

PROJET MODÈLES À STRUCTURES LATENTES

Les auto-encodeurs



MOUHAMED BA

ABDOULAYE KOROKO

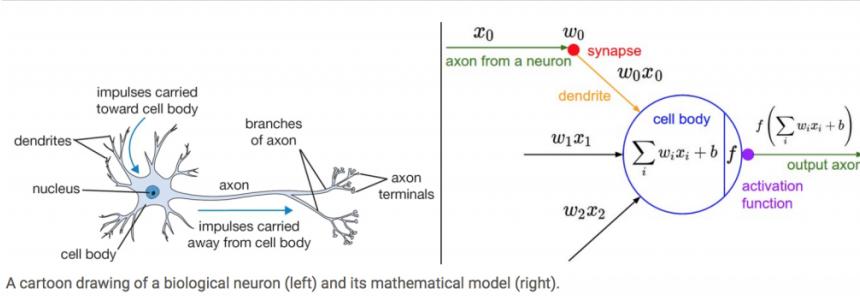
RAMATOULAYE NDIAYE

Sommaire

1 Réseaux de neurones artificiels	2
2 Les auto-encodeurs	2
2.1 L'intuition	2
2.2 L'encodage et le décodage	5
3 Problèmes et limitations liés aux auto-encodeurs	5
3.1 Problème de la couche cachée trop grande	5
3.2 Limitation des auto-encodeurs pour la génération de contenu	6
4 Auto-encodeur Variationnel (VAE)	7
4.1 Définition et représentation des auto-encodeurs Variationnels	7
4.2 Détails mathématiques de l'auto-encodeur variationnel	8

1 Réseaux de neurones artificiels

En 1943, McCulloch, un neurophysiologiste, et Pitts, un logicien, ont proposé la notion de neurones formels. Ces derniers étaient censés imiter les neurones biologiques et être capables de mémoriser des fonctions booléennes simples. Les réseaux de neurones artificiels conçus à partir de ce type de neurones furent inspirés du système nerveux. Ces réseaux doivent être capables d'apprendre, de mémoriser, de généraliser de l'information dans les connexions entre les neurones et de traiter de l'information incomplète. Beaucoup de travaux ont eu lieu sur ce sujet et de nos jours nous avons des modèles performants faisant des réseaux neuronaux un des modèles de calcul les plus prisés en intelligence artificielle.

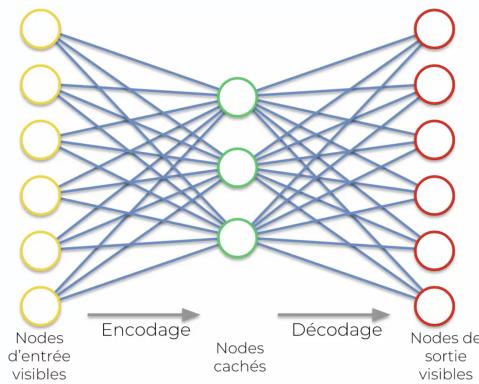


Ce qui nous intéressera dans ce projet, c'est l'étude et la mise en pratique d'un réseau de neurone particulier appelé Auto-encodeurs.

2 Les auto-encodeurs

2.1 L'intuition

Un auto-encodeur, ou auto-associateur est un réseau de neurones artificiels utilisé pour l'apprentissage non supervisé de caractéristiques discriminantes. L'objectif d'un auto-encodeur est d'apprendre une représentation (encodage) d'un ensemble de données, généralement dans le but de réduire la dimension de cet ensemble.



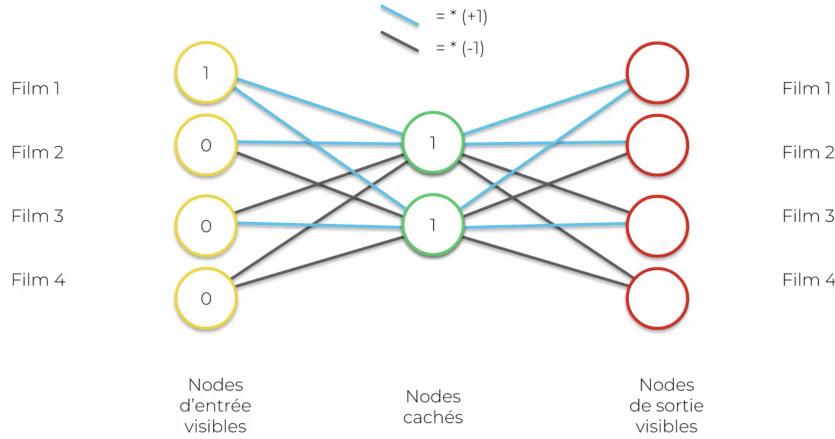
Dans le réseau ci-dessus, les noeuds d'entrées correspondent à une donnée d'entrée constituée de 6 variables. On voit qu'on a autant d'entrées que de sorties, et chaque sortie correspond

à un décodage d'une entrée qui était codée. Il y a deux grandes étapes dans la construction d'un auto-encodeur : L'encodage et le décodage.

L'intuition derrière ces deux étapes est dû à l'explosion de la dimension d'entrée qui peut devenir trop grand pour l'entraîner. Ce qui fait qu'on l'encode en résumant toute l'information sur une dimension plus petite (phase d'encodage qui permet de détecter des features) et de décoder les éléments encodés pour essayer de reconstruire la valeur d'entrée. Plusieurs exemples d'applications des auto-encodeurs sont connus de nos jours comme :

La classification d'image: Une image représente plusieurs milliers de pixels, de ce fait pour classifier par exemple une image de chien chat on peut utiliser les auto-encodeurs pour transformer notre image (matrice de pixel) en un vecteur ligne dont chaque variable représentera un pixel, puis on encode ce vecteur en détectant des features (des pixels propres à un chien ou un chat), ce qui réduit la dimension de l'entrée, une fois toute l'information résumée sur les neurones, on passe à l'étape de décodage qui permettra grâce aux features détectés de dire avec une certaine précision si on a une image de chien ou de chat.

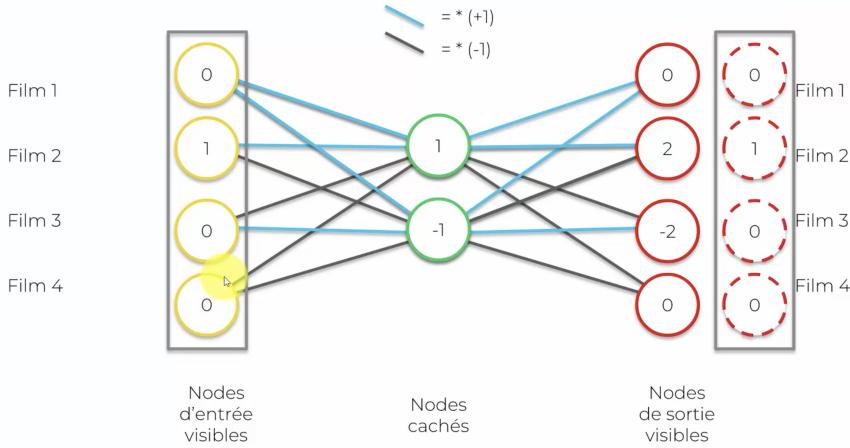
Système de recommandation Pour avoir une idée de cette application on va considérer un utilisateur qui regarde plusieurs films avec certains qu'il a aimé et d'autres non et on va essayer de deviner les films qu'il a aimé grâce à un réseau d'auto-encodage. Donc notre donnée d'entrée va alors correspondre à un de ces 5 films regardés, par l'utilisateur avec la valeur (1) qui signifie qu'il a aimé le film et (0) qui signifie qu'il n'a pas aimé. Notre réseau est représenté ci après.



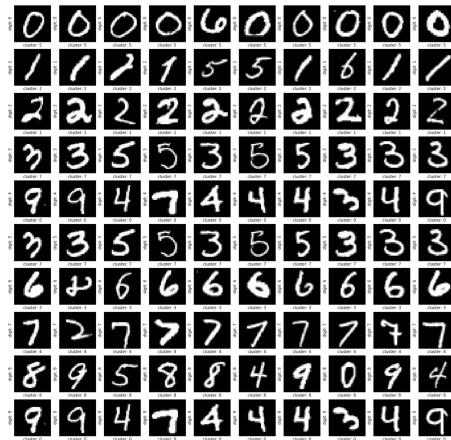
En choisissant au hasard nos poids (lignes bleues(+1) et noires(-1)) on construit les deux noeuds cachés en faisant une pondération entre les poids et les valeurs d'entrées (par exemple $\text{NoeudEnHaut} = ((+1)) + ((+1)) + ((-1)) + ((-1)) = 1$).

On obtient les même valeurs de sortie en effectuant la même opération, ce qui donne la figure suivante et après on applique à la fonction de sortie la valeur softmax (si la valeur de sortie ≥ 0.5 alors elle passe à 1 sinon elle passe à 0). On voit que l'on obtient exactement la même valeur de sortie. Ainsi on est arrivé à deviner la valeur d'entrée. Cet exemple est utilisé pour savoir si un client va aimer ou pas un film en lui recommandant des films qu'il pourrait

aimer.



K-means encodeurs : Supposons qu'on veuille classifier des images correspondant à des chiffres de la table MNIST, donc un moyen d'utiliser les auto-encodeurs est d'encoder chaque image brute comme on l'a vu dans la classification d'image ; au lieu de décoder l'image, on va essayer grâce aux features qu'on a créé dans l'encodage de classifier(avec le k-means) cette image en minimisant une fonction de perte qui calcule la distance entre la quantité d'informations perdues entre la représentation compressée de nos données et la représentation décompressée. Ceci permettra de classifier chaque image codée sur un ensemble d'image de 0 à 9.



La figure suivante représente des images de la base de donnée MNIST qui peuvent être utilisées pour tester l'algorithme de k-means auto-encoders. Chaque image est transformée en vecteurs de pixels avant d'être encodée en vue de la réduction du vecteur puis avec les features formées, on essaye de la classifier entre 0 et 9.

2.2 L'encodage et le décodage

Un auto-encodeur se compose essentiellement de deux parties comme on l'a vu toute à l'heure : l'encodage et le décodage qui peuvent être définies comme des transitions. φ et ψ peuvent être définies de la façon suivante :

$$\begin{aligned}\varphi &: \Omega \rightarrow \Sigma \\ \psi &: \Sigma \rightarrow \Omega \\ \psi, \varphi &= \underset{\psi, \varphi}{\operatorname{argmin}} \|X - (\psi \circ \varphi)X\|^2\end{aligned}$$

Dans le cas où il y a qu'une seule couche cachée, l'étape d'encodage prend l'entrée $x \in \mathbb{R}^d = \Omega$ et l'associe à $z \in \mathbb{R}^p = \Sigma$: $z = \sigma(Wx + b)$.

- z : code, variables latentes ou représentation latent,
- σ : une fonction d'activation (sigmoïde, RELU, redresseur etc)
- W : une matrice de poids
- b : un vecteur biais associé à l'encodage

Ensuite l'étape de décodage z associe à la reconstruction x' de forme identique à x :

$$x' = \sigma(W'z + b')$$

Un auto-encodeur est aussi entraîné pour minimiser l'erreur de reconstruction (erreur quadratique) :

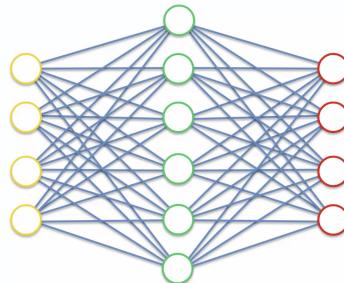
$$\mathcal{L}(x, x') = \|x - x'\|^2 = \|x - \sigma(W'(\sigma(Wx + b)) + b')\|^2$$

Si l'espace caractéristique Σ possède une dimension inférieure à l'espace d'entrée Ω , on peut considérer φ comme étant une représentation compressée de x .

3 Problèmes et limitations liés aux auto-encodeurs

3.1 Problème de la couche cachée trop grande

Considérons la représentation suivante :

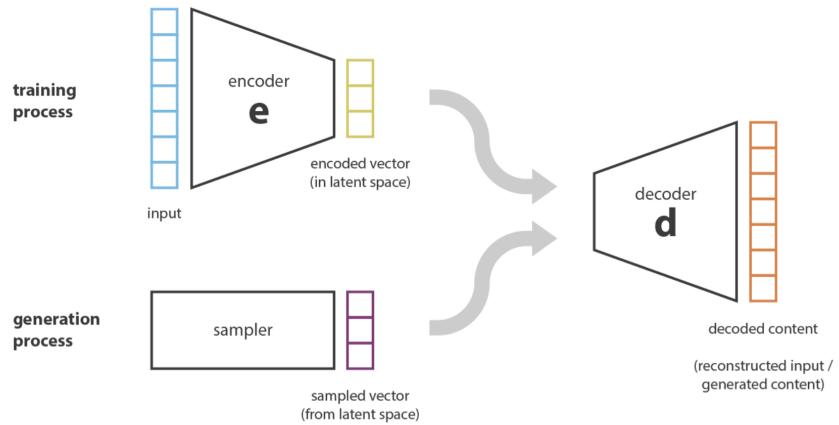


Dans ce réseau on voit que le nombre de neurones de la couche cachée (6 neurones) est plus grande que les variables de la couche d'entrée (4 variables). Maintenant un exemple simple pour avoir la couche de sortie, est de remplacer les 4 premières neurones de la couche cachée par les 4 variables de la couche d'entrée (possible vu que la couche cachée est construite pour restituer au mieux toute la couche d'entrée) et on ignore les deux dernières neurones de la couche cachée, ce qui permet en sortie d'avoir exactement une couche identique à la couche d'entrée. Ce qui était le but , mais dans ce cas le réseau souffrira d'un problème de sur-apprentissage (overfitting), ce qui veut dire que le réseau apprend tellement bien de nos données que si on lui donne une nouvelle donnée, il risque de donner de mauvaises prédictions.

3.2 Limitation des auto-encodeurs pour la génération de contenu

À ce stade, une question qui vient naturellement à l'esprit est : "Quel est le lien entre les auto-codeurs et la génération de contenu". En effet, une fois que l'auto-encodeur a été formé, nous disposons à la fois d'un codeur et d'un décodeur, mais toujours pas de véritable moyen de produire un nouveau contenu. À première vue, on pourrait être tenté de penser que, si l'espace latent est suffisamment régulier (bien organisé par l'encodeur pendant le processus de formation), nous pourrions prendre un point au hasard de cet espace latent et le décoder pour obtenir un nouveau contenu. Le décodeur se comporterait alors plus ou moins comme le générateur d'un réseau génératif.

Nous pouvons générer de nouvelles données en décodant des points qui sont prélevés au hasard dans l'espace latent (comme le montre la figure suivante). La qualité et la pertinence des données générées dépendent de la régularité de l'espace latent.



Cependant, la régularité de l'espace latent pour les auto-codeurs est un point difficile qui dépend de la distribution des données dans l'espace initial, de la dimension de l'espace latent et de l'architecture du codeur. Il est donc assez difficile (voire impossible) de garantir, à priori, que le codeur organisera l'espace latent d'une manière intelligente et compatible avec le processus de génération que nous venons de décrire.

En y réfléchissant une minute, ce manque de structure parmi les données encodées dans l'espace latent est assez normal. En effet, rien dans la tâche pour laquelle l'auto-encodeur est formé ne permet d'obtenir une telle organisation : l'auto-encodeur est uniquement formé pour encoder et décoder avec le moins de perte possible, quelle que soit la façon dont l'espace latent est organisé. Ainsi, si l'on n'est pas attentif à la définition de l'architecture, il est naturel que, lors de la formation, le réseau profite de toutes les possibilités de suréquipement pour accomplir sa tâche au mieux à moins de la régulariser explicitement!

Maintenant pour parer ce problème de non-régularisation des données d'entrées, nous introduisons les auto-encodeurs variationnels que l'on étudiera dans le paragraphe succinct et qu'on mettra en pratique dans notre algorithme.

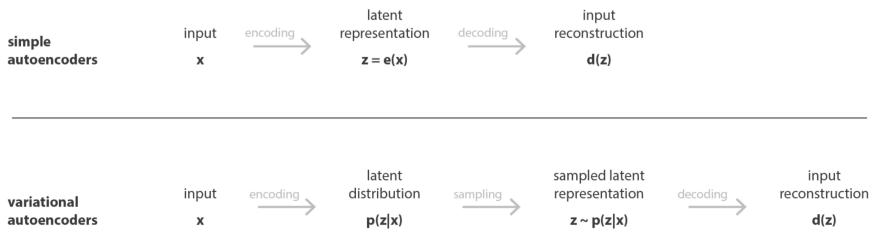
4 Auto-encodeur Variationnel (VAE)

4.1 Définition et représentation des auto-encodeurs Variationnels

Un auto-encodeur variationnel peut être défini comme étant un auto-encodeur dont l'apprentissage est régularisé pour éviter le sur-ajustement et garantir que l'espace latent a de bonnes propriétés qui permettent le processus de génération. Tout comme un auto-encodeur standard, un auto-encodeur variationnel est une architecture composée à la fois d'un codeur et d'un décodeur et qui est formée pour minimiser l'erreur de reconstruction entre les données codées-décodées et les données initiales. Cependant, afin d'introduire une certaine régularisation de l'espace latent, nous procédons à une légère modification du processus d'encodage et de décodage : au lieu d'encoder une entrée comme un point unique, nous l'encodons comme une distribution sur l'espace latent.

Le modèle est ensuite formé comme suit :

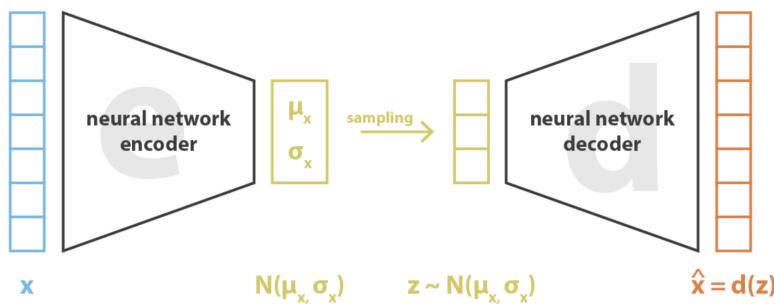
- l'entrée est encodée comme une distribution sur l'espace latent,
- un point de l'espace latent est échantillonné à partir de cette distribution,
- le point échantillonné est décodé et l'erreur de reconstruction peut être calculée,
- Enfin, l'erreur de reconstruction est rétropropagée à travers le réseau



En pratique, les distributions codées sont choisies pour être normales afin que l'encodeur puisse être entraîné à renvoyer la moyenne et la matrice de covariance qui décrivent ces

gaussiens.

La fonction de perte qui est minimisée lors de l'apprentissage d'une VAE est composée d'un "terme de reconstruction" (sur la couche finale), qui tend à rendre le schéma d'encodage et de décodage aussi performant que possible, et d'un "terme de régularisation" (sur la couche latente), qui tend à régulariser l'organisation de l'espace latent en rendant les distributions retournées par l'encodeur proches d'une distribution normale standard. Ce terme de régularisation est exprimé par la divergence de Kulback-Leibler entre la distribution rentrée et une distribution gaussienne standard.



$$\text{loss} = \|x - \hat{x}\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)] = \|x - d(z)\|^2 + \text{KL}[N(\mu_x, \sigma_x), N(0, I)]$$

4.2 Détails mathématiques de l'auto-encodeur variationnel

Nous allons utiliser une approche par inférence variationnelle en appliquant une méthode bayesienne.

Commençons par définir un modèle graphique probabiliste pour décrire nos données: x représente nos données et nous supposons que x est généré à partir d'une variable latente z (la représentation codée) qui n'est pas directement observée. Ainsi, pour chaque point de données, on suppose le processus de génération en deux étapes suivant :

- d'abord, une représentation latente z est échantillonnée à partir de la distribution antérieure $p(z)$,
- les données x sont échantillonnées à partir de la distribution de vraisemblance conditionnelle $p(x|z)$

Ainsi, nous définissons notre encodeur comme une distribution de $p(z|x)$ et notre décodeur comme une distribution de $p(x|z)$. Ce qui rend ces modèles probabiliste et non déterministe comme dans le cas des auto-encodeurs simples.

A ce stade, notre aspect de régulation latente qui nous manquait dans les auto-encodeurs apparaît naturellement ici: les représentations codées z dans l'espace latent sont en effet supposées suivre la distribution $p(z)$ antérieure. Sinon, on peut aussi rappeler

le célèbre théorème de Bayes qui fait le lien entre le $p(z)$ antérieur, la probabilité $p(x|z)$ et le $p(z|x)$ postérieur.

$$p(z|x) = \frac{p(x|z)p(z)}{\int p(x|u)p(u)du}$$

Supposons maintenant que :

$$\begin{aligned} p(z) &\equiv \mathcal{N}(0, 1) \\ p(x|z) &\equiv \mathcal{N}(f(x), cI) \quad f \in F \quad c > 0 \end{aligned}$$

On considère pour l'instant que f est bien définie. En théorie comme nous connaissons $p(z)$ et $p(x|z)$ nous pouvons à l'aide de la formule de Bayes déterminer $p(z|x)$, mais en pratique ceci est difficile en raison de l'intégrale au dénominateur et nécessite l'utilisation de technique d'approximation telle que l'inférence variationnelle.

Inférence Variationnelle : En statistique, l'inférence variationnelle est une technique permettant d'approcher des distributions complexes. L'idée est de définir une famille de distribution paramétrée (par exemple la famille des Gaussiens, dont les paramètres sont la moyenne et la covariance) et de rechercher la meilleure approximation de notre distribution cible parmi cette famille. Le meilleur élément de la famille est celui qui minimise une mesure donnée de l'erreur d'approximation (la plupart du temps la divergence de Kullback-Leibler entre l'approximation et la cible) .

Nous allons ici approximer $p(z | x)$ par une distribution gaussienne $q_x(z)$ dont la moyenne et la covariance sont définies par deux fonctions, g et h , du paramètre x . Ces deux fonctions sont censées appartenir, respectivement, aux familles de fonctions G et H qui sont censées être paramétrées. On peut ainsi désigner.

$$q_x(z) \equiv \mathcal{N}(g(x), h(x)) \quad g \in G \quad h \in H$$

Maintenant viens l'étape d'optimisation des fonctions g et h (en fait leurs paramètres) pour minimiser la divergence de Kullback-Leibler entre l'approximation et la cible $p(z | x)$. En d'autres termes, nous recherchons les fonctions g^* et h^* optimales telles que:

$$\begin{aligned} (g^*, h^*) &= \underset{(g,h) \in G \times H}{\operatorname{argmin}} \text{KL}(q_x(z), p(z|x)) \\ &= \underset{(g,h) \in G \times H}{\operatorname{argmin}} \left(\mathbb{E}_{z \sim q_x}(\log q_x(z)) - \mathbb{E}_{z \sim q_x}(\log \frac{p(x|z)p(z)}{p(x)}) \right) \\ &= \underset{(g,h) \in G \times H}{\operatorname{argmin}} (\mathbb{E}_{z \sim q_x}(\log q_x(z)) - \mathbb{E}_{z \sim q_x}(\log p(x|z)) - \mathbb{E}_{z \sim q_x}(\log p(z)) + \mathbb{E}_{z \sim q_x}(\log p(x))) \\ &= \underset{(g,h) \in G \times H}{\operatorname{argmax}} (\mathbb{E}_{z \sim q_x}(\log p(x|z)) - \text{KL}(q_x(z), p(z))) \\ &= \underset{(g,h) \in G \times H}{\operatorname{argmax}} \left(\mathbb{E}_{z \sim q_x}(-\frac{\|x - f(x)\|^2}{2c}) - \text{KL}(q_x(z), p(z)) \right) \quad \text{car } p(x|z) \equiv \mathcal{N}(f(x), cI) \end{aligned}$$

Jusqu'à présent, nous avons supposé que la fonction f était connue et fixe et nous avons montré que, sous de telles hypothèses, nous pouvons approximer le $p(z|x)$ postérieur en utilisant la technique d'inférence variationnelle. Toutefois, dans la pratique, cette fonction f , qui définit le décodeur, n'est pas connue et doit également être choisie. Pour ce faire, rappelons que notre objectif initial est de trouver un schéma d'encodage-décodage performant dont l'espace latent est suffisamment régulier pour être utilisé à des fins génératives. Si la régularité est principalement régie par la distribution préalable supposée sur l'espace latent, la performance du schéma global d'encodage-décodage dépend fortement du choix de la fonction f . En effet, comme $p(z|x)$ peut être approchée (par inférence variationnelle) de $p(z)$ et $p(x|z)$ et comme $p(z)$ est un gaussien standard simple, les deux seuls leviers dont nous disposons dans notre modèle pour faire des optimisations sont le paramètre c (qui définit la variance de la vraisemblance) et la fonction f (qui définit la moyenne de la vraisemblance).

nous recherchons un schéma de codage-décodage aussi efficace que possible et, ensuite, nous voulons choisir la fonction f qui maximise la log-vraisemblance attendue de x étant donné z lorsque z est échantillonné à partir de $q_x^*(z)$. Ainsi, nous recherchons le f^* optimal tel que :

$$\begin{aligned} f^* &= \underset{f \in F}{\operatorname{argmax}} (\mathbb{E}_{z \sim q^*} (\log p(x|z))) \\ &= \underset{f \in F}{\operatorname{argmax}} \left(\mathbb{E}_{z \sim q^*} \left(-\frac{\|x - f(x)\|^2}{2c} \right) \right) \end{aligned}$$

où $q_x^*(z)$ dépend de la fonction f . Ainsi en rassemblant les éléments, nous cherchons les valeurs optimales f^* , g^* et h^* telles que :

$$(f^*, g^*, h^*) = \underset{(f,g,h) \in F \times G \times H}{\operatorname{argmax}} \left(\mathbb{E}_{z \sim q_x} \left(-\frac{\|x - f(x)\|^2}{2c} \right) - KL(q_x(z), p(z)) \right)$$

Nous pouvons identifier dans cette fonction objective les éléments introduits dans la description intuitive des VAE donnée dans la section précédente : l'erreur de reconstruction entre x et $f(z)$ et le terme de régularisation donné par la divergence KL entre $q_x(z)$ et $p(z)$ (qui est un gaussien standard). On peut également noter la constante c qui régit l'équilibre entre les deux termes précédents. Plus c est élevé, plus nous supposons une variance élevée autour de $f(z)$ pour le décodeur probabiliste de notre modèle et, par conséquent, plus nous privilégions le terme de régularisation par rapport au terme de reconstruction (et l'inverse si c est faible).