Genrative Adversarial Networks

GAN, WGAN and Sig-WGAN

Le-Rong Hsu

Aug. 2023

GAN AND WGAN

1 Generative Adversarial Networks . . . 4
- 1.1 Motivations . . . 4
- 1.2 Framework . . . 5
- 1.3 Loss Functions . . . 7
- 1.4 Training Process . . . 12
- 1.5 Problems . . . 18
- 1.6 More about GAN . . . 20

2 Wasserstein Generative Adversarial Networks . . . 22
- 2.1 Likelihood and KL divergence . . . 22
- 2.2 Motivation . . . 28
- 2.3 W_1 Distance . . . 29
- 2.4 WGAN . . . 33
- 2.5 Comparison . . . 37

Sig-WGAN

1 Sig-WGAN 40

1.1 Log-signature 40

1.2 Sig-W_1 Distance 44

1.3 Logsig-RNN 46

1.4 RNN and LSTM 52

2 References 58

Part I

Generative Models

Generative Adversarial Networks

Motivations

- Model-based reinforcement learning
- Simulate possible futures
- Generate good examples for training

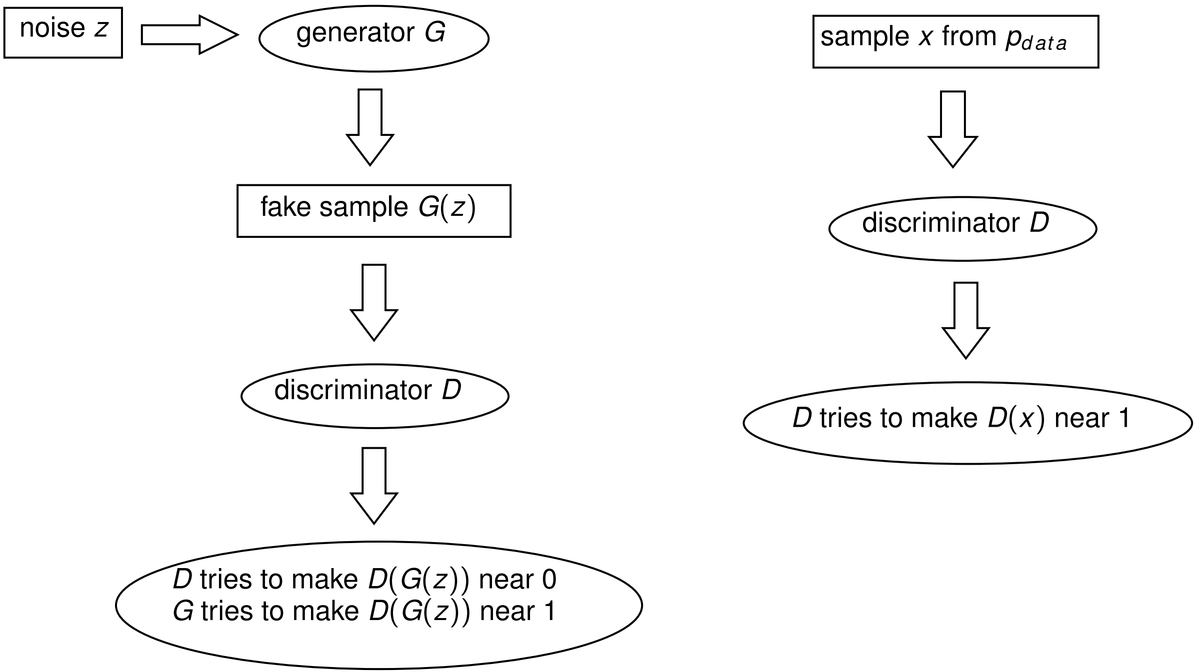
Generative Adversarial Networks

Framework

- **generator** : counterfeiter, trying to make fake money
  - $G : \mathcal{Z} \rightarrow \mathcal{X}$ inputs a latent vector (or noise) z and outputs a fake sample
  - $\mathcal{Z}$: latent space
  - z variable in $\mathcal{Z}$ with known distribution $\mathbb{P}_Z$
- **discriminator** : police, trying to allow legitimate money and catch counterfeit money
  - D: inputs a fake or real sample and outputs the probability that the sample is **real**
- **differentiable**
- p_{data}: data generating distribution
- G (and also D) is a neural network

Generative Adversarial Networks

Framework



Generative Adversarial Networks

Loss Functions

- The discriminator wishes to minimize $J^{(D)}(\theta^{(D),\theta^{(G)}})$.
- The generator wishes to minimize $J^{(G)}(\theta^{(D)}, \theta^{(G)})$.
- All of the different games designed for GANs so far use the same cost for the discriminator, $J^{(D)}$. They differ only in terms of the cost used for the generator, $J^{(G)}$. (Ian Goodfellow, 2016)

For discriminator:

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z} \log(1 - D(G(z)))$$

Generative Adversarial Networks

Loss Functions

$$J^{(D)}(\theta^{(D)}, \theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_{z} \log(1 - D(G(z)))$$

- $\log$: negative on $(0, 1]$, avoid too long precision
- $D(x)$: the probability that x is real
- $1 - D(G(z))$: the probability that D tells the fake sample $G(z)$
- We take all data into our consideration $\longrightarrow$ expectation
- If the probability is close to 1, then J is close to 0

Generative Adversarial Networks

Loss Functions

In the zero-sum game, the loss function of the generator could be

$$J^{(G)} = -J^{(D)}.$$

In this case, we can describe the entire game by a value function $V(\theta^{(D)}, \theta^{(G)})$, defined by

$$V\left(\theta^{(D)}, \theta^{(G)}\right) = -J^{(D)}(\theta^{(D)}, \theta^{(G)}).$$

The optimal parameters of G is thus

$$\theta^{(G)*} = \arg\min_{\theta^{(G)}} \max_{\theta^{(D)}} V\left(\theta^{(D)}, \theta^{(G)}\right)$$

What is the problem?

- Note that the protagonist of GAN is the generator.
- At the starting stage, the generator is poor. The discriminator can tell the fake samples easily.
- If the discriminator is able to tell the real data with high probability, that is, the loss $J^{(D)}$ is small, then the gradient of $J^{(G)} = -J^{(D)}$ vanishes.
- Cannot improve the generator

Generative Adversarial Networks

Loss Functions

Thus, we often choose

$$J^{(G)} = -\frac{1}{2}\mathbb{E}_z \log D(G(z))$$

where $D(G(z))$ is the probability that the discriminator believes that the fake sample is real. The generator aims to **maximize** this probability so that $-\log D(G(z))$ is small, i.e. minimize $J^{(G)}$.

More precisely, if the discriminator can tell with high confidence, then $D(G(z))$ is small. In other words, $-\log D(G(z))$ is large.

Algorithm General GAN training algorithm with alternating updates

Initialize $\theta^{(D)}, \theta^{(G)}$
for each training iteration **do**
    for K steps **do**
        Update the discriminator parameters $\theta^{(D)}$ using the gradient $\nabla J^{(D)}(\theta^{(D)}, \theta^{(G)})$
    end for
    Update the generator parameters $\theta^{(G)}$ using the gradient $\nabla J^{(G)}(\theta^{(D)}, \theta^{(G)})$
end for

How to calculate the gradient?

We want to update $\theta^{(D)}$ and $\theta^{(G)}$, so we take the gradient with respect to $\theta^{(D)}$ and $\theta^{(G)}$. The expectation is taken over all x or z.

Theorem 1 (Leibniz integral rule)

If $a(x)$ and $b(x)$ and $f(x,y)$ are C^1, then

$$\frac{d}{dx}\int_{a(x)}^{b(x)} f(x,y)dy = f(x,b(x))b'(x) - f(x,a(x))a'(x) + \int_{a(x)}^{b(x)} \frac{\partial f}{\partial x}(x,y)dy.$$

Special case: $a(x)$ and $b(x)$ are constant. Then

$$\frac{d}{dx}\int_{a}^{b} f(x,y)dy = \int_{a}^{b} \frac{\partial f}{\partial x}(x,y)dy.$$

How to calculate the gradient?
The above theorem also holds for random variable $y = X$ similarly.

Theorem 2

Let X be a random variable, $g\colon \mathbb{R} \times \mathcal{X} \longrightarrow \mathbb{R}$ a function such that $g(t, X)$ is integrable for all t and g is continuously differentiable w.r.t. t. Assume that there is a random variable Z such that $\frac{\partial}{\partial t} g(t, X) \leq Z$ a.s. for all t and $\mathbb{E}Z < \infty$. Then

$$\frac{\partial}{\partial t}\mathbb{E}[g(t, X)] = \mathbb{E}[\frac{\partial}{\partial t}g(t, X)]$$

- You can replace t by parameters θ
- There's a more general theorem with similar form of the conditions, but we're going to mention it.

Generative Adversarial Networks

Training Process

Thus, we have

$$\begin{aligned}
\nabla J^{(D)}(\theta^{(D)}, \theta^{(G)}) &= \nabla \left(-\frac{1}{2}\mathbb{E}_{x \sim p_{data}} \log D(x) - \frac{1}{2}\mathbb{E}_z \log(1 - D(G(z))) \right) \\
&= -\frac{1}{2}\mathbb{E}_{x \sim p_{data}} \nabla \log D(x) - \frac{1}{2}\mathbb{E}_z \nabla \log(1 - D(G(z)))
\end{aligned}$$

and

$$\begin{aligned}
\nabla J^{(G)} &= \nabla \left(-\frac{1}{2}\mathbb{E}_z \log D(G(z)) \right) \\
&= -\frac{1}{2}\mathbb{E}_z \nabla \log D(G(z))
\end{aligned}$$

$\nabla \log D(x)$ and $\nabla \log D(G(z))$ can be calculated with **backpropagation** , and each expectation can be estimated using **Monte Carlo estimation** .

Generative Adversarial Networks

Training Process

Algorithm GAN training algorithm

Initialize $\theta^{(D)}, \theta^{(G)}$
for each training iteration **do**
  for K steps **do**
    Sample batch of m noise vectors z_i
    Sample batch of m examples x_i
    Update the discriminator by performing stochastic gradient descent using
    $\nabla_{\theta^{(D)}} \frac{1}{m} \sum_{i=1}^{m} \left[-\frac{1}{2} \log D(x_i) - \frac{1}{2} \log(1 - D(G(z_i))) \right]$
  end for
  Sample batch of m noise vectors z_i
  Update the generator by performing stochastic gradient descent using
  $\nabla_{\theta^{(G)}} \frac{1}{m} \sum_{i=1}^{m} \left[-\frac{1}{2} \log(D(G(z_i))) \right]$
end for

Generative Adversarial Networks

Training Process

In practice, there are better choices of the training algorithm

- Adam gradient descent
- Simultaneous gradient descent: one step for each player (Ian Goodfellow (2016) thinks this is the best one)

Generative Adversarial Networks

Problems

There are some theoretical and practical problems of training GAN:

- Non-convergence:
  - GANs require finding the equilibrium to a game with two players.
  - In practice, GANs often seem to oscillate. The equilibrium is like the saddle point.
  - There is neither a theoretical argument that GAN games should converge when the updates are made to parameters of deep neural networks, nor a theoretical argument that the games should not converge.
- Balance D and G

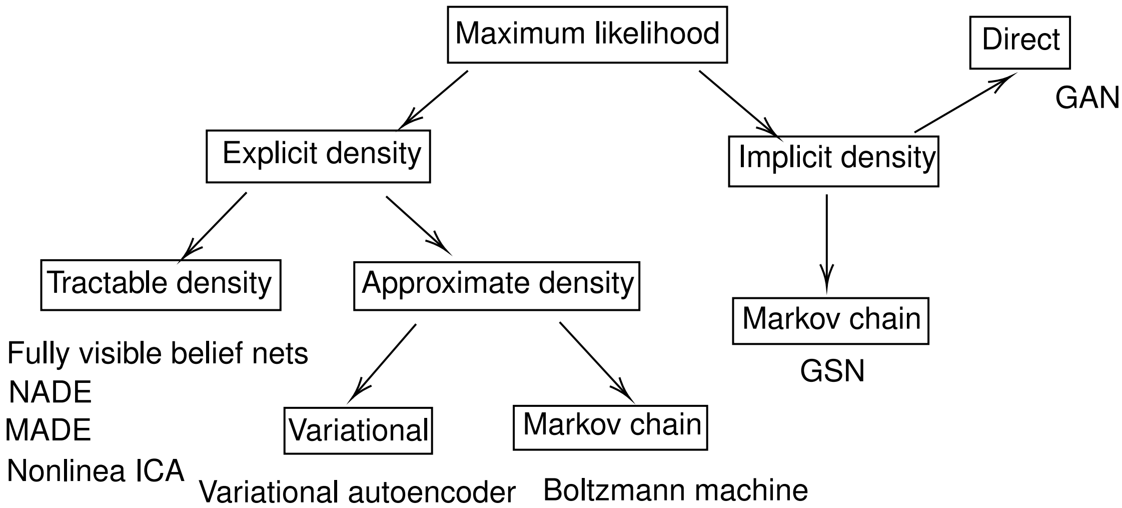
Generative Adversarial Networks

Problems

- Mode collapse
  - The generator learns to map several different input z values to the same output point.
  - Lack diversity
  - Mode collapse does not seem to be caused by any particular cost function
  - Limited to problems where it is acceptable for the model to produce a small number of distinct outputs

Generative Adversarial Networks

More about GAN



Generative Adversarial Networks

More About GAN

- Boltzmann machines: few probability distributions admit tractable Markov chain sampling
- Nonlinear independent component analysis: generator must be invertible; z must have the same dimension as the samples x
  - GAN admits z with larger dimension
- No Markov chains are needed
  - slow convergence, inefficient in high-dimensional spaces
- No variational bound is needed
  - Good likelihood but bad samples
- GANs are subjectively regarded as producing better samples than other methods.

Wasserstein Generative Adversarial Networks

Likelihood and KL divergence

For generative models,

- maximize the **likelihood** of our training data, that is, to generate training data as possible
- minimize the **distance** of the two distributions

What is likelihood?

- 4 patients among 10 people
- infected patients $X \sim Bin(10, \theta)$
- Patients are independent

$$\mathbb{P}(X = 4|\theta) = \binom{10}{4}\theta^4(1-\theta)^{10-4}$$

What is the θ such that $\mathbb{P}(X = 4|\theta)$ is largest?

WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS

LIKELIHOOD

- **Likelihood** : $L(\theta|X=4) = \mathbb{P}(X=4|\theta)$
  - a function of θ, determining the probability of the observation $X=4$
  - $\mathbb{P}(X=4|\theta)$: given θ, the probability of 4 infected patients
- **Maximum likelihood** : $\max_\theta L(\theta|X=4)$

In this problem, the optimal parameter θ is $\arg\max_\theta L(\theta|X=4) = 4$.

Wasserstein Generative Adversarial Networks

Likelihood in Generative Models

- training data: x^i, $i = 1, ..., m$
- likelihood: the probability that the model assigns to the training data $\Pi_{i=1}^m p_{model}(x^i; \theta)$
- take log

$$\begin{aligned} \theta^* &= \arg\max_\theta \Pi_{i=1}^m p_{model}(x^i; \theta) \\ &= \arg\max_\theta \Sigma_{i=1}^m \log p_{model}(x^i; \theta) \end{aligned}$$

Intuitively, the purpose of KL divergence is to measure the **difference** or " **distance** " of two probability distributions.

Definition 2.1 (KL divergence)

Given two distributions $p(x)$ and $q(x)$. The discrete KL divergence is defined by

$$KL(p||q) \doteq \sum_{k=1}^{m} \log \frac{p_k}{q_k},$$

and the continuous KL divergence is

$$\begin{aligned} KL(p||q) &\doteq \int_{-\infty}^{\infty} p(x) \log \frac{p(x)}{q(x)} d\mu(x) \\ &= \mathbb{E}_{x \sim p(x)} \log \frac{p(x)}{q(x)} \end{aligned}$$

Wasserstein Generative Adversarial Networks

Likelihood and KL divergence

Proposition 1

Minimizing KL divergence is equivalent to maximizing likelihood.

Proof.

Let θ^* be the optimal parameters and $\hat{\theta}$ be the approximate parameters.

$$
\begin{aligned}
&\arg\min_{\hat{\theta}} KL(p(x|\theta^*)||p(x|\hat{\theta})) \\
&= \arg\min_{\hat{\theta}} \mathbb{E}_{x\sim p(x|\theta^*)}\left[\log \frac{p(x|\theta^*)}{p(x|\hat{\theta})}\right] \\
&= \arg\min_{\hat{\theta}} \mathbb{E}_{x\sim p(x|\theta^*)}\left[\log p(x|\theta^*) - \log p(x|\hat{\theta})\right] \\
&= \arg\min_{\hat{\theta}} \mathbb{E}_{x\sim p(x|\theta^*)}\left[- \log p(x|\hat{\theta})\right] \\
&= \arg\max_{\hat{\theta}} \mathbb{E}_{x\sim p(x|\theta^*)}\left[\log p(x|\hat{\theta})\right]
\end{aligned}
$$

Why does the choice of distance matter?

Theorem 3

A sequence of distributions $\mathbb{P}_t$ converges with respect to ρ if and only if there exists a distribution $\mathbb{P}_\infty$ such that $\rho(\mathbb{P}_t, \mathbb{P}_\infty) \longrightarrow 0$ as $t \rightarrow 0$.

- In order to optimize the parameter θ, we hope the distance of distributions is **continuous** , that is, if θ converges to θ^*, then $\mathbb{P}_\theta$ converges to $\mathbb{P}_{\theta^*}$.
- Gradient descent on KL divergence? No, we'll give an counterexample later.

Wasserstein Generative Adversarial Networks

W_1 Distance

Let $\mathcal{X}$ be a compact metric set and let Σ denote the set of all the Borel subsets of $\mathcal{X}$.

- The *Total Variation* (TV) distance

$$\delta(\mathbb{P}_r, \mathbb{P}_g) = \sup_{A \in \Sigma} |\mathbb{P}_r(A) - \mathbb{P}_g(A)|$$

- The KL divergence

$$KL(\mathbb{P}_r || \mathbb{P}_g) = \int \log \frac{P_r(x)}{P_g(x)} P_r(x) d\mu(x)$$

  This is asymmetric.

- The *Jensen-Shannon* (JS) divergence

$$JS(\mathbb{P}_r, \mathbb{P}_g) = KL(\mathbb{P}_r || \mathbb{P}_m) + KL(\mathbb{P}_g || \mathbb{P}_m)$$

  where $\mathbb{P}_m = \frac{\mathbb{P}_g + \mathbb{P}_g}{2}$. This divergence is symmetrical and always defined because we can choose $\mu = \mathbb{P}_m$.

Wasserstein Generative Adversarial Networks

EM Distance

- The *Earth-Mover* (EM) distance or Wasserstein-1

$$W(\mathbb{P}_r, \mathbb{P}_g) = \inf_{\gamma \sim \Pi(\mathbb{P}_r, \mathbb{P}_g)} \mathbb{E}_{(x,y)\sim\gamma} \left[||x - y|| \right]$$

where $\Pi(\mathbb{P}_r, \mathbb{P}_g)$ denotes the set of all joint distributions $\gamma(x, y)$ whose marginals are respectively $\mathbb{P}_r$ and $\mathbb{P}_g$. Intuitively, $\gamma(x, y)$ indicates how much "mass" must be transported from x to y in order to transform the distributions $\mathbb{P}_r$ into the distribution $\mathbb{P}_g$. The EM distance then is the **"cost"** of the optimal transport plan.

Wasserstein Generative Adversarial Networks

EM Distance

Example

- $Z \sim U[0,1]$
- $\mathbb{P}_0 = (0, Z)$
- $g_\theta = (\theta, z), z \sim Z$

In this case,

- $$W(\mathbb{P}_0, \mathbb{P}_g) = |\theta|$$
- $$JS(\mathbb{P}_0, \mathbb{P}_g) = \begin{cases} \log 2 & \text{if } \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$
- $$KL(\mathbb{P}_0 || \mathbb{P}_g) = KL(\mathbb{P}_g || \mathbb{P}_0) \begin{cases} \infty & \text{if } \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$
- $$\delta(\mathbb{P}_0, \mathbb{P}_g) = \begin{cases} 1 & \text{if } \theta \neq 0 \\ 0 & \theta = 0 \end{cases}$$

- Among all distances, only Wasserstein-1 distance is continuous in this example.

Fix a distribution $\mathbb{P}_r$ and let $\mathbb{P}_\theta$ denote the distribution of $g_\theta(Z)$.

Proposition 2 (easily understood version)

For any feedforward neural network parametrized by θ, noise z sampled from a desired distribution (e.g. Gaussian), $W(\mathbb{P}_r, \mathbb{P}_\theta)$ is continuous everywhere and differentiable almost everywhere.

- desired distribution $\longrightarrow \mathbb{E}_{z \sim p(z)}[\|z\|] < \infty$

Wasserstein Generative Adversarial Networks

WGAN

By the Kantorovich-Rubinstein duality,

$$W(\mathbb{P}_r, \mathbb{P}_\theta) = \sup_{||f||_L \leq 1} \mathbb{E}_{x \sim \mathbb{P}_r}[f(x)] - \mathbb{E}_{x \sim \mathbb{P}_\theta}[f(x)].$$

If we replace $||f||_L \leq 1$ by $||f||_L \leq K$, then we end up with $K \cdot W(\mathbb{P}_r, \mathbb{P}_\theta)$. The gradient is scaled but its direction does not change.

WASSERSTEIN GENERATIVE ADVERSARIAL NETWORKS

WGAN

Let f_w denote the neural network with parameters w. We could consider solving

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim \mathbb{P}_r}[f_w(x)] - \mathbb{E}_{z \sim \mathbb{P}_z}[f_w(g_\theta(z))]$$

where we restrict the range of parameters. For example, we restrict $\mathcal{W} \in [-0.01, 0.01]^l$ so that $\frac{\partial f_w}{\partial w_i}$ are bounded and the gradient is Lipschitz.

Wasserstein Generative Adversarial Networks

WGAN

Algorithm WGAN. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{critic} = 5$.

Initialize α: learning rate, c: clipping parameter, m: batch size, n_{critic}: the number of iterations of the critic per generator iteration
Initialize w_0: discriminator parameters, θ_0: generator parameters
for each training iteration **do**
  for t=0,1,...,n_{critic} **do**
    Sample $\{x^{(i)}\}_{i=1}^{m} \sim \mathbb{P}_r$ a batch of real data
    Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples
    $g_w \leftarrow \nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
    $w \leftarrow w + \alpha \cdot \mathrm{RMSProp}(w, g_w)$
    $w \leftarrow \mathrm{clip}(w, -c, c)$
  end for
  Sample $\{z^{(i)}\}_{i=1}^{m} \sim p(z)$ a batch of prior samples
  $g_\theta \leftarrow -\nabla_\theta \left[\frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
  $\theta \leftarrow \theta + \alpha \cdot \mathrm{RMSProp}(w, g_\theta)$
end for

Wasserstein Generative Adversarial Networks

WGAN

- We clip the domain of parameters to enforce Lipchitz
- **We get rid of the sigmoid in the last layer since the discriminator of WGAN is not a logistic regression.** Instead, the discriminator is an approximate to W-1 distance, so this is an regression and we don't need sigmoid.
- No $\log$

- **In no experiment did we see evidence of mode collapse for the WGAN algorithm.**

Figure. Algorithms trained with a DCGAN generator. Left: WGAN algorithm. Right: standard GAN formulation. Both algorithms produce high quality samples.

Figure. Algorithms trained with a generator without batch normalization and constant number of filters at every layer. Standard GAN failed to learn while the WGAN still was able to produce samples.

Wasserstein Generative Adversarial Networks

WGAN

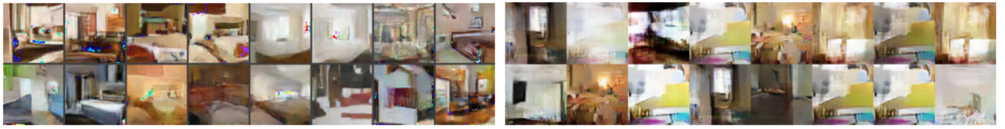


Figure. Algorithms trained with an MLP generator with 4 layers and 512 units with ReLU nonlinearities. WGAN: lower quality, GAN: worse quality and mode collapse

Part II

Signature and GAN

SIG-WGAN

LOG-SIGNATURE

Piecewise linear transformation

- time-joined transformation
- lead-lag transformation (uniquely determines the path)
- rectlinear interpolation

Definition 1.1 (Signature)

For a continuous path $X = (f_1(t), \ldots, f_d(t)) \colon [0,1] \longrightarrow \mathbb{R}^d$*, the signature of* X *is defined by an infinite sequence*

$$S(X) = \left(\int_0^1 \cdots \int_0^{t_{k-1}} dX_{t_1} \cdots dX_{t_k} \right)_{k \in \mathbb{N}} \in (\mathbb{R}^d)^{\otimes k}$$

SIG-WGAN

LOG-SIGNATURE

Definition 1.2 (Logarithm map)

Let $a = (a_0, a_1, \ldots) \in T((\mathbb{R}^d))$ be such that $a_0 = 1$, $t = a - 1$. Then the logarithm map denoted by log is defined as follows:

$$\log(a) = \log(1 + t) = \sum_{n=1}^{\infty} \frac{(-1)^{n-1}}{n} t^{\otimes n}$$

- a can be viewed as the signature of a path

Definition 1.3 (The Log Signature of a Path))

The log signature of path X by $\log S(X)$ is the logarithm of the signature of the path X, denoted by $LogSig(X)$.

- degree $M \longrightarrow LogSig^M(X)$

Sig-WGAN

Log-signature

- **Uniqueness** : the log-signature is bijective to the signature
- **Dimension reduction** : the dimension of the truncated log-signature is no greater than that of the truncated signature
- Invariance under time parameterization
- Missing Data and unequally time spacing

SIG-WGAN

SIG-W$_1$ DISTANCE

- $\mathcal{X} = \Omega_0^1(J, \mathbb{R}^d)$: all paths with 1-variation that maps a compact interval J to $\mathbb{R}^d$
- μ, ν: measures (probability distributions) on $\Omega_0^1(J, \mathbb{R}^d)$

The W-1 distance on the signature space is

$$W_1^{\text{Sig space}}(\mu, \nu) = \sup_{\|f\|\_\text{Lip} \leq 1} \mathbb{E}_{X\sim\mu}[f(S(X))] - \mathbb{E}_{X\sim\nu}[f(S(X))].$$

universality $\implies$

$$\text{Sig-W}_1^{\text{Sig space}}(\mu, \nu) = \sup_{L \text{ is linear}, \|L\| \leq 1} \mathbb{E}_{X\sim\mu}[L(S(X))] - \mathbb{E}_{X\sim\nu}[L(S(X))].$$

SIG-WGAN

SIG-W_1 DISTANCE

Factorial decay $\implies$

$$\text{Sig-W}_1^{(M)}(\mu, \nu) = \sup_{L \text{ is linear}, \|L\| \leq 1} L\left(\mathbb{E}_{X\sim\mu}[S^M(X)] - \mathbb{E}_{X\sim\nu}[S^M(X)]\right).$$

When the norm of L is chosen as the l_2 norm of the linear coefficients of L, this reduced optimization problem admits the analytic solution

$$\text{Sig-W}_1^{(M)}(\mu, \nu) := |\mathbb{E}_\mu[S^M(X)] - \mathbb{E}_\nu[S^M(X)]|$$

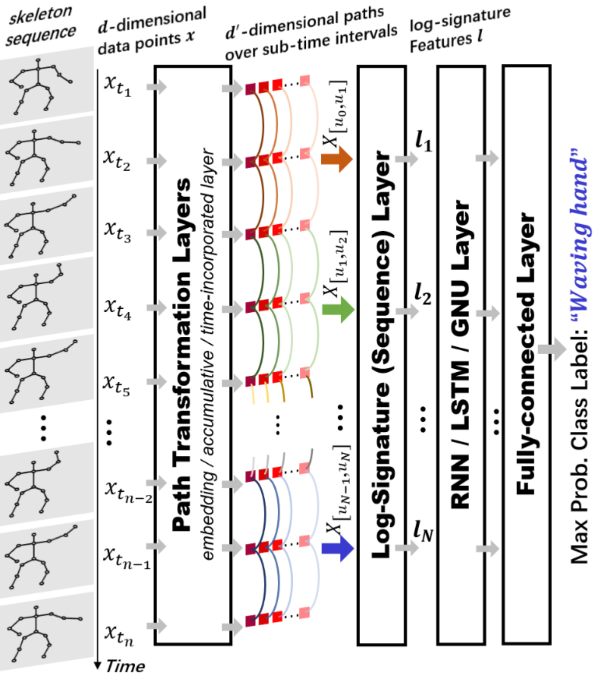
SIG-WGAN

LOGSIG-RNN

- time-series data $\longrightarrow$ path transformation $\longrightarrow$ log-signature transformation $\longrightarrow$ RNN/LSTM

SIG-WGAN

LOGSIG-RNN



SIG-WGAN

PATH TRANSFORMATION LAYER

- **Embedding Layer** : maps $(X_{t_i})_{i=1}^n$ to $(LX_{t_i})_{i=1}^n$, where $X_{t_i} \in \mathbb{R}^d$, $LX_{t_i} \in \mathbb{R}^{d'}$ with $d' < d$, L trainable matrix (dimension reduction)
- **Accumulative Layer** : $Y_i = \sum_{j=1}^{i} X_{t_j}$, $i = 1, ..., n$ (extract the quadratic variation and other higher order statistics of an input path X effectively)
- **Time-incorporated Layer** : $(t_i, X_{t_i})_{i=1}^n$

SIG-WGAN

LOG-SIGNATURE LAYER

Consider

- $x^{\hat{\mathcal{D}}} = (x_{t_i})_{i=1}^{n}$: discrete d-dimensional time series over some time interval J, $\hat{\mathcal{D}}$ partition of J
- linear interpolation
- $\mathcal{D} := (u_k)_{k=0}^{N} \subset \hat{\mathcal{D}}$: coarser partition of J

The Log-Signature Layer transforms an input $x^{\hat{\mathcal{D}}}$ to a sequence of the log signature of $x^{\hat{\mathcal{D}}}$ over a coarser time partition $\mathcal{D}$.

Sig-WGAN

Log-Signature Layer

$\mathbb{R}^{d \times n}$ $\qquad\qquad$ $\mathbb{R}^{d_{ls} \times N}$

$$x^{\hat{D}} \longrightarrow \boxed{\text{Log-Signature Layer}} \longrightarrow (\mathrm{LogSig}_k^M)_{k=0}^{N-1}, \mathrm{LogSig}_k = \mathrm{LogSig}^M(x)_{[u_k, u_{k+1}]}$$

- d_{ls}: dimension of truncated log-signature
- No weights
- $(d, n) \longrightarrow (d_{ls}, N)$ where $N \leq n$ and $d_{ls} \geq d$
  - shrinks time dimension by using the more informative spatial features of a higher dimension

Sig-WGAN

Backpropogation

By chain rule, the derivative of F is

$$\frac{\partial F(l_0, \ldots, l_{N-1})}{\partial x_{t_i}} = \sum_{k=0}^{N-1} \frac{\partial F(l_0, \ldots, l_{N-1})}{\partial l_k} \frac{\partial l_k}{\partial x_{t_i}}.$$

- If $t_i \notin [u_k, u_{k+1}]$, $\frac{\partial l_k}{\partial x_{t_i}} = 0$. Otherwise, $\frac{\partial l_k}{\partial x_{t_i}}$ is the derivative of the single log-signature l_k w.r.t. path $x_{u_k, u_{k+1}}$ where $t_i \in \mathcal{D} \cap [u_k, u_{k+1}]$.
- The log signature $LogSig(x^{\hat{\mathcal{D}}})$ with respect to x_{t_i} is proved differentiable.

SIG-WGAN

RNN

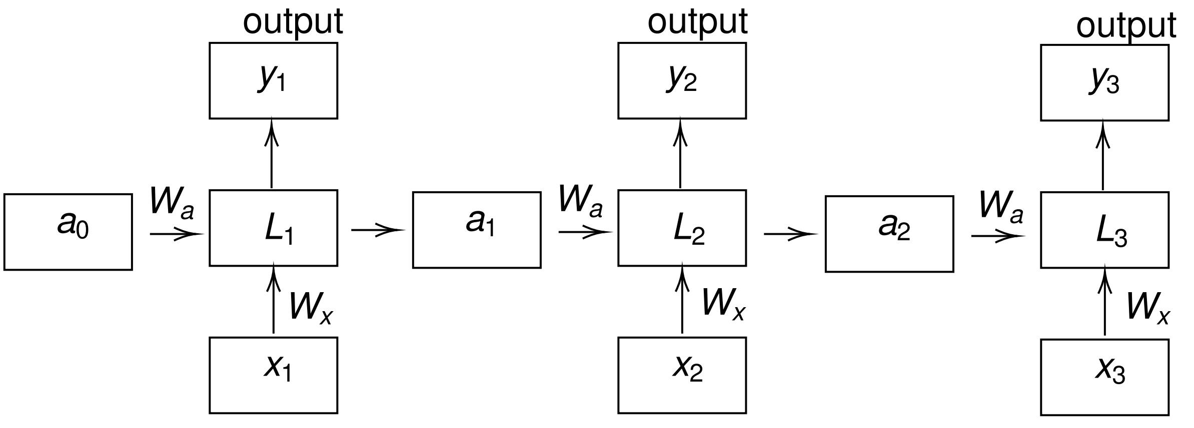
- The inspiration of Logsig-RNN comes from the numerical approximation of SDE, which is of the form

$$a_{n+1} = f_\theta(a_n, x_n).$$

- Backpropagation
- Gradient vanishing/exploding

SIG-WGAN

RNN



- $a_1 = \sigma(W_a \cdot a_0 + W_x \cdot x_1 + b_1)$
- $a_2 = \sigma(W_a \cdot a_1 + W_x \cdot x_2 + b_2)$
- $\vdots$

SIG-WGAN

RNN

- The same weight matrices are multiplied many times in the hidden layers.
- The gradient decays/cumulates

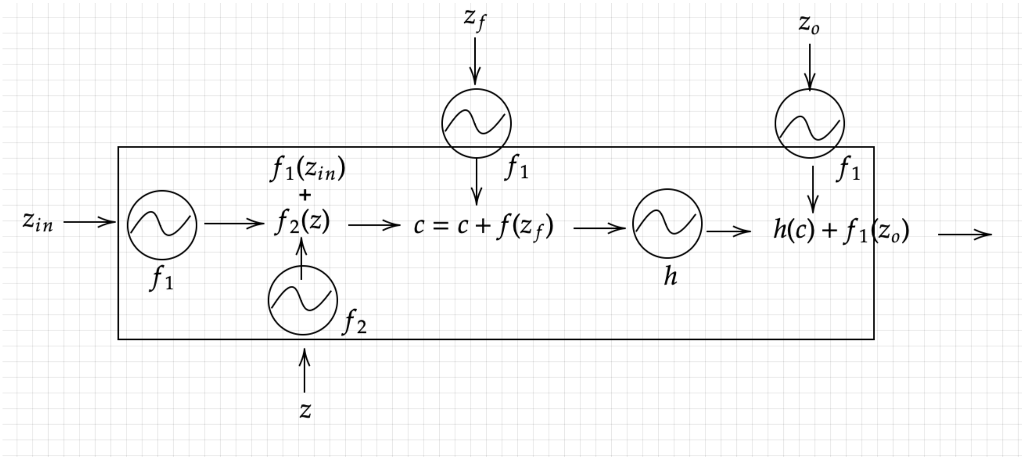
SIG-WGAN

LSTM

- Long Short-term memory (LSTM) is a unit ($\approx$ neuron).
- Use gates to control input

SIG-WGAN

LSTM



- sigmoid: between 0 and 1, the degree of opening

Sig-WGAN

Logsig-RNN vs. LSTM

- few time steps $\longrightarrow$ LSTM is better
- high frequency $\longrightarrow$ Logsig-RNN is better

References

- Probabilistic Machine Learning: Advanced Topics, MIT Press, 2023.
- NIPS 2016 Tutorial: Generative Adversarial Networks. arXiv:1701.00160
- Wasserstein GAN. arXiv:1701.07875
- Learning from the past, predicting the statistics for the future, learning an evolving system. arXiv:1309.0260
- Sig-Wasserstein GANs for Time Series Generation. arXiv:2111.01207
- Learning stochastic differential equations using RNN with log signature features. arXiv:1908.08286
- zhihu