\theta(\mathbf{x}^0, \tilde{\mathbf{a}}, \mathbf{w}_a)\right]_+$$
$$\tilde{\mathbf{x}} \sim \pi_x\left(\cdot \mid \mathbf{a}, \mathbf{w}_x\right), \quad \tilde{\mathbf{a}} \sim \pi_a\left(\cdot \mid \mathbf{x}^0, \mathbf{w}_a\right)$$

Where $[\cdot]_+ = \log(1 + \exp(\cdot))$ is the softplus function. This form is similar to contrastive divergence [8] and structured SVM forms [1] and is a special case of guided cost learning formulation [5]. The approximation comes from sample-based estimates of the partition functions for $p(\mathbf{x})$ and $p(\mathbf{a})$.

The above equation makes use of sampling distributions π_x and π_a to estimate the respective partition functions. The approximation error in these estimates is minimal when KL divergence between π and $\exp\{-E\}/Z$ is minimized, which intuitively encourages sampling distributions to generate samples from low-energy regions and is expressed by the following loss

$$\mathcal{L}_\pi^{\text{KL}}(X, \mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim X}\left[E(\tilde{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x) + E(\mathbf{x}^0, \tilde{\mathbf{a}}, \mathbf{w}_a)\right] + \mathrm{H}\left[\pi_x\right] + \mathrm{H}\left[\pi_a\right] \tag{6}$$
$$\tilde{\mathbf{x}} \sim \pi_x\left(\cdot \mid \mathbf{a}, \mathbf{w}_x\right), \quad \tilde{\mathbf{a}} \sim \pi_a\left(\cdot \mid \mathbf{x}^0, \mathbf{w}_a\right)$$

Execution-Time Inference of Concepts Given a set of example events, the concept codes can be inferred at execution-time via maximum likelihood estimation over codes $\mathbf{w}$

$$\mathbf{w}_\theta^*(X) = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}_\theta^{\mathrm{ML}}(X, \mathbf{w}) \tag{7}$$

This minimization is similar to execution-time parameter adaptation and the inner update of meta-learning approaches [4]. We perform the optimization with stochastic gradient updates similar to equation (3). Note that this optimization is not over meta-level parameters θ, but is instead over function inputs that are unique for every batch entry.

Meta-Level Parameter Optimization We seek parameters θ of the energy function E that maximize the likelihood of training data X^{train} given demonstration data X^{demo}. Additionally, we seek sampling distributions π that generates samples corresponding to low energies. However, the sampling distribution is implicitly a function of energy parameters θ via equations (3), a dependence which we denote as $\pi(\theta)$. The meta-level parameters are found via the following optimization problem

$$\min_\theta \mathcal{L}_\theta^{\mathrm{ML}}(X^{\text{train}}, \mathbf{w}_\theta^*(X^{\text{demo}})) + \mathcal{L}_{\pi(\theta)}^{\mathrm{KL}}(X^{\text{train}}, \mathbf{w}_\theta^*(X^{\text{demo}}))$$

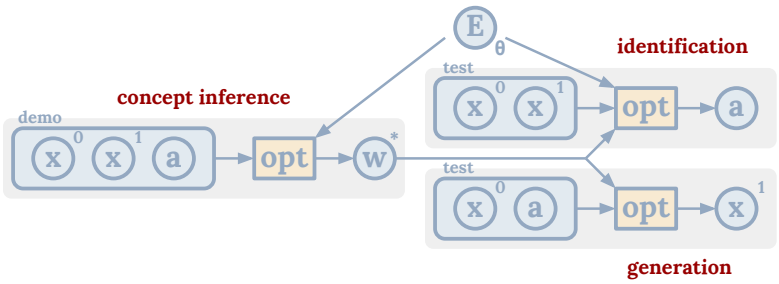

Figure 3: Execution-time inference in our method and the resulting nested optimization problems.

The difficulty is that even evaluation of $\mathcal{L}$ requires two nested levels of gradient-based optimization problems - first optimization level to infer $\mathbf{w}$ and second optimization level to generate samples $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{a}}$ (see Figure 3). In order to backpropagate over such procedure, we turn nested optimizations into a sequence of alternating optimizations.

4 Experiments

Due to space constraints, we are unable to describe our experiments in detail. Briefly, we use a two-dimensional particle-based environment to investigate execution-time concept generation, identification and inference and via optimization. We found success in spatial and relational concepts similar to examples shown in the figures. We have also investigated transfer of concept generation between domains, learning concept models in a the simple particle domain and then successfully enacting the concepts in a different environment with a physically simulated robot arm using model-predictive control [16].

5 Conclusion

We believe that execution-time optimization plays a crucial role in acquisition and generalization of knowledge, planning and abstract reasoning, and communication. In this preliminary work, we proposed energy-based concept models as basic building blocks over which such optimization procedures can fruitfully operate. Interestingly, we observed the need for multiple nested levels of optimization problems. This presents difficulties in meta-level training which must take these procedures into account. We currently unroll nested optimizations into one alternating sequence, but perhaps there may be a more principled approaches for solving nested optimizations problems. In the current work we also used a simple concept structure, but more complex structure with multiple arguments or recursion would be interesting to investigate in the future.

References

[1] D. Belanger, B. Yang, and A. McCallum. End-to-end learning for structured prediction energy networks. *arXiv preprint arXiv:1703.05667*, 2017.

[2] P. Dayan, G. E. Hinton, R. M. Neal, and R. S. Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

[3] K. Dvijotham and E. Todorov. Inverse optimal control with linearly-solvable mdps. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 335–342, 2010.

[4] C. Finn, P. Abbeel, and S. Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *arXiv preprint arXiv:1703.03400*, 2017.

[5] C. Finn, S. Levine, and P. Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pages 49–58, 2016.

[6] K. Friston. The free-energy principle: a unified brain theory? *Nature Reviews Neuroscience*, 11(2):127–138, 2010.

[7] I. Higgins, N. Sonnerat, L. Matthey, A. Pal, C. P. Burgess, M. Botvinick, D. Hassabis, and A. Lerchner. Scan: Learning abstract hierarchical compositional visual concepts. *arXiv preprint arXiv:1707.03389*, 2017.

[8] G. E. Hinton. Training products of experts by minimizing contrastive divergence. *Training*, 14(8), 2006.

[9] T. Kim and Y. Bengio. Deep directed generative models with energy-based probability estimation. *arXiv preprint arXiv:1606.03439*, 2016.

[10] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[11] B. M. Lake, T. D. Ullman, J. B. Tenenbaum, and S. J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, pages 1–101, 2016.

[12] G. Lakoff and M. Johnson. The metaphorical structure of the human conceptual system. *Cognitive science*, 4(2):195–208, 1980.

[13] C. Olah and S. Carter. Attention and augmented recurrent neural networks. *Distill*, 1(9):e1, 2016.

[14] E. Rosch, C. B. Mervis, W. D. Gray, D. M. Johnson, and P. Boyes-braem. Basic objects in natural categories. *COGNITIVE PSYCHOLOGY*, 1976.

[15] R. Salakhutdinov and G. Hinton. Deep boltzmann machines. In *Artificial Intelligence and Statistics*, pages 448–455, 2009.

[16] Y. Tassa, T. Erez, and E. Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4906–4913. IEEE, 2012.

[17] M. Welling and Y. W. Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 681–688, 2011.

[18] B. D. Ziebart, A. L. Maas, J. A. Bagnell, and A. K. Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

Appendix

Derivation of Joint Log-Likelihood Approximation

The derivation is similar to guided cost learning [5] and cost learning in linearly-solvable MDP [3] formulations. Joint negative log-likelihood of observing tuple $\left(\mathbf{x}^0, \mathbf{x}^1, \mathbf{a}\right)$ is $-\log p\left(\mathbf{x}^1, \mathbf{a} \mid \mathbf{x}^0, \mathbf{w}\right)$

$$= -\log\left(p\left(\mathbf{x}^1 \mid \mathbf{a}, \mathbf{w}_x\right) p\left(\mathbf{a} \mid \mathbf{x}^0, \mathbf{w}_a\right)\right) \tag{8}$$

$$= -\log \frac{\exp\left\{-E(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_x)\right\}}{\int_{\tilde{\mathbf{x}}} \exp\left\{-E(\tilde{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x)\right\}} - \log \frac{\exp\left\{-E(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_a)\right\}}{\int_{\tilde{\mathbf{a}}} \exp\left\{-E(\mathbf{x}^0, \tilde{\mathbf{a}}, \mathbf{w}_a)\right\}} \tag{9}$$

Consider a more general form of the two individual terms above with non-negative function f

$$-\log \frac{\exp\left\{-f(\mathbf{x})\right\}}{\int_{\tilde{\mathbf{x}}} \exp\left\{-f(\tilde{\mathbf{x}})\right\}} = f(\mathbf{x}) + \log \mathbb{E}_{\tilde{\mathbf{x}} \sim q}\left[\frac{\exp\left\{-f(\tilde{\mathbf{x}})\right\}}{q(\tilde{\mathbf{x}})}\right] \tag{10}$$

The equality follows due to importance sampling under distribution q. There are a number of choices for sampling distribution q, but a choice that simplifies the above expression and we found to give stable results in practice is $q(X) = \frac{1}{2}\mathbb{I}\left[X = \mathbf{x}\right] + \frac{1}{2}\mathbb{I}\left[X = \tilde{\mathbf{x}}\right]$ where $\tilde{\mathbf{x}} \sim \pi$ and π is a distribution that minimizes $\mathrm{KL}\left(\pi(X) || \exp\left\{-f(X)\right\}/Z\right)$.

In this case, sample-based approximation of equation (10) leads to

$$f(\mathbf{x}) + \log \mathbb{E}_{\tilde{\mathbf{x}} \sim q}\left[\frac{\exp\left\{-f(\tilde{\mathbf{x}})\right\}}{q(\tilde{\mathbf{x}})}\right] \approx f(\mathbf{x}) + \log\left(\exp\left\{-f(\mathbf{x})\right\} + \exp\left\{-f(\tilde{\mathbf{x}})\right\}\right) \tag{11}$$

$$= \log\left(1 + \exp\left\{f(\mathbf{x}) - f(\tilde{\mathbf{x}})\right\}\right) = \left[f(\mathbf{x}) - f(\tilde{\mathbf{x}})\right]_+ \tag{12}$$

Using the above approximation in equation (9), gives the desired result $-\log p\left(\mathbf{x}^1, \mathbf{a} \mid \mathbf{x}^0, \mathbf{w}\right)$

$$\approx \left[E(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_x) - E(\tilde{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x)\right]_+ + \left[E(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_a) - E(\mathbf{x}^0, \tilde{\mathbf{a}}, \mathbf{w}_a)\right]_+ \tag{13}$$

$$\text{where } \tilde{\mathbf{x}} \sim \pi_x\left(\,\cdot \mid \mathbf{a}, \mathbf{w}_x\right), \quad \tilde{\mathbf{a}} \sim \pi_a\left(\,\cdot \mid \mathbf{x}^0, \mathbf{w}_a\right)$$