Rapport d'analyse

Benotsmane Ismat

November 2020

Table des matières

1 Descente de gradient

Ici nous voulons comparer les deux méthodes de descente de gradient, la descente en *stochastique* et en *Mini-batch*.

Ces descentes de gradient ont été effectué sur le dataset **Boston Housing**. Ce dataset a été normalisé.

1.1 Résultat

Pour les 2 descentes de gradient nous avons utilisé 1000 itérations et 1e-4 en pas de gradient. La taille des mini batch est de 32. A noter que la taille des mini-batch est un hyper-parametre tout comme le nombre d'itérations et le pas de gradient.

La courbe rose concerne la descente en stochastique, quant à la verte c'est la descente en mini-batch.
On remarquera que en stochastique on converge plus rapidement vers l'optimum mais cet optimum est local dû au pas de gradient qui nous fait osciller autour. En baissant le pas de gradient on convergera vers une meilleur solution.
Tandis qu'en mini-batch en converge plus lentement vers un meilleur optimum.

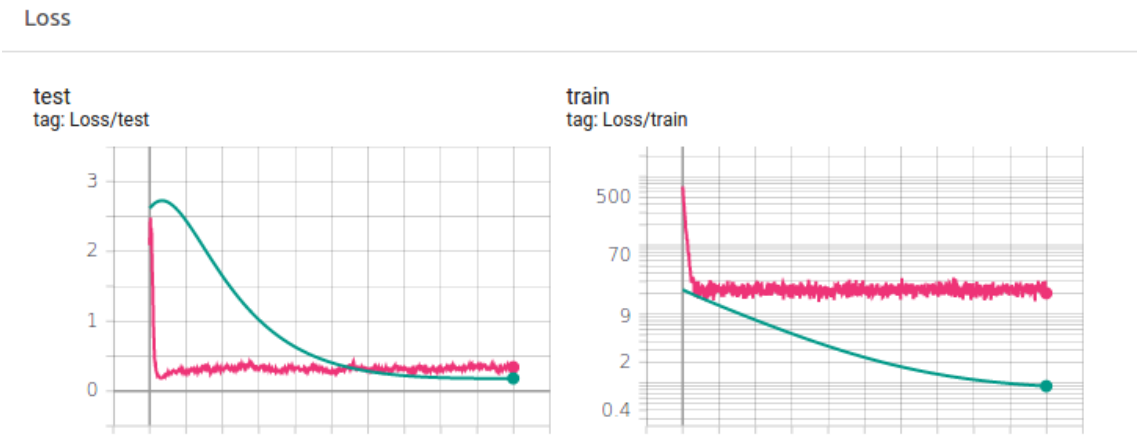

FIGURE 1 – Descente stochastique VS descente mini-batch

2 Autoencodeur

2.1 Definition

L'autoencodur est un reseau de nerones composé d'une ou plusieurs couches cachées. Il prend en entré un image, qu'il réduit dans un espace de dimension inferieur, et en sortie il retourne une image qui doit etre la plus fidéle possible

de l'image passée en entré.
Ce reseau de neurones a pour but de réduire la dimension de l'image dans un espace beaucoup plus réduit(**encodage**). Cette encodage est important pour pouvoir traiter une image dans des taches comme la classification dans un espace de dimension raisonnable.
La phase d'apprentissage de l'autoencodeur s'appuit sur l'image renvoyée en sortie(**decodage**) pour corriger les parametres du modele.

2.2 Les données

Les données utilisées pour le modéle autoencodeur sont issues du dataset d'image MNIST. Une image est une matrice 28×28 que nous transformant en un vecteur de taille 784.

2.3 Le modele

Le modéle est un reseau de neurones composé de :
— Une couche *input* : on envoie une image
— Une couche *caché* : Une fonction **lineaire** suivie d'une fonction d'activation **Relu**.
— Une couche *output* : On applique une fonction **lineaire** suivie d'une fonction d'activation **sigmoid**.

2.4 Apprentissage

Pour l'apprentissage on utilise une descente de gradient avec deux fonctions loss : MSE et Binary Cross-Entropy. On utilise l'optimisation SGD en back propagation.
On va utiliser une descente de gradient en $Mini - batch$ avec des batchs de 32 images.

2.5 Resultat

Dans les résultats présentés ici sauf si précisé autrement, nous avons fait un encodage dans un espace latent de dimension 100.
Ajoutons égalemment que le modéle est entrainé sur un dataset train, et qu'il testé sur un dataset test different du train.

Dans la figure 2 on obsèrve la performance de notre modéle en train et en test. Ici nous avons utilisé MSE comme fonction de loss pour les 2 courbes.
La courbe gris est une descente de gradient avec un pas de gradient de 1 et 100 itérations. Quant à la courbe orange c'est une descente de gradient avec un pas de gradient de 0.5 et 500 itérations.

Notons que pour les deux courbes le modéle se comporte bien en apprentissage et en test. La courbe de test suit la courbe de train, cela est un indicateur que notre modéle s'améliore en apprentissage sans être dans le sur-apprentisage.

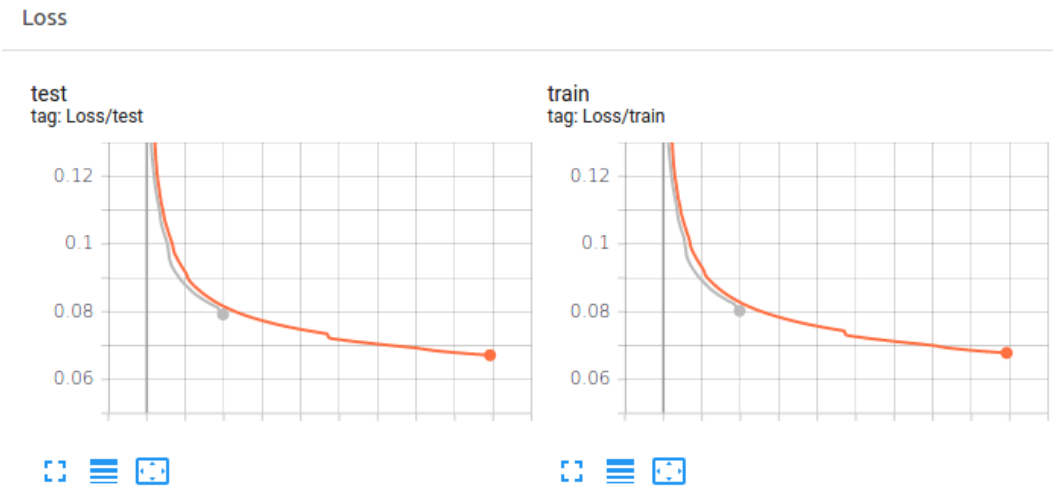

FIGURE 2 – Comparaison resultat de la MSE loss train/test

On peut observer égalemment que notre modéle ne converge pas encore et peut être ameliorer avec plus d'itérations.

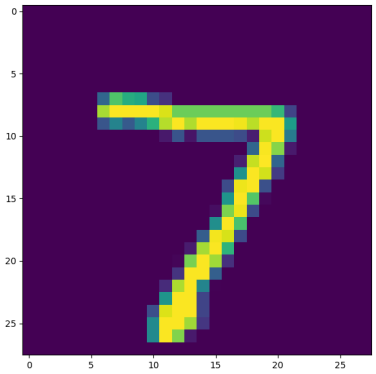

FIGURE 3 – Image original

La figure 3 est l'exemple d'une image des données test qu'on va essayer d'encoder puis decoder.

La figure 4 est le resultat de l'auto-encodage sur l'image à la figure 3 avec le modéle appris sur 100 itérations et un pas de gradient de 1.

La figure 5 est le resultat de l'auto-encodage sur l'image à la figure 3 avec le modéle appris sur 500 itérations et un pas de gradient de 0.5.

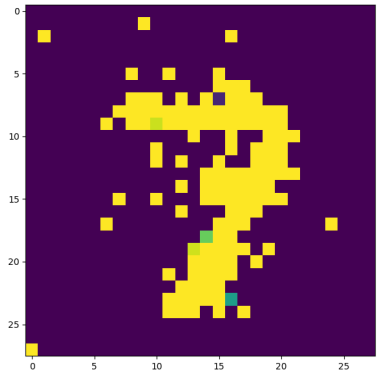

FIGURE 4 – Autoencodage avec MSE : 100 iterations et eps=1

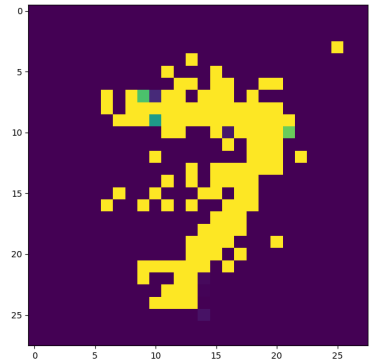

FIGURE 5 – Autoencodage avec MSE : 500 iterations et eps=0.5

On remarque que le resultat de la figure 5 est meilleur que la figure 4 car on a moins de bruit(point éloigné de la forme originale), et la forme est plus fine.

La figure 6 décrit la performance du modéle en train et en test avec Binary Cross-Entropy comme fonction de loss pour les 2 courbes.
La courbe bleu est la loss de la descente de gradient avec un pas de gradient de 1 et 100 itérations. Quant à la courbe orange, suivi de la marron puis de la bleue clair, c'est la loss de la descente de gradient avec un pas de gradient de 0.5 et 500 itérations. Cette derniére a été calculer grace a des checkpoints.

Les 2 modéles ici ne sont pas également en sur-apprentissage et peuvent être encore améliorer.

Les resultats de la figure 7 et 8 sont issues des modéles corrigés pas une loss Binary Cross-entropy. 8 nous montre qu'avec plus d'itérations on converge vers un résultat plus précis.

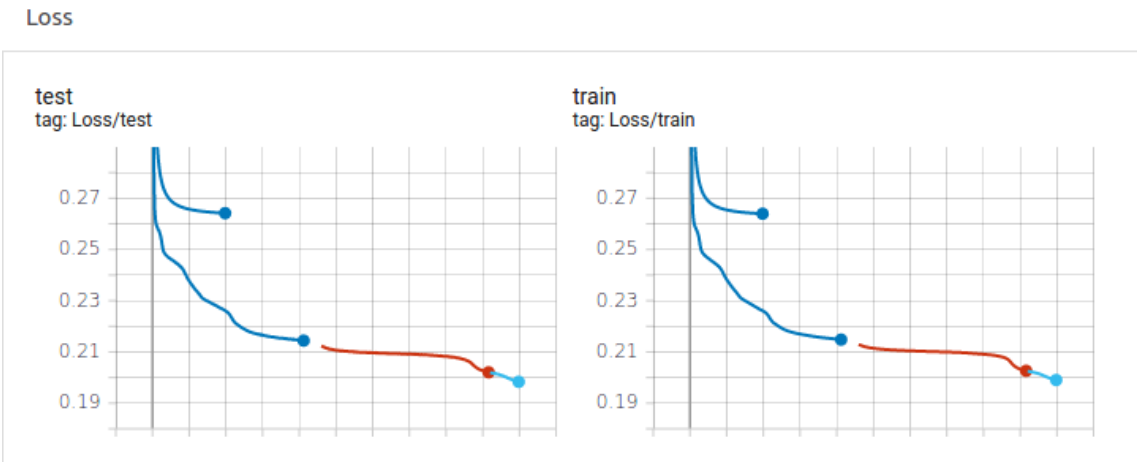

FIGURE 6 – Comparaison resultat de la BCE loss train/test

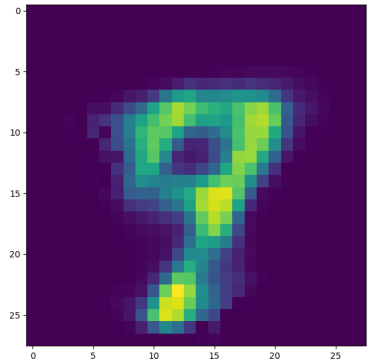

FIGURE 7 – Autoencodage avec BCE : 100 iterations et eps=1

Globalement les modéles avec une fonction de coût Binary Cross-Entropy sont plus de proche en reproduction de l'image d'origine que les modéles utilisant une MSE.
Cela peut s'expliquer par le fait que l'application d'une MSE sur une fonction sigmoid on peut se retrouver dans un probleme non convexe à resoudre, et dans ce cas-là on se retrouve avec un minimum local. Or avec une Cross Entropy on est sûr de se retrouver dans un problème convexe avec un minimum global.

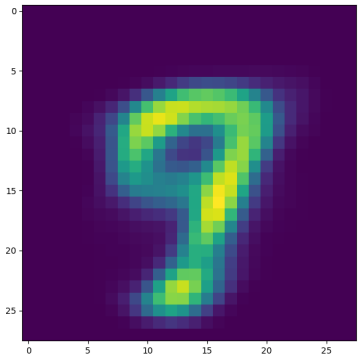

FIGURE 8 – Autoencodage avec BCE : 500 iterations et eps=0.5

2.6 Optimisation Adam

Adam est un algorithme d'optimisation de la descente de gradient appliqué lors de la back propagation. Il adapte son pas de gradient en fonction de chaque parametre w_j du vecteur $W \in R^{100}$ en compte la moyenne et la variance des mises à jours précedentes.

Comme on le voit à la figure 9, les resultats sont bien meilleur sur la BCE Loss avec une optimisation Adam qu'avec une optimisation SGD. Nous avons itéré sur 500 et un pas de gradient de 0.5 pour SGD et 1e-3 pour Adam.

La diffrence se voit directement sur la figure 10 où le rendu est largement meilleur que précedemment.

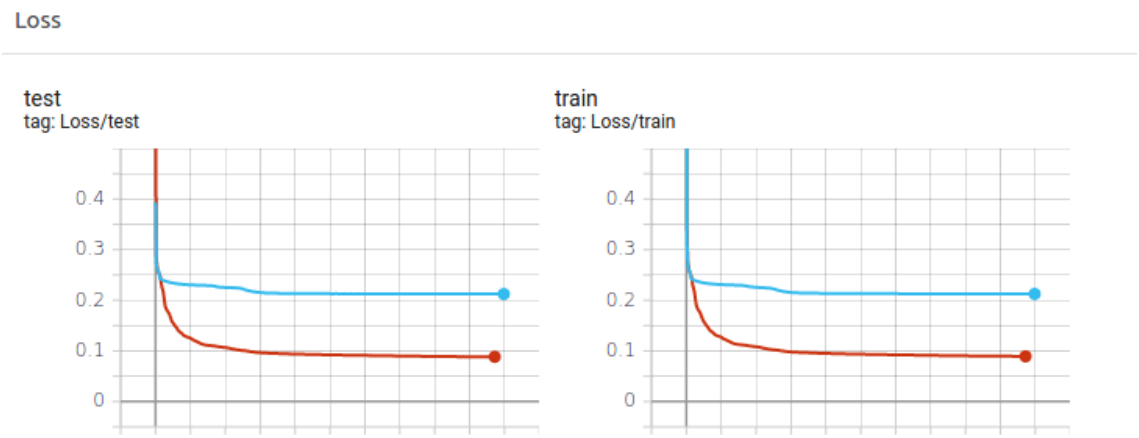

FIGURE 9 – BCE Loss SGD vs ADAM

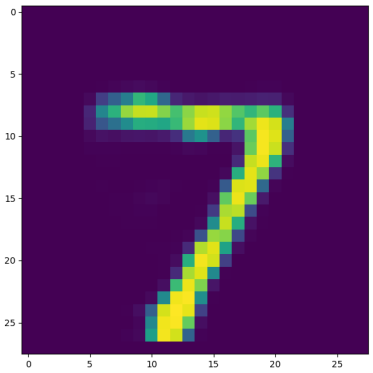

FIGURE 10 – Autoencodage avec BCE Loss par optimisation Adam