

Machine learning under physical constraints

Technical details of DAN

Sixin Zhang
(sixin.zhang@toulouse-inp.fr)

Outline

Model design of DAN

Pytorch implementation of DAN

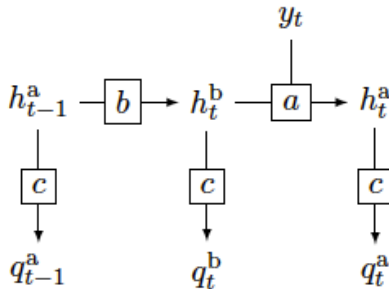
Outline

Model design of DAN

Pytorch implementation of DAN

DAN: model design in detail

- ▶ How to parameterize \mathbf{a} , \mathbf{b} , \mathbf{c} so that $p_t^{\mathbf{a}} \approx q_t^{\mathbf{a}}$ and $p_t^{\mathbf{b}} \approx q_t^{\mathbf{b}}$?
- ▶ What is the training and test procedure?



Design of procoder c : Gaussian model

- ▶ Let $\{c(h)\}_{h \in \mathbb{H}}$ represent a family of Gaussian distributions.
- ▶ How to parameterize a Gaussian distribution

$$\mathcal{N}(\mu, \Sigma)?$$

- ▶ $x \in \mathbb{R}^n \Rightarrow \mu \in \mathbb{R}^n$.
- ▶ Σ is a positive-definite matrix on $\mathbb{R}^{n \times n}$.
- ▶ **Challenge:** How to parameterize Σ ?

Design of procoder c: parameterize Σ

- ▶ Cholesky representation of covariance matrices by **lower-triangular** matrix Λ

$$\mathcal{N}(\mu, \Lambda \Lambda^\top).$$

- ▶ If $\det(\Lambda) > 0$, then $\Lambda \Lambda^\top$ is positive definite.
- ▶ If Λ has **strictly positive diagonal** elements, then Λ is invertible. Denote $\mu = (v_0, \dots, v_{n-1})^\top$, and

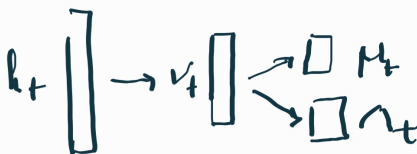
$$\Lambda = \begin{pmatrix} e^{v_n} & 0 & \dots & 0 \\ v_{2n} & e^{v_{n+1}} & \dots & 0 \\ \dots & \dots & \dots & 0 \\ v_{n+\frac{n(n+1)}{2}-1} & \dots & v_{3n-2} & e^{v_{2n-1}} \end{pmatrix}$$

Design of procoder \mathbf{c} : One-layer model

- ▶ How to build $(\mu_t, \Lambda_t) = \mathbf{c}(h_t)$ for $h_t \in \mathbb{H}$?
- ▶ Assume $\mathbb{H} \in \mathbb{R}^{mn}$ (m as ensemble size, n dim. of x).
- ▶ Associate (μ, Λ) with a vector
 $\mathbf{v} = (v_0, \dots, v_{n+\frac{n(n+1)}{2}-1}) \in \mathbb{R}^{n+n(n+1)/2}$.
- ▶ \Rightarrow use one linear layer with **parameter** $\theta^c = (W, b)$

$$(\mu_t, \Lambda_t) := \mathbf{v}_t = \mathbf{c}(h_t) = Wh_t + b.$$

Procoder \mathbf{c}

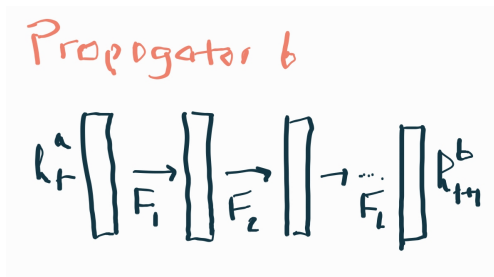


Design of propagator **b**: need non-linearity

- ▶ How to build $h_{t+1}^b = \mathbf{b}(h_t^a)$?
- ▶ Why using **non-linear** multiple-layers of NN?

$$h_{t+1}^b = F_L \circ F_{L-1} \circ \cdots \circ F_1(h_t^a).$$

- ▶ Problem of gradient vanishing/explosion when L is big



Design of propagator **b**: Residual networks

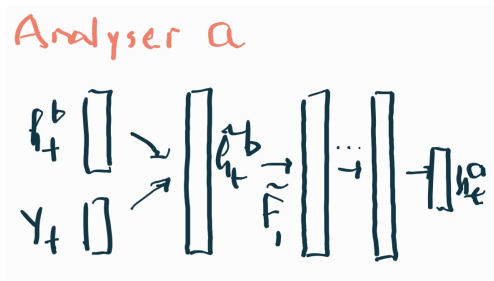
- ▶ For large L , we use residual networks with depth L to represent **b**.
- ▶ The layer ℓ is a mapping of $\mathbb{H} \rightarrow \mathbb{H}$, with parameter $(W_\ell, b_\ell, \alpha_\ell)$:

$$h \mapsto F_\ell(h) = h + \alpha_\ell \rho(W_\ell h + b_\ell).$$

- ▶ The non-linearity ρ is element-wise and it can be chosen to be tanh or (leaky-)relu.
- ▶ To be close to the identity mapping, initialize $\alpha_\ell = 0$.
- ▶ The **parameter** of **b** is $\theta^{\mathbf{b}} = (W_\ell, b_\ell, \alpha_\ell)_{\ell \leq L}$.

Design of analyser **a**: Augmented residual networks

- ▶ How to build $h_t^a = a(h_t^b, y_t)$?
- ▶ Consider an augmented input: $\tilde{h}_t^b = (h_t^b, y_t) \in \mathbb{H} \times \mathbb{R}^d$.
- ▶ Use a similar residual networks with depth L to transform \tilde{h}_t^b .
- ▶ Need an extra layer to map from $\mathbb{H} \times \mathbb{R}^d$ to \mathbb{H} .



Design of analyser **a**: Augmented residual networks

- ▶ For layer $\ell \leq \tilde{L}$, with parameter $(\tilde{W}_\ell, \tilde{b}_\ell, \tilde{\alpha}_\ell)$, is a mapping of $\mathbb{H} \times \mathbb{R}^d \rightarrow \mathbb{H} \times \mathbb{R}^d$:

$$\tilde{h} \mapsto \tilde{F}_\ell(\tilde{h}) = \tilde{h} + \tilde{\alpha}_\ell \rho(\tilde{W}_\ell \tilde{h} + \tilde{b}_\ell).$$

- ▶ As in **c**, define the extra layer to be a linear mapping from $\mathbb{H} \times \mathbb{R}^d \rightarrow \mathbb{H}$ with parameters (\tilde{W}, \tilde{b}) .
- ▶ The **parameter** of **a** is $\theta^{\mathbf{a}} = (\tilde{W}_\ell, \tilde{b}_\ell, \tilde{\alpha}_\ell, \tilde{W}, \tilde{b})_{\ell \leq L}$.

The objective $L_0(q_0^a)$

- ▶ Since q_0^a is Gaussian, we **evaluate the negative log-likelihood of Gaussian distributions**.
- ▶ Let $(\mu_0^a, \Lambda_0^a) := c(h_0^a) = Wh_0^a + b$, then

$$L_0(q_0^a) = \int \frac{(x_0 - \mu_0^a)^\top (\Lambda_0^a (\Lambda_0^a)^\top)^{-1} (x_0 - \mu_0^a)}{2} p(x_0) dx_0 \\ + \log |2\pi \Lambda_0^a (\Lambda_0^a)^\top|^{1/2}.$$

- ▶ As Λ_0^a is a triangular matrix, it is easy to compute $|\Lambda_0^a|$.
- ▶ How about $(\Lambda_0^a)^{-1}$? e.g. given $(a, b, c) \in \mathbb{R}^3$,

$$\Lambda = \begin{pmatrix} 1 & 0 & 0 \\ a & 1 & 0 \\ b & c & 1 \end{pmatrix}, \quad \Lambda^{-1} = \begin{pmatrix} 1 & 0 & 0 \\ -a & 1 & 0 \\ ac - b & -c & 1 \end{pmatrix}$$

Training procedure: supervised learning

- ▶ Optimize the **training objective** from I training sequences $\{(x_t(i), y_t(i))\}_{i \leq I, t \leq T}$,

$$\mathcal{L}_{train} = \frac{1}{T} \sum_{t \leq T} (\mathcal{L}_t(q_t^b) + \mathcal{L}_t(q_t^a)) + \mathcal{L}_0(q_0^a)$$

- ▶ Each term in \mathcal{L}_{train} is a Monte-Carlo estimation of $L_t(q_t^b)$, $L_t(q_t^a)$ or $L_0(q_0^a)$.
- ▶ For example, $L_0(q_0^a)$ is estimated by

$$\mathcal{L}_0(q_0^a) = \frac{1}{I} \sum_{i \leq I} \frac{(x_0(i) - \mu_0^a(i))^T (\Lambda_0^a(i) (\Lambda_0^a(i))^T)^{-1} (x_0(i) - \mu_0^a(i))}{2} \\ + \log |2\pi \Lambda_0^a(i) (\Lambda_0^a(i))^T|^{1/2}.$$

Training procedure: full mode

- ▶ Step 1 Pre-training: Optimize \mathbf{c} from I training sequences at $t = 0$,

$$\min_{\theta^{\mathbf{c}}} \mathcal{L}_0(q_0^{\mathbf{a}})$$

- ▶ Step 2 Full-training: Optimize $\mathbf{a}, \mathbf{b}, \mathbf{c}$ from I training sequences $t = 1, \dots, T$,

$$\min_{\theta^{\mathbf{a}}, \theta^{\mathbf{b}}, \theta^{\mathbf{c}}} \frac{1}{T} \sum_{t \leq T} (\mathcal{L}_t(q_t^{\mathbf{b}}) + \mathcal{L}_t(q_t^{\mathbf{a}})) + \mathcal{L}_0(q_0^{\mathbf{a}})$$

- ▶ Deterministic optimization: solved by GD, L-BFGS, etc.

Test procedure

- ▶ Goal: evaluate the trained model
- ▶ **Generalization (loss)**: use / **test sequences** to estimate how small the **objective function** is

$$\frac{1}{T} \sum_{t \leq T} (L_t(q_t^a) + L_t(q_t^b)) + L_0(q_0^a)$$

- ▶ **Generalization (error)**: for a **test trajectory** (x_t, y_t) of ODS, compute (μ_t^a, μ_t^b) for $t \leq T$, and then evaluate **RMSE**

$$\frac{1}{T} \sum_{t \leq T} \|x_t - \mu_t^a\|, \quad \frac{1}{T} \sum_{t \leq T} \|x_t - \mu_t^b\|$$

- ▶ **Prediction**: what happens for $t > T$?

Training procedure: online mode

- ▶ Minimization of $\mathcal{L}_t(q_t^{\mathbf{b}}) + \mathcal{L}_t(q_t^{\mathbf{a}})$ at each t , by truncated BPTT.
- ▶ Example, $\mathcal{L}_t(q_t^{\mathbf{a}}) = \frac{1}{I} \sum_i \mathcal{L}_t(q_t^{\mathbf{a}}(i))$, the truncated gradients $\tilde{\nabla} \mathcal{L}_t(q_t^{\mathbf{a}}(i))$ with respect to $(\theta^{\mathbf{a}}, \theta^{\mathbf{b}}, \theta^{\mathbf{c}})$ at time-step t are

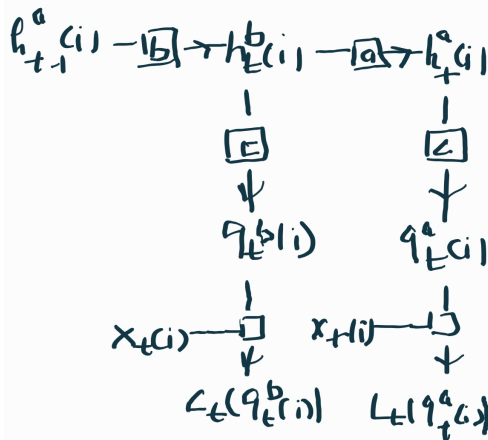
$$\tilde{\nabla}_{\theta^{\mathbf{c}}} \mathcal{L}_t(q_t^{\mathbf{a}}(i)) = \left(\frac{\partial q_t^{\mathbf{a}}(i)}{\partial \theta_t^{\mathbf{c}}} \right)^{\top} \nabla_{q_t^{\mathbf{a}}(i)} \mathcal{L}_t(q_t^{\mathbf{a}}(i))$$

$$\tilde{\nabla}_{\theta^{\mathbf{b}}} \mathcal{L}_t(q_t^{\mathbf{a}}(i)) = \left(\frac{\partial h_t^{\mathbf{b}}(i)}{\partial \theta_t^{\mathbf{b}}} \right)^{\top} \nabla_{h_t^{\mathbf{b}}(i)} \mathcal{L}_t(q_t^{\mathbf{a}}(i))$$

$$\tilde{\nabla}_{\theta^{\mathbf{a}}} \mathcal{L}_t(q_t^{\mathbf{a}}(i)) = \left(\frac{\partial h_t^{\mathbf{a}}(i)}{\partial \theta_t^{\mathbf{a}}} \right)^{\top} \nabla_{h_t^{\mathbf{a}}(i)} \mathcal{L}_t(q_t^{\mathbf{a}}(i))$$

Training procedure: online mode

- Computational graph (truncated) from i -th sample sequence $(x_s(i), y_s(i))_{s \leq t}$



Training procedure: online mode

- ▶ Let $\mathcal{L}_t(\theta) = \mathcal{L}_t(q_t^{\mathbf{b}}) + \mathcal{L}_t(q_t^{\mathbf{a}})$ with parameters $\theta = (\theta^{\mathbf{a}}, \theta^{\mathbf{b}}, \theta^{\mathbf{c}})$
- ▶ Online: update model parameters using truncated gradients.
- ▶ Truncated GD optimiser with learning rate $\eta_k > 0$:

$$\theta^{(k+1)} = \theta^{(k)} - \eta_k \tilde{\nabla}_{\theta} \mathcal{L}_{k+1}(\theta^{(k)})$$

- ▶ Truncated Adam optimiser: adaptive control of η_k :

$$\theta^{(k+1)} = \theta^{(k)} - \text{Adam}_k(\tilde{\nabla}_{\theta} \mathcal{L}_{k+1}(\theta^{(k)}))$$

Outline

Model design of DAN

Pytorch implementation of DAN

Data generation

- ▶ Propagation step: $x_t = Mx_{t-1} + \eta_t$
 - ▶ Linear 2d: Periodic (Hamiltonian) dynamics
 - ▶ Lorentz (non-linear) 40d: Chaotic dynamics
 - ▶ η_t white noise
- ▶ Observation step: $y_t = Hx_t + \epsilon_t$
 - ▶ Identity case: $H = I$
 - ▶ ϵ_t white noise

Forward steps of $(\mathbf{a}, \mathbf{b}, \mathbf{c})$ in DAN

- ▶ Compute the loss (and the gradient) over t .
- ▶ Initial $h_0^{\mathbf{a}}$
- ▶ Input: $h_{t-1}^{\mathbf{a}}, x_t, y_t$
- ▶ Output: $\mathcal{L}_t(q_t^{\mathbf{b}}) + \mathcal{L}_t(q_t^{\mathbf{a}}), h_t^{\mathbf{a}}$
- ▶ Key internal steps:
 - ▶ Compute $h_t^{\mathbf{b}} = \mathbf{b}(h_{t-1}^{\mathbf{a}})$
 - ▶ Compute $q_t^{\mathbf{b}} = \mathbf{c}(h_t^{\mathbf{b}})$
 - ▶ Compute $h_t^{\mathbf{a}} = \mathbf{a}(h_t^{\mathbf{b}}, y_t)$
 - ▶ Compute $q_t^{\mathbf{a}} = \mathbf{c}(h_t^{\mathbf{a}})$

Summary: Training

- ▶ **Input:** net, data, optimizer, dimensions, training time
- ▶ **Output:** a trained net
- ▶ Optimize and compute how fast the training loss decreases.
 - ▶ 2 modes: full vs. online
- ▶ Full mode
 - ▶ Generate training data: I sequences of $\{(x_t(i), y_t(i))\}_{i \leq I, t \leq T}$.
 - ▶ Optimize the total loss $\frac{1}{T} \sum_{t \leq T} (\mathcal{L}_t(q_t^b) + \mathcal{L}_t(q_t^a)) + \mathcal{L}_0(q_0^a)$.
- ▶ Online mode
 - ▶ At each $t \in \mathbb{Z}_+$, generate $\{(x_t(i), y_t(i))\}_{i \leq I}$ on the fly, and optimize the loss $\mathcal{L}_t(q_t^b) + \mathcal{L}_t(q_t^a)$ with truncated gradients.

Summary: Test

- ▶ **Input:** net, data, dimensions, test time
- ▶ **Output:** RMSE over $t \leq T$ (or $t > T$)
- ▶ Generate test data, e.g. I sequences of $\{(\tilde{x}_t(i), \tilde{y}_t(i))\}_{i \leq I, t \leq T}$ (independent of the training sequences).
- ▶ Use net to compute $\tilde{\mu}_t^a(i)$ from $\tilde{Y}_t(i)$, resp. $\tilde{\mu}_t^b(i)$ from $\tilde{Y}_{t-1}(i)$.
- ▶ Compute the RMSE based on $\tilde{x}_t(i)$,

$$\frac{1}{I} \sum_i \|\tilde{x}_t(i) - \tilde{\mu}_t^a(i)\|, \quad \frac{1}{I} \sum_i \|\tilde{x}_t(i) - \tilde{\mu}_t^b(i)\|$$