Modèles génératifs profonds

Guillaume Bourmaud

PLAN

I) Introduction

I)

Principe d'un modèle génératif profond

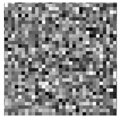
Base de données **non-étiquetées** : $\{\mathtt{X}_i\}_{i=1\ldots N}$

"est un échantillon de"

Point de vue probabiliste : $\mathtt{X}_i \sim p_{\text{data}}(\mathtt{X})$

Inconnue

Objectif : Optimiser les paramètres d'un réseau de neurones profond afin de générer de **nouveaux échantillons** de $p_{\text{data}}(\mathtt{X})$

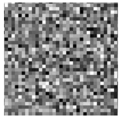


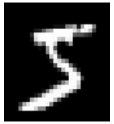
$$\mathtt{Z}_{\text{ech}} \sim \mathcal{N}(0;\mathtt{I}) \longrightarrow \boxed{f} \xrightarrow{} \mathtt{X}_{\text{ech}} \sim p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$$

$\boldsymbol{\theta}$

Difficulté de la tâche d'apprentissage

Base de données **non-étiquetées** : $\{\mathtt{X}_i\}_{i=1\ldots N}$



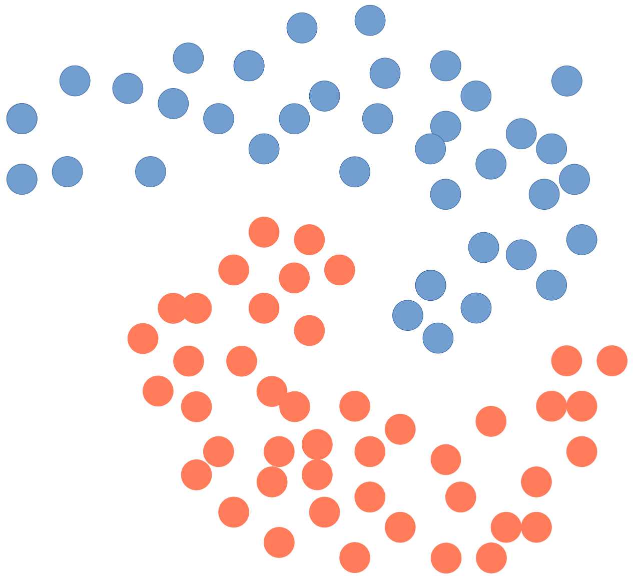


$$\mathtt{Z}_{\text{ech}} \sim \mathcal{N}(\mathtt{0};\mathtt{I}) \longrightarrow \boxed{f} \longrightarrow \mathtt{X}_{\text{ech}}, \qquad \boldsymbol{\theta} \to f$$

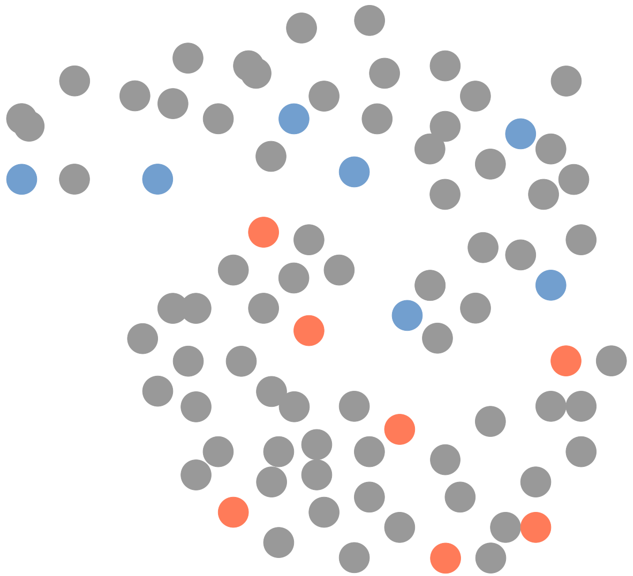
On ne dispose pas de couples $\{\mathtt{Z}_i, \mathtt{X}_i\}_{i=1\ldots N}$

→ problème d'apprentissage non-supervisé

Exemple d'utilisation : Apprentissage semi-supervisé



Supervisé



Semi-Supervisé

I)

Principe d’un modèle génératif profond conditionnel

Base de données **étiquetées** : $\{\mathtt{Y}_i, \mathtt{X}_i\}_{i=1\dots N}$

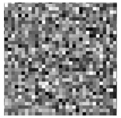
Point de vue probabiliste : $\mathtt{X}_i \sim p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y}_i)$ **Inconnue**

Objectif : Optimiser les paramètres d’un réseau de neurones afin de générer, à partir d’un $\mathtt{Y}$, de **nouveaux échantillons** de $p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$

$\boldsymbol{\theta}$

$\mathtt{Y}$

$\mathtt{Z}_{\mathrm{ech}} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$



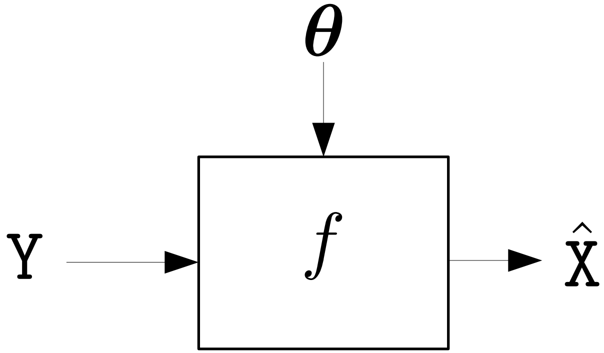
f

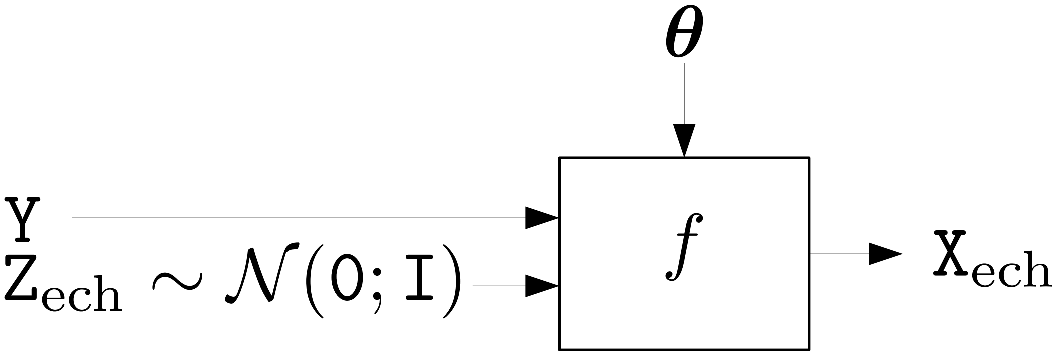
$\mathtt{X}_{\mathrm{ech}} \sim p_{\mathrm{modele}}(\mathtt{X}|\mathtt{Y},\boldsymbol{\theta}) \approx p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$

Différence vis-à-vis de l'apprentissage supervisé

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$

de forme complexe
(ex : multimodale)





Supervisé → Régresseur ou classifieur

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$ de forme “simple”
(ex : unimodale)

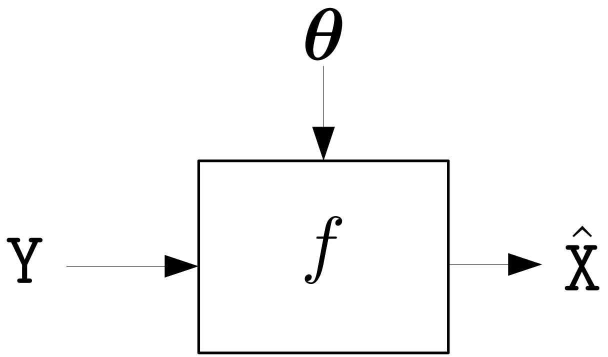
Modèle génératif → Échantillonneur

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$ de forme “complexe”

Différence vis-à-vis de l’apprentissage supervisé (suite)

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$

de forme complexe
(ex : multimodale)



Question : Que se passe-t-il si on apprend un régresseur pour colorer une image ?

Supervisé → Régresseur ou classifieur

$p_{\mathrm{data}}(\mathtt{X}|\mathtt{Y})$ de forme “simple”
(ex : unimodale)

Exemples d’utilisation

- Colorisation d’image
- Super-résolution
- Édition/Synthèse de données (ex : retouche d’image)

https://openai.com/blog/dall-e/

Génération de faux contenus :

- faux article de presse
- fausse vidéo
- faux signal sonore

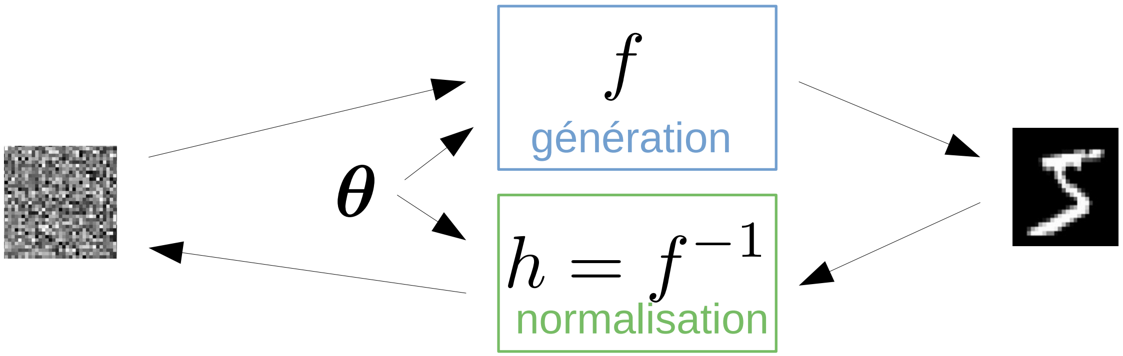
Comment entraîner un modèle génératif profond ?

Beaucoup de solutions possibles, nous allons en étudier trois :

- le réseau inversible (NF)
- l’auto-encodeur variationnel (VAE)
- le réseau antagoniste (GAN)

II) Réseau inversible (NF)

Réseau inversible : idée générale

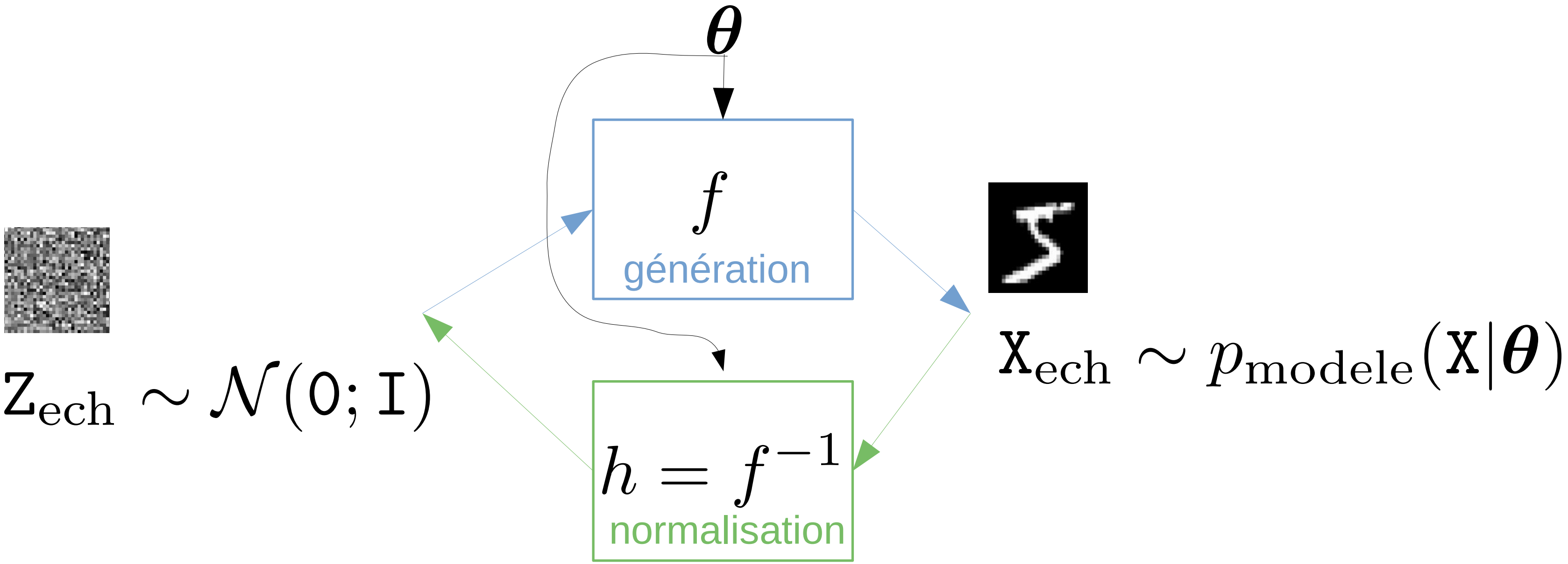


Ne pas disposer de couples $\{\mathtt{Z}_i, \mathtt{X}_i\}_{i=1 \ldots N}$ n'est plus un problème.

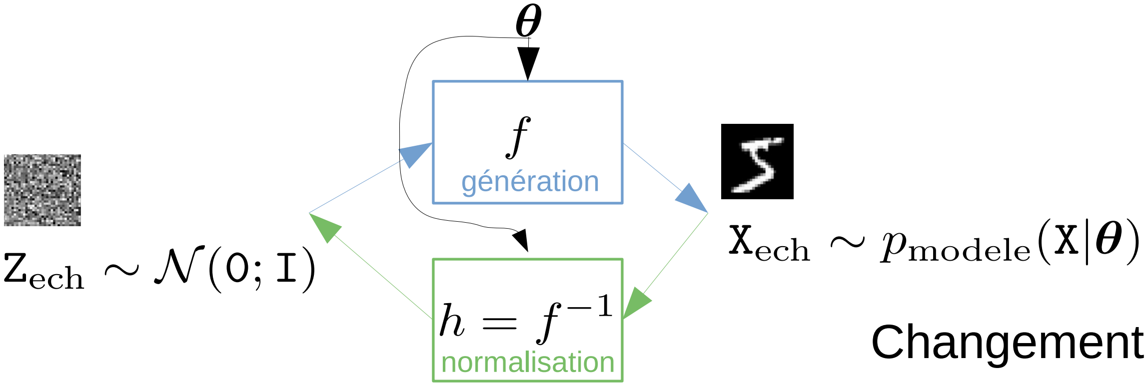
Terminologie : “Invertible neural network”, “Normalizing Flow” (NF)

II)

Réseau inversible : idée générale (suite)



Réseau inversible : formalisme



Changement de variable : $\mathtt{Z} = h(\mathtt{X};\boldsymbol{\theta})$

$$1 = \int p(\mathtt{Z})d\mathtt{Z} = \int \mathcal{N}(\mathtt{Z};\mathtt{0},\mathtt{I})d\mathtt{Z} = \int \underbrace{\mathcal{N}(h(\mathtt{X};\boldsymbol{\theta});\mathtt{0},\mathtt{I})|\det J_h(\mathtt{X})|}_{p_{\mathrm{modele}}(\mathtt{X}|\boldsymbol{\theta})}d\mathtt{X}$$

$$-\ln p_{\mathrm{modele}}(\mathtt{X}|\boldsymbol{\theta}) = ||h(\mathtt{X};\boldsymbol{\theta})||^2 - \ln|\det J_h(\mathtt{X};\boldsymbol{\theta})| + \mathrm{cst}_{\boldsymbol{\theta}}$$

Réseau inversible : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ pour que $p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\text{data}}(\mathtt{X})$

Nécessité de choisir une fonction de similarité (“divergence”).

Pour un NF on choisit généralement la divergence de Kullback-Leibler suivante :

$$KL(p_{\text{data}}(\mathtt{X})||p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) = -\int p_{\text{data}}(\mathtt{X}) \ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta}) d\mathtt{X} + \text{cst}_{\boldsymbol{\theta}}$$

$$L(\boldsymbol{\theta}) = \mathbb{E}_{p_{\text{data}}(\mathtt{X})}(-\ln p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})) \approx \frac{1}{N}\sum_{i=1}^{N} -\ln p_{\text{modele}}(\mathtt{X}_i|\boldsymbol{\theta}))$$

= maximisation de la vraisemblance

“Flow-GAN” : exemple d’utilisation d’un apprentissage antagoniste pour un NF

Réseau inversible : apprentissage (suite)

$$L(\boldsymbol{\theta}) \approx \frac{1}{N} \sum_{i=1}^{N} ||h(\mathtt{X}_i; \boldsymbol{\theta})||^2 - \ln|\det J_h(\mathtt{X}_i; \boldsymbol{\theta})|$$

Essaye de transformer $\mathtt{X}_i$ proche du tenseur “zéro”

Empêche que tout le monde se transforme en un tenseur “zéro”

Descente de gradient

$\boldsymbol{\theta}$

$\mathtt{X}_i \rightarrow \boxed{h = f^{-1}}$ normalisation

$\mathtt{Z}_i = h(\mathtt{X}_i)$

$- \ln|\det J_h(\mathtt{X}_i)|$

$L_i(\boldsymbol{\theta})$

Réseau inversible : apprentissage (suite)

$$L(\boldsymbol{\theta}) \approx \frac{1}{N}\sum_{i=1}^{N} ||h(\mathtt{X}_i;\boldsymbol{\theta})||^2 - \ln|\det J_h(\mathtt{X}_i;\boldsymbol{\theta})|$$

Essaye de transformer $\mathtt{X}_i$ proche du tenseur “zéro”

Empêche que tout le monde se transforme en un tenseur “zéro”

Descente de gradient

$\boldsymbol{\theta}$

$\mathtt{X}_i$ → $h = f^{-1}$ normalisation → $\mathtt{Z}_i = h(\mathtt{X}_i)$, $-\ln|\det J_h(\mathtt{X}_i)|$ → $L_i(\boldsymbol{\theta})$

En pratique : arrêt prématuré (“early stopping”) sur un ensemble de validation.

Réseau inversible : architecture

Besoin d'une architecture très spécifique, chaque couche doit :

1. avoir la même taille en entrée et en sortie

2. être inversible, et pour pouvoir échantillonner efficacement cette fonction inverse doit se calculer efficacement

3. avoir comme propriété que le “logdet” de sa jacobienne se calcule efficacement et soit différentiable.

Couche la plus souvent utilisée : “Affine coupling layer”

ResNet

$$X_1 = Z_1$$

$$X_2 = \exp(s_{\boldsymbol{\theta}}(Z_1)) \odot Z_2 + m_{\boldsymbol{\theta}}(Z_1)$$

jacobienne triangulaire !

Réseau inversible : avantages et inconvénients

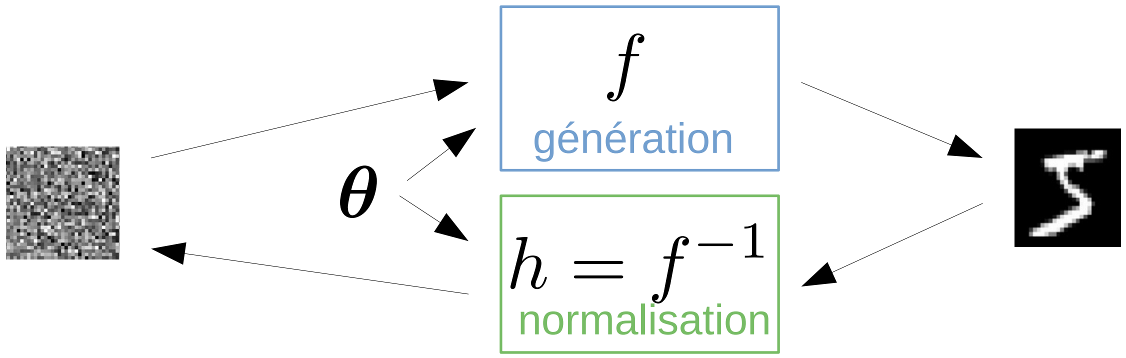
+ capable d’échantillonner efficacement
+ capable de calculer la (log-)probabilité d’une donnée
+ apprentissage possible par maximum de vraisemblance

- contrainte d’architecture inversible
- contrainte du même nombre de dimension entrée/sortie
- la forme de la distribution qu’on transforme est fortement contrainte
- **contrainte sur le calcul du log-déterminant**

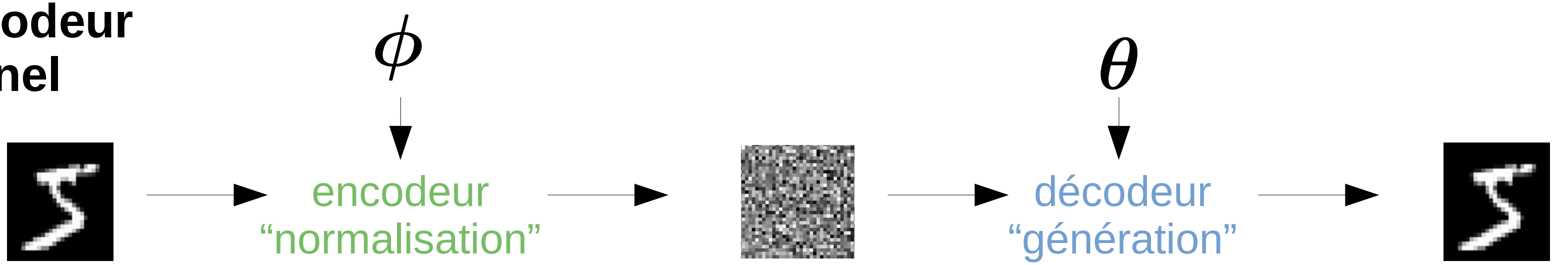
III) Auto-encodeur variationnel (VAE)

Auto-encodeur variationnel : idée générale

Réseau inversible

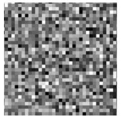


Auto-encodeur variationnel



Terminologie : “Variational Auto-Encoder” (VAE)

Auto-encodeur variationnel : décodeur



$$\mathtt{Z}_{\mathrm{ech}} \sim p(\mathtt{Z}) = \mathcal{N}(\mathtt{0}; \mathtt{I}) \longrightarrow \boxed{\overset{\boldsymbol{\theta}\,\downarrow}{f \ \text{décodeur}}} \longrightarrow \boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}$$

$$p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}, \Sigma_{\mathtt{X}|\mathtt{Z}_{\mathrm{ech}}}) \xrightarrow{\text{échantillon}}$$

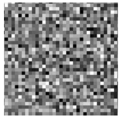
$$p_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z}, \boldsymbol{\theta})$$

Auto-encodeur variationnel : encodeur

$$\phi$$

$$\mathtt{X}_{\mathrm{ech}} \sim p_{\mathrm{data}}(\mathtt{X}) \longrightarrow \boxed{h \text{ encodeur}} \longrightarrow \boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\mathrm{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\mathrm{ech}}}$$

$$q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X}_{\mathrm{ech}}, \phi) = \mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_{\mathrm{ech}}}, \Sigma_{\mathtt{Z}|\mathtt{X}_{\mathrm{ech}}}) \xrightarrow{\text{échantillon}}$$



$$q_{\mathrm{m}}(\mathtt{X}, \mathtt{Z}|\phi) = p_{\mathrm{data}}(\mathtt{X}) q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X}, \phi)$$

Auto-encodeur variationnel : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ et ϕ pour que

$$p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}) = p(\mathtt{Z})p_{\mathrm{m}}(\mathtt{X}|\mathtt{Z},\boldsymbol{\theta}) \approx q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi}) = p_{\mathrm{data}}(\mathtt{X})q_{\mathrm{m}}(\mathtt{Z}|\mathtt{X},\boldsymbol{\phi})$$

Pour un VAE on choisit généralement la divergence de Kullback-Leibler suivante :

$$KL(q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi})||p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}))$$

Auto-encodeur variationnel : apprentissage (suite)

$$KL(q_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\phi})||p_{\mathrm{m}}(\mathtt{X},\mathtt{Z}|\boldsymbol{\theta}))$$

$$=\mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}\mathbb{E}_{\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}},\Sigma_{\mathtt{Z}|\mathtt{X}})}\left(-\ln\mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|\mathtt{Z}},\Sigma_{\mathtt{X}|\mathtt{Z}})\right)$$

“Erreur de reconstruction” :
Incite la sortie du décodeur à ressembler aux X

$$+\mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}},\Sigma_{\mathtt{Z}|\mathtt{X}})||\mathcal{N}(\mathtt{0},\mathtt{I})))$$

“Régularisation” :
Incite la sortie de l’encodeur a être centrée réduite → logique car lorsqu’on générera des z, on les tirera selon une gaussienne centrée réduite

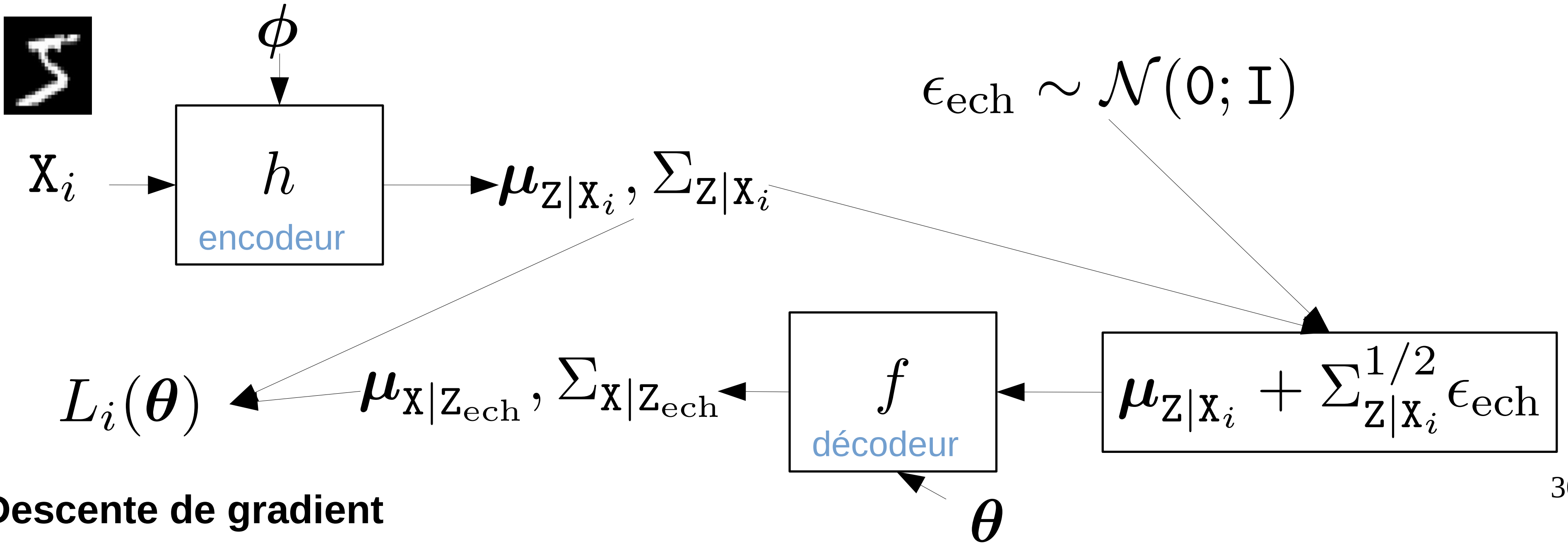
$$+\mathrm{cst}_{\boldsymbol{\phi},\boldsymbol{\theta}}$$

III)

Auto-encodeur variationnel : apprentissage (suite)

$$\epsilon_{\text{ech}} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$$

$$L_i(\boldsymbol{\theta}) = KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}, \Sigma_{\mathtt{Z}|\mathtt{X}_i})||\mathcal{N}(\mathtt{0},\mathtt{I})) - \ln\mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}+\Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i}\epsilon_{\text{ech}})}, \Sigma_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}+\Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i}\epsilon_{\text{ech}})})$$

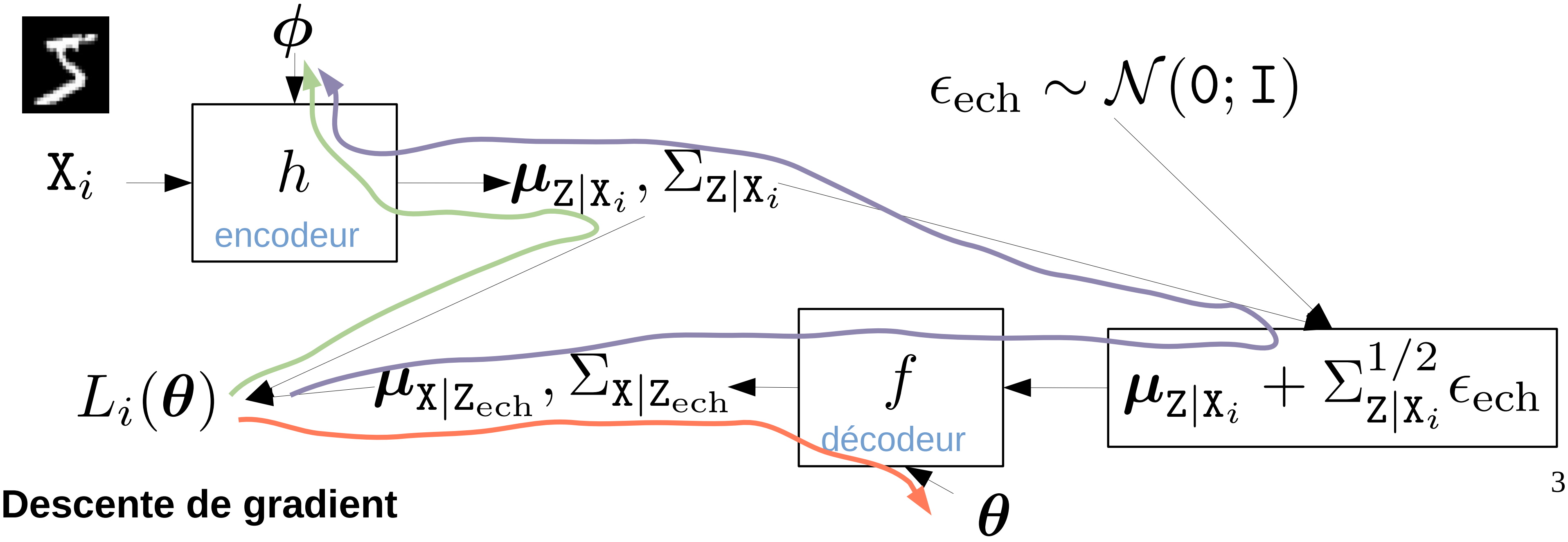


Descente de gradient

Auto-encodeur variationnel : apprentissage (suite)

$$\epsilon_{\text{ech}} \sim \mathcal{N}(0; \mathtt{I})$$

$$L_i(\boldsymbol{\theta}) = KL(\mathcal{N}(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i}, \Sigma_{\mathtt{Z}|\mathtt{X}_i}) || \mathcal{N}(0, \mathtt{I})) - \ln \mathcal{N}(\boldsymbol{\mu}_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i}\epsilon_{\text{ech}})}, \Sigma_{\mathtt{X}|(\boldsymbol{\mu}_{\mathtt{Z}|\mathtt{X}_i} + \Sigma^{1/2}_{\mathtt{Z}|\mathtt{X}_i}\epsilon_{\text{ech}})})$$



Descente de gradient

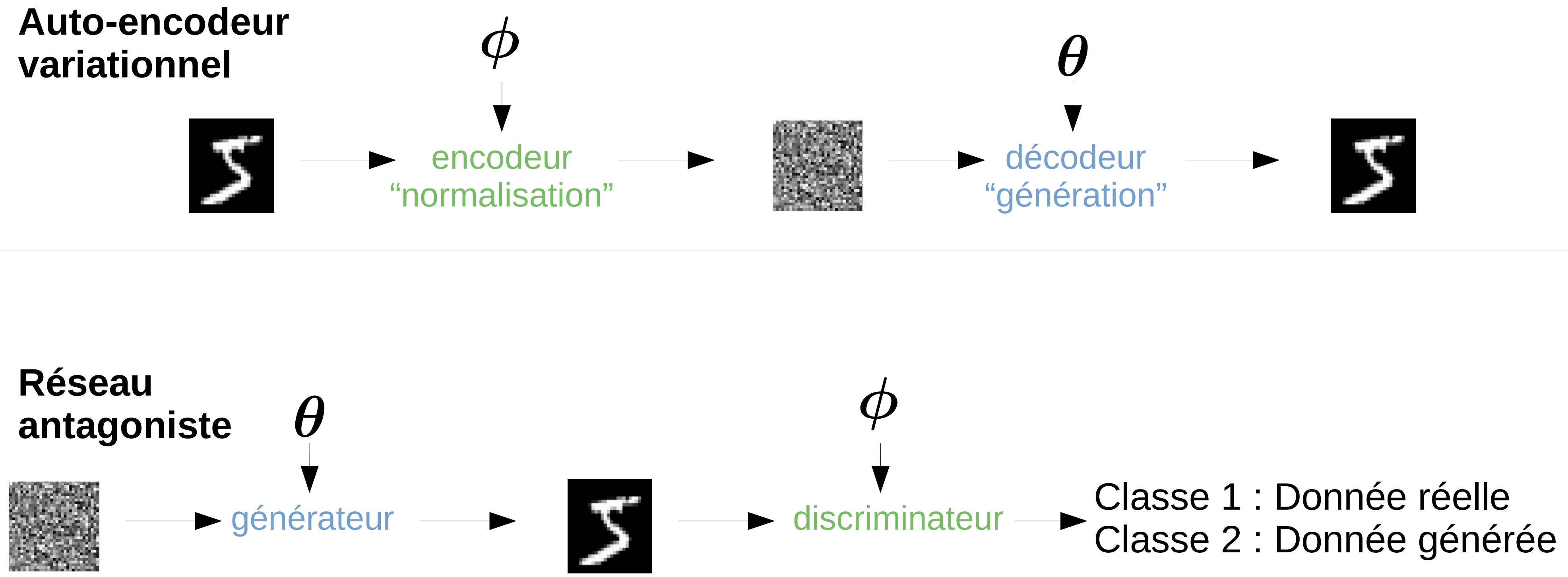
Auto-encodeur variationnel : avantages et inconvénients

+ capable d'échantillonner efficacement
+ pas de contrainte particulière sur l'architecture
+ taille de Z non contrainte par celle de X

- pas d'expression exacte de la (log-)probabilité d'une donnée
- impossible d'apprendre par maximum de vraisemblance (mais borne inférieure quand même)
- contrainte sur la forme des distributions conditionnelles (exemple : covariance diagonale pour avoir un calcul rapide)

IV) Réseau antagoniste (GAN)

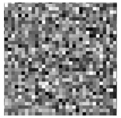
Réseau antagoniste : idée générale



Terminologie : “Generative Adversarial Network” (GAN)

Réseau antagoniste : générateur

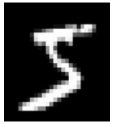
$\boldsymbol{\theta}$



$\mathtt{Z}_{\text{ech}} \sim \mathcal{N}(\mathtt{0};\mathtt{I})$

f

génération



$\mathtt{X}_{\text{ech}} \sim p_{\text{modele}}(\mathtt{X}|\boldsymbol{\theta})$

Définition implicite, le générateur n’est pas inversible

Réseau antagoniste : discriminateur

$\mathtt{X}_{\mathrm{ech}} \sim p_{\mathrm{modele}}(\mathtt{X}|\boldsymbol{\theta})$

ϕ

d

discrimination

Probabilité “Donnée réelle” ≈ 0

$\mathtt{X}_{\mathrm{ech}} \sim p_{\mathrm{data}}(\mathtt{X})$

ϕ

d

discrimination

Probabilité “Donnée réelle” ≈ 1

Réseau antagoniste : apprentissage

Objectif : optimiser $\boldsymbol{\theta}$ et ϕ pour que $p_{\mathrm{modele}}(\mathtt{X}|\boldsymbol{\theta}) \approx p_{\mathrm{data}}(\mathtt{X})$

Pour un GAN on choisit généralement une divergence de la famille suivante :

$$D(p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})||p_{\mathrm{data}}(\mathtt{X})) =$$

$$\max_{\boldsymbol{\phi}} \mathbb{E}_{p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})}(h'(\mathtt{X},\boldsymbol{\phi})) - \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(h''(\mathtt{X},\boldsymbol{\phi}))$$

Exemple :

$$\max_{\boldsymbol{\phi}} \mathbb{E}_{p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})}(\ln(1 - d(\mathtt{X},\boldsymbol{\phi})) + \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(\ln(d(\mathtt{X},\boldsymbol{\phi}))$$

Maximum atteint quand $d(\mathtt{X},\boldsymbol{\phi}) = \dfrac{p_{\mathrm{data}}(\mathtt{X})}{p_{\mathrm{data}}(\mathtt{X}) + p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})}$

Si $p_{\mathrm{m}} = p_{\mathrm{data}}$

Valeur div. $-\ln(4)$

Réseau antagoniste : apprentissage (suite)

$$\min_{\boldsymbol{\theta}} D(p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})||p_{\mathrm{data}}(\mathtt{X}))$$

$$= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{p_{\mathrm{m}}(\mathtt{X}|\boldsymbol{\theta})}(\ln(1 - d(\mathtt{X}, \boldsymbol{\phi})) + \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(\ln(d(\mathtt{X}, \boldsymbol{\phi}))$$

$$= \min_{\boldsymbol{\theta}} \max_{\boldsymbol{\phi}} \mathbb{E}_{\mathcal{N}(\mathtt{0},\mathtt{I})}(\ln(1 - d(f(\mathtt{Z}, \boldsymbol{\theta}), \boldsymbol{\phi})) + \mathbb{E}_{p_{\mathrm{data}}(\mathtt{X})}(\ln(d(\mathtt{X}, \boldsymbol{\phi}))$$

Problème minmax → potentiellement beaucoup plus difficile à optimiser qu’un VAE ou qu’un NF

Réseau antagoniste : apprentissage (suite)

En pratique on alterne :

Un pas gradient pour optimiser ϕ

$$L'_i(\phi) = -\ln(1 - d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \phi)) - \ln(d(\mathtt{X}_i, \phi))$$

Objectif : Améliorer les paramètres du discriminateur afin qu'il prédise une valeur proche de 0 pour une donnée générée et proche de 1 pour une donnée réelle

Suivi d'un pas de gradient pour optimiser $\boldsymbol{\theta}$

$$L''_i(\boldsymbol{\theta}) = \ln(1 - d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \phi)) \longrightarrow$$

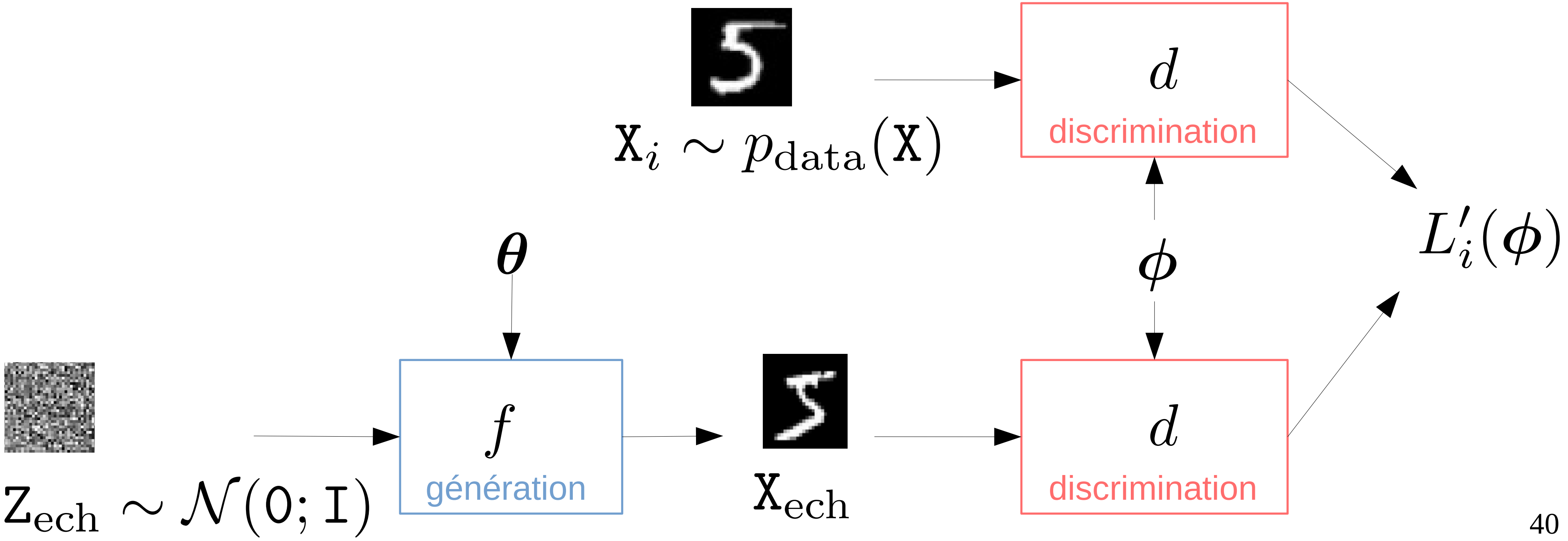
En pratique remplacé par :

$$-\ln(d(f(\mathtt{Z}_{\mathrm{ech}}, \boldsymbol{\theta}), \phi))$$

Objectif : Améliorer les paramètres du générateur afin que le discriminateur se trompe et prédise une valeur proche de 1

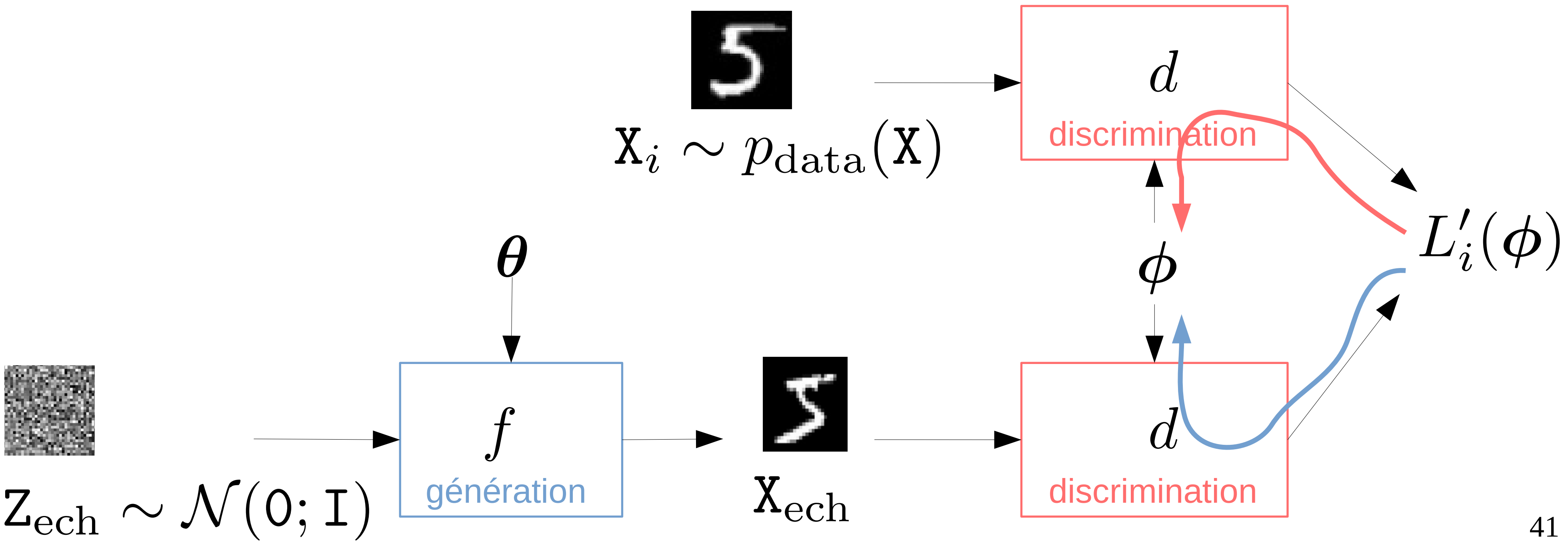
Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du discriminateur



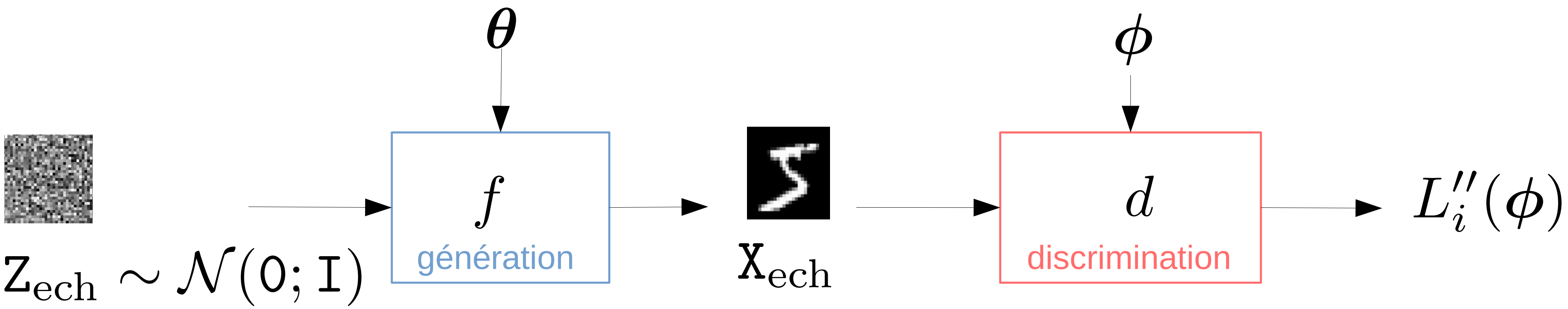
Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du discriminateur



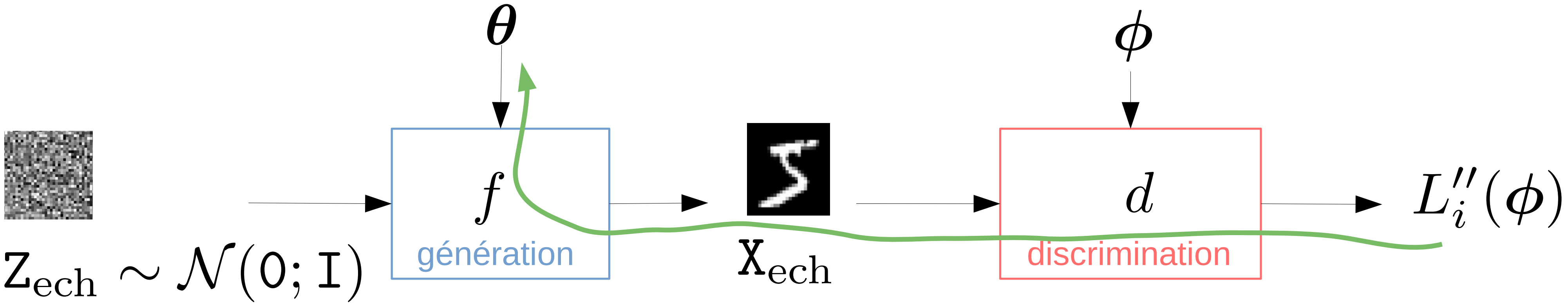
Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du générateur



Réseau antagoniste : apprentissage (suite)

Pas de gradient sur les paramètres du générateur

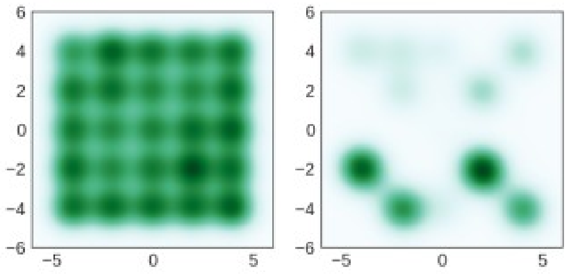


Réseau antagoniste : avantages et inconvénients

+ capable d'échantillonner efficacement
+ pas de contrainte particulière sur l'architecture
+ taille de Z non contrainte par celle de X

- pas d'expression de la (log-)probabilité d'une donnée
- problème minmax difficile à optimiser

** problèmes de convergence
** échantillonnage uniquement d'une partie de la distribution ("mode collapse")



(f) True Data (g) GAN