Concept Learning with Energy-Based Models

[Mordatch, 2018]

Presented by Aga Dobrowolska

June 22, 2020

UCL NLP Reading Group

Overview

An energy-based model that can quickly learn to identify and generate instances of concepts, such as *near*, *above*, *between*, *closest* and *furthest*, expressed as sets of 2D points. The model learns these concepts after only five demonstrations.

Introduction

Why learn concepts?

- Concepts act as basic building blocks of human understanding and reasoning.
- Essential for generalizing from limited experience, abstract reasoning and planning, analogical reasoning, creative problem solving, and capacity for language and explanation - and more.
- Combining concepts compositionally and recursively enhances expressivity.
- Learning occurs ‘on-the-fly’ - an execution-time optimisation problem?

Why Energy Based Models?

- Typically functions that assign low energy values to inputs in the data distribution and high energy values to other inputs.
- Can then be used to discriminate whether or not a query input comes from the data distribution.
- Key property: it is a *smooth* function.
- This paper: uses sampling procedure based on gradient of the energy (which mixes much faster than gradient-free MCMC) while training the energy function and attains a gradient field that produces good samples.

Briefly on Related Work (1)

Inspiration for this work:

- Energy-based models: used mostly for density modelling.
- Concept learning:
  - Recent work on learning visual concepts such as color and shape, where concepts are defined in terms of distributions over latent variables produced by a variational autoencoder. [1]
  - Focuses solely on visual concepts from pixel input.
  - Some work on Bayesian Reasoning applied for learning numerical concepts. [2]
  - This paper explores learning concepts that involve **complex interaction of multiple entities**.

Briefly on Related Work (2)

- Inverse Reinforcement Learning
  - IRL: learning the cost function underlying the observed behaviour.
  - Concept energy function are analogous to the cost / negative value functions in IRL (but not the same).
- Meta-learning
  - Here we want to optimize 1) concept generation/identification and 2) which concepts take part in the event.
  - Hence, in training we take into account the behaviour of these inner optimisation process, as in meta-learning.

Definitions: States

- T events are denoted as a trajectory of T states: $\mathbf{x} = \left[\mathbf{x}^0, \ldots, \mathbf{x}^T\right]$.
- Each state is a collection of some N entities: $\mathbf{x}^t = [\mathbf{x}_0, \ldots, \mathbf{x}_N]$.
- Each entity vector $\mathbf{x}_i^t$ contains information such as position coordinates, colour, shape etc.

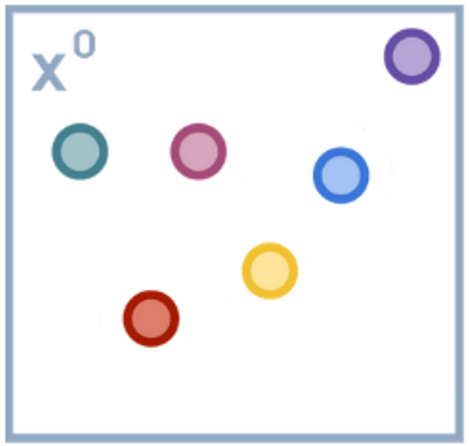


Figure 1: Example state x^0 with $N = 6$ entities

Definitions: Attention

- Attention is a mask $a \in \mathbb{R}^N$ specifying which entities participate in the given concept.

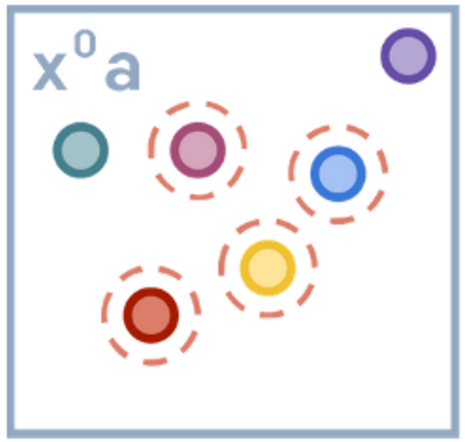
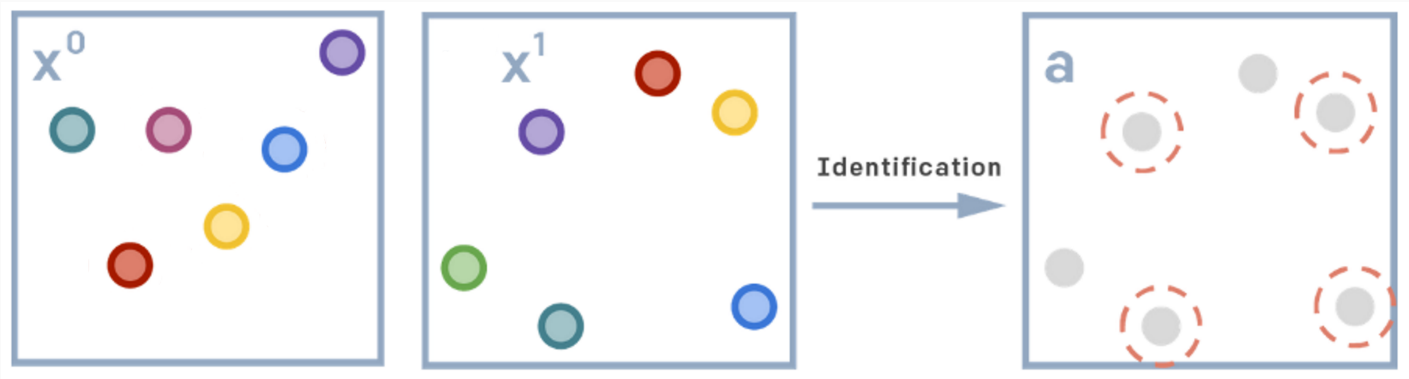


Figure 2: An attention mask applied to entities participating in the given concept

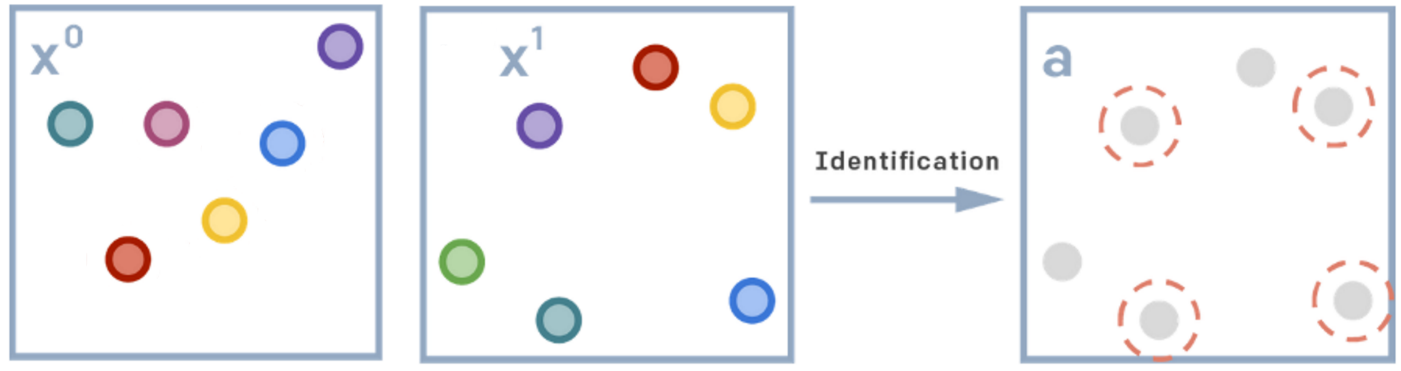
Tasks of Interest

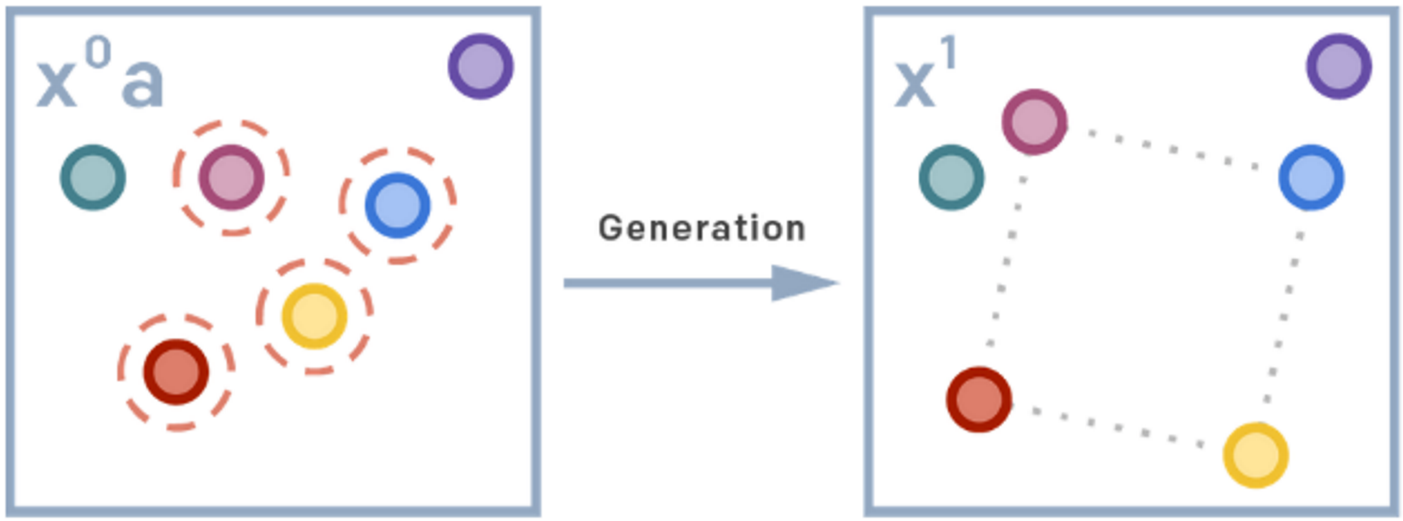
Two tasks which we would like to perform: Identification and Generation



Tasks of Interest

Two tasks which we would like to perform: Identification and Generation





Definitions: Concept Codes

- Concept code, **w**, is a vector learned by the model which identifies a given concept.
- Here, we perform execution-time optimisation over the concept code and thus use it to encode event and attention configurations.
- Hence, concept codes are essential for **concept transfer**
- **w** can be thought of as analogous to the code in an autoencoder ie. a low-dimensional representation of some high-dimensional information.
- *"We believe there is a strong link between concept codes and language, but leave it unexplored in this work".*

Definitions: Concept Codes

Why not just use attention to transfer concepts?

- the number of entities may differ between events.

Definitions: The Energy Model

- Specify an Energy Function over the events, where the energy function is a function of the 3 components: the states, the attention mask and the concept code: $E(\mathbf{x}, \mathbf{a}, \mathbf{w}) \in \mathbb{R}^+$.
- $E(\mathbf{x}, \mathbf{a}, \mathbf{w}) = 0$ when state trajectory $\mathbf{x}$ under attention mask $\mathbf{a}$ over the entities satisfies the concept $\mathbf{w}$ and $E(\mathbf{x}, \mathbf{a}, \mathbf{w}) > 0$ otherwise.

Definitions: The Energy Model

Here, the Energy Function is a Relational Neural Network:

$$E_\theta(\mathbf{x}, \mathbf{a}, \mathbf{w}) = f_\theta \left(\sum_{t,i,j} \sigma\left(\mathbf{a}_i\right) \sigma\left(\mathbf{a}_j\right) \cdot g_\theta \left(\mathbf{x}_i^t, \mathbf{x}_j^\iota, \mathbf{w} \right), \mathbf{w} \right)^2$$

where f and g are multi-layer neural networks that each take concept code as part of their input, and σ is the sigmoid function, used to gate the entity pairs by their attention masks.

Intuition 1: SGD as part of the model

- Suppose we already have a 'good' energy function
- We can use keep 2 of the 3 arguments constant and use gradient descent eg. $\nabla_{\mathbf{x}} E(\mathbf{x}, \mathbf{a}, \mathbf{w})$ or $\nabla_{\mathbf{a}} E(\mathbf{x}, \mathbf{a}, \mathbf{w})$
- Obtain a better attention or state, which leads to a lower value of the energy function.

Intuition 2: A training sample

- Usually, to learn model parameters θ: obtain $(\mathbf{x}, y)$, predict $\hat{y}$ and use loss to measure distance between y and $\hat{y}$, then backpropagate.
- In this model, a *single* training sample X consists of a *demo* example: $\{\mathbf{x}^0, \mathbf{x}^1, \mathbf{a}\}^{demo}$ and a *test example*, $\{\mathbf{x}^0, \mathbf{x}^1, \mathbf{a}\}^{test}$.

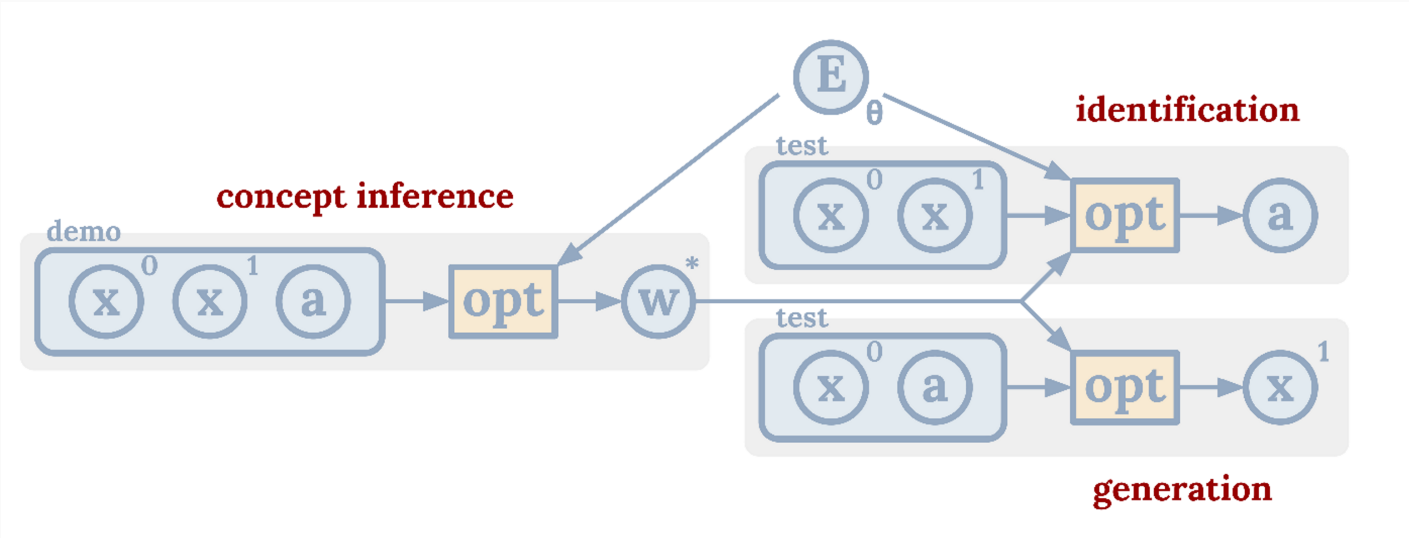


Figure 3: A single training sample

Stochastic Optimisation at Execution Time

- Similarities here to MCMC sampling, which uses conditional distributions to approximate the posterior.
- Conditional probabilities are given as Boltzmann distributions:
  $p(\mathbf{x}|\mathbf{a}, \mathbf{w}) \propto \exp\{-E(\mathbf{x}, \mathbf{a}, \mathbf{w})\}$
  $p(\mathbf{a}|\mathbf{x}, \mathbf{w}) \propto \exp\{-E(\mathbf{x}, \mathbf{a}, \mathbf{w})\}$
- Goal: $\mathbf{x}(\mathbf{a}) = \underset{\mathbf{x}}{\operatorname{argmin}} E(\mathbf{x}, \mathbf{a}, \mathbf{w}) \quad \mathbf{a}(\mathbf{x}) = \underset{\mathbf{a}}{\operatorname{argmin}} E(\mathbf{x}, \mathbf{a}, \mathbf{w})$

Stochastic Optimisation at Execution Time

- Similarities here to MCMC sampling, which uses conditional distributions to approximate the posterior.
- Conditional probabilities are given as Boltzmann distributions:
  $p(\mathbf{x}|\mathbf{a}, \mathbf{w}) \propto \exp\{-E(\mathbf{x}, \mathbf{a}, \mathbf{w})\}$
  $p(\mathbf{a}|\mathbf{x}, \mathbf{w}) \propto \exp\{-E(\mathbf{x}, \mathbf{a}, \mathbf{w})\}$
- Goal: $\mathbf{x}(\mathbf{a}) = \underset{\mathbf{x}}{\operatorname{argmin}} E(\mathbf{x}, \mathbf{a}, \mathbf{w}) \quad \mathbf{a}(\mathbf{x}) = \underset{\mathbf{a}}{\operatorname{argmin}} E(\mathbf{x}, \mathbf{a}, \mathbf{w})$
- Stochastic Gradient Langevin Dynamics updates:

$$\tilde{\mathbf{x}} \sim \pi_x(\cdot|\mathbf{a}, \mathbf{w}) = \mathbf{x}^K, \quad \mathbf{x}^k = \mathbf{x}^{k-1} + \frac{\alpha}{2}\nabla_{\mathbf{x}} E(\mathbf{x}, \mathbf{a}, \mathbf{w}) + \omega^k$$

$$\tilde{\mathbf{a}} \sim \pi_a(\cdot|\mathbf{x}, \mathbf{w}) = \mathbf{a}^K, \quad \mathbf{a}^k = \mathbf{a}^{k-1} + \frac{\alpha}{2}\nabla_{\mathbf{a}} E(\mathbf{x}, \mathbf{a}, \mathbf{w}) + \omega^k$$

Learning Model Parameters: $\mathcal{L}_p^{\mathrm{ML}}$

- Assume that the demonstrations, X^{demo} are samples from the posterior distributions, p given by the energy function E. (Following the maximum entropy formulation from IRL)
- Given an inferred concept code, the problem of finding θ is a ML estimation problem, resulting in the ML loss of:

$$\mathcal{L}_p^{\mathrm{ML}}(X, \mathbf{w}) = \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim X}\left[-\log p\left(\mathbf{x}^1, \mathbf{a} | \mathbf{x}^0, \mathbf{w}\right)\right]$$

where
$\log p\left(\mathbf{x}^1, \mathbf{a} | \mathbf{x}^0, \mathbf{w}\right) = \log p\left(\mathbf{x}^1 | \mathbf{a}, \mathbf{w}_x\right) + \log p\left(\mathbf{a} | \mathbf{x}^0, \mathbf{w}_a\right), \quad \mathbf{w} = [\mathbf{w}_x, \mathbf{w}_a]$

- Derivation in Appendix:

$$\begin{aligned}
\log p\left(\mathbf{x}^1 | \mathbf{a}, \mathbf{w}_x\right) &\approx -\left[E\left(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_x\right) - E\left(\tilde{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x\right)\right]_+ \quad \tilde{\mathbf{x}} \sim \pi_x\left(\cdot | \mathbf{a}, \mathbf{w}_x\right) \\
\log p\left(\mathbf{a} | \mathbf{x}^0, \mathbf{w}_a\right) &\approx -\left[E\left(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_a\right) - E\left(\mathbf{x}^0, \tilde{\mathbf{a}}, \mathbf{w}_a\right)\right]_+ \quad \tilde{\mathbf{a}} \sim \pi_a\left(\cdot | \mathbf{x}^0, \mathbf{w}_a\right)
\end{aligned}$$

Where $[\cdot]_+ = \log(1 + \exp(\cdot))$ is the softplus operator.

Learning Model Parameters: $\mathcal{L}_{\pi}^{\mathrm{KL}}$

- In $\mathcal{L}_{p}^{\mathrm{ML}}$ we use the truncated, biased distributions π_x and $\pi_a$to approximate their partitions functions.
- The approximation error in these estimates is minimised when KL divergence between biased distribution π and true distribution $\exp\{-E\}/Z$ is minimized.
- Introduce :

$$\begin{aligned}\mathcal{L}_{\pi}^{\mathrm{KL}}(X, \mathbf{w}) &= \mathrm{KL}\left(\pi_x \| p_x\right) + \mathrm{KL}\left(\pi_a \| p_a\right) \\ &= \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim X}\left[E\left(\tilde{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x\right) + E\left(\mathbf{x}^0, \tilde{\mathbf{a}}, \mathbf{w}_a\right)\right] + \mathrm{H}\left[\pi_x\right] + \mathrm{H}\left[\pi_a\right] \\ \tilde{\mathbf{x}} &\sim \pi_x\left(\cdot|\mathbf{a}, \mathbf{w}_x\right), \quad \tilde{\mathbf{a}} \sim \pi_a\left(\cdot|\mathbf{x}^0, \mathbf{w}_a\right)\end{aligned}$$

- Intuitively encourages sampling distributions π to generate samples from low energy regions

Execution-Time Inference of Concepts

- NB: the authors only consider positive examples when adapting $\mathbf{w}$ and ignore the effect that changing $\mathbf{w}$ has on the sampling distribution π.
- Given a set of example events X, the concept codes can be inferred at execution-time via finding codes $\mathbf{w}$ that minimize both, $\mathcal{L}^{\mathrm{ML}}$ and $\mathcal{L}^{\mathrm{KL}}$
- Equivalent to simply minimizing the energy functions wrt $\mathbf{w}$ over the concept example events

$$\mathbf{w}_\theta^*(X) = \underset{\mathbf{w}}{\operatorname{argmin}} \mathbb{E}_{(\mathbf{x},\mathbf{a})\sim X}\left[E_\theta\left(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_x\right) + E_\theta\left(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_a\right)\right]$$

Update samples $\mathbf{w}_x$ and $\mathbf{w}_a$ via stochastic gradient Langevin dynamics step:

$$\mathbf{w}_x(\theta) \leftarrow \mathbf{w}_x(\theta) + \tfrac{\alpha}{2}\nabla_{\mathbf{w}} E_\theta\left(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_x(\theta)\right) + \omega^k$$
$$\mathbf{w}_a(\theta) \leftarrow \mathbf{w}_a(\theta) + \tfrac{\alpha}{2}\nabla_{\mathbf{w}} E_\theta\left(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_a(\theta)\right) + \omega^k$$

Learning Model Parameters

- We seek to find the probability density functions p that maximize the likelihood of training data X via $\mathcal{L}_p^{\mathrm{ML}}$ and, at the same time, we also seek the sampling distributions π that generate samples from p via $\mathcal{L}_\pi^{\mathrm{KL}}$.
- The authors here draw a parallel to inverse reinforcement learning, where these two objectives correspond to cost and policy function optimization, and are treated as separate alternating optimization problems because they operate over two different functions.
- However, in our case both p and π are implicitly a functions of the energy model and its parameters θ, a dependence which we denote as $p(\theta)$ and $\pi(\theta)$.
- Consequently we can pose the problem as a single joint optimization

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

  The above optimization problem is solved via end-to-end backpropagation, differentiating through the gradient-based sampling procedures... ouch. Scalability issues?

Training Sample Again

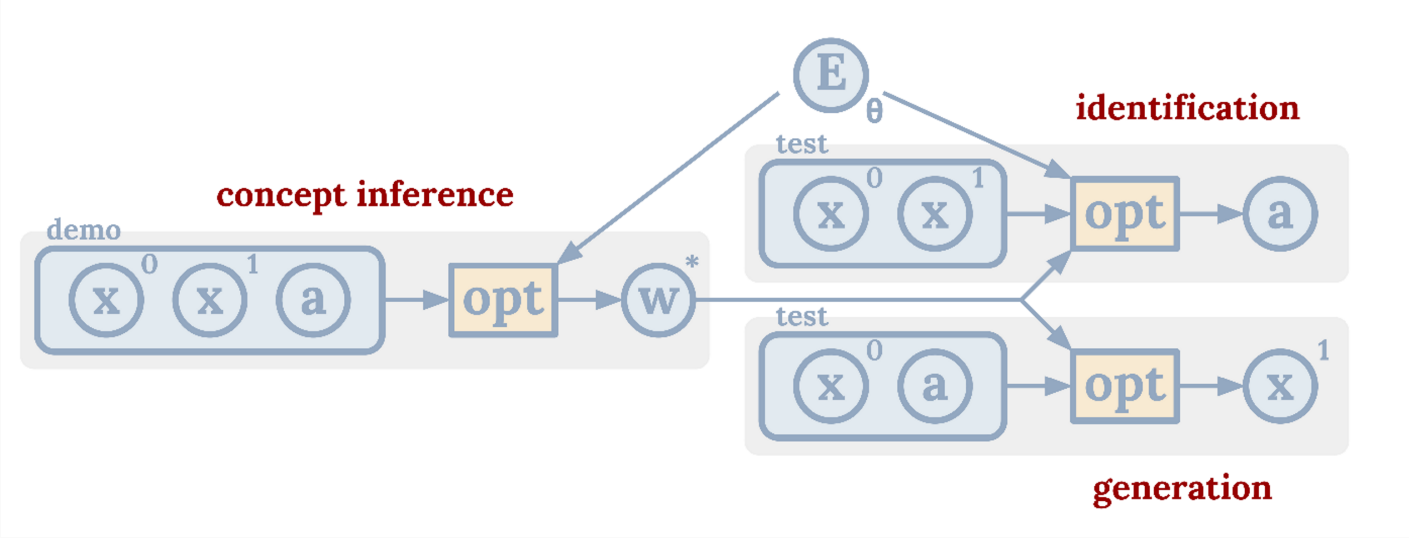


Figure 4: A single training sample

Algorithm Details

Algorithm 1: Energy-based model learning from demonstration events

Initialize energy model parameters θ
for events X^{train} and X^{demo} sampled from the same concept **do**
  Randomly sample event $\left(\mathbf{x}^0, \mathbf{x}^1, \mathbf{a}\right)$ from X^{demo}
  Initialize $\mathbf{w}_x$ and $\mathbf{w}_a$ from unit Gaussian
  for sampling iteration $k = 1$ to K **do**
    Update samples $\mathbf{w}_x$ and $\mathbf{w}_a$ via stochastic gradient Langevin dynamics step:

$$\mathbf{w}_x(\theta) \leftarrow \mathbf{w}_x(\theta) + \frac{\alpha}{2}\nabla_{\mathbf{w}} E_\theta(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_x(\theta)) + \omega^k$$
$$\mathbf{w}_a(\theta) \leftarrow \mathbf{w}_a(\theta) + \frac{\alpha}{2}\nabla_{\mathbf{w}} E_\theta(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_a(\theta)) + \omega^k$$

  end for
  Randomly sample event $\left(\mathbf{x}^0, \mathbf{x}^1, \mathbf{a}\right)$ from X^{train}
  Initialize $\tilde{\mathbf{x}}$ to $\mathbf{x}^0$ and initialize $\tilde{\mathbf{a}}$ from unit Gaussian
  for sampling iteration $k = 1$ to K **do**
    Update samples $\tilde{\mathbf{x}}$ and $\tilde{\mathbf{a}}$ via stochastic gradient Langevin dynamics step:

$$\tilde{\mathbf{x}}(\theta) \leftarrow \tilde{\mathbf{x}}(\theta) + \frac{\alpha}{2}\nabla_{\mathbf{x}} E_\theta(\tilde{\mathbf{x}}(\theta), \mathbf{a}, \mathbf{w}_x(\theta)) + \omega^k$$
$$\tilde{\mathbf{a}}(\theta) \leftarrow \tilde{\mathbf{a}}(\theta) + \frac{\alpha}{2}\nabla_{\mathbf{a}} E_\theta(\mathbf{x}^0, \tilde{\mathbf{a}}(\theta), \mathbf{w}_a(\theta)) + \omega^k$$

  end for
  Set $\bar{\mathbf{x}}$ be the result of applying gradient stopping operator on $\tilde{\mathbf{x}}$ (and similarly for $\bar{\mathbf{a}}$, $\bar{\mathbf{w}}$ and $\bar{E}$)
  Formulate two losses operating over p and π holding other function fixed:

$$\mathcal{L}^{\text{ML}}(\theta) = \left[E_\theta(\mathbf{x}^1, \mathbf{a}, \mathbf{w}_x(\theta)) - E_\theta(\bar{\mathbf{x}}, \mathbf{a}, \mathbf{w}_x(\theta))\right]_+ + \left[E_\theta(\mathbf{x}^0, \mathbf{a}, \mathbf{w}_a(\theta)) - E_\theta(\mathbf{x}^0, \bar{\mathbf{a}}, \mathbf{w}_a(\theta))\right]_+$$
$$\mathcal{L}^{\text{KL}}(\theta) = \bar{E}(\tilde{\mathbf{x}}(\theta), \mathbf{a}, \bar{\mathbf{w}}_x) + \bar{E}(\mathbf{x}^0, \tilde{\mathbf{a}}(\theta), \bar{\mathbf{w}}_a)$$

  Update θ based on gradient $\nabla_\theta(\mathcal{L}^{\text{ML}}(\theta) + \mathcal{L}^{\text{KL}}(\theta))$ via Adam optimizer
end for

Experiments

Dataset

This dataset is not meant to be an exhaustive list of all possible concepts, but rather a varied sampling of interesting concepts.

- Changing color of entities or attending to entities of a particular color.
- Regional placement of entities in a particular spatial area - either a point, horizontal or vertical line, circle, or corners of a square.
- Placement relations of a one entity either north, south, east, west of another entity or between two entities.
- Shape relations between entities of either joining together, or forming a line, triangle, or square shapes.
- Proximity relations bring attention to the entity closest or furthest to another entity or to bring that entity to be closest or furthest to the attended entity.
- Quantity relations bring attention to any one, two, three, or more than three entities.
- See paper's Appendix for full list.

Main Goals of Experiments

The authors stipulate that they designed experiments with 3 main goals in mind.

To investigate

1. Whether the model is able to learn understanding of wide variety of concepts under multiple contexts
2. The utility of iterative optimization-based inference processes
3. Ability to reuse learned concepts on different environments and actuation platforms.

Key Task

Given N demonstration events (usually N = 5) that involve identical attention and state changes under different situations, perform analogous behavior under N novel test situations (by attending to analogous entities and performing analogous state changes).

- ie. learn to transfer concepts
- NB: Such behavior is not unique and there may be multiple possible solutions.

Concepts in Multiple Contexts

Is the model able to produce qualitatively 'sensible' predictions in both, identification and generation scenarios?

Consider events with proximity relations "farthest":

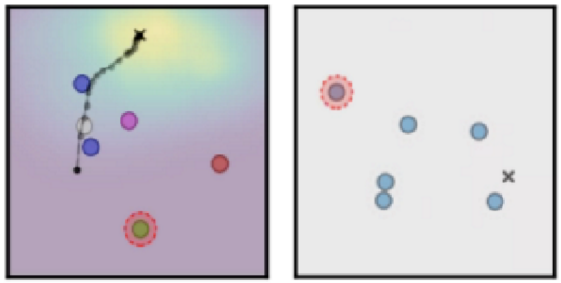


Figure 4: Outcomes of generation (left) and identification (right) for the concept of being farthest to cross-shaped entity. Path in left image is the optimization trajectory for the cross entity with the energy heatmap is overlaid.

Transfer of Learning

Can the model transfer of learning between generation and identification contexts?

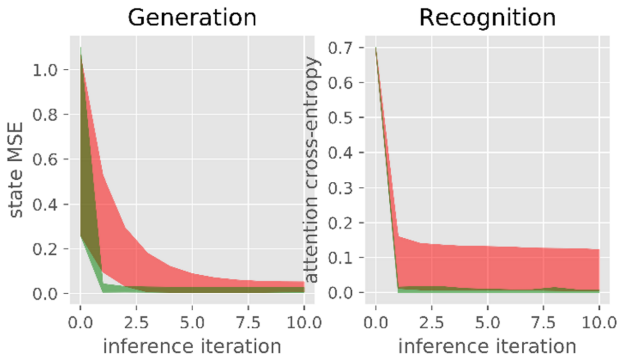


Figure 5: Accuracy of transfer between contexts for an absolute position concept. **Red** is error of the model trained only in one context (generation or identification) evaluated on the opposite context. **Green** is error of the model trained in both contexts.

Condition	Generation Error	Attention Error
Untrained Network	0.621	0.698
Trained on Generation	0.006	0.061
Trained on Identification	0.016	0.001
Trained on Both	0.007	0.001

Sharing of Concept Codes across contexts

- PCA projection of the components $\mathbf{w}_a$ and $\mathbf{w}_x$ for identification and generation finds they are interchangeable.
- "Thus we see evidence that a concept code is reused across contexts, similarly to how words in a language are used in multiple contexts. This property presents exciting opportunities in applying our model to grounded language understanding."

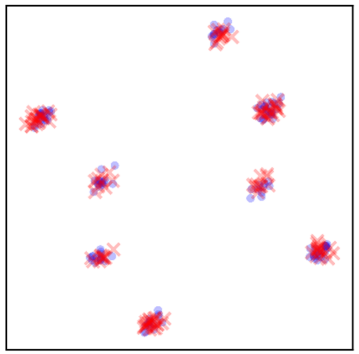
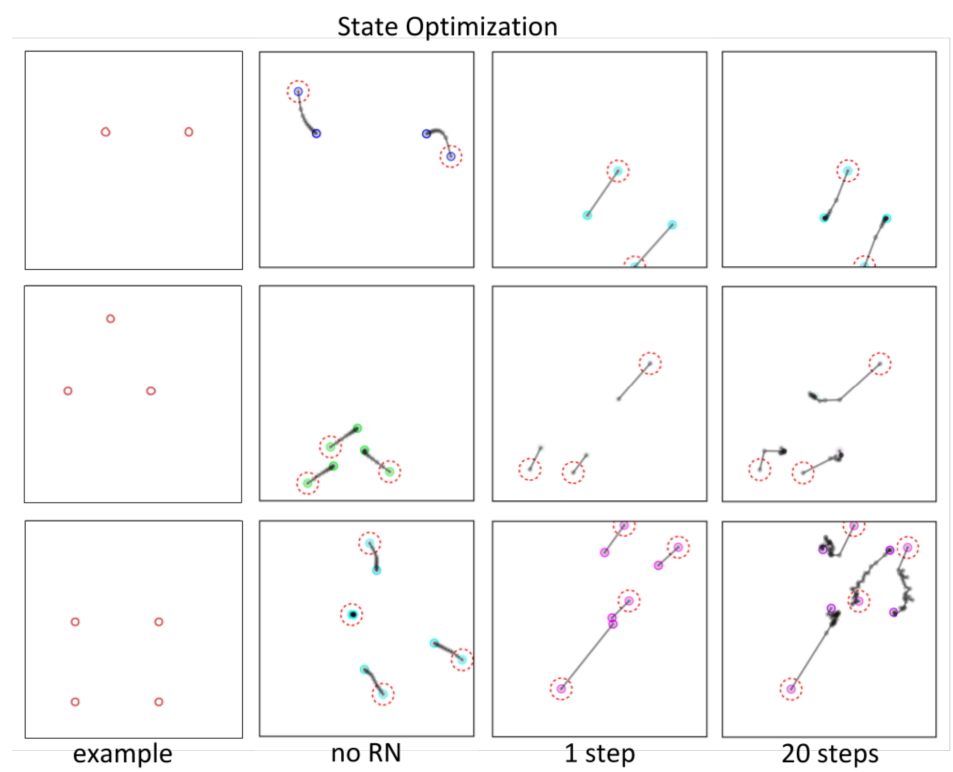
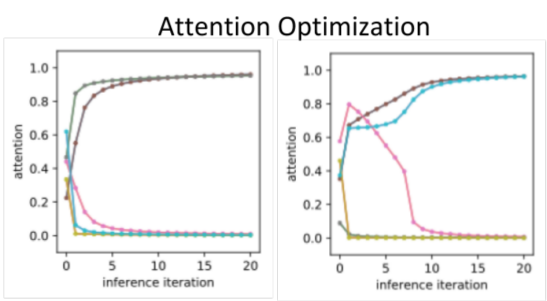
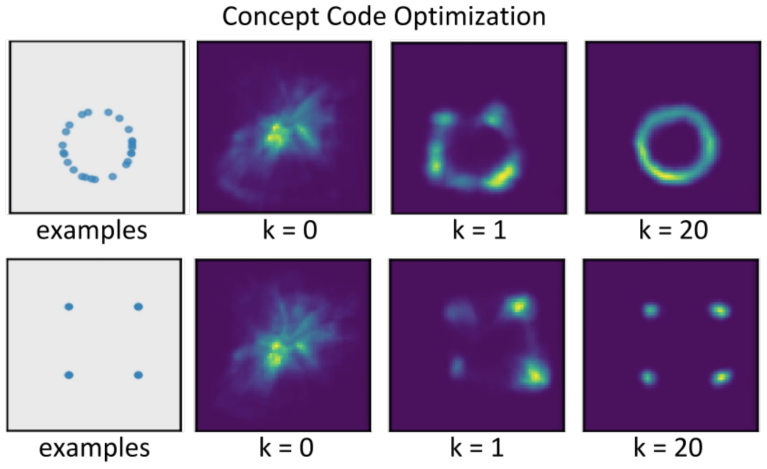


Figure 6: Projected concept codes for color events. **red** are generation codes $\mathbf{w}_x$ and **blue** are attention codes $\mathbf{w}_a$.

Optimisation-based Inference







Is the $\mathcal{L}^{\mathrm{KL}}$ Objective necessary?

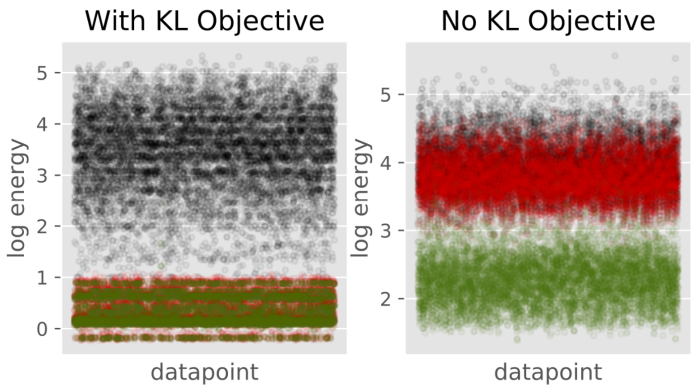


Figure 8: Energy values from models trained with and without KL objective. **Green** and positive example events, **red** are sample events generated by our run-time inference process, and **black** are random events from the initial distribution.

- Trained only on $\mathcal{L}^{\mathrm{ML}}$: can discriminate between true example events (green) and sampled (red) and random (black) events but **unable to produce sample events that match energy of examples**.
- Network trained with both objectives $\mathcal{L}^{\mathrm{ML}}$ and $\mathcal{L}^{\mathrm{KL}}$ can do both.

Reuse of Concepts between Environments

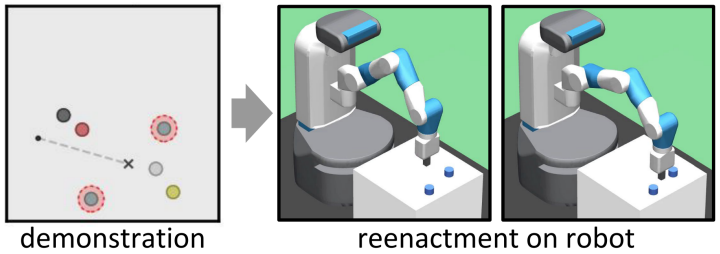


Figure 9: Energy function of reaching between blue objects learned from demonstration in 2D particle environment reused in a 3D robot simulator under a novel arrangement.

- Can transfer concepts without re-training
- But... simply a question of changing the joint torques of a robotic arm rather than directly changing the coordinates of the entities?

Some Thoughts

- The energy function is the only function that needs to be learned. In generative models or inverse reinforcement learning approaches, you typically also learn an explicit generator/policy function.
- This **implicit** manner of learning concepts is what makes this approach more transferable and versatile.
- Combining concepts can be easily achieved by summing energies of participating events.

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. In the current work we used a simple concept structure, but more complex structure with multiple arguments or recursion would be interesting to investigate in the future. It would also be interesting to test compositionality of concepts, which is very suited to our model as compositions corresponds to the summation of the constituent energy functions."

References

I. Higgins, N. Sonnerat, L. Matthey, A. Pal, C. Burgess, M. Botvinick, D. Hassabis, and A. Lerchner, *Scan: Learning abstract hierarchical compositional visual concepts*, (2017).

E. Todorov, T. Erez, and Y. Tassa, *Mujoco: A physics engine for model-based control*, 10 2012, pp. 5026–5033.