

RCP211 – Modèles génératifs

Auto-encodeurs variationnels avancés

Arnaud Breloy `arnaud.breloy@lecnam.net`

28 octobre 2025

Conservatoire national des arts & métiers

Table of contents

1. Rappels

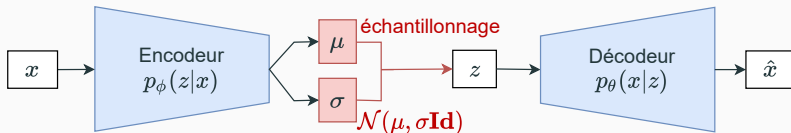
2. VAE conditionnels

3. VQ-VAE

4. Modèles autorégressifs

Rappels

Auto-encodeur variationnel



Fonction de coût du VAE

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\mathbb{E}_{q_\phi(z|x)} [\|\hat{x} - x\|]}_{\text{vraisemblance (reconstruction)}} + \underbrace{D_{\text{KL}}(q_\phi(z|x) | p(z))}_{\text{régularisation (distance au prior)}}$$

- **Reconstruction** : « similarité » entre l'entrée et la sortie
 - dépend du problème (MSE, MAE, BCE...)
- **Régularisation** : divergence-KL (contrainte de proximité au *prior*)
 - Expression analytique dans le cas où tout est gaussien

VAE conditionnels

Modèles génératifs

Le décodeur des auto-encodeurs forme un modèle génératif $p_{\theta}(x|z)$.

- on ne connaît (généralement) pas $p(z) \implies$ VAE
- comment spécifier une **connaissance a priori** sur l'observation x que l'on souhaite générer?

Exemples de problématiques

- générer une séquence cohérente d'objets,
- réaliser une prédiction probabiliste,
- générer un exemple d'une certaine classe,
- générer un exemple présentant des propriétés de plusieurs classes.

Modèles génératifs

Le décodeur des auto-encodeurs forme un modèle génératif $p_{\theta}(x|z)$.

- on ne connaît (généralement) pas $p(z) \implies$ VAE
- comment spécifier une **connaissance a priori** sur l'observation x que l'on souhaite générer?

Exemples de problématiques

- générer une séquence cohérente d'objets,
- réaliser une prédiction probabiliste,
- générer un exemple d'une certaine classe,
- générer un exemple présentant des propriétés de plusieurs classes.

Modèles génératifs

Le décodeur des auto-encodeurs forme un modèle génératif $p_{\theta}(x|z)$.

- on ne connaît (généralement) pas $p(z) \implies$ VAE
- comment spécifier une **connaissance a priori** sur l'observation x que l'on souhaite générer?

Exemples de problématiques

- générer une séquence cohérente d'objets,
- réaliser une prédiction probabiliste,
- générer un exemple d'une certaine classe,
- générer un exemple présentant des propriétés de plusieurs classes.

Modèles génératifs

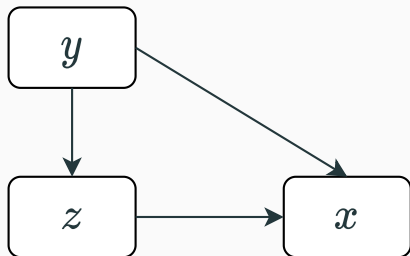
Le décodeur des auto-encodeurs forme un modèle génératif $p_{\theta}(x|z)$.

- on ne connaît (généralement) pas $p(z) \implies$ VAE
- comment spécifier une **connaissance a priori** sur l'observation x que l'on souhaite générer?

Exemples de problématiques

- générer une séquence cohérente d'objets,
- réaliser une prédiction probabiliste,
- générer un exemple d'une certaine classe,
- générer un exemple présentant des propriétés de plusieurs classes.

VAE conditionnel



Conditionnement

Construire le modèle génératif

$$p_{\theta}(x|y, z)$$

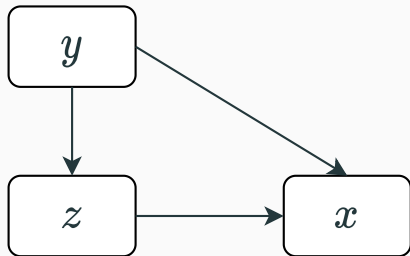
→ on injecte une information spécifique (y) dans l'espace latent ($\mathcal{Y} \times \mathcal{Z}$)

Cadre formel

On cherche un modèle θ permettant de générer x à partir de z sachant y . On maximise la (log-)vraisemblance *conditionnelle* sur $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$:

$$\hat{\theta} = \arg \max_{\theta} \mathbb{E}_{p_{\mathcal{D}}} \log p_{\theta}(x|y) = \frac{1}{N} \sum_i \log p_{\theta}(x_i|y_i)$$

VAE conditionnel

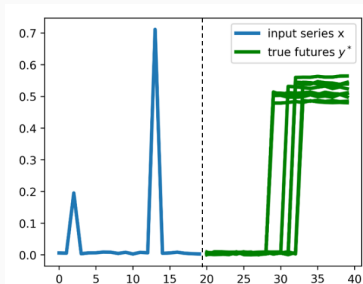


Conditionnement

Construire le modèle génératif

$$p_{\theta}(x|y, z)$$

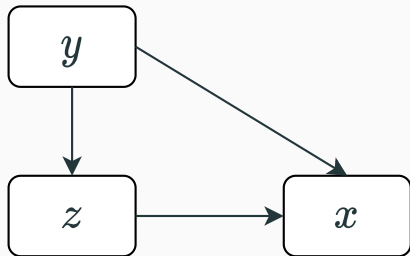
→ on injecte une information spécifique (y) dans l'espace latent ($\mathcal{Y} \times \mathcal{Z}$)



Cadre formel

On cherche un modèle θ permettant de générer x à partir de z

VAE conditionnel

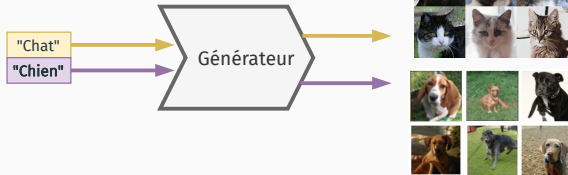


Conditionnement

Construire le modèle génératif

$$p_{\theta}(x|y, z)$$

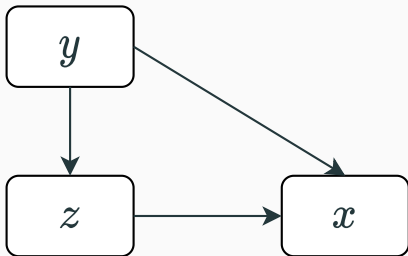
→ on injecte une information spécifique (y) dans l'espace latent ($\mathcal{Y} \times \mathcal{Z}$)



Cadre formel

On cherche un modèle θ permettant de générer x à partir de z

VAE conditionnel



Conditionnement

Construire le modèle génératif

$$p_{\theta}(x|y, z)$$

→ on injecte une information spécifique (y) dans l'espace latent ($\mathcal{Y} \times \mathcal{Z}$)

Cadre formel

On cherche un modèle θ permettant de générer x à partir de z sachant y . On maximise la (log-)vraisemblance *conditionnelle* sur $\mathcal{D} = \{(x_1, y_1), \dots, (x_n, y_n)\}$:

$$\hat{\theta} = \arg \max_{\theta} \mathbb{E}_{p_{\mathcal{D}}} \log p_{\theta}(x|y) = \frac{1}{N} \sum_i \log p_{\theta}(x_i|y_i)$$

VAE conditionnel

Soit x les données, z les variables latentes, y le *conditionnement*.

Le VAE conditionnel s'intéresse :

- au modèle génératif $p_{\theta}(x|z, y)$ (\simeq décodeur)
- au modèle conditionnel $q_{\phi}(z|x, y)$ (\simeq encodeur)

Objectif

Déterminer les paramètres θ et ϕ tels que :

- $p_{\theta}(x|y)$ soit une bonne approximation de $p(x|y) \rightarrow$ génération conditionnelle,
- $q_{\phi}(z|x, y)$ soit une bonne approximation de $p_{\theta}(z|x, y) \rightarrow$ encodage dans l'espace latent.

VAE conditionnel

Soit x les données, z les variables latentes, y le *conditionnement*.

Le VAE conditionnel s'intéresse :

- au modèle génératif $p_{\theta}(x|z, y)$ (\simeq décodeur)
- au modèle conditionnel $q_{\phi}(z|x, y)$ (\simeq encodeur)

Objectif

Déterminer les paramètres θ et ϕ tels que :

- $p_{\theta}(x|y)$ soit une bonne approximation de $p(x|y) \rightarrow$ génération conditionnelle,
- $q_{\phi}(z|x, y)$ soit une bonne approximation de $p_{\theta}(z|x, y) \rightarrow$ encodage dans l'espace latent.

VAE conditionnel

Soit x les données, z les variables latentes, y le *conditionnement*.

Le VAE conditionnel s'intéresse :

- au modèle génératif $p_{\theta}(x|z, y)$ (\simeq décodeur)
- au modèle conditionnel $q_{\phi}(z|x, y)$ (\simeq encodeur)

Objectif

Déterminer les paramètres θ et ϕ tels que :

- $p_{\theta}(x|y)$ soit une bonne approximation de $p(x|y) \rightarrow$ génération conditionnelle,
- $q_{\phi}(z|x, y)$ soit une bonne approximation de $p_{\theta}(z|x, y) \rightarrow$ encodage dans l'espace latent.

VAE conditionnel

Soit x les données, z les variables latentes, y le *conditionnement*.

Le VAE conditionnel s'intéresse :

- au modèle génératif $p_{\theta}(x|z, y)$ (\simeq décodeur)
- au modèle conditionnel $q_{\phi}(z|x, y)$ (\simeq encodeur)

Objectif

Déterminer les paramètres θ et ϕ tels que :

- $p_{\theta}(x|y)$ soit une bonne approximation de $p(x|y) \rightarrow$ génération conditionnelle,
- $q_{\phi}(z|x, y)$ soit une bonne approximation de $p_{\theta}(z|x, y) \rightarrow$ encodage dans l'espace latent.

VAE conditionnel

Soit x les données, z les variables latentes, y le *conditionnement*.

Le VAE conditionnel s'intéresse :

- au modèle génératif $p_{\theta}(x|z, y)$ (\simeq décodeur)
- au modèle conditionnel $q_{\phi}(z|x, y)$ (\simeq encodeur)

Objectif

Déterminer les paramètres θ et ϕ tels que :

- $p_{\theta}(x|y)$ soit une bonne approximation de $p(x|y) \rightarrow$ génération conditionnelle,
- $q_{\phi}(z|x, y)$ soit une bonne approximation de $p_{\theta}(z|x, y) \rightarrow$ encodage dans l'espace latent.

VAE conditionnel

Soit x les données, z les variables latentes, y le *conditionnement*.

Le VAE conditionnel s'intéresse :

- au modèle génératif $p_{\theta}(x|z, y)$ (\simeq décodeur)
- au modèle conditionnel $q_{\phi}(z|x, y)$ (\simeq encodeur)

Objectif

Déterminer les paramètres θ et ϕ tels que :

- $p_{\theta}(x|y)$ soit une bonne approximation de $p(x|y) \rightarrow$ génération conditionnelle,
- $q_{\phi}(z|x, y)$ soit une bonne approximation de $p_{\theta}(z|x, y) \rightarrow$ encodage dans l'espace latent.

Définition

$$\text{ELBO}(x, \theta, \phi|y) = \log p_{\theta}(x|y) - \text{KL}(q_{\phi}(z|x, y) || p_{\theta}(z|x, y))$$

Équivalence

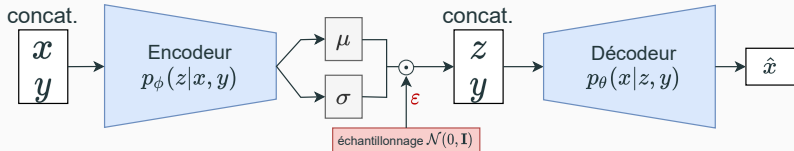
$$\arg \max_{\theta} \log p_{\theta}(x|y)$$



$$\arg \max_{\theta, \phi} \text{ELBO}(\theta, \phi; x, y)$$

(extension naturelle du cas non-conditionnel)

Entraînement du VAE conditionnel

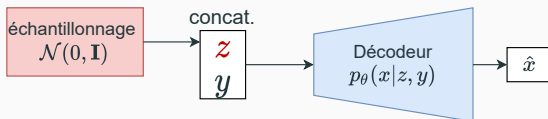


Optimisation

Analogue au VAE classique :

- Astuce de la reparamétrisation,
- Concaténation du conditionnement au code latent.

Inférence dans le VAE conditionnel



Génération

1. Choix de y (ou tirage aléatoire),
2. Échantillonnage de $z \sim p(z)$,
3. Concaténation $[y, z]$,
4. Génération de \hat{x} par décodage.

Évaluer des modèles génératifs est difficile : comment quantifier la qualité des observations synthétiques produites ?

Fréchet Inception Distance

On compare la distribution des observations générées \tilde{x} et réelles x . Supposons un modèle (par exemple, discriminatif) f qui produit *features* des observations réelles et générées calculées.

- Calculer $f(\tilde{x})$ et $f(x)$ pour n observations réelles et générées,
- Estimer la moyenne μ et la variance Σ sur les deux distributions
- Calculer la distance de Fréchet entre les distributions :

$$\text{FID} = |\mu - \tilde{\mu}|^2 + \text{tr} \left(\Sigma + \tilde{\Sigma} - 2(\Sigma\tilde{\Sigma})^{1/2} \right)$$

Intuitivement, FID faible \implies distributions proches.

Dans le cas des images, on utilise les *features* extraites par le modèle Inception v3 préentraîné sur Imagenet.

Évaluation des VAE conditionnels

Si le modèle est conditionnel, on voudrait aussi mesurer son respect du conditionnement. Pour un modèle génératif $p(x|y)$, on suppose que l'on a un modèle discriminatif $p(y|x)$ entraîné sur le même jeu de données.

Inception score

- Générer n observations synthétiques \tilde{x}
- Calculer pour chaque observation la prédiction du modèle discriminatif $p(y|\tilde{x})$
 - Idéalement, cette prédiction devrait être proche du vecteur one-hot associé à y
- Calculer la divergence KL entre la distribution a priori des classes $p(y)$ et la distribution observée $p(y|\tilde{x})$:

$$IS = \exp \left(\mathbb{E}_{\tilde{x} \sim p(x|y)} [\text{KL}(p(y|\tilde{x})|p(y))] \right)$$

VQ-VAE

Distribution postérieure dans le VAE

Approximation du postérieur

$q_\phi(z|x_i) = \mathcal{N}(\mu_{x_i}, \sigma_{x_i} \mathbf{I}) \rightarrow$ obtenue par l'encodeur

Avantages

- Simple à calculer, dérivation facile pour la rétropropagation
- Échantillonnage aisé à l'aide du *reparametrization trick*
- Possibilité d'approcher $q_\phi(z|x) = \sum_i q_\phi(z|x_i)$ (par ex., modèle de mélange gaussien)

Inconvénients

- Capacité d'approximation limitée (gaussienne...)
- La régularisation (proximité au *prior*) limite les possibilités pour μ_x
- Pas de corrélations entre les dimensions de $z \rightarrow$ interprétabilité limitée

Distribution postérieure dans le VAE

Approximation du postérieur

$q_\phi(z|x_i) = \mathcal{N}(\mu_{x_i}, \sigma_{x_i} \mathbf{I}) \rightarrow$ obtenue par l'encodeur

Avantages

- Simple à calculer, dérivation facile pour la rétropropagation
- Échantillonnage aisé à l'aide du *reparametrization trick*
- Possibilité d'approcher $q_\phi(z|x) = \sum_i q_\phi(z|x_i)$ (par ex., modèle de mélange gaussien)

Inconvénients

- Capacité d'approximation limitée (gaussienne...)
- La régularisation (proximité au *prior*) limite les possibilités pour μ_x
- Pas de corrélations entre les dimensions de $z \rightarrow$ interprétabilité limitée

Discrétiser l'espace latent

Pourquoi discrétiser ?

- Facilite la manipulation
- Permet d'apprendre la forme de l'espace latent

Codebook

L'espace latent gaussien \mathcal{Z} est remplacé par un dictionnaire de K codes latents appris :

$$\{e_1, e_2, \dots, e_K\}$$

Une observation est représentée par un vecteur d'entiers, correspondant aux « mots » du dictionnaire qui la décrivent, par exemple :

$$x \rightarrow [1, 7, 104, 12] \rightarrow [e_1, e_7, e_{104}, e_{12}]$$

Distribution catégorielle

Considérons un ensemble latent de K vecteurs $\{e_1, \dots, e_K\}$. On définit alors la distribution postérieure $q_\phi(z|x)$ telle que :

$$q_\phi(z = k|x) = \begin{cases} 1 & \text{si } k = \arg \min_j \|z_e(x) - e_j\|_2 \\ 0 & \text{sinon} \end{cases}$$

Autrement dit, z prend pour valeur le vecteur e_i qui est le plus proche du code $z_e(x)$ obtenu en sortie de l'encodeur.

- l'espace latent est divisé en K entrées d'un dictionnaire,
- la catégorie latente est obtenue par recherche du plus proche voisin.

VQ-ELBO

$$q_{\phi}(z = k|x) = \begin{cases} 1 & \text{si } k = \arg \min_j \|z_e(x) - e_j\|_2 \\ 0 & \text{sinon} \end{cases}$$

$q_{\phi}(z = k|x)$ est déterministe. En supposant l'a priori sur z **uniforme** :

$$\text{KL}(q_{\phi}(z|x)||p_{\theta}(z)) = \log K$$

La fonction objectif du VQ-VAE est alors :

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\mathbb{E}_{q_{\phi}(z|x)}[\log p_{\theta}(x|z)]}_{\log p_{\theta}(x|z) \text{ car } q_{\phi}(z|x) \text{ est déterministe}} - \underbrace{\text{KL}(q_{\phi}(z|x)||p_{\theta}(z))}_{\log K}$$

Quantification

Le décodeur reçoit $z_q(x) = e_i$ qui est le plus proche voisin de $z_e(x)$:

$$z_q(x) = \arg \min_{e_j \in \{e_1, \dots, e_K\}} \|z_e(x) - e_j\|$$

Non-dérivabilité

L'opérateur $\arg \min$ n'est pas dérivable :

- on « copie » le gradient en entrée du décodeur D_θ à la sortie de l'encodeur E_ϕ ,

→ on calcule les gradients de $z_e(x)$ comme si c'était $z_q(x)$

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\log p_\theta(x|z_q(x))}_{\text{vraisemblance}} + \underbrace{\|\text{sg}[z_e(x)] - e\|_2^2}_{\text{apprentissage des atomes, sg = "stop gradient"}}$$

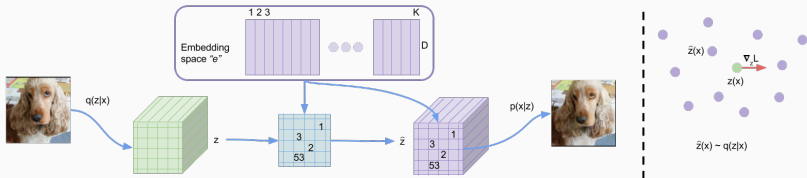
Problème

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\log p_{\theta}(x|z_q(x))}_{\text{vraisemblance}} + \underbrace{\|\text{sg}[z_e(x)] - e\|_2^2}_{\text{apprentissage des atomes, sg = "stop gradient"}}$$

Rien n'empêche l'encodeur de dériver par rapport au dictionnaire ou d'osciller entre plusieurs valeurs pour e .

Régularisation

$$\mathcal{L}(\theta, \phi; x) = \underbrace{\log p_{\theta}(x|z_q(x))}_{\text{vraisemblance}} + \underbrace{\|\text{sg}[z_e(x)] - e\|_2^2}_{\text{apprentissage des atomes}} + \beta \underbrace{\|z_e(x) - \text{sg}[e]\|_2^2}_{\text{commitment loss}}$$



Encodage en pratique

En réalité, une observation x est encodée en N vecteurs latents. Par exemple, une image $224 \times 224 \rightarrow 32 \times 32$ codes entiers.

Échantillonnage ?

Selon quelle distribution échantillonner les valeurs des codes z pour la génération ?

→ apprentissage d'un modèle autorégressif (PixelCNN) pour apprendre la distribution des z

Modèles autorégressifs

Supposons un processus :

- dont on peut estimer les prochaines réalisations à partir des observations déjà rencontrées,
- dont les prochaines réalisations ne dépendent pas du futur (i.e. **causal**).

Processus autorégressif

Supposons un processus :

- dont on peut estimer les prochaines réalisations à partir des observations déjà rencontrées,
- dont les prochaines réalisations ne dépendent pas du futur (i.e. **causal**).

Définition

On appelle AR(p) ou *processus autorégressif d'ordre p* un processus $\{x_t\}_{0 \leq t \leq N}$ qui vérifie :

$$x_t = c + \varphi_1 X_{t-1} + \varphi_2 X_{t-2} + \cdots + \varphi_p X_{t-p} + \epsilon_t$$

avec ϵ_t un bruit. On appelle $\varphi_1, \varphi_2, \dots, \varphi_p$ les paramètres du modèle.

Processus autorégressif

Supposons un processus :

- dont on peut estimer les prochaines réalisations à partir des observations déjà rencontrées,
- dont les prochaines réalisations ne dépendent pas du futur (i.e. **causal**).

Propriété d'autorégression

Un processus $\{x_t\}_{0 \leq t \leq N}$ satisfait la *propriété d'autorégression* si :

$$p(x_t) = p(x_t | x_{t-1}, \dots, x_1)$$

Exemple : les chaînes de Markov sont des processus autorégressifs d'ordre 1 ($p(x_t) = p(x_t | x_{t-1})$).

Processus autorégressif

Supposons un processus :

- dont on peut estimer les prochaines réalisations à partir des observations déjà rencontrées,
- dont les prochaines réalisations ne dépendent pas du futur (i.e. **causal**).

Propriété d'autorégression

Un processus $\{x_t\}_{0 \leq t \leq N}$ satisfait la *propriété d'autorégression* si :

$$p(x_t) = p(x_t | x_{t-1}, \dots, x_1)$$

Exemple : les chaînes de Markov sont des processus autorégressifs d'ordre 1 ($p(x_t) = p(x_t | x_{t-1})$).

Ordonnancement

Soit \mathcal{D} un jeu de données contenant les observations d'une variable aléatoire $\mathbf{x} \in \mathbb{R}^d$. À l'aide de la *chain rule** :

$$\begin{aligned} p(x_1, x_2, \dots, x_d) &= p(x_1, x_2, \dots, x_{d-1}) \cdot p(x_d | x_1, x_2, \dots, x_{d-1}) \\ &= p(x_{<d-1}) \cdot p(x_d | x_{<d-1}) \\ &= p(x_{<d-2}) \cdot p(x_{d-1} | x_{<d-2}) \cdot p(x_d | x_{<d-1}) \\ &= \dots \end{aligned}$$

→ \mathbf{x} devient un processus autorégressif pour peu que l'on accepte d'imposer un *ordonnancement* sur ses dimensions

Modèle génératif autorégressif

On suppose $p_{\theta_i}(x_i | \mathbf{x}_{<i})$: les lois conditionnelles sont paramétrées par θ_i .

* Formule des probabilités composées : $P(A \cap B) = P(B | A) \cdot P(A)$

Ordonnancement

Soit \mathcal{D} un jeu de données contenant les observations d'une variable aléatoire $\mathbf{x} \in \mathbb{R}^d$. À l'aide de la *chain rule** :

$$p(\mathbf{x}) = \prod_{i=1}^d p(x_i | \mathbf{x}_{<i}) = \prod_{i=1}^d p(x_i | x_1, x_2, \dots, x_{i-1})$$

→ \mathbf{x} devient un processus autorégressif pour peu que l'on accepte d'imposer un *ordonnancement* sur ses dimensions

Modèle génératif autorégressif

On suppose $p_{\theta_i}(x_i | \mathbf{x}_{<i})$: les lois conditionnelles sont paramétrées par θ_i .

* Formule des probabilités composées : $P(A \cap B) = P(B | A) \cdot P(A)$

Exemple

Soit $\{x_t\}_{0 \leq t \leq N}$ avec $x_i \in \{0, 1\}$:

$$p_{\theta_i}(x_i | \mathbf{x}_{<i}) = \text{Bern}(f_i(x_1, x_2, \dots, x_{i-1}))$$

c'est-à-dire que la probabilité de passer d'obtenir $x_i = 1$ est donnée par une loi de Bernouilli dont la probabilité est une fonction f_i de x_1, x_2, \dots, x_{i-1} , paramétrée par θ_i .

Cas simple

$$f_i(x_1, x_2, \dots, x_{i-1}) = \sigma(\alpha_0^{(i)} + \alpha_1^{(i)}x_1 + \dots + \alpha_{i-1}^{(i)}x_{i-1})$$

avec σ la sigmoïde et $\theta_i = \{\alpha_0^{(i)}, \dots, \alpha_{i-1}^{(i)}\}$ les paramètres.

\implies *fully-visible sigmoid belief network (FVSBN)*

Neural Autoregressive Density Estimator

f_i est remplacé par un perceptron multi-couche :

$$\mathbf{h}_i = \sigma(A_i \mathbf{x}_{<i} + \mathbf{c}_i)$$

$$f_i(x_1, x_2, \dots, x_{i-1}) = \sigma(\alpha^{(i)} \mathbf{h}_i + b_i)$$

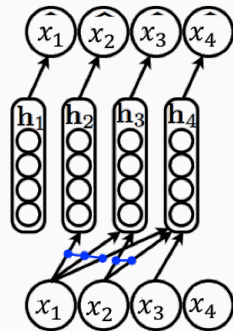
Partage des poids

En pratique, tous les MLP utilisent la même matrice de poids W et on utilise $A_i = W_{\cdot, <i}$:

$$\mathbf{h}_i = \sigma(\mathbf{a}_i)$$

$$\mathbf{a}_{i+1} = \mathbf{a}_i + W[:, i] x_i$$

avec $\mathbf{a}_1 = \mathbf{c}$.



NADE

Comment rendre un autoencodeur autorégressif?

Pour un ordonnancement donné x_1, \dots, x_n , il ne doit pas y avoir de chemin dans le réseau qui mène de $\mathbf{x}_{<i}$ à \hat{x}_i .

Masked Autoencoders for Density Estimation

Autoencodeur : *multi-layer perceptron*

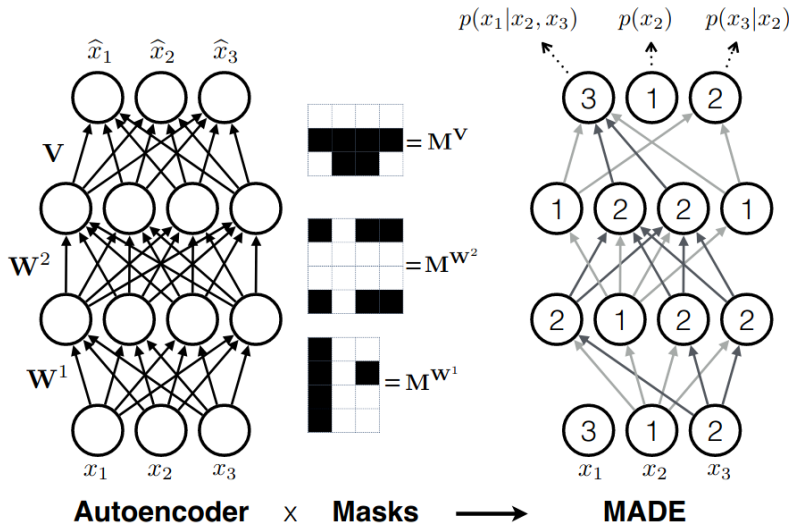
$$\mathbf{z} = f(\mathbf{b} + (\mathbf{W} \odot \mathbf{M}^{\mathbf{W}})\mathbf{x})$$

$$\hat{\mathbf{x}} = \sigma(\mathbf{c} + (\mathbf{V} \odot \mathbf{M}^{\mathbf{V}})\mathbf{z})$$

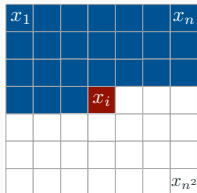
On assigne un numéro $m(k) < D$ à chaque neurone et on “masque” tous les neurones d’un numéro supérieur au neurone courant :

$$\mathbf{M}_{m(k),d}^{\mathbf{W}} = \begin{cases} 1 & \text{si } m(k) \geq d \\ 0 & \text{sinon} \end{cases} \quad \text{et} \quad \mathbf{M}_{m(k),d}^{\mathbf{V}} = \begin{cases} 1 & \text{si } m(k) > d \\ 0 & \text{sinon} \end{cases}$$

MADE : illustration



Génération d'images



Context

Pixels = séquence

$$p(\mathbf{x}) = \prod_{i=1}^{n^2} p(x_i | x_1, \dots, x_{i-1})$$

Image couleur : trois canaux RVB

$$(x_i = (x_{i,R}, x_{i,V}, x_{i,B}))$$

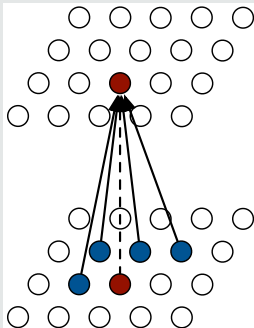
$$p(x_i | \mathbf{x}_{<i}) = p(x_{i,R} | \mathbf{x}_{<i}) p(x_{i,V} | \mathbf{x}_{<i}, x_{i,R}) p(x_{i,B} | \mathbf{x}_{<i}, x_{i,R}, x_{i,V})$$

Pixels discrets ou continus ?

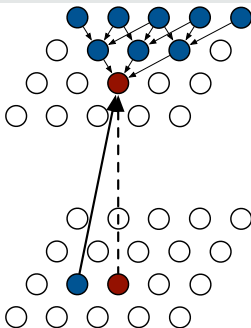
Deux possibilités :

- $p(\mathbf{x})$ est une distribution continue (\sim régression)
- $p(\mathbf{x})$ est une distribution discrète (256 valeurs) \rightarrow softmax

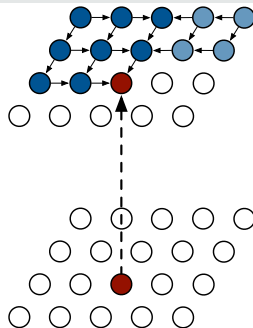
Architectures autorégressives



PixelCNN



Row LSTM



Diagonal BiLSTM