Attend, Infer, Repeat: Notes

Adam Kosiorek

December 29, 2017

1 Introduction

Attend, Infer, Repeat (AIR; Eslami et al., 2016) is a VAE-type model capable of decomposing a static scene into its constituent parts. This is useful, since it explicitly represent distinct parts of the scene, which provides useful and actionable representations for down-stream processing. An example application are proposal location for object detection algorithms. While not guaranteed to represent objects, parts of the scene represented by separate representations are bound to be statistically significant. The purpose of this document is to explore the model in more detail.

2 Generative Model and Priors

Let $\mathbf{x}$ an image, n a number of statistically interesting separate parts in the image, $\mathbf{z} = \{\mathbf{z}^1, \dots, \mathbf{z}^n\}$ a group of variables, where each variable describes a part of the scene. Let $p_\theta(\mathbf{z}, n) = p_\theta(\mathbf{z} \mid n) p_\theta(n)$ be a prior over latent variables describing our assumptions about possible layouts and complexity of the scene and let $p_\theta(\mathbf{x} \mid \mathbf{z})$ be a *generating* model. The prior and the generating model together form a generative model of the scene, where the prior describes it in terms of latent variables and the generating model uses the latent variables to paint the scene. We can write down the marginal distribution of images as

$$p_\theta(\mathbf{x}) = \sum_{n=1}^{N} p_{\theta,N}(n) \int p_\theta(\mathbf{z} \mid n) p_\theta(\mathbf{x} \mid \mathbf{z}) \, d\mathbf{z}. \tag{1}$$

We take the prior to have the following form:

$$p_\theta(n) = \text{Geom}(n \mid \theta), \tag{2}$$

$$p_\theta(\mathbf{z} \mid n) = \prod_{i=1}^{n} p_\theta\left(\mathbf{z}^i\right) = \prod_{i=1}^{n} \mathcal{N}(\mathbf{0}, \mathbf{I}), \tag{3}$$

which assumes that $\mathbf{z}^i$ are independent under the prior. The success probability for the geometric distribution is chosen to encourage sparse (as few steps as possible) solutions. In practice, it is annealed over 100k training steps to a very low value on the order of $p = 10^{-e}$, $e \in [5, 10]$.

3 Inference through an Approximate Posterior

Since the latent variables $\mathbf{z}^i$ are independent under the prior, they are exchangeable. It introduces symmetries, which increases the volume of the search space in a combinatorial way. To address this issue, the original work formulates inference as a sequential problem, where latent variables describing a part of a scene depend on previously inferred latent variables. Specifically, it parametrises n as a vector of n ones followed by a zero $\mathbf{z}^p$. With this representation and a sequential implementation of inference, it is enough to output a single *presence* indicator variable at each processing step. The approximate posterior is given by

$$q_\theta(\mathbf{z}, \mathbf{z}^p \mid \mathbf{x}) = q_\phi\left(z^{p,n+1} = 0 \mid \mathbf{z}^{1:n}, \mathbf{x}\right) \prod_{i=1}^{n} q_\phi\left(\mathbf{z}^i, z^{p,i} = 1 \mid \mathbf{z}^{1:i-1}, \mathbf{x}\right), \quad (4)$$

with q_ϕ implemented as a neural network. To avoid explaining the same object twice, it is vital to capture the dependency $bz^i, z^{p,i} \mid \mathbf{z}^{1:i-1}, \mathbf{x}$, which is implemented as a recurrent neural network R_ϕ with hidden state $\boldsymbol{h}^i$, where the initial hidden state $\boldsymbol{h}^0$ is randomly initialised and learnable, and specifically as

$$\omega^i, \boldsymbol{h}^i = R_\phi(\mathbf{x}, \mathbf{h}^{i-1}). \quad (5)$$

The variable ω^i specifies parameters of the probability distribution over $\mathbf{z}^i$ and $z^{p,i}$ and introduces a conditional independence property, namely $\mathbf{z}^i \perp\!\!\!\perp z^{p,i} \mid \omega^i$. We can use this property to factorise the approximate posterior distribution as

$$q_\theta(\mathbf{z}, \mathbf{z}^p \mid \mathbf{x}) = q_\phi\left(z^{p,n+1} = 0 \mid \omega^{n+1}\right) \prod_{i=1}^{n} q_\phi\left(\mathbf{z}^i \mid \omega^i\right) q_\phi\left(z^{p,i} \mid \omega^i\right). \quad (6)$$

Since the hidden state $\boldsymbol{h}^i$ does not depend on any of the latent variables, there is no need to integrate over the hidden state . All of the distribution parameters ω are computed before any of the distributions is used or even instantiated. That hints at what the implementation might look like: a recurrent neural network should compute the parameters of all the probability distributions, but all sampling can be done outside in a feed-forward fashion.

check with the STORN paper

4 Learning by Maximising the ELBO

AIR is trained by maximising the evidence lower bound $\mathcal{L}_{\mathrm{AIR}}$ (ELBO) given by

$$\mathcal{L}_{\mathrm{AIR}}(\phi, \theta) = \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(\mathbf{z}, n, \mathbf{x})}{q_\phi(\mathbf{z}, n \mid \mathbf{x})}\right] \tag{7}$$

$$= \mathbb{E}_{q_\phi}[\log p_\theta(\mathbf{x} \mid \mathbf{z})] - \mathrm{KL}\big(q_\phi(\mathbf{z}, n \mid \mathbf{x}) \parallel p_\theta(\mathbf{z}, n)\big). \tag{8}$$

The first term is a probabilistic analog of the reconstruction error, while the second term is a complexity penalty. In this case it encourages minimum-length encoding of every part of the scene as well as decomposing the scene into a minimal number of parts. The KL term can be rewritten according to the product rule as

$$\mathrm{KL}\big(q_\phi(\mathbf{z}, n \mid \mathbf{x}) \parallel p_\theta(\mathbf{z}, n)\big) = \tag{9}$$

$$= \mathrm{KL}\big(q_\phi(n \mid \mathbf{x}) \parallel p_\theta(n)\big) + \mathbb{E}_{q_\phi(n \mid \mathbf{x})}\big[\mathrm{KL}\big(q_\phi(\mathbf{z} \mid n, \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid n)\big)\big]. \tag{10}$$

5 Policy Gradient Techniques

Policy gradients are used here to backprop through samples from a discrete probability distribution.

5.1 Score Function

Williams, 1992

5.2 NVIL

Mnih and Gregor, 2014 implemented but doesn't work that well, presumably because of different effective weightings of the continuous and discrete components of the loss. I still have to check with the lower learning rate.

5.3 Discrete IWAE

To be done... Originally introduced by Burda, Grosse, and Salakhutdinov, 2015, but extended to a discrete case by Tucker et al., 2017.

5.4 Convex Relaxation

To be done... concrete distribution/gumbel Jang, Gu, and Poole, 2016; Maddison, Mnih, and Teh, 2016

5.5 REBAR

This seems terribly complicated Tucker et al., 2017

6 FIVO for Better Bounds

Introduced by Maddison, Lawson, et al., 2017; ywt says its much better for sequential estimation such as AIR than the usual ELBO.

7 Complex Backgrounds: Behaviour and Extenstions

AIR depends on statistical differences between pixels belonging to an object and the ones belonging to the background. Moreover, the way it has been defined, it can reconstruct only objects while neglecting the background completely. It is natural to expect that it simply wouldn't work in the case of non-empty backgrounds. To mitigate this issue, it is necessary to augment the model with a background-explaining component g. In a simple case, it could produce a background $\boldsymbol{b}$ based on the last hidden state of the controller RNN R_ϕ, specifically:

$$\boldsymbol{b} = g_\phi(\boldsymbol{h}^N) \tag{11}$$

$$y = h^{\text{dec}}(\mathbf{z}, \boldsymbol{b}) \tag{12}$$

For the case of MNIST, g can be implemented as a simple MLP. For more complicated model, it would have to be a more complicated model based on up-convolutions, but this is also true about the glimpse decoder.

Design an experiment to test this behaviour.

8 An Extension to Timeseries

AIR displays an inconsistent behaviour depending on initialisation. It is randomly initialised and it tends to converge to different solutions when training is restarted. Only a small subset of the solutions is close to what the user would want, namely explaining one object at a time. This is understandable, since a dataset of single images cannot encode an *consistency* prior, whereby objects would have to be preserved when they move. Extending AIR to sequential data has a potential of solving this issue, since it should be much easier to explain an object by changing its position parameter than to inferring its appearance description from scratch at every time-step. Moreover, it prevents explaining two objects moving separately with a single glimpse, since it would be impossible to preserve them consistently while reconstructing.

An extension to timeseries is not trivial, however. It requires some form of dependency between consecutive timestep. It is important to distinguish temporal and within-timestep dependencies. We dub the latter *sequential* dependencies. In the original work, the only sequential dependencies exist between the hidden states $\boldsymbol{h}^i \mid \boldsymbol{h}^{i-1}$ and the presence variables $z^{p,i} \mid z^{p,1:i-1}$, while there is no sequential dependence between appearance and position $\mathbf{z}^i$. A temporal version of the algorithm requires (or might benefit from) the following:

- between-timestep hidden state dependence e. g., $\boldsymbol{h}^t \mid \boldsymbol{h}^{t-1}$, where this timestep might or might not be the same as the sequential hidden state
- temporal dependence on object position
- temporal dependence on object appearance
- temporal dependence on object presence

8.1 Indian Buffer Process for Object Counts

Read and write (Gael, Teh, and Ghahramani, 2009).

9 Sequential AIR

Let us start by describing the model from the point of view of its generative story. Images are generated by first assuming that, at every time-step, objects are propagated from previous time-step and some new objects can be introduced. Let $k \in \{0, 1, \ldots, K\}$, $K \in \mathcal{N}_+$ the number of objects propagated from the previous time-step and let $n \in \{0, 1, \ldots, N-k\}$, $N \in \mathcal{N}_+$ the number of objects discovered at the current time-step. The maximum of K objects can be propagated and the model can handle up to N total objects, therefore $K \leqslant N$.

Let superscripts D and P denote latent variables generated by discovery and propagation models, respectively. At the first time-step $t = 1$ there are no objects to propagate, so we sample up to N objects from a discovery prior $p\big(n_1, \mathbf{z}_1^D \mid N\big)$. Starting from the second time-step $t = 2$, the model first propagates objects by sampling from a propagation prior $p\big(k_2, \mathbf{z}_2^P \mid n_1 + k_1, \mathbf{z}_1\big)$, where $\mathbf{z}_1 = \mathbf{z}_1^D$. The model also samples n_2 new objects from the prior $p\big(n_2, \mathbf{z}_2^D \mid N - k_2\big)$. From now on, that is for $t \geqslant 2$, we set the aggregated latent variable $\mathbf{z}_t = \{\mathbf{z}_t^P, \mathbf{z}_t^D\}$. This process continues up to the final time-step T. The images are generated by passing the generated latent variables $\mathbf{z}_{1:T}$ to the generative model $p_\theta(\mathbf{x}_t \mid \mathbf{z}_t)$ one at a time. Note that $\mathbf{z}_t$ is a set of up to N latent variable, where each latent variable represents a separate objects. The generating model acts separately on every latent variables in the set, and the output random variable of the generating model consists of the sum of outputs of the generative model for each latent variable in the set. Figure 1 shows

the graphical model of the generative story. Please note that the prior for the number of new objects stays the same for every time-step and it is analogous to the prior used by AIR, with the exception that the number of maximum objects can differ.

To make it more formal, we will now give all equations governing the generative process. The prior for the discovery model is defined as follows.

$$\begin{aligned} &p\left(n_t, \mathbf{z}_t^D \mid N\right) = \\ &p(n_t \mid N) p(\mathbf{z}_t \mid n_t) = \text{TruncatedGeom}(n_t \mid N) \prod_{i=1}^{n_t} \mathcal{N}(\mathbf{0}, \mathbf{I}), \end{aligned} \tag{13}$$

where $\text{TruncatedGeom}(n_t \mid N)$ is a geometric distribution, whose support is truncated to $\{0, 1, \ldots, N\}$. The propagation model is defined by the following.

$$\begin{aligned} &p\left(k_t, \mathbf{z}_t^P \mid m_{t-1}, \mathbf{z}_{1:t-1}\right) = \prod_{j=1}^{m} p\left(b_t^j \mid \mathbf{z}_{1:t-1}^j\right) p\left(\mathbf{z}_t^j \mid \mathbf{z}_{1:t-1}^j\right) = \\ &\prod_{j=1}^{m_{t-1}} \text{Bernoulli}\left(b_t^j \mid \psi(\mathbf{z}_{1:t-1}^j)\right) \mathcal{N}\left(\mathbf{z}_t^j \mid \boldsymbol{\mu}(\mathbf{z}_{1:t-1}^j), \boldsymbol{\sigma}^2(\mathbf{z}_{1:t-1}^j)\right), \end{aligned} \tag{14}$$

with $\mathbf{z}_{1:t-1}^j$ denoting the history of latent variables belonging to object j and $\mathbf{z}_{1:0}^j = \emptyset$. In the above we reparamterise k_t as a sum of Bernoulli random variables b_t^k, that is $k_t = \sum_{i=1}^{m} b_t^j$. Since for every object the probability of propagation ψ is different, k_t follows a PoissonBinomial distribution with m trials. Direct conditioning on the whole history of latent variables is impractical, and therefore we use a learned deterministic RNN to estimate ψ, μ and σ^2. the RNN shares parameters but maintains a separate hidden state for every object j. We can rewrite Equation (14) in terms of a simple recurrence as

$$\begin{aligned} &p\left(k_t, \mathbf{z}_t^P \mid m_{t-1}, \mathbf{z}_{1:t-1}\right) = p\left(k_t, \mathbf{z}_t^P \mid m_{t-1}, \boldsymbol{h}_t^j\right) = \\ &\prod_{j=1}^{m_{t-1}} \text{Bernoulli}\left(b_t^j \mid \psi(\boldsymbol{h}_t^j)\right) \mathcal{N}\left(\mathbf{z}_t^j \mid \boldsymbol{\mu}(\boldsymbol{h}_t^j), \boldsymbol{\sigma}^2(\boldsymbol{h}_t^j)\right), \end{aligned} \tag{15}$$

$$\boldsymbol{h}_t^j = f_\theta^{\text{RNN}}\left(\mathbf{z}_{t-1}^j, \boldsymbol{h}_{t-1}^j\right). \tag{16}$$

Any parameters in the generating model, e. g., the parameters of the RNN, can be learnt jointly with the inference model (to be described shortly). Alternatively, one could use a non-parametric prior for the discrete latent variable $k_{1:T}$, e. g., the Indian Buffet Process.

Images are created by decoding the latent variables into small 'glimpses' that are then transformed to match the size of the original image and summed

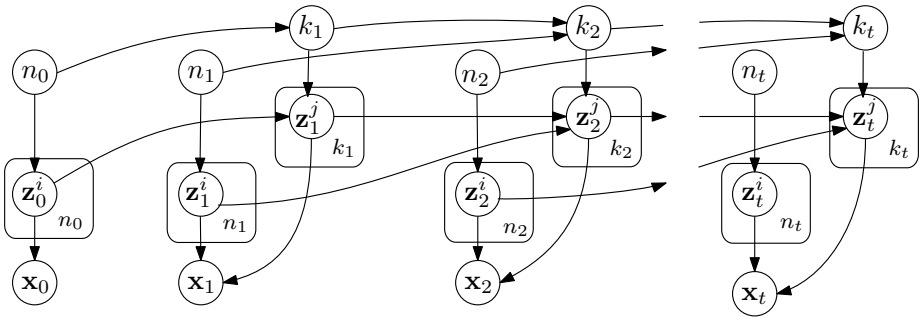

Figure 1: The generative story of sequential AIR

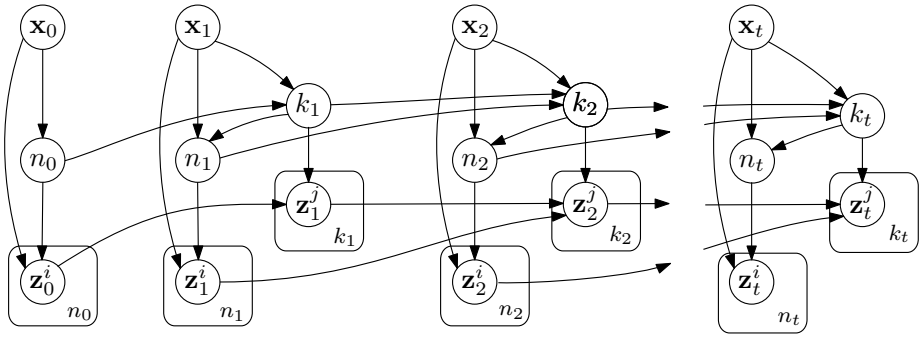

Figure 2: Graphical model for inference in sequential AIR

up. Each latent variable $\mathbf{z}_t^j$ can be decomposed as $\mathbf{z}_t^j = \left\{\mathbf{z}_t^{j,\text{where}}, \mathbf{z}_t^{j,\text{what}}\right\}$. More formally,

$$\widehat{\mathbf{g}}_t^j = f_\theta^{\text{dec}}\left(\mathbf{z}_t^{j,\text{what}}\right), \tag{17}$$

$$\mathbf{y}_t^j = f^{\text{STN}}\left(\widehat{\mathbf{g}}_t^j, \mathbf{z}_t^{j,\text{where}}\right), \tag{18}$$

$$\widehat{\mathbf{x}}_t \sim \mathcal{N}\left(\mathbf{x}_t \mid \sum_{j=1}^{N_t} \mathbf{y}_t^j, \sigma_x^2\right), \tag{19}$$

with the decoder f^{dec}, spatial transformer f^{STN} and a fixed per-pixel standard deviation σ_x^2.

Inference To infer the latent variables, we have to invert the generative model. To do so, the inference model has to take into account the observations $\mathbf{x}_{1:T}$ and produce sequences of sets of latent variables $\mathbf{z}_{1:T}$.

References

Burda, Yuri, Roger Grosse, and Ruslan Salakhutdinov (2015). "Importance Weighted Autoencoders". In: ISSN: 1312.6114v10. arXiv: 1509.00519. URL: http://arxiv.org/abs/1509.00519.

Eslami, S. M. Ali et al. (2016). "Attend, Infer, Repeat: Fast Scene Understanding with Generative Models". In: *NIPS*. arXiv: 1603.08575. URL: http://arxiv.org/abs/1603.08575.

Gael, Jurgen Van, Yee Whye Teh, and Zoubin Ghahramani (2009). "The Infinite Factorial Hidden Markov Model". In: *NIPS*, pp. 1697–1704. ISBN: 9781605609492. URL: https://papers.nips.cc/paper/3518-the-infinite-factorial-hidden-markov-model.

Jang, Eric, Shixiang Gu, and Ben Poole (2016). "Categorical Reparameterization with Gumbel-Softmax". In: ISSN: 1611.01144. arXiv: 1611.01144. URL: http://arxiv.org/abs/1611.01144.

Maddison, Chris J., Dieterich Lawson, et al. (2017). "Filtering Variational Objectives". In: *NIPS*. arXiv: 1705.09279. URL: http://arxiv.org/abs/1705.09279.

Maddison, Chris J., Andriy Mnih, and Yee Whye Teh (2016). "The Concrete Distribution: A Continuous Relaxation of Discrete Random Variables". In: arXiv: 1611.00712. URL: http://arxiv.org/abs/1611.00712.

Mnih, Andriy and Karol Gregor (2014). "Neural Variational Inference and Learning in Belief Networks". In: *ICML*. ISBN: 9781634393973. arXiv: arXiv:1402.0030v2. URL: http://arxiv.org/abs/1402.0030.

Tucker, George et al. (2017). "REBAR: Low-variance, unbiased gradient estimates for discrete latent variable models". In: *NIPS*. arXiv: 1703.07370. URL: http://arxiv.org/abs/1703.07370.

Williams, Ronald J. (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Mach. Learn.* 8.3-4, pp. 229–256. ISSN: 0885-6125. DOI: 10.1007/BF00992696. URL: http://link.springer.com/10.1007/BF00992696.