Modeling

Monday, 29 June 2020 17:27

§ 0. Basics of Quantum Mechanics

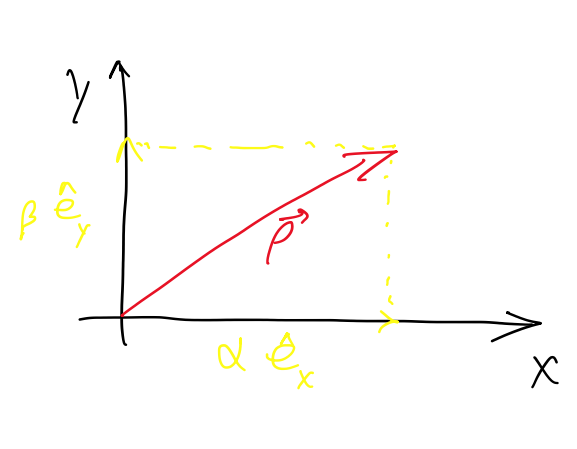
- Superposition ⤳ linear combination for possible states of qubits.

Bits 0 or 1

Quantum bits (qubits) $\alpha(0) + \beta(1)$ ⤳ like a probability dist. ($20\% = \alpha$, $80\% = \beta$)

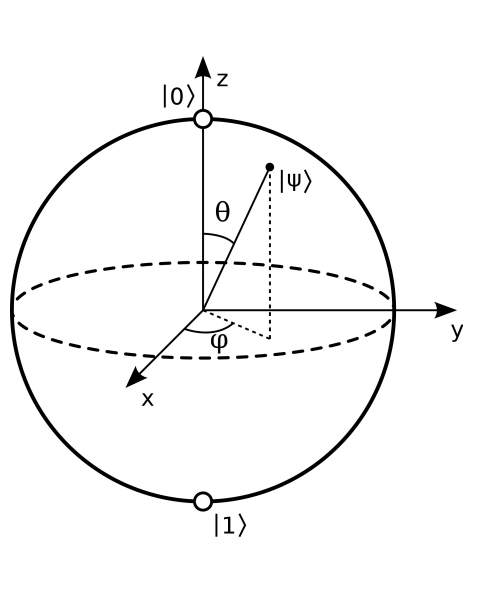
- Computational basis ⤳ defined by the number of qubits.

(*) Cartesian Coord.



$$\vec{\rho} = \alpha\hat{e}_x + \beta\hat{e}_y$$

(*) computational basis for 1 qubit:



$$|\Psi\rangle = \overbrace{\cos(\theta/2)}^{\alpha}\,|0\rangle + \overbrace{e^{i\varphi}\sin(\theta/2)}^{\beta}\,|1\rangle$$

$$|\Psi\rangle = \begin{pmatrix}\alpha\\ \beta\end{pmatrix} = \alpha\begin{pmatrix}1\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\end{pmatrix} = \alpha|0\rangle + \beta|1\rangle$$

(*) 2-qubits: $|\Psi\rangle = \begin{pmatrix}\alpha\\ \beta\\ \gamma\\ \delta\end{pmatrix} = \gamma\begin{pmatrix}1\\0\\0\\0\end{pmatrix} + \beta\begin{pmatrix}0\\1\\0\\0\end{pmatrix} + \gamma\begin{pmatrix}0\\0\\1\\0\end{pmatrix} + \delta\begin{pmatrix}0\\0\\0\\1\end{pmatrix}$

$$= \alpha|00\rangle + \beta|01\rangle + \gamma|10\rangle + \delta|11\rangle$$

(*) n-qubits state : $|\Psi\rangle = \alpha_1|0\cdots0\rangle + \alpha_2\cdots + \alpha_N|1\cdots1\rangle$

$N = 2^n$ ⤷ W-State : $|\Psi_W\rangle = \frac{1}{\sqrt{N}}\sum_{J=1}^{N}|J\rangle$

(*) $\frac{1}{\sqrt{N}}|7\rangle = \underbrace{\frac{1}{\sqrt{N}}}_{\alpha_7}\underbrace{|0\ldots0111\rangle}$ ⤷ $1\times2^0 + 1\times2^1 + 1\times2^2 + 0\cdots$

- Evolution of the state ⤳ Unitary evolution : $UU^\dagger = \mathbb{1}$

$$|\Psi(t_F)\rangle = U(t_F, t_I)\,|\Psi(t_I)\rangle$$

- Measurements ⤳ we want to know $\alpha_i, \forall i$, but in each single measurement we just get one of the α_i (wave-function collapse) ⤳ therefore we need to statistically find the α_i $\forall i$ ⤳ sampling the state $|\Psi\rangle$ for a number of times and calculating the probability distribution.

- Entanglement : $|\Psi\rangle_{AB} \neq |\Psi\rangle_A|\Psi\rangle_B$

Ex: Proof of contradiction :

$$\frac{1}{\sqrt{2}}\left(|00\rangle + |11\rangle\right) = \left(\alpha_1|0\rangle + \beta_1|1\rangle\right)\left(\alpha_2|0\rangle + \beta_2|1\rangle\right)$$

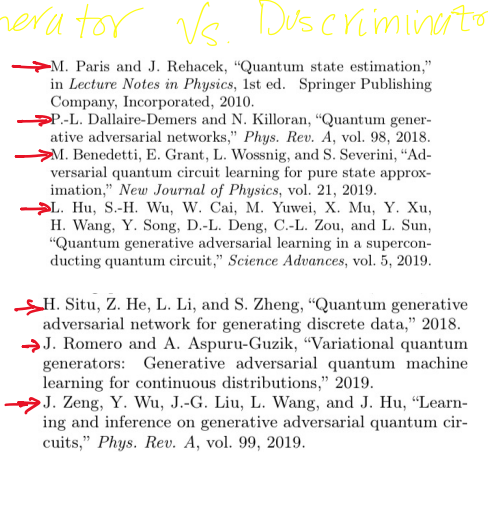
$\alpha_i, \beta_i = ?$

$$\begin{cases}\alpha_1\alpha_2 = \beta_1\beta_2 = 1\\ \alpha_1\beta_2 = \alpha_2\beta_1 = 0\end{cases} \leadsto \text{✗}$$

§ 1. Models

A) arxiv: 1904.00043 V2 : Quantum Generative Adversarial Networks for Learning and Loading Random Distributions.

- The best known method requires $O(2^n)$ gates to load an exact representation of a generic data structure into n-qubit state.
- Therefore, we need to use an approximate hybrid quantum-classical algorithm to load the generic dataset using $O(\text{poly}(n))$ gates.
- We look at a generative model: Generator Vs Discriminator



M. Paris and J. Rehacek, "Quantum state estimation," in *Lecture Notes in Physics*, 1st ed. Springer Publishing Company, Incorporated, 2010.

P.-L. Dallaire-Demers and N. Killoran, "Quantum generative adversarial networks," *Phys. Rev. A*, vol. 98, 2018.

M. Benedetti, E. Grant, L. Wossnig, and S. Severini, "Adversarial quantum circuit learning for pure state approximation," *New Journal of Physics*, vol. 21, 2019.

L. Hu, S.-H. Wu, W. Cai, M. Yuwei, X. Mu, Y. Xu, H. Wang, Y. Song, D.-L. Deng, C.-L. Zou, and L. Sun, "Quantum generative adversarial learning in a superconducting quantum circuit," *Science Advances*, vol. 5, 2019.

H. Situ, Z. He, L. Li, and S. Zheng, "Quantum generative adversarial network for generating discrete data," 2018.

J. Romero and A. Aspuru-Guzik, "Variational quantum generators: Generative adversarial quantum machine learning for continuous distributions," 2019.

J. Zeng, Y. Wu, J.-G. Liu, L. Wang, and J. Hu, "Learning and inference on generative adversarial quantum circuits," *Phys. Rev. A*, vol. 99, 2019.

two use cases
- ① Quantum state estimation :
- ② Generating classical data samples in accordance with an underlying PDF :

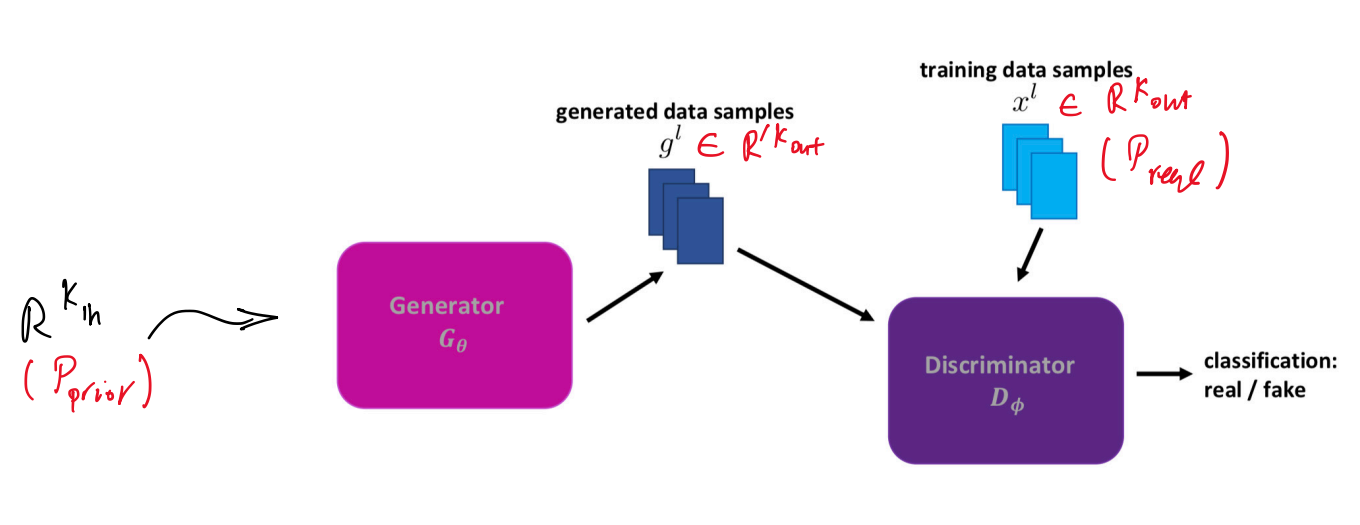
- In this proposal they load PDFs into quantum states.

Q-GAN Model:

$X = \{x^0, \ldots, x^{s-1}\} \subset R^{k_{out}}$ ⤳ sampled from an unknown probability dist. P_{real}.

Let $\underset{\text{gen.}}{G_\theta} : R^{k_{in}} \to R^{k_{out}}$ & $\underset{\text{Dis.}}{D_\varphi} : R^{k_{out}} \to \{0,1\}$

where $\theta \in R^{k_g}$ & $\varphi \in R^{k_d}$ are network parameters.



Loss functions:

$$\begin{cases}\mathcal{L}_G(\varphi,\theta) = -\mathbb{E}_{z\sim P_{prior}}\left[\log\left(D_\varphi\left(G_\theta(z)\right)\right)\right]\\ \mathcal{L}_D(\varphi,\theta) = \mathbb{E}_{x\sim P_{real}}\left[\log D_\varphi(x)\right] + \mathbb{E}_{z\sim P_{prior}}\left[\log\left(1 - D_\varphi\left(G_\theta(z)\right)\right)\right]\end{cases}$$

(*) we can approximate the expected values by batches of size m :

$$\begin{cases}\mathcal{L}_G(\varphi,\theta) = -\frac{1}{m}\sum_{l=1}^{m}\left[\log\left(D_\varphi\left(G_\theta(z^l)\right)\right)\right]\\ \mathcal{L}_D(D_\varphi, G_\theta) = \frac{1}{m}\sum_{l=1}^{m}\left[\log D_\varphi(x^l) + \log\left(1 - D_\varphi\left(G_\theta(z^l)\right)\right)\right]\end{cases} \quad \text{for}\begin{cases}x^l \in X\\ z^l \sim P_{prior}\end{cases}$$

Then, training of GAN is equivalent to finding the Nash-equilibrium of a two-player game, using ADAM or AMSGRAD optimizers:

$$\begin{cases}\max_\theta \mathcal{L}_G(\varphi,\theta)\\ \max_\varphi \mathcal{L}_D(\varphi,\theta)\end{cases}$$

§ Distribution Learning

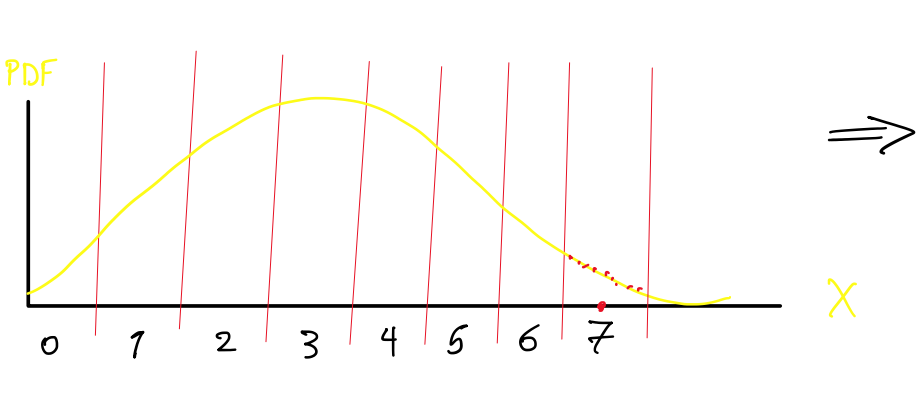
- A classical discriminator and a quantum generator.
- The generators' task is to transform an n-qubit input state to an n-qubit output state :

$$G_\theta|\Psi_{in}\rangle = |g_\theta\rangle = \sum_{j=0}^{2^n-1}\sqrt{P_\theta^j}\,|j\rangle$$

⤷ basis states.

- The distribution loading schemes' resolution, is determined by the number of qubits :

For n=3 :

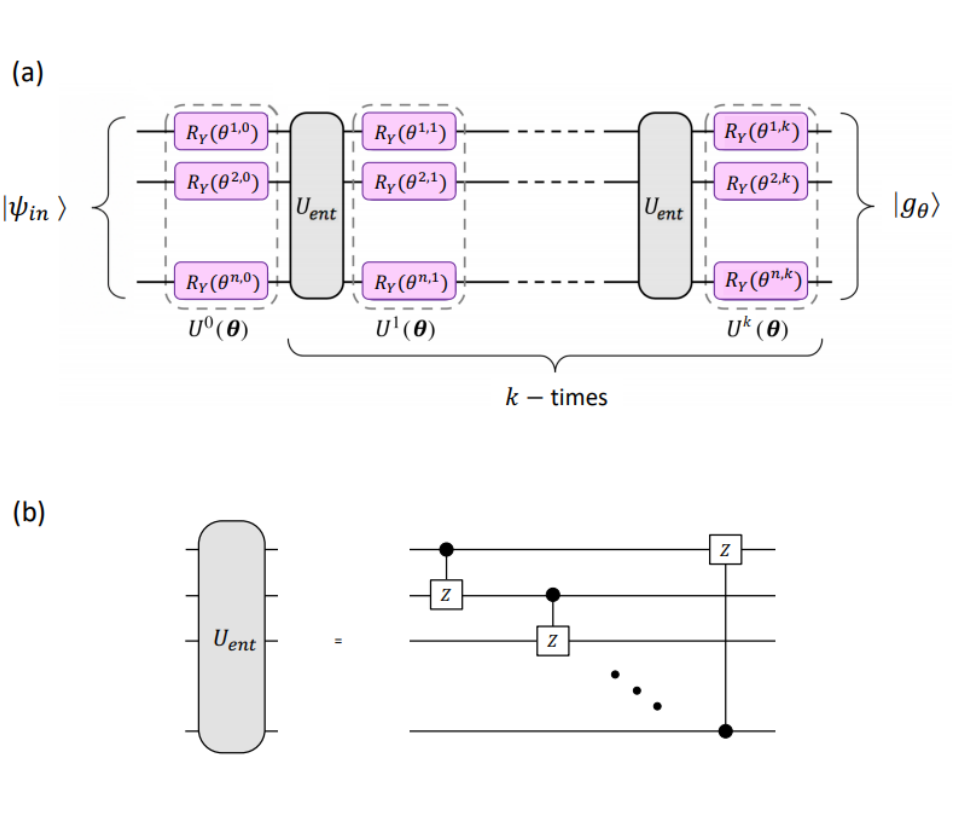


⟹

The quantum generator as a parametrized quantum circuit:

$$R_Y(\theta) = \begin{pmatrix}\cos\theta/2 & -\sin\theta/2\\ \sin\theta/2 & \cos\theta/2\end{pmatrix}$$

$$CZ = \begin{pmatrix}1 & & & 0\\ & 1 & & \\ 0 & & 1 & \\ & & & -1\end{pmatrix}$$



Total number of gates :

$(k+1)\,n$ single-qubit gates

kn two-qubit "

- The rational behind using R_Y and CZ is that at $\theta^{i,j} = 0$, these gates do not touch the state amplitudes, but only flip the phases. (remember modeled distributions just depend on state amplitudes).
- Then for the outcome of generator ($|g_\theta\rangle$) we have :

$$|g_\theta\rangle = G_\theta|\Psi_{in}\rangle = \prod_{p=1}^{k}\left[\bigotimes_{q=1}^{n}\left(R_Y(\theta^{p,q})\right)U_{ent}\right]\bigotimes_{q=1}^{n}\left(R_Y(\theta^{q,0})\right)|\Psi_{in}\rangle$$

$$= \sum_{j=0}^{2^n-1}\sqrt{P_\theta^j}\,|j\rangle$$

- The state $|g_\theta\rangle$ is measured m times to obtain data samples $g^l, l \in \{1, \ldots, m\}$, from which the P_θ^j can be calculated.

- The generator loss function can be written as :

$$\mathcal{L}_G(\varphi,\theta) = -\frac{1}{m}\sum_{l=1}^{m}\log\left(D_\varphi(g^l)\right) = -\sum_{j=0}^{2^n-1}P_\theta^j\log\left(D_\varphi(g^j)\right)$$

with $P_\theta^j = \left|\langle j|g_\theta\rangle\right|^2$

- Derivative of loss function for gradient based optimization can be evaluated as :

$$\frac{\partial\mathcal{L}_G(\varphi,\theta)}{\partial\theta^{i,l}} = -\sum_{j=1}^{m}\frac{\partial P_\theta^j}{\partial\theta^{i,l}}\log\left(D_\varphi(g^j)\right)$$

where $\frac{\partial P_\theta^j}{\partial\theta^{i,l}} = \frac{1}{2}\left(P^j_{\theta_+^{i,l}} - P^j_{\theta_-^{i,l}}\right)$ & $\theta_\pm^{i,l} = \theta^{i,l} \pm \frac{\pi}{2}e_{i,l}$

(i,l)- unit vector of respective parameter space.

§ Statistical Measures

Consider two PDFs P, Q. We can define measures to find the distance of these two PDFs.

① Kolmogorov-Smirnov : $D_{KS}(P\|Q) = \sup_{x\in X}\left|P(X \leq x) - Q(X \leq x)\right|$

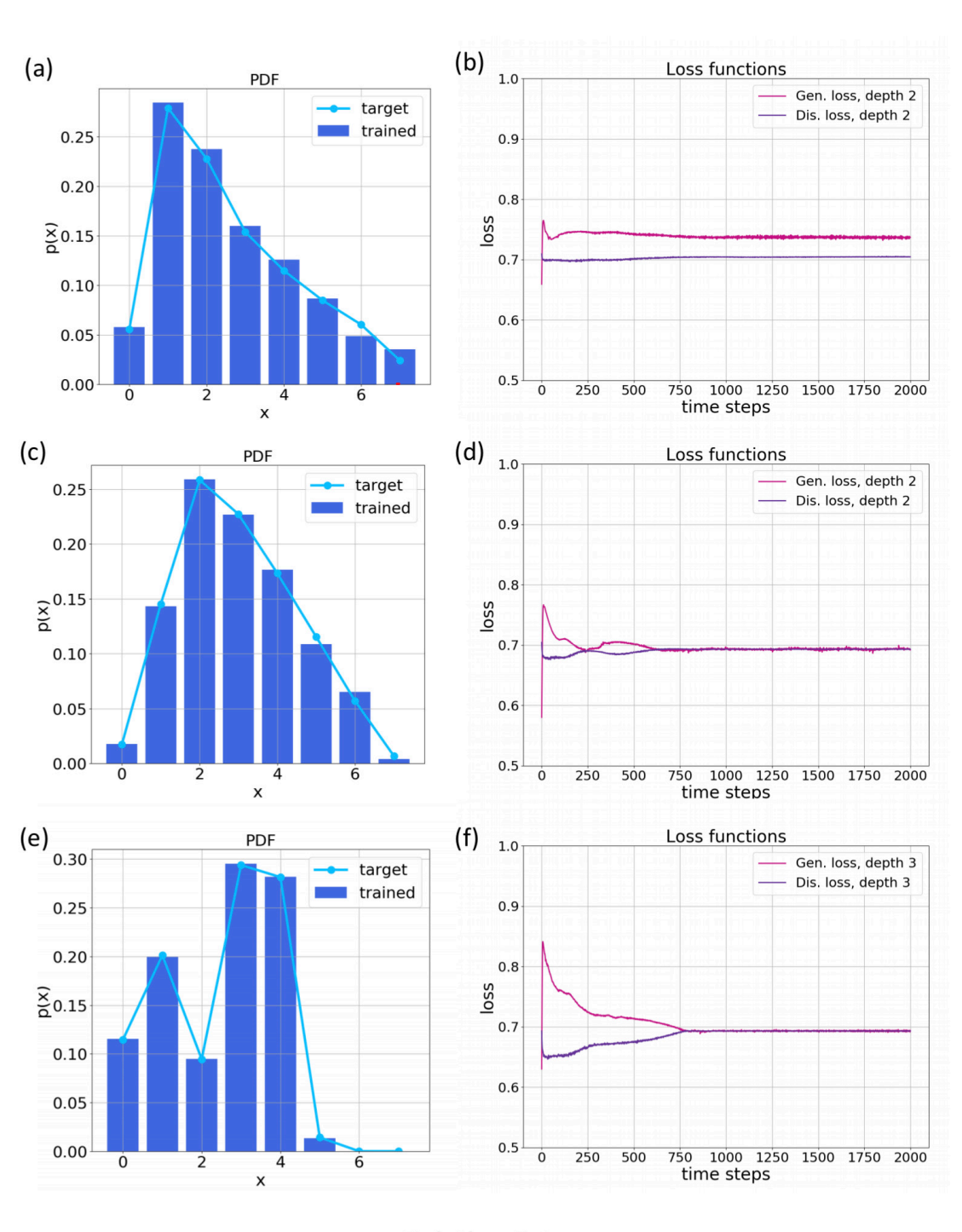
The null-hypothesis is accepted if $D_{KS}(P\|Q) \leq \sqrt{\frac{\ln 2/\alpha}{s}}$ → e.g. $= 0.05$

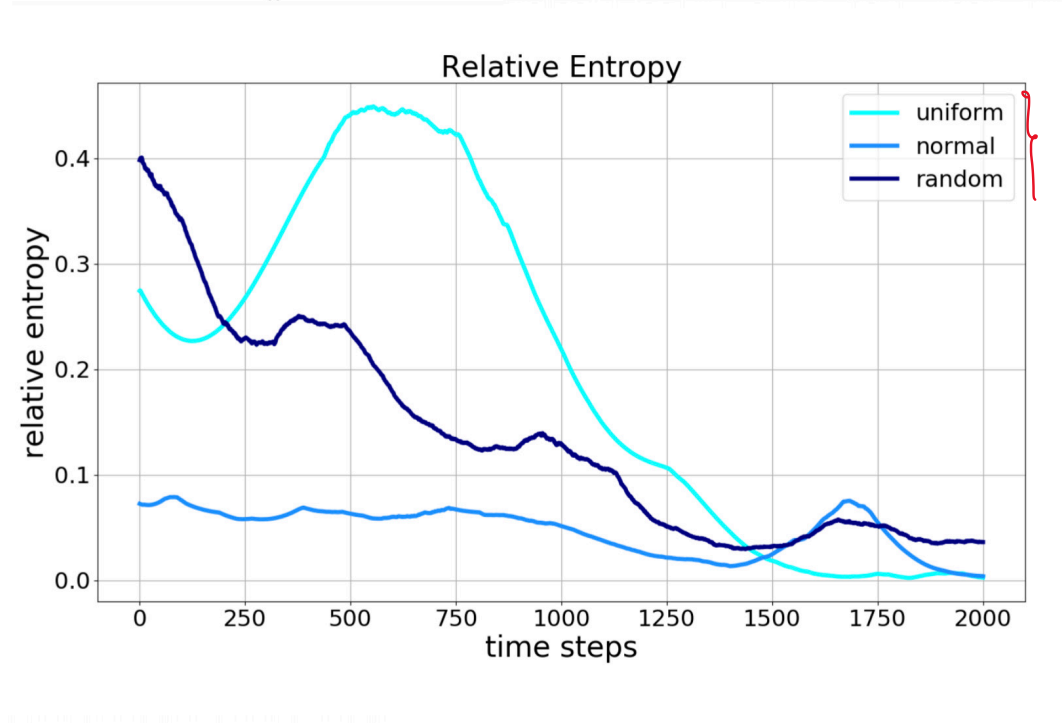
s: number of samples

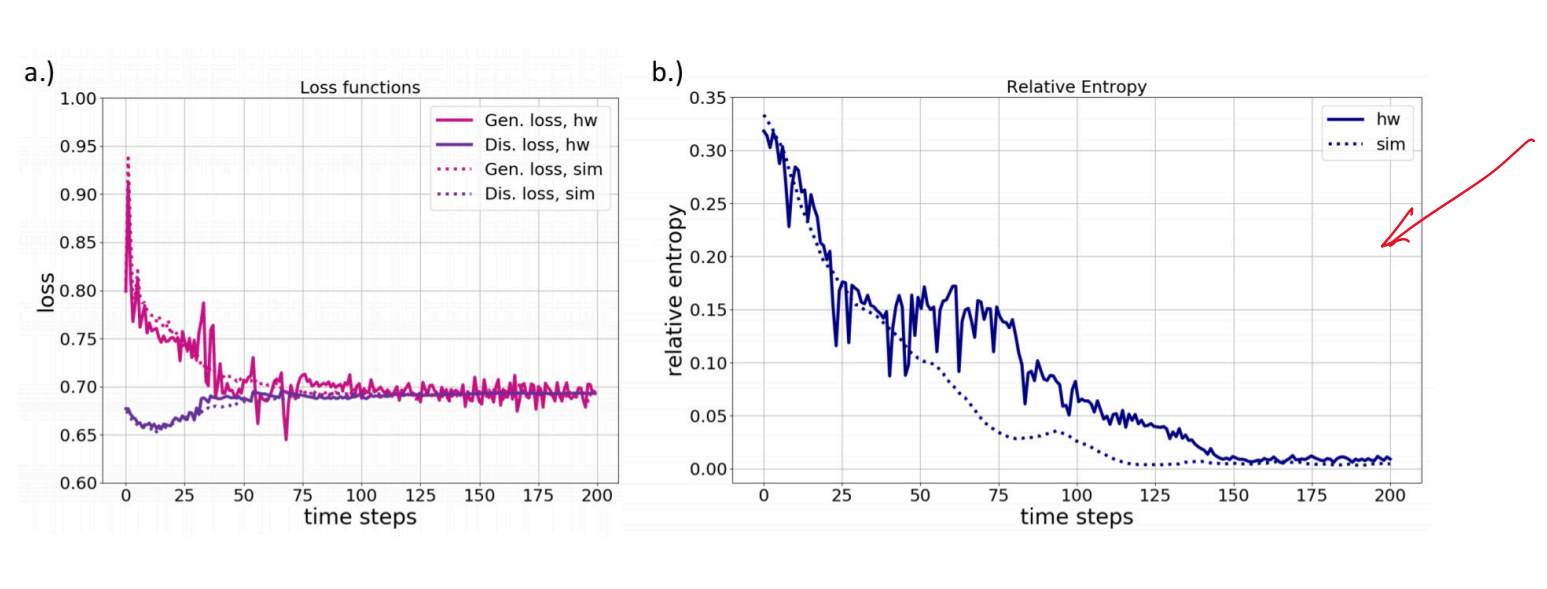
② Relative Entropy : $D_{RE}(P\|Q) = \sum_{x\in X}P(x)\log\left(\frac{P(x)}{Q(x)}\right)$

$D_{RE} \leadsto 0$, $P \sim Q$

§ Results :







Aside:

§ Discriminator network architecture dynamical analysis :

Consider a network with one or more dynamical variables $x_i, y_i, \ldots$ on each node i. These variables are coupled to eachother only along the edges of the network.

def. Adjacency matrix : $A_{ij} = \begin{cases}1 & \exists \text{ edge connecting nodes } i, j\\ 0 & \text{otherwise.}\end{cases}$

- edges may have weights ⤳ $A_{ij} = \begin{cases}W_{ij} & "\\ 0 & "\end{cases}$

def. Degree of nodes : $k_i = \#$ of edges connected to i.

$k_i = \sum_{j=1}^{n}A_{ij}$ n-nodes , total # edges = m

$$\Rightarrow 2m = \sum_{i=1}^{n}k_i = \sum_{i=1}^{n}\sum_{j=1}^{n}A_{ij}$$

def. Graph Laplacian (for simple, undirected network): n×n symmetric matrix $\mathcal{L}$:

$$\mathcal{L}_{ij} = \begin{cases}k_i & \text{if } i = j\\ -1 & \text{if } i \neq j \text{ \& } \exists \text{ edge connecting } i \text{ and } j\\ 0 & \text{otherwise}\end{cases}$$

or $\mathcal{L}_{ij} = k_i\delta_{ij} - A_{ij}$, For weighted networks $\mathcal{L}_{ij} = \sum_{j=1}^{n}A_{ij}\delta_{ij} - A_{ij}$

- In a network, we are interested to find the fixed point x^* :

$\underline{X} = \{x^i\}, i = 1, \ldots, n$, $\frac{d\underline{X}}{dt} = F(\underline{X}, t)$

- One can remove the time dependence in F :

$\frac{d\underline{X}}{dt} = F(\underline{X}, \underline{Y})$, $\frac{dY}{dt} = 1$ & $\underline{X}' = \underline{X} + \underline{Y}$

then at the fixed point $\underline{x}^*$: $F(\underline{x}^*) = 0$

- For a network with a single variable on each node :

$$\frac{dx_i}{dt} = \underbrace{f_i(x_i)}_{\text{intrinsic dynamics of a node}} + \sum_j A_{ij}\,g_{ij}(x_i, x_j)$$

⤷ describes the contribution from the connections.

(*) Assuming that all of the nodes obey a similar form of dynamics, we have :

$$\frac{dx_i}{dt} = f(x_i) + \sum_j A_{ij}\,g(x_i, x_j)$$

(*) For an undirected network, A_{ij} is symmetric, however we cannot assume that g is symmetric in its arguments : $g(u,v) \neq g(v,u)$

Linear Stability Analysis

Supposing that we are able to find a fixed point $\{x_i^*\}$ by simultaneously solving equations :

$$f(x_i^*) + \sum_j A_{ij}\,g(x_i^*, x_j^*) = 0 \quad , \forall i$$

Now, I will linearize about this fixed point by writing $x_i = x_i^* + \epsilon_i$ and performing multi-variable Taylor expansion and just keeping up to $O(\epsilon_i)$ terms:

$$\frac{dx_i}{dt} = \frac{d\epsilon_i}{dt} = f(x_i^* + \epsilon_i) + \sum_j A_{ij}\,g(x_i^* + \epsilon_i, x_j^* + \epsilon_j)$$

$$= f(x_i^*) + \epsilon_i\left.\frac{df}{dx}\right|_{x=x_i^*} + \sum_j A_{ij}\,g(x_i^*, x_j^*) + $$

$$+ \epsilon_i\sum_j A_{ij}\left.\frac{\partial g(u,v)}{\partial u}\right|_{u=x_i^*, v=x_j^*} +$$

$$+ \sum_j A_{ij}\,\epsilon_j\left.\frac{\partial g(u,v)}{\partial v}\right|_{u=x_i^*, v=x_j^*} + O(\epsilon^2)$$

$$\Rightarrow \frac{d\epsilon_i}{dt} = \epsilon_i\left.\frac{df}{dx}\right|_{x=x_i^*} + \epsilon_i\sum_j A_{ij}\left.\frac{\partial g(u,v)}{\partial u}\right|_{u=x_i^*, v=x_j^*} +$$

$$+ \sum_j A_{ij}\,\epsilon_j\left.\frac{\partial g(u,v)}{\partial v}\right|_{u=x_i^*, v=x_j^*}$$

- Derivatives evaluated at fixed points are simply numbers, therefore we define :

$$\alpha_i = \left.\frac{\partial f}{\partial x}\right|_{x=x_i^*} ,$$

$$\beta_{ij} = \left.\frac{\partial g(u,v)}{\partial u}\right|_{u=x_i^*, v=x_j^*} ,$$

$$\gamma_{ij} = \left.\frac{\partial g(u,v)}{\partial v}\right|_{u=x_i^*, v=x_j^*} .$$

$$\Rightarrow \frac{d\epsilon_i}{dt} = \left[\alpha_i + \sum_j\beta_{ij}A_{ij}\right]\epsilon_i + \sum_j\gamma_{ij}A_{ij}\epsilon_j .$$

which in the matrix form can be written as :

$$\frac{d\underline{\epsilon}}{dt} = M\underline{\epsilon} \quad \text{where } M_{ij} = \delta_{ij}\left[\alpha_i + \sum_k\beta_{ik}A_{ik}\right] + \gamma_{ij}A_{ij}$$

> This equation can be solved by writing $\underline{\epsilon}$ as a linear combination of evs, V_r of M :

$$\underline{\epsilon}(t) = \sum_r c_r(t)\,\underline{V}_r \Rightarrow \frac{d\underline{\epsilon}}{dt} = \sum_r\frac{dc_r}{dt}V_r = M\sum_r c_r(t)V_r$$

$= \sum_r \mu_r c_r(t)\,\underline{V}_r$ where μ_r is the eigen value corresponding to the ev V_r. Comparing terms in each ev :

$$\frac{dc_r}{dt} = \mu_r c_r(t) \longrightarrow c_r(t) = c_r(0)\,e^{\mu_r t}$$

(*) We see that, in order to have an attracting fixed point for the network, the real part of μ_r should be negative, then $c_r(t)$ – and hence $\underline{\epsilon}$ – is decaying in time for all r. Otherwise (a repelling fixed point or a saddle point) the network will never converge to a solution.

(*) The matrix M, for a given set of f & g, only depends on the architecture of the network (through adjacency matrix A_{ij})

Special Cases

① symmetric fixed point : $x_i^* = x^*$, $\forall i$

One can derive the stability condition for this type of fixed points to be :

$$\underset{\substack{\text{most negative eigenvalue}\\ \text{of } A_{ij} \text{ in } M \text{ evs}}}{\frac{1}{k_n}} < -\left[\frac{dg}{dx}\Big/\frac{df}{dx}\right]_{x=x^*} < \underset{\substack{\text{most positive eigenvalue}\\ \text{of } A_{ij} \text{ in } M \text{ evs}}}{\frac{1}{k_1}}$$

② if $g(x_i, x_j) = g(x_i) - g(x_j)$ and considering the fact that the smallest eigenvalue of Laplacian matrix is always zero, we have a master stability condition for network :

$$\underset{\substack{\text{largest eigenvalue of}\\ \text{Laplacian in } M \text{ evs}}}{\frac{1}{\lambda_n}} > -\left[\frac{dg}{dx}\Big/\frac{df}{dx}\right]_{x=x^*}$$