

報酬が殆ど得られない環境での 強化学習

2019/03/12 mabo

自己紹介

金融機関や販社でデータ分析

- 20年以上データ分析一筋
- 正当に特徴量を扱えば予測が当たることに驚く
 - 特徴量とGoalの設定だけで適切に動作する強化学習の能力にも驚く
- 深層強化学習の実装の経験は少ない
- RLアーキテクチャの高度で分り易い講演に感謝

強化学習の全般

ある環境で報酬最大化のため最適行動をするモデル

- 現実の環境(摩擦・制御誤差・ノイズある世界)

誤差を前提にモデル化 SLAM GPS(Guided Policy Search)

- 不完全情報の環境(相互に情報秘匿がある環境)

ナッシュ均衡モデル

- 理想の環境(エージェントからは状態は見える環境)

- a. 複数エージェントの環境 相互協調モデル

- b. 単独エージェントの環境

- 1. 報酬が逐次得られる環境 深層学習Actor-critic(A3C)が有力

- 2. 報酬が殆ど得られない環境

- I. 最後に勝敗が決する環境(数億の遷移がある) 将棋 囲碁

- II. ゴールに達するまで報酬が得られない環境 迷路 倉庫問題



今回の対象

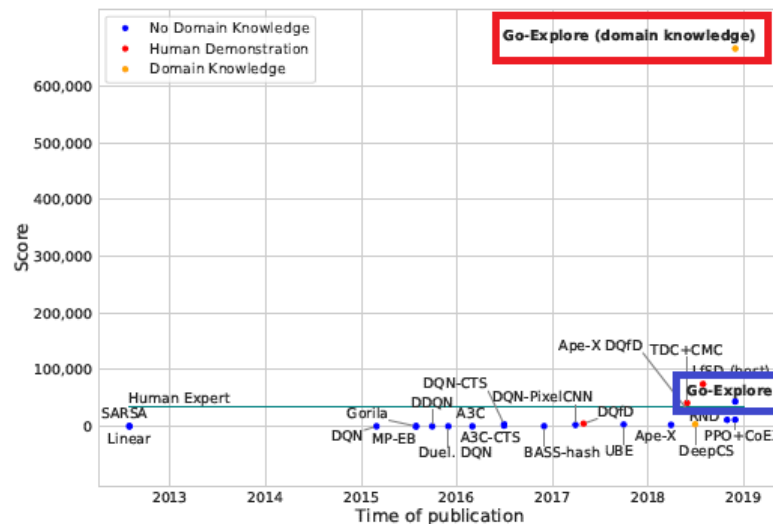
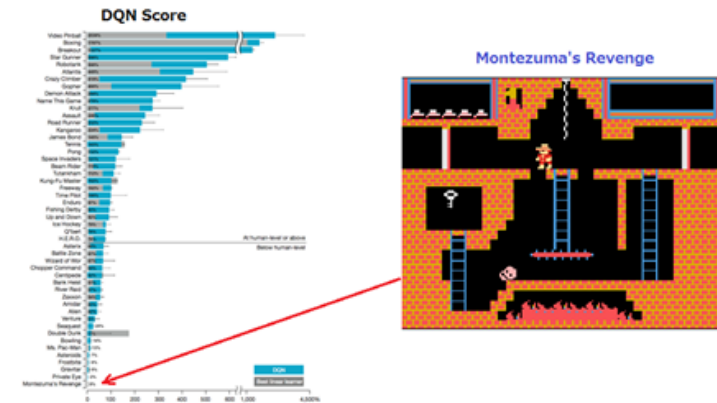
報酬が殆ど得られない環境

- ゴールから逆向きに解くモデル
- 擬似報酬を設定するモデル
 - 逆強化学習(熟練者の経路を教師データとする)
 - 内的動機を報酬とするモデル
 - 蒸留
 - 好奇心
 - 擬似カウント
 - 階層的モデル(広い視野を持つモデル)
 - Optionのサブゴール
 - 封建的階層モデル
 - メタ学習
- 状態表現による環境を理解するモデル
 - GQN TD_VAE Wold Models
- まとめ

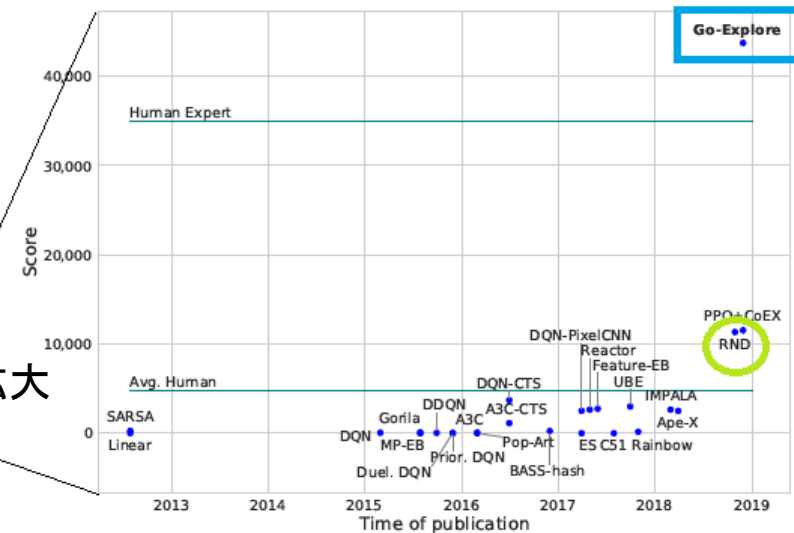
ゴールから逆向きに解くモデル

[Go-Explore: a New Approach for Hard-Exploration Problems \(2019\)](#)

- 「Montezumaの逆襲」はDQNでは最難関ゲーム
- ぶっち切で高得点をだしたUberのモデル(Go-explorer)は様々な擬似報酬モデルを合成したもの



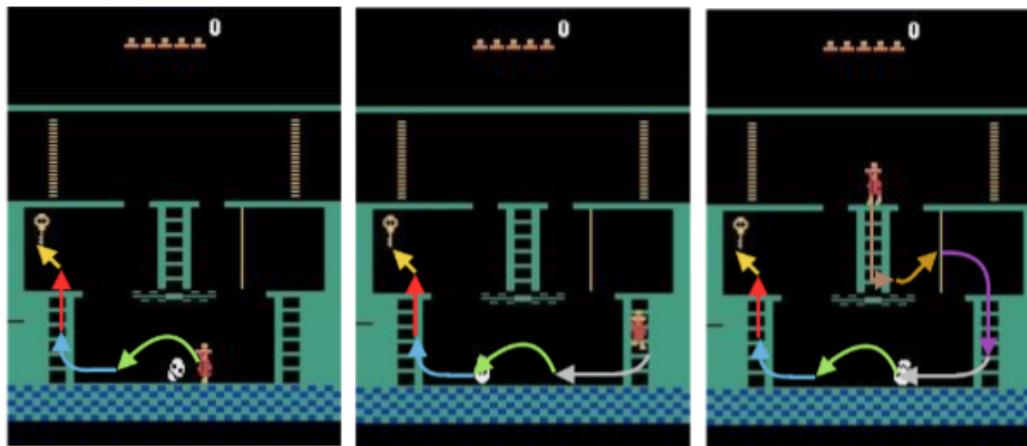
人間のゲーム知識を入れたモデル



人間のゲーム知識を入れ無いモデル

ゴールから逆向きに解くモデル

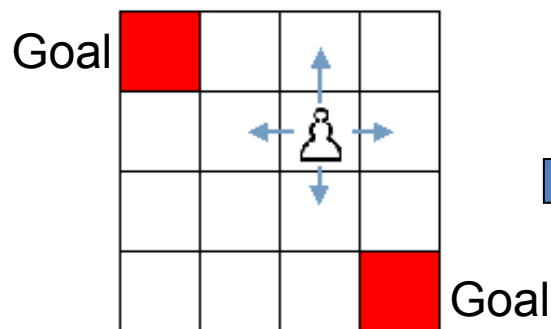
- 2つのPhaseで構成
- Phase1 開始点からの探索
 - 内的動機で探索(新しい場所、高報酬先)
 - 価値が高い場所Option(cell)として選定
 - Option間連結を木構造で構成(別の選択子容易に移動)
- Phase2 ゴールから逆向き経路より逆強化学習
 - ゴール迄達したOption経路毎に強化学習して連結する



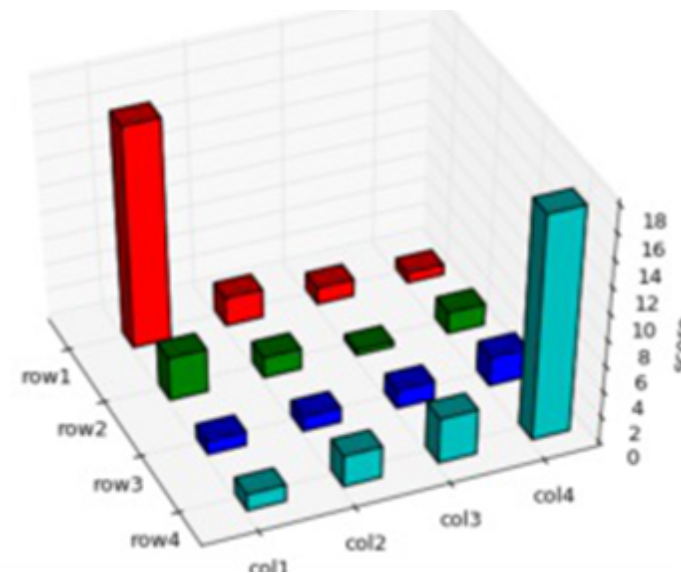
逆強化学習

Maximum Entropy Deep Inverse Reinforcement Learning(2015)

- 報酬は無くとも熟練者の行動経路を教師データとして状態の価値を計算
- 要は熟練者の多く訪れる状態に価値を高く設定している
- しかし経路だけからは熟練者の動機や意図(方策)の変更は判断できない



Open-AIを熟練者として
多数の経路を生成



状態の価値

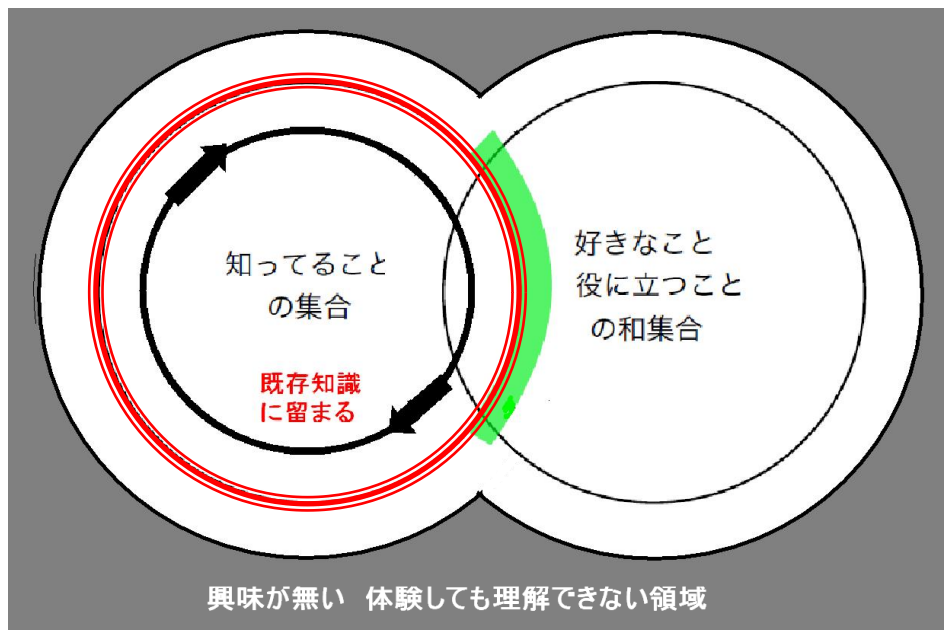
報酬が殆ど得られない環境

- ゴールから逆向きに解くモデル
- 擬似報酬を設定するモデル
 - 逆強化学習(熟練者の経路を教師データとする)
 - 内的動機を報酬とするモデル
 - 蒸留
 - 好奇心
 - 擬似カウント
 - 階層的モデル(広い視野を持つモデル)
 - Optionのサブゴール
 - 封建的階層モデル
 - メタ学習
- 状態表現による環境を理解するモデル
 - GQN TD_VAE Wold Models
- まとめ

蒸留

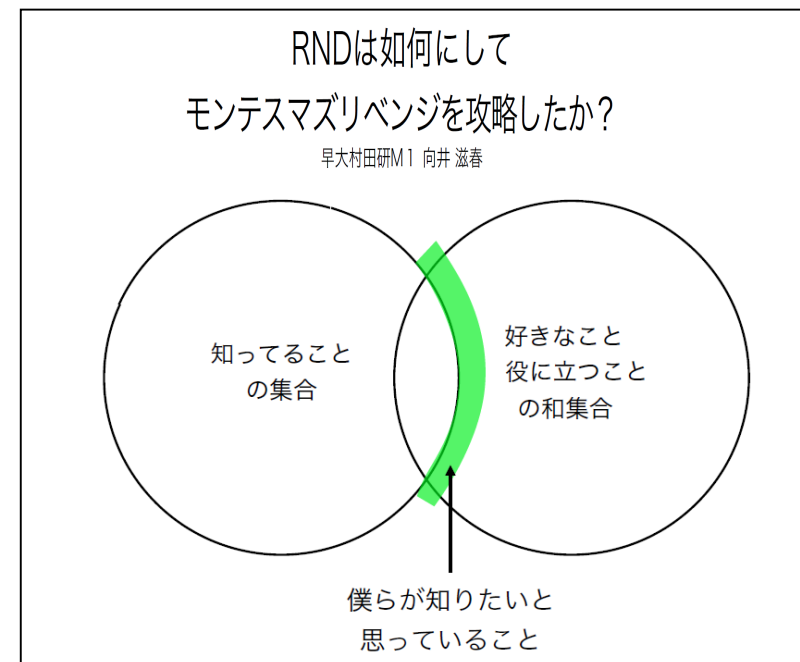
Exploration by Random Network(2018)

- 成功や失敗の体験を通じて知識(Q関数)を獲得する
- 既存の知識では探索が広がらない(未知の道はリスクが高い)
 - 未知の道を選択するにしても興味のある方向に進みたい



2019-9-4

既存知識の壁

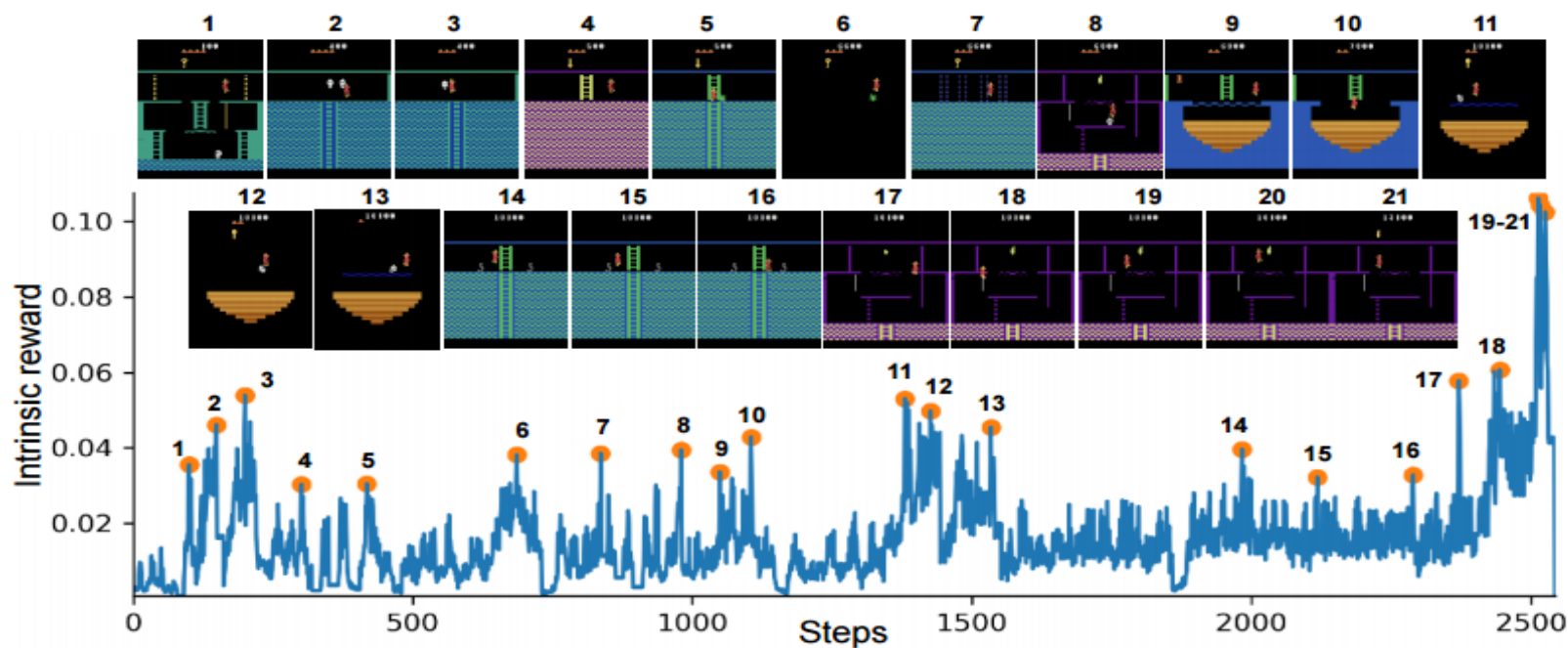


向井滋春氏の資料

蒸留

蒸留を用いたGNDモデルとは

- 深層学習で異なる組成のエージェントを多数生成し、**同じ体験で同じ選択をする**様に訓練する
- 蒸留エージェントが異なる選択した場合、既存知識の延長として選択しなかった場所として擬似報酬を高める
 - 湯上り、夢の中、別人格になると新しい発想が湧く場合がある



2019-9-4 ゲーム場面ではGNDが新規の経験した時、高い擬似報酬となっている

好奇心

- [Curiosity-driven Exploration by Self-supervised Prediction\(2017\)](#)

– 予測との相違の量を擬似報酬とする

$$r_i^t = \frac{\eta}{2} \|\hat{\phi}(s_{t+1}) - \phi(s_{t+1})\|_2^2$$

$\hat{\phi}(s_{t+1})$: 予想した次の状態のencode特徴

$\phi(s_{t+1})$: 実際の次の状態のencode特徴

予想した次の状態の特徴は深層ネットワーク f で算出する (順モデル)

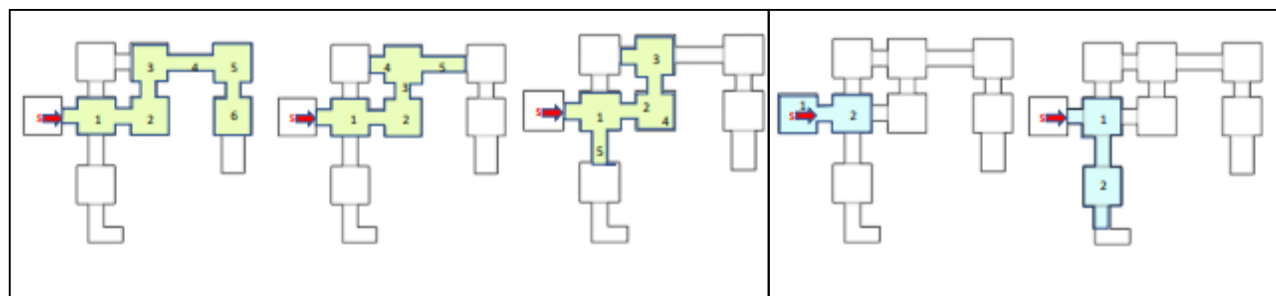
$$\hat{\phi}(s_{t+1}) = f(\phi(s_t), \hat{a}_t; \theta_F)$$

次の行動予測も別の深層ネットワーク g で算出する (逆モデル)

$$\hat{a}_t = g(s_t, s_{t+1}; \theta_I) \quad \leftarrow \theta_I \text{ と } \theta_F \text{ は相違が最小になるよう調整される}$$



(a) Input snapshot in VizDoom



2100歩後の好奇心モデル

2100歩後の探索

擬似カウント

[Unifying Count-Based Exploration and Intrinsic Motivation\(2016\)](#)

- 状態(x)の再訪回数を擬似カウント(pseudo-count) $N(x)$ で表現

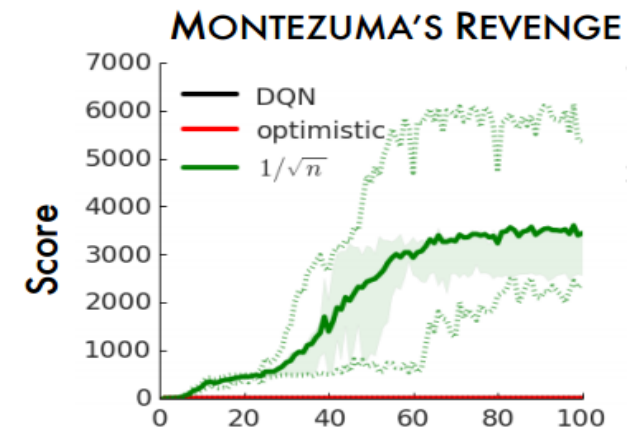
$$\begin{aligned} \rho_n(x) &= \frac{\hat{N}_n(x)}{\hat{n}} \\ \rho'_n(x) &= \frac{\hat{N}_n(x) + 1}{\hat{n} + 1} \end{aligned} \quad \xrightarrow{\text{方程式を解く}} \quad \hat{N}_n(x) = \frac{\rho_n(x)(1 - \rho'_n(x))}{\rho'_n(x) - \rho_n(x)} = \hat{n}\rho_n(x).$$

状態(x)に1回追加する

- 擬似カウントと**反比例**する擬似報酬Rを与える

$$R_n^+(x, a) := \beta(\hat{N}_n(x) + 0.01)^{-1/2},$$

再訪が少ない場所を選択し易くなる



報酬が殆ど得られない環境

- ゴールから逆向きに解くモデル
- 擬似報酬を設定するモデル
 - 逆強化学習(熟練者の経路を教師データとする)
 - 内的動機を報酬とするモデル
 - 蒸留
 - 好奇心
 - 擬似カウント
 - 階層的モデル(広い視野を持つモデル)
 - Optionのサブゴール
 - 封建的階層モデル
 - メタ学習
- 状態表現による環境理解するモデル
 - GQN TD_VAE Wold Models
- まとめ

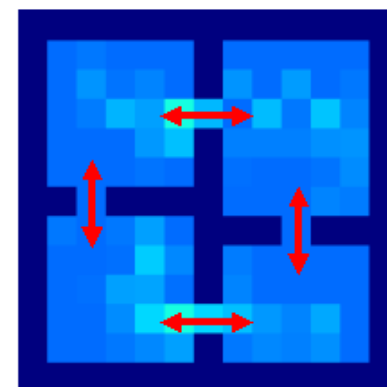
Optionのサブゴール

- OptionとはR.Suttonが1998年に発表した状態遷移Pから求めるサブゴール
<http://www-anw.cs.umass.edu/~barto/courses/cs687/Sutton-Precup-Singh-AIJ99.pdf>
- Option-Critic Optionの終端の確率 $\beta_{w,\theta}(s)$ の**按分**でQ関数を決定
[The Option-Critic Architecture\(2016\)](#)

$$Q_{\Omega}(s, \omega) = \sum_a \pi_{\omega, \theta}(a|s) Q_U(s, \omega, a)$$

$$Q_U(s, \omega, a) = r(s, a) + \gamma \sum_{s'} P(s'|s, a) \boxed{U(\omega, s')}$$

$$\boxed{U(\omega, s')} = (1 - \boxed{\beta_{w, \theta}(s')}) Q_{\Omega}(s', \omega) + \boxed{\beta_{w, \theta}(s')} V_{\Omega}(s')$$



s' がOptionの非終端($\beta_{w, \theta}(s)=0.0$)ならQ関数は将来のOptionで決まる
 s' がOptionの終端($\beta_{w, \theta}(s)=1.0$)ならQ関数は状態 s' の価値で決まる
即ち状態 s' 以降のOption価値は少ない

Optionのサブゴール

Q関数を微分して終端確率 $\beta_{\omega, \theta}(s)$ のパラメータ θ を決定

$$\frac{\partial Q_{\Omega}(s, \omega)}{\partial \theta} = \left[\sum_a \frac{\partial \pi_{\omega, \theta}(a|s) Q_U(s, \omega, a)}{\partial \theta} \right] + \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \frac{\partial U(\omega, s')}{\partial \theta}$$

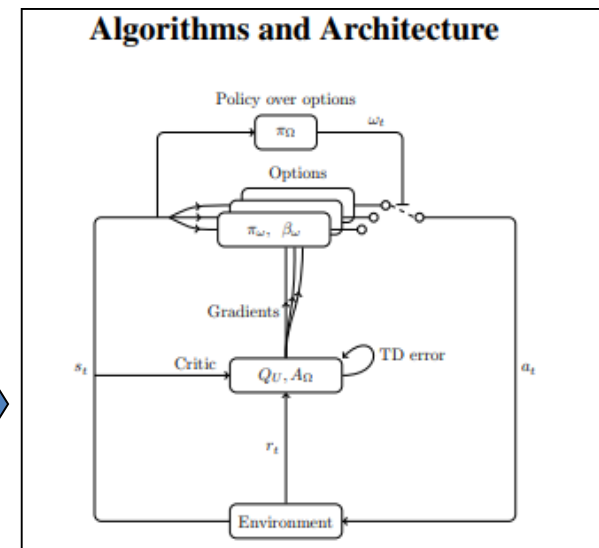
$$\frac{\partial Q_{\Omega}(s, \omega)}{\partial \theta} = \sum_a \pi_{\omega, \theta}(a|s) \sum_{s'} \gamma P(s'|s, a) \frac{\partial U(\omega, s')}{\partial \theta}$$

$$\frac{\partial U(\omega, s')}{\partial \theta} = - \boxed{\frac{\beta_{\omega, \theta}(s')}{\partial \theta}} A_{\Omega}(s', \omega) + \gamma \sum_{\omega'} \sum_{s''} P(s'', \omega'|s', \omega) \frac{\partial U(\omega', s'')}{\partial \theta}$$

$$A_{\Omega}(s', \omega) = Q_{\Omega}(s', \omega) - V_{\Omega}(s')$$

$$\frac{\partial U(\omega, s')}{\partial \theta} = - \sum_{s', \omega} \mu_{\Omega}(s', \omega | s_0, \omega_0) \boxed{\frac{\partial \beta_{\omega, \theta}(s')}{\partial \theta}} A_{\Omega}(s', \omega)$$

Actor部分がOptionになっている



封建的階層

[FeUdal Networks for Hierarchical Reinforcement Learning\(2017\)](#)

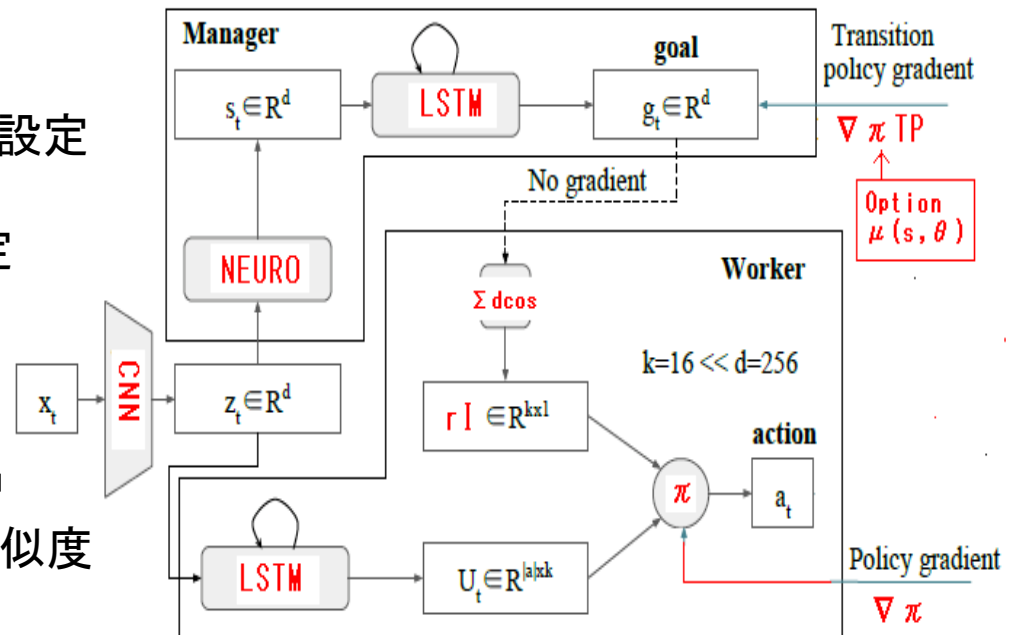
- Option (但し固定区間)を使って上位下位で連携して学習

- 上位の強化学習

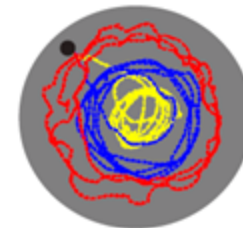
- 固定区間 (C step) 毎に Option を設定
- Actor により最適方策を決定
- 下位の擬似報酬 g_t を RNN で設定

- 下位の強化学習

- 上位の Option 間のみ強化学習
- 擬似報酬を上下経路の Cosin 類似度 (方向の同一性) で調整
- 方策のパラメータは RNN で学習



実験例: 上位は渦の位置 下位は渦の中の探索



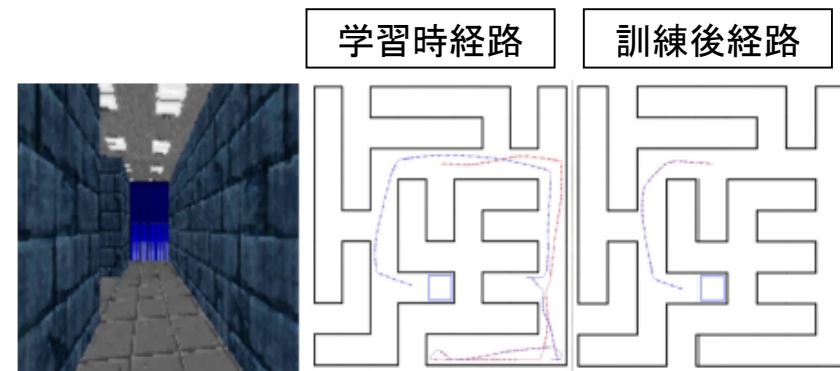
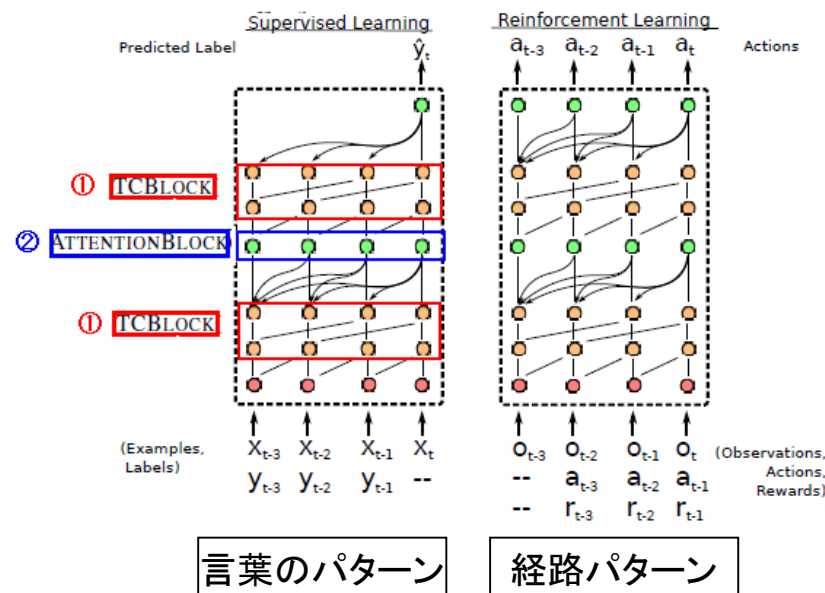
報酬が殆ど得られない環境

- ゴールから逆向きに解くモデル
- 擬似報酬を設定するモデル
 - 逆強化学習(熟練者の経路を教師データとする)
 - 内的動機を報酬とするモデル
 - 蒸留
 - 好奇心
 - 擬似カウント
 - 階層的モデル(広い視野をもつモデル)
 - Optionのサブゴール
 - 封建的階層モデル
 - メタ学習
 - Attentionによるメタ学習
 - RNNによるメタ学習
- 状態表現による環境を理解するモデル
 - GQN TD_VAE Wold Models
- まとめ

メタ学習: Attention

A Simple Neural Attentive Meta-Learner(2017)

- 言葉間の同時生起確率を学習する自己Attentionモデルを使う
- 強化学習の経路より行動の同時生起確率を経路パターンとしてメタ学習
- モデルはゴールに近いパターンを覚えているので、これに似れば近道する



Method	Large Maze	
	Episode 1	Episode 2
Random	420.2 ± 1.2	420.8 ± 1.2
LSTM	180.1 ± 6.0	150.6 ± 5.9
SNAIL (ours)	140.5 ± 4.2	105.9 ± 2.4

RNNによるメタ学習(1)

[Learning to learn by gradient descent by gradient descent\(2016\)](#)

- 1万次元ほどのパラメータの収束を2系列の深層学習で実現

$$\theta^* = \operatorname{argmin}_{\theta \in \Theta} f(\theta) \quad \text{最小化}$$

• Learning to learn

• SDG

$$\theta_{t+1} = \theta_t + \boxed{g_t}(\nabla f(\theta_t), \phi) \quad \theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)$$

θ_t は最適化パラメータ(Optimizee)で1万次元を想定

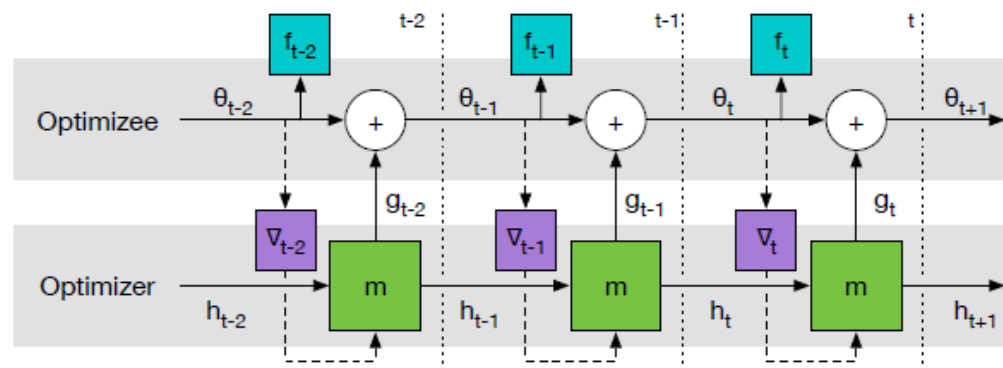
g_t は最適化関数(Optimizer)

ϕ はOptimizerのパラメータ

損失関数Lが最小になる様に Φ をmで解く

$$\mathcal{L}(\phi) = \mathbb{E}_f[\sum_{t=1}^T w_t f(\theta_t)] \quad \text{実装上 } w_t = 1$$

$$[g_t, h_{t+1}] = m(\nabla_t, h_t, \phi) \quad \nabla_t = \nabla_{\theta} f(\theta_t)$$



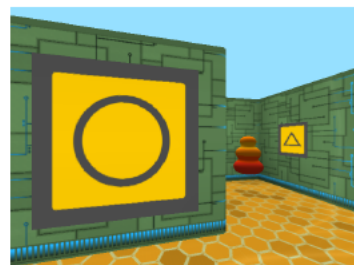
RNNによるメタ学習(2)

[Learning to reinforcement learn](#)(2016)

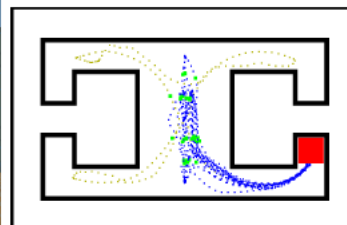
- RNNで学習のパラメータ収束状況をメタ知識として強化学習に適用

	Learning to Learn	Meta-RL
目的	関数の最適化のパラメータ探索	報酬最大化するパラメータ探索
利点	最適化対象の関数に依存しない	Model-BaseでなくModel-Free
過程	パラメータの勾配の収束過程	MDP過程
方法	パラメータの勾配改善	A3Cでの方策とQ関数のパラメータ改善
次元	パラメータ1万次元	画像によるゲームでは高次元

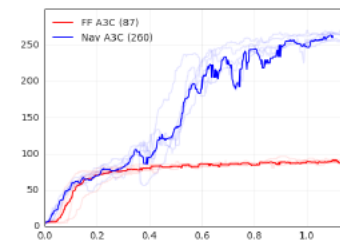
- このメタ知識を用いてノイズや環境の変化に耐えうる学習を達成
 - 看板に応じて宝のある場所をMeta-RLで記憶



(a) Labyrinth I-maze



(b) Illustrative Episode



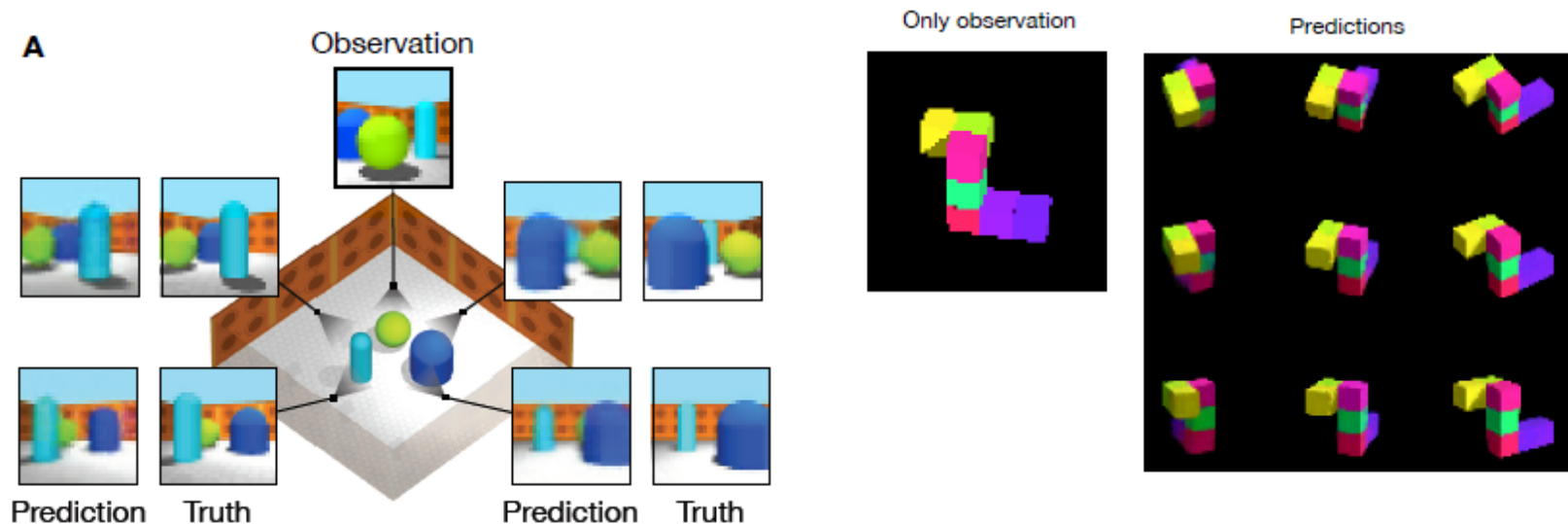
(c) Performance

報酬が殆ど得られない環境

- ゴールから逆向きに解くモデル
- 擬似報酬を設定するモデル
 - 逆強化学習(熟練者の経路を教師データとする)
 - 内的動機を報酬とするモデル
 - 蒸留
 - 好奇心
 - 擬似カウント
 - 階層的モデル(広い視野を持つモデル)
 - Optionのサブゴール
 - 封建的階層モデル
 - メタ学習
- **状態表現による環境理解するモデル**
 - **GQN**
 - **TD_VAE**
 - **Wold Models**
- **まとめ**

VAEによる内部表現(GQN)

- [Neural Scene Representation and Rendering\(2018\)](#)
- 複数の2D画面から3D構造を推定するGQN



VAEによる内部表現(GQN)

- 観測 $p(x)$ を実態 $q(z)$ で近似する変分ベイズモデル
- PRMLの10章(持橋大地訳)変分定理 (復習)

$$\log p(x) = \int q(z) \log \frac{p(x,z)}{q(z)} dz - \int q(z) \log \frac{p(z|x)}{q(z)} dz$$

$$\mathcal{L}(q) = \int q(z) \log \frac{p(x,z)}{q(z)} dz$$

$$\mathcal{KL}(q||p) = - \int q(z) \log \frac{p(z|x)}{q(z)} dz$$

$$\log p(x) = \mathcal{L}(q) + \mathcal{KL}(q||p) \quad \text{qの出来とpとqの近さで表せられる}$$

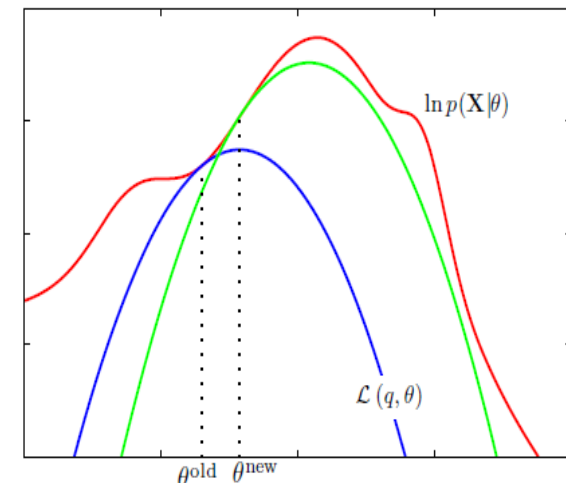
これは次の様に式を展開すると証明できる。

$$\mathcal{L}(q) + \mathcal{KL}(q||p) = \int q(z) \log \frac{p(x,z)}{q(z)} \frac{q(z)}{p(z|x)} dz$$

$$\mathcal{L}(q) + \mathcal{KL}(q||p) = \int q(z) \log \frac{p(x,z)}{p(z|x)} dz$$

ところで $p(z|x) = p(x,z)/p(x)$ なので

$$\mathcal{L}(q) + \mathcal{KL}(q||p) = \int q(z) \log p(x) dz = \log(x)$$



VAEによる内部表現(GQN)

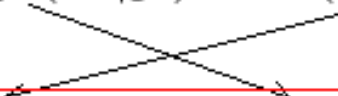
- 見る場所(y)の条件付変分ベイズモデルに拡張
- Encode側 $q(z|x,y)$ とdecoder側 $g(z|x,y)$ のでのKL距離を一致させて損失関数ELBOを最小化する

見る場所を条件 y とした条件付き変分式を解いている,

$$\log p(x|y) = \mathcal{L}q(z|x,y) + \mathcal{KL}(q(z|x,y)||p(x|y))$$

観測点が複数ある場合、損失関数 $\mathcal{F}(\theta, \phi)$ を使って

$$\sum_i \log g_\theta(x_i|y_i) = \mathcal{F}(\theta, \phi) + \sum_i \mathcal{KL}(q_\phi(z_i|x_i, y_i)||g_\theta(z_i|x_i, y_i))$$


$$\mathcal{F}(\theta, \phi) = \sum_i \log g_\theta(x_i|y_i) + \sum_i \mathcal{KL}(q_\phi(z_i|x_i, y_i)||g_\theta(z_i|x_i, y_i))$$

$$\mathcal{F}(\theta, \phi) \geq \sum_i \log g_\theta(x_i|y_i)$$

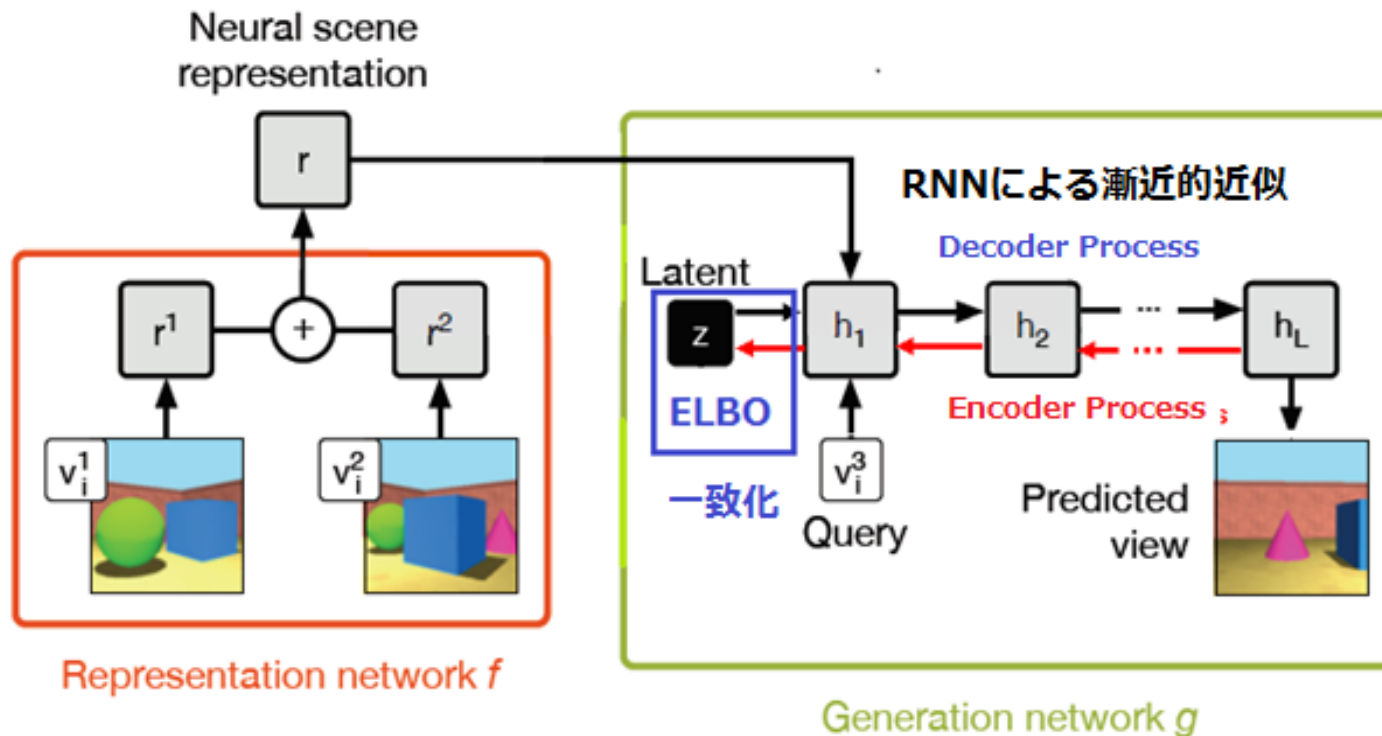
Encode

Decode

$$\mathcal{F}(\theta, \phi) \geq -\mathcal{L}(\theta) \quad \text{ELBO(Evidence lower bound)}$$

VAEによる内部表現(GQN)

- GQNの下位限界ELBOのKL距離は解析的に解けない
 - 複数の2D画面(v)から3D構造(z)の分布は複雑である可能性が高い
 - 条件付VAEの場合は実態(z)を混合ガウスなので解析的に解ける
- 潜在構造(z)を L 個に分割してRNNにより漸近的にKL距離を最小化する



VAEによる内部表現(GQN)

特徴量の合成 但し実装では f は単純和

$r = f(x^1, \dots, x^M, v^1, \dots, v^M)$ は画像群の特徴情報

x^i は観測画像 v^i は観測画像のカメラ位置と傾き

v^q は推定したい画像のカメラ位置

実体 z を L 個に分解

$$\pi_{\theta}(z|v^q, r) = \prod_{l=1}^L \pi_{\theta_l}(z_l|v^q, r, z_{z>l})$$

L 個に分解された実体 z を多次元正規分布 N で近似

(a-1) Decoder側(Generation architecture)

$$g_{\theta_l}(z_l|v^q, r, z_{z>l}) = \mathcal{N}(z_l|\eta_{\theta}^g(h_l^g))$$

正規分布のパラメータはLSTMで漸近近似

$$(c_{l+1}^g, h_{l+1}^g, u_{l+1}) = \text{ConvLSTM}_{\theta}^g(v_q, r, c_l^g, h_l, u_l, z_l)$$

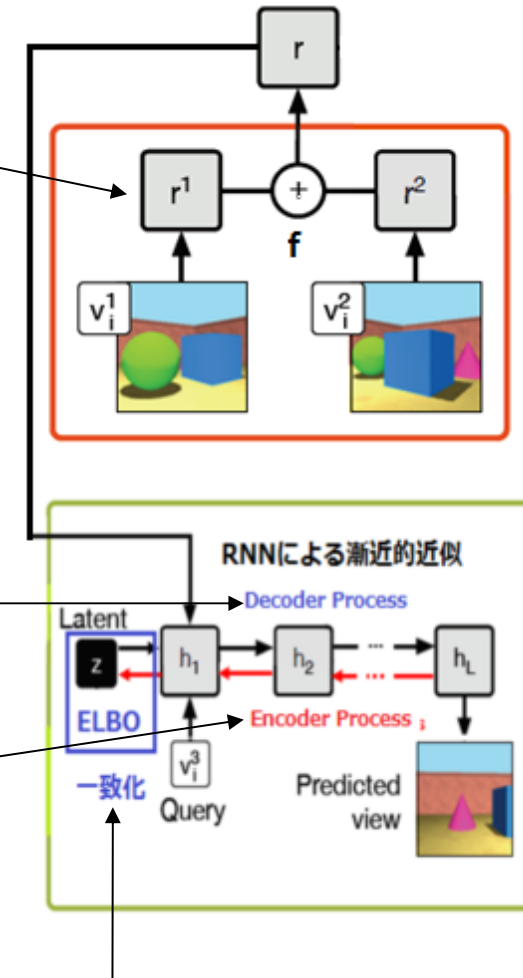
(a-2) Encoder側(inference architecture)

$$q_{\phi_l}(z_l|x_q, v_q, r, z_{z>l}) = \mathcal{N}(z_l|\eta_{\phi}^e(h_l^e))$$

$$(c_{l+1}^e, h_{l+1}^e) = \text{ConvLSTM}_{\phi}^e(x_q, v_q, r, c_l^e, h_l^e, u_l)$$

(a-3) ELBOによる z_l の一致

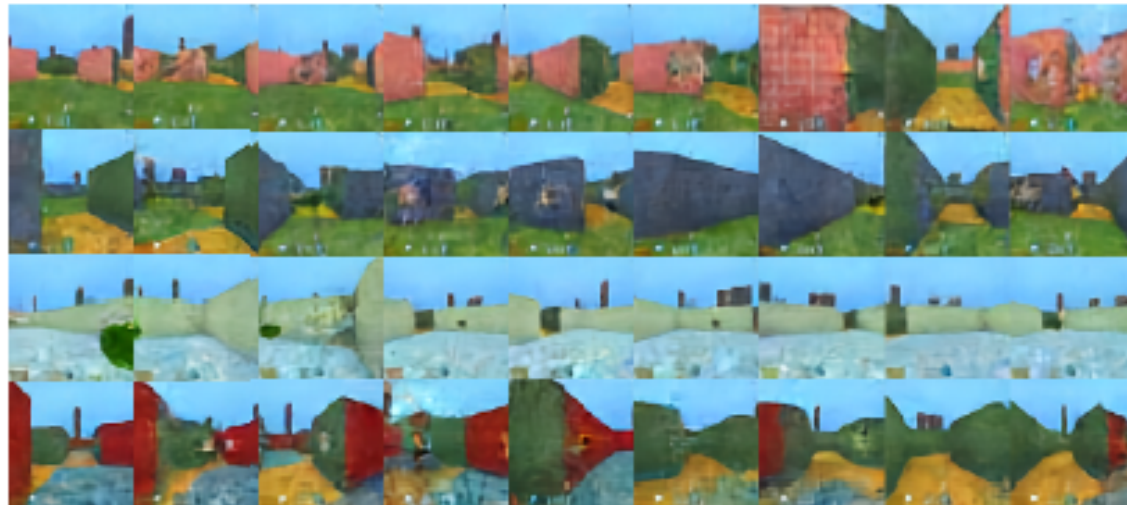
$$\mathcal{F}(\theta, \phi) = \mathbb{E}_{x,v,z \sim \psi(\phi)} [-\log N(x^q|\eta_{\theta}^g(u_L)) + \sum_{l=1}^L \mathcal{KL}(N(z|\eta_{\phi}^e(h_l^e))||N(z|\eta_{\theta}^g(h_l^g)))]$$



時系列のVAE(TD_VAE)

[Temporal Difference Variational Auto-Encoder\(2018\)](#)

- 観察の時系列 $[x_1, \dots, x_N]$ から実体 $[z_1, \dots, z_N]$ をEncodeする変分ベイズ
- 実体 $[z_1, \dots, z_N]$ の推移から観察の時系列をDecodeして生成する



時系列のVAE(TD_VAE)

- 実体 $[z_1, \dots, z_N]$ の推移は観察 $[x_1, \dots, x_N]$ から変分ベイズで推定できる
- 損失関数は Smoother Encoder Transformer Decodeの和で表現できる

実体の推移確率 $p(z_t|z_{t-1})$ を全観察 $[x_1, \dots, x_t]$ から推定するEncoder $q(z|x)$ を導入

$$\log p(x) = \int \log p(x, z) dz = \int \log p(x|z) p(z) dz = \mathbb{E}_{z \sim p(z|x)} [\log p(x|z)]$$

$z \sim p(z|x)$ の代わりに $z \sim q(z|x)$

$$\log p(x) \geq \mathbb{E}_{z \sim q(z|x)} [\sum_t \log p(x_t|z_t) + \boxed{\log p(z_t|z_{t-1}) - \log q(z_t|z_{t-1}, \phi_t(x))}]$$

ELBO(Evidence Lower Band Optimizer)モデル

過去の観察 $x_{<t}$ に依存し、2回の z_t, z_{t-1} の両方が $x_{<t}$ に依存する

$$\log p(x_t|x_{<t}) \geq \mathbb{E}_{z \sim q(z_t, z_{t-1}|x_{<t})} [\log p(x_t|z_t) + \log(z_t|x_{<t}) + \log p(z_t|z_{t-1}) - \log q(z_t|x_{<t}) - \log q(z_{t-1}|z_t, x_{<t})]$$

記憶の概念 b_t をRNNで導入する

t 時点までの観測 $x_{<t}$ を $b_t = f_B(b_t, x_t)$ とする

$$\log p(x_t|x_{<t}) \geq \mathbb{E}_{z_t, z_{t-1} \sim \psi(z, b)} [\log p(x_t|z_t) + \log p_B(z_t|b_t) + \log p(z_t|z_{t-1}) - \log p_B(z_t|b_t) - \log q(z_{t-1}|z_t, b_{t-1}, b_t)]$$

TD_VAE Jumpyモデル

ステップ間 $[t_1 \sim t_2]$ でのELBOモデルに変換

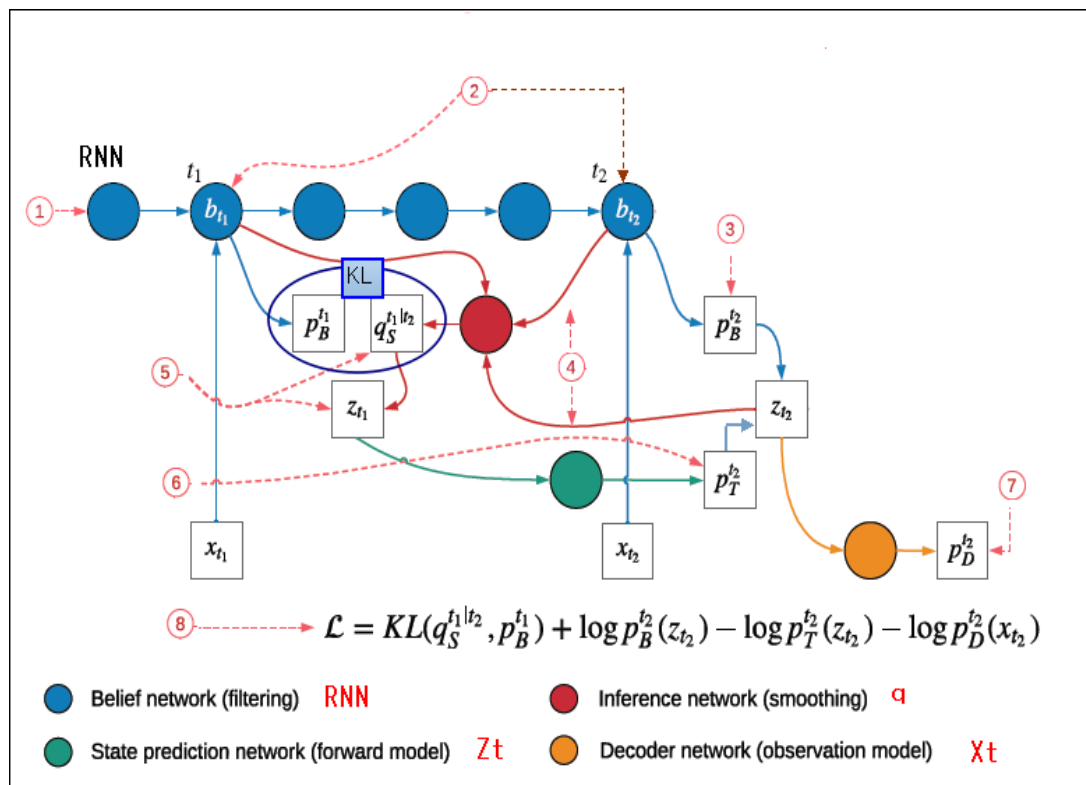
$$\mathcal{L}_{t_1, t_2} = \mathbb{E}_{z_{t_1}, z_{t_2} \sim \psi(z, b)} [\underbrace{\log p(x_{t_2}|z_{t_2})}_{\text{Decoder}} + \underbrace{\log p_B(z_{t_1}|b_{t_1})}_{\text{Transform}} + \underbrace{\log p(z_{t_2}|z_{t_1})}_{\text{Transform}} - \underbrace{\log p_B(z_{t_2}|b_{t_2})}_{\text{Encoder}} - \underbrace{\log q(z_{t_1}|z_{t_2}, b_{t_1}, b_{t_2})}_{\text{Smoother}}]$$

TD_VAEのアルゴリズム

$$\mathcal{L} = \boxed{\mathcal{KL}(q_S^{t_1|t_2} || p_B^{t_1})} + \underbrace{\log p_B^{t_2}(z_{t_2})}_{\text{Encoder}} - \underbrace{\log p_T^{t_2}(z_{t_2})}_{\text{Transform}} - \underbrace{\log p_D^{t_2}(x_{t_2})}_{\text{Decoder}}$$

時系列のVAE(TD_VAE)

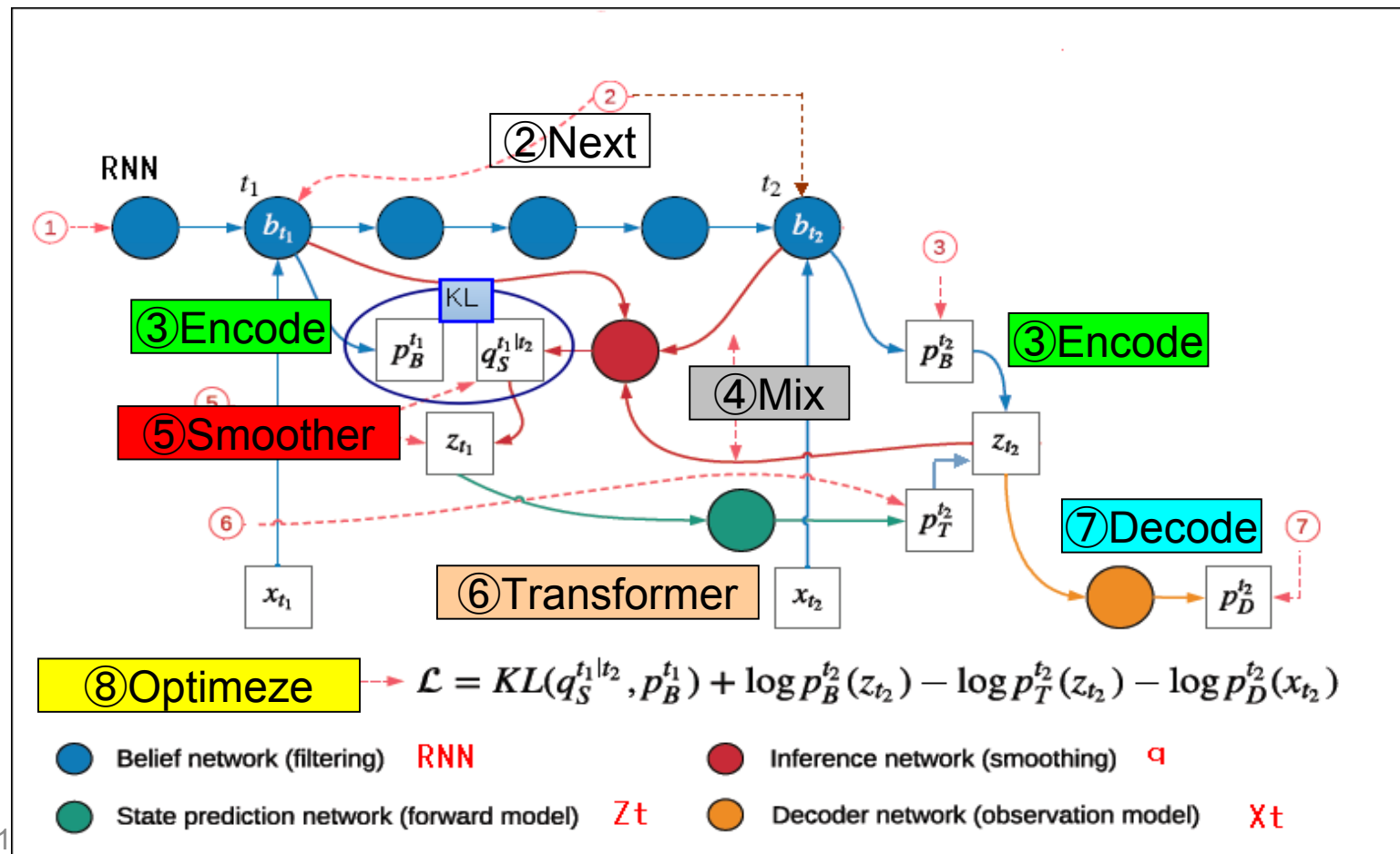
- 観察(x)の履歴をRNNの記憶(b)にして2時点[z1,z2]で逐次的に解く
- **Encoder Transformer Smoother**を使って実体[z]を計算
- 損失関数を最小化するため**Encoder** $q(\phi)$ と**Decoder** $p(\theta)$ のパラメータ更新



- ① 観察(x1)よりRNN記憶(b1)を生成
- ② 次の道標を選択
- ③ RNN記憶(b2)より**Encoder**で実体(z2)を予測
- ④ 2点間の記憶(b1とb2)から**Encoder**で予測実体(z1とz2)を求める**Smoother**で予測実体(z1)を計算
- ⑤ KL距離を使って**Encoder**と**Smoother**の予測実体の相違を計算
- ⑥ **Transformer**を使って次の予測実体(z2)を計算
- ⑦ 予測実体(z2)を**Decoder**して観察(x2)を表現
- ⑧ 損失関数Lを最小化するため**Encoder** $q(\phi)$ と**Decoder** $p(\theta)$ のパラメータを最適化

時系列のVAE(TD_VAE)

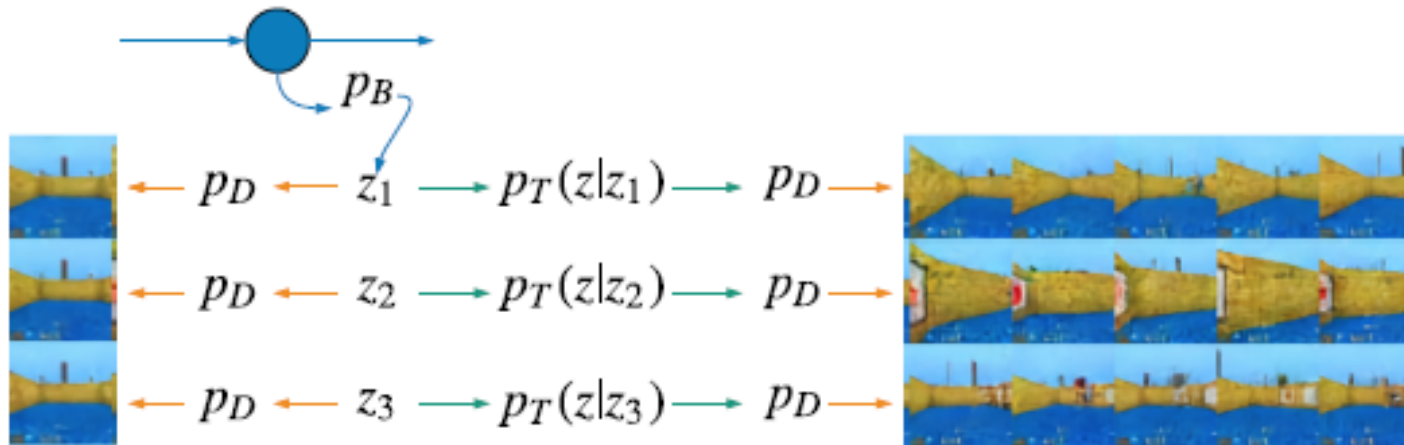
- 観察(x)の履歴をRNNの記憶(b)にして2時点[z1,z2]で逐次的に解く
- **Encoder Transformer Smoother(2時点)**を使って実体[z]を計算
- 損失関数を最小化するため**Encoder $q(\phi)$** と**Decoder $p(\theta)$** のパラメータ更新



時系列のVAE(TD_VAE)

実験例

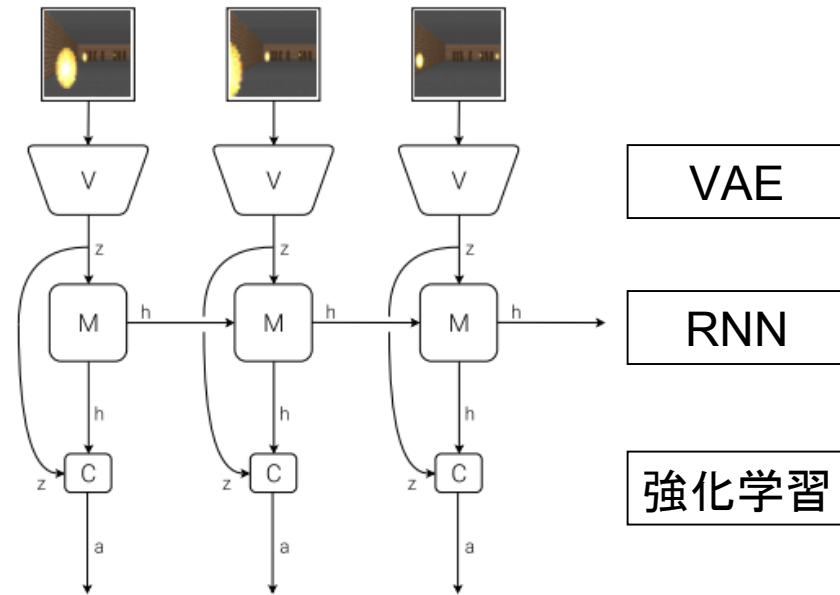
- RNN系列の記憶(b)から実体(z_1, z_2, z_3)をサンプリング
- 左図は実体(z_1, z_2, z_3)からDecodeした画像は大体同じ画像
- しかし右図は実体(z_1, z_2, z_3)から推移としてサンプリングされた実体をDecodeした画像は連続の様子が各々異なる



World Models

[World Models\(2018\)](#)

- ゲーム画面→VAEで抽象化
→仮想ゲーム画面を生成
- VAEで5000個の混合正規分布で抽象化される
- 仮想的な画面での訓練が
実ゲームでも寄与すること
発見
 - 敵が壁の中から出現、敵が
宙に浮く等
- GDNと同様に現実の壁に
収まらない訓練を行う



現実のゲーム



仮想のゲーム

まとめ

- Go-exploreは「Montezuma逆襲」でぶち切の得点を出した
 - 報酬が殆ど得られない場合のモデルの合成
 - 逆強化学習、内的動機探索、階層的探索
 - しかしゴールは確認できるので、逆向きに学習が適用しやすい
- 人間が持つ課題は一般にゴールが定かでない
 - 内的動機探索
 - 好奇心 GND:蒸留(異なる組成)での探索 擬似カウント
 - 階層的探索
 - 隘路や重大な局面をサブゴールとするOption メタ学習
 - 状態表現:現象から実体を捉える
 - GQN TD-VAE World_Models 全てVAEの発展形
- ゴールが見えない場合での計算機が効果的に解くモデルは人間が課題に対処する方法に大きな示唆を与える様になっている