Glorot's Initialization[3]

Random Initialization

The simplest way to initialize weights and biases is to set them to small uniform random values. **This works well for NNs with single hidden layer**. But this simple approach does not work sufficient well for deep NNs(especially those that use ReLu activation).

Glorot's Initialization

One common initialization scheme for deep NNs. Two ways:

- In this we initialize each weight with $N\left(0.0, \frac{2.0}{(fan-in+fan-out)}\right)$. Here fan-in is number of input nodes and fan-out is number of hidden notes.
- Can choose weights from a uniform distribution $U\left[\frac{-\sqrt{6}}{\sqrt{fan-in+fan-out}}, \frac{\sqrt{6}}{\sqrt{fan-in+fan-out}}\right]$

Some notes[4]

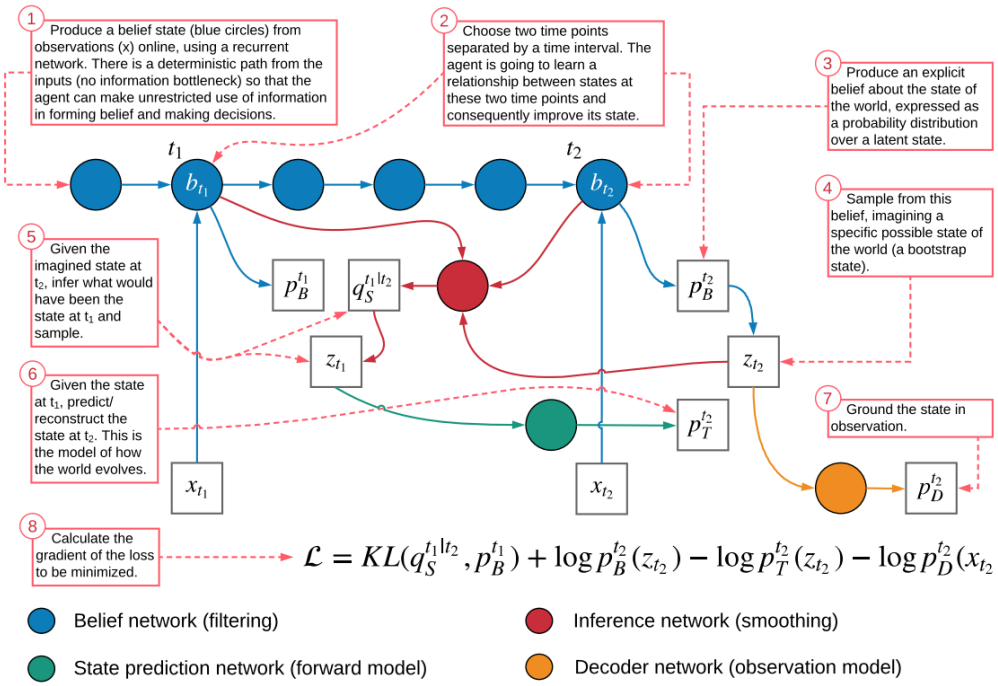
TEMPORAL DIFFERENCE VARIATIONAL AUTO-ENCODER

In partially observed environments, in order to act and explore optimally, agents need to build a representation of the uncertainty about the world, computed from the information they have gathered so far.
To successfully address this, a model of the agent's experience should exhibit the following properties:

- The model should learn an abstract state representation of the data and be capable of making predictions at the state level, not just the observation level.
- The model should learn a belief state, i.e. a deterministic, coded representation of the filtering posterior of the state given all the observations up to a given time. A belief state contains all the information an agent has about the state of the world and thus about how to act optimally.
- The model should exhibit temporal abstraction, both by making 'jumpy' predictions (predictions several time steps into the future), and by being able to learn from temporally separated time points without back propagating through the entire time interval.

Their work

The propose TD-VAE, a generative sequence model that learns representations containing explicit beliefs about states several steps into the future, and that can be rolled out directly without single-step transitions. TD-VAE is trained on pairs of temporally separated time points, usign an analogue of temporal difference learning used in RL.



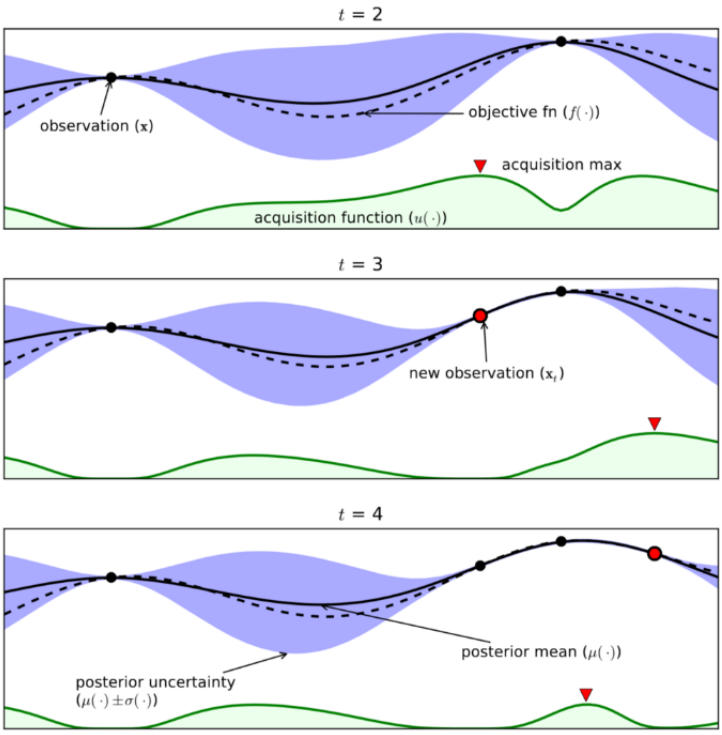
How is this useful for our work

- In reinforcement learning, the state of an agent represents a belief about the sum of discounted rewards. In our case as well as the paper, the model expresses a *belief* about possible future **states** instead of the *sum* of discounted *rewards*.
  Their model trains the belief at time t_1 using belief at some time t_2 in the future. It accomplishes this by (variationally) auto-encoding a sample of the future state into a sample of present of current state.

Bayesian Optimization[1][2]

Introduction

Bayesian optimization is an approach to optimizing objective functions that take a long time to evaluate. It is best-suited for optimization over continuous domains of less that 20 dimensions, and tolerates stochastic noise in function evaluations.



Sample data increases & Curve fitting gets improved.

Relevance: In the paper- Learning to learn without GD with GD, the authors used **Expected Improvement** function to incorporate explicit *exploration* term which will encourage the learning algorithm to explore in addition to exploitation.
Even though in our paper, we are not using black-box function(as used in the above paper), we can still use their EI function to encourage exploration in HJB equation based optimization.

We have a function $f : X \longrightarrow R$ that we want to minimize on some sub-domain of X. That is wish to find

$$x^* = argmin_{x \in X} f(x)$$

Although not strictly required, Bayesian optimization almost always reasons about f by choosing an appropriate Gaussian process prior:

$$p(f) = GP\,(f; \mu, \kappa)$$

Acquisition Functions: Many acquisition functions can be interpreted in the framework of Bayesian decision theory as evaluating an expected loss associated with evaluating f at a point x. We then select the point with the lowest expected loss.

Expected Improvement

Before considering EI, consider the following terms,

- *Improvement*: Random variable defined by

$$I(x) = \max\{f_{min} - f(x), 0\}$$

  ie, $I(x)$ assigns a reward of $f_{min} - f(x)$ if $f(x) < f_{min}$ and zero otherwise.

- *Probability of Improvement(PI)*: Acquisition function defined by

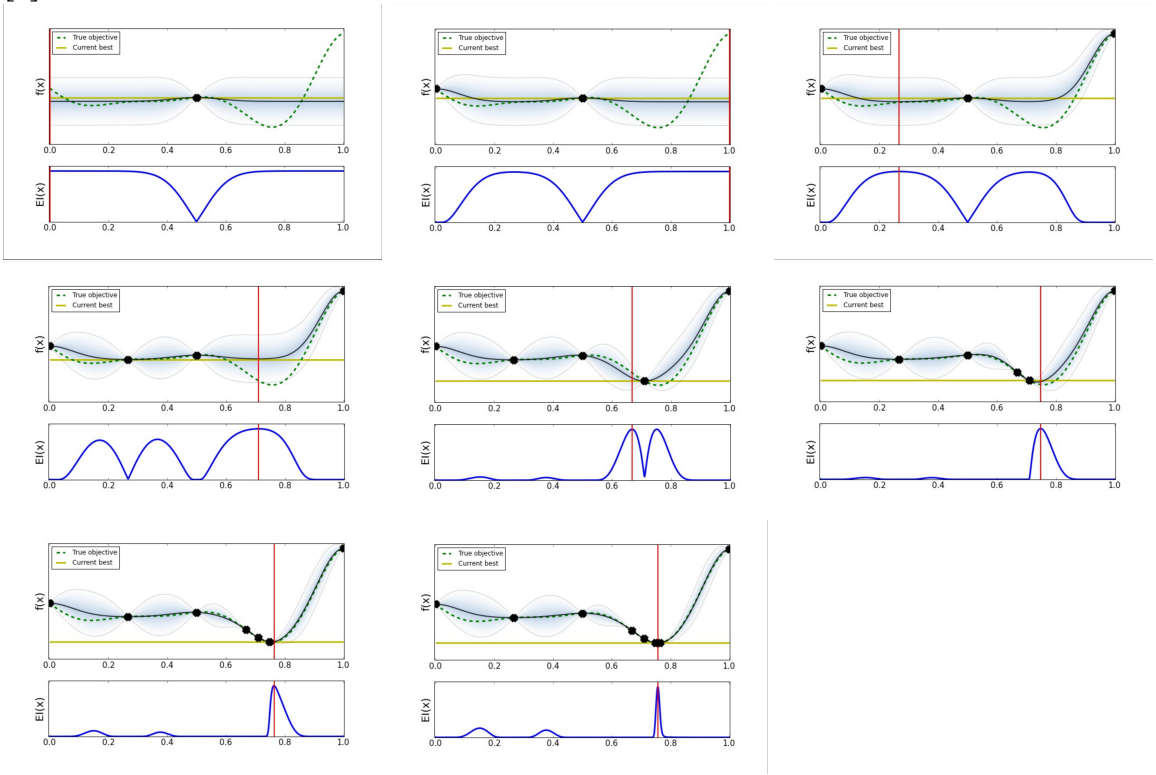
$$\begin{aligned} PI(x) &= P\{I(x) > 0\} \\ &= P\{f(x) < f_{min}\} \end{aligned}$$

  The above function assigns a reward of 1 when we have an improvement, irrespective of the magnitude of this improvement, and 0 otherwise. It seems to be a naive method to assign a rewards always equal to 1, irrespective of the value.

- *Expected Improvement*: It evaluates f at the point most likely to improve upon this value, while also taking into account for the size of the improvement. Hence the name EI.

$$EI(x) = E\{I(x)\}$$

[5]



References

[1] Eric Brochu, Vlad M Cora, and Nando De Freitas. "A tutorial on Bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning". In: *arXiv preprint arXiv:1012.2599* (2010).

[2] Peter I Frazier. "A tutorial on Bayesian optimization". In: *arXiv preprint arXiv:1807.02811* (2018).

[3] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. 2010, pp. 249–256.

[4] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[5] *Krasser, M. (2019). Bayesian optimization - Martin Krasser's Blog. [online]*. `http://krasserm.github.io/2018/03/21/bayesian-optimization/`. Accessed 11 Mar. 2019.