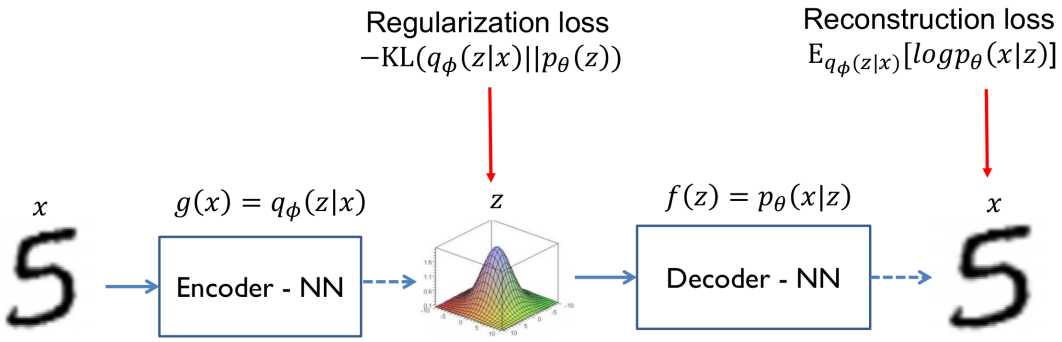
# TME M2DAC RLD : Variational Auto-Encoder

N. Thome, B. Piwowarski

2022-2023

Un auto-encodeur variationnel est un modèle génératif, i.e. visant à modéliser la distribution des données. L'hypothèse fondamentale des VAEs est de considérer que la distribution des données $p_\theta$ peut être obtenue par la transformation d'une variable aléatoire latente $z$ suivant une distribution *simple* $p(z)$ (comme une gaussienne multivariée) à l'aide d'une fonction (complexe) $f$ - le décodeur. Inversement, à partir d'une donnée $x$, il est possible d'obtenir la distribution simple latente $q_\phi(z|x)$ à l'aide d'une fonction $g(x)$ - l'encodeur. Le fonctionnement est illustré par le schéma ci-dessous tiré du cours.




Pour une donnée d'entraînement $x_i$, on peut montrer que la log-vraisemblance de $x_i$ s'écrit :

$$log(P_\theta(x_i)) = KL(q_\phi(z|x)||p(z|x)) + [\mathbb{E}_{z\sim q_{\phi(z|x)}} log(p_\theta(x|z) - KL(q_\phi(z|x)||p(z))]$$

La loss de l'auto-encodeur variationnel s'écrit donc :

$$\mathcal{L}_{VI}(x, \theta, \phi) = KL(q_\phi(z|x)||p(z)) - \mathbb{E}_{z\sim q_{\phi(z|x)}} log(p_\theta(x|z)$$

Quelques points importants :

— l'encodeur ne permet pas d'obtenir directement un vecteur $z$ dans l'état latent, mais les paramètres de la distribution $q_\phi(z|x)$ : $z$ est obtenu en tirant au hasard un vecteur selon cette distribution ;

— de même, le décodeur ne permet pas d'obtenir directement une donnée, mais une distribution de probabilités. Dans le cas d'une image en noir et blanc par exemple, on peut modéliser chaque pixel par une bernoulli ; le décodeur nous fournira les paramètres de chaque bernoulli. En échantillonnant selon ces distributions, on obtient une image.

— plutôt que de modéliser directement la distribution de probabilités, l'encodeur (et le décodeur) infère les paramètres des lois de probabilités : dans le cas d'une gaussienne, l'encodeur infère deux vecteurs, le vecteur des moyennes et le vecteur des variances ; dans le cas d'une image, le décodeur infère le vecteur des paramètres des bernoulli (compris entre 0 et 1, de taille la taille de l'image).

Les fonctions $f$ et $g$ sont des réseaux de neurones. L'apprentissage est semblable à la procédure pour un auto-encodeur (d'où le nom) : $f$ et $g$ sont apprises de façon à reconstruire au mieux les données passées en entrée. Par ailleurs, il faut contraindre la fonction $g$ (la distribution $q_\theta(z|x)$) à ressembler à la distribution $p(z)$ fixée comme a priori. La fonction de coût fait intervenir ainsi deux termes :

— le terme de reconstruction, une log-vraissemblance : $-\mathbb{E}_{z\sim q_{\phi(z|x)}} log(p_\theta(x|z)$

— une pénalisation de $q_\theta$ à l'aide d'une KL-divergence : $KL(q_\phi(z|x)||p(z))$

## Expérimentations

On considère dans la suite que la probabilité a priori $p(z)$ est une loi normale de covariance diagonale. Afin d'effectuer une descente de gradient pour apprendre les réseaux, on utilise un *reparametrization trick* pour se débarrasser du tirage stochastique de l'encodeur : une variable aléatoire $z$ suivant une loi normale $\mathcal{N}(\mu, \sigma)$ peut être obtenue par $z = \mu + \sigma \odot \epsilon$ avec $\epsilon \sim \mathcal{N}(0, 1)$. Ainsi ce n'est plus directement $z$ qui est stochastique, mais le bruit $\epsilon$ injecté, non paramétrisé : cela nous permet donc d'effectuer une rétro-propagation usuelle sur les modules qui produisent $\mu$ et $\sigma$.

Pour des raisons de stabilité, plutôt que d'inférer le vecteur $\sigma$, on infère $logsigma = log(\sigma^2)$. Dans ce cas, la KL-divergence en $d$ dimension s'écrit : $KL(q_\phi(z|x)||p(z)) = -\frac{1}{2}\sum_{i=1}^{d}(1 + logsigma_i - \mu_i^2 - e^{logsigma_i})$.

Vous expérimenterez sur le jeu de données MNIST. N'oubliez pas de transformer les images en niveau de gris et des valeurs entre 0 et 1. Vous pouvez tester un encodeur avec une architecture : `linéaire` $\rightarrow$ `ReLU` $\rightarrow$ `linéaire`, la dernière couche produisant deux vecteurs, l'un pour $\mu$ l'autre pour $\sigma$ ; et un décodeur : `linéaire` $\rightarrow$ `ReLU` $\rightarrow$ `linéaire` $\rightarrow$ `sigmoïde`. Vous pouvez également tester une architecture avec convolution/déconvolution. Etudiez l'influence de la dimension de l'espace latent sur la reconstruction.

Vous pouvez visualiser au fur et à mesure des itérations les images originales et reconstruites ( en apprentissage et en test). Vous pouvez échantillonner des points dans l'espace latent selon la loi a priori $p(z)$ et appliquer le décodeur $p_\theta(x|z)$ pour observer l'évolution des images générées.

Vous pouvez également échantillonner des points dans l'espace latent pour observer une signification possible des dimensions (particulièrement si l'espace est de dimension 2). Vous pouvez également tracer les embeddings dans l'espace latent des images en fonction de leur label (en 2D par exemple les deux premières coordonnées du vecteur de paramètre $\mu$).