DEEP LEARNING

Inférence Variationnel

Jean Marius KOMBOU

Université de Lyon 2

29 mai 2024

backgrounds

Vous retouverez dans ce rapport une explication de l'algorithme VAEB, constitué principalement de 3 classes: Encodeur, Decodeur et VAE. Egalement des étapes de calcul de la fonction de perte, de minimisation de la fonction de perte totale à l'aide d'algorithmes d'optimisation, et de chargement de données. Nous allons utiliser Flax sur JAX, qui est une bibliothèque de réseaux neuronaux développée par Google.

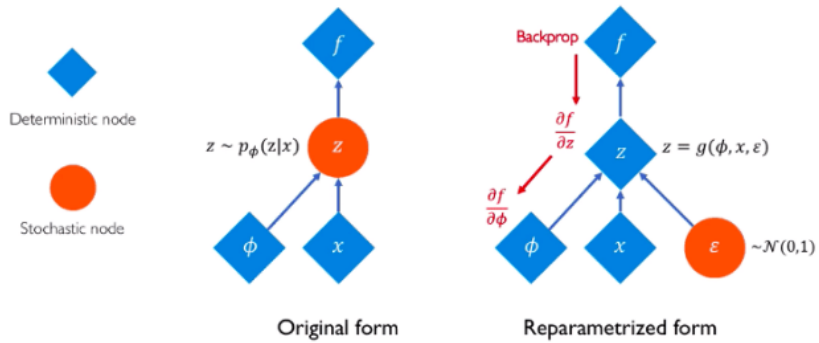
Pour l'encodeur, une simple couche linéaire suivie d'une activation RELU. La sortie de la couche sera à la fois la moyenne et l'écart type de la distribution de probabilité.
On utilise le package *nn.linen* de Flax qui contient la plupart des couches et opérations d'apprentissage profond telles que Dense, relu.

Le Decodeur sera fait de deux couche linéaires qui reçoivent la représentation latente z et aura en sortie la reconstruction de la donnée d'entrée.

Frame Title

Pour combiner l'Encodeur et et le Decodeur on aura la classe VAE, ici on implémente l'astuce de reparamétrisation.

L'image ci-dessous explique l'astuce de reparametrisation:



Pour tirer pleinement parti des capacités de JAX, nous devons ajouter la vectorisation automatique et la compilation XLA à notre code. Cela peut être fait facilement avec l'aide des annotations *vmap* et *jit*. De plus, nous devons activer la différentiation automatique, ce qui peut être accompli avec la transformation *grad fn*. Nous utilisons le package *flax.optim* pour les algorithmes d'optimisation.
Les VAE sont entraînés en maximisant la borne inférieure de l'évidence, connue sous le nom d'ELBO

$$L_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - \mathrm{KL}(q_\phi(z|x)\|p_\theta(z)) \qquad (1)$$

Ici, on exécute l'ensemble de la boucle d'entraînement qui exécutera de manière itérative la fonction *trainstep*. Le modèle doit être initialisé avant l'entraînement, ce qui est fait par la fonction d'initialisation *init*. On utilise *jax.deviceput* pour transférer l'optimiseur dans la mémoire du GPU

Flax n'inclut pas encore de packages de manipulation de données en dehors des opérations de base de jax.numpy. Pour l'instant, la meilleure solution est d'emprunter des packages à d'autres frameworks tels que TensorFlow Datasets (tfds) ou Torchvision. On utilise les données MNIST binarisées pour l'entraînement.

Merci!