Année 2021-2022

TP : Restricted Bolzmann Machines

Réalisé par:
Yoldoz TABEI

Table des matières

Table des figures

1 Construction d'un RBM et test sur Binary AlphaDigits

1.1 Construction d'un RBM

— lire alpha digit : permet de récupérer les données sous forme matricielle.

— init_RBM : permet de construire et d'initialiser les poids et les biais d'un RBM.

— entree_sortie_RBM : qui prend en argument une structure RBM et des données d'entrée et qui retourne la valeur des unit´es de sortie calculées à partir de la fonction sigmoide.

— sortie_entree_RBM : qui prend en argument un RBM, des donn´ees de sortie et qui retourne la valeur des unit´es d'entrée à partir de la fonction sigmoide.

— train_RBM : permettant d'apprendre de manière non supervisée un RBM par l'algorithme Contrastive-Divergence-1.

— generer_image_RBM : permet de générer des échantillons suivant un RBM.

1.2 Test sur Binary AlphaDigits

1.2.1 Hyperparamètres

Afin de faire une analyse des erreurs, on fait varier les hyperparamètres (nombre d'unités cachées q, learning rate espsilon, et le batch_size) afin de trouver la meilleure combinaison possible.

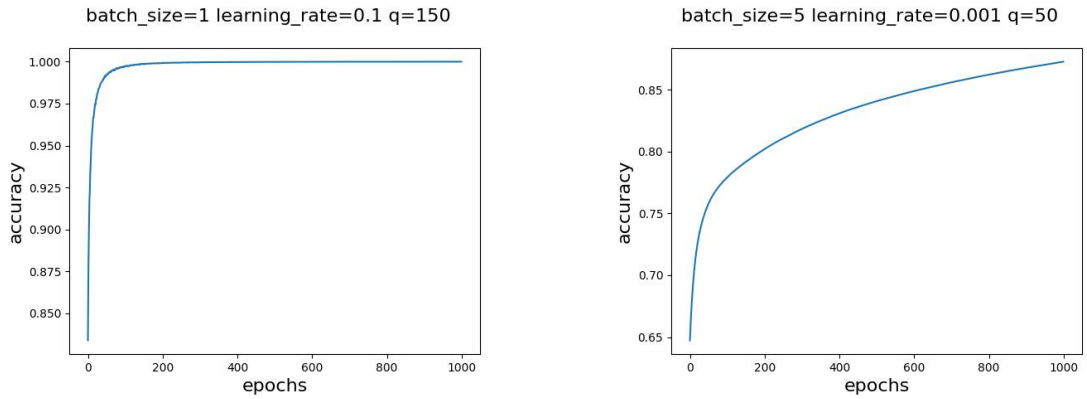


FIGURE 1 – Exemples de courbes de précision en variant les hyperparamètres

Après avoir tracé plusieurs courbes de précision, on trouve les meilleurs résultats lorsqu'on choisit :

— BATCH_SIZE = 1

— EPSILON = 0.1

— q = 150

On s'intéressera à cette étape à l'influence de la variation de chaque paramètre sur les résultats.

1. BATCH_SIZE : On varie le BATCH_SIZE en gardant EPSILON = 0.1 et q = 150.

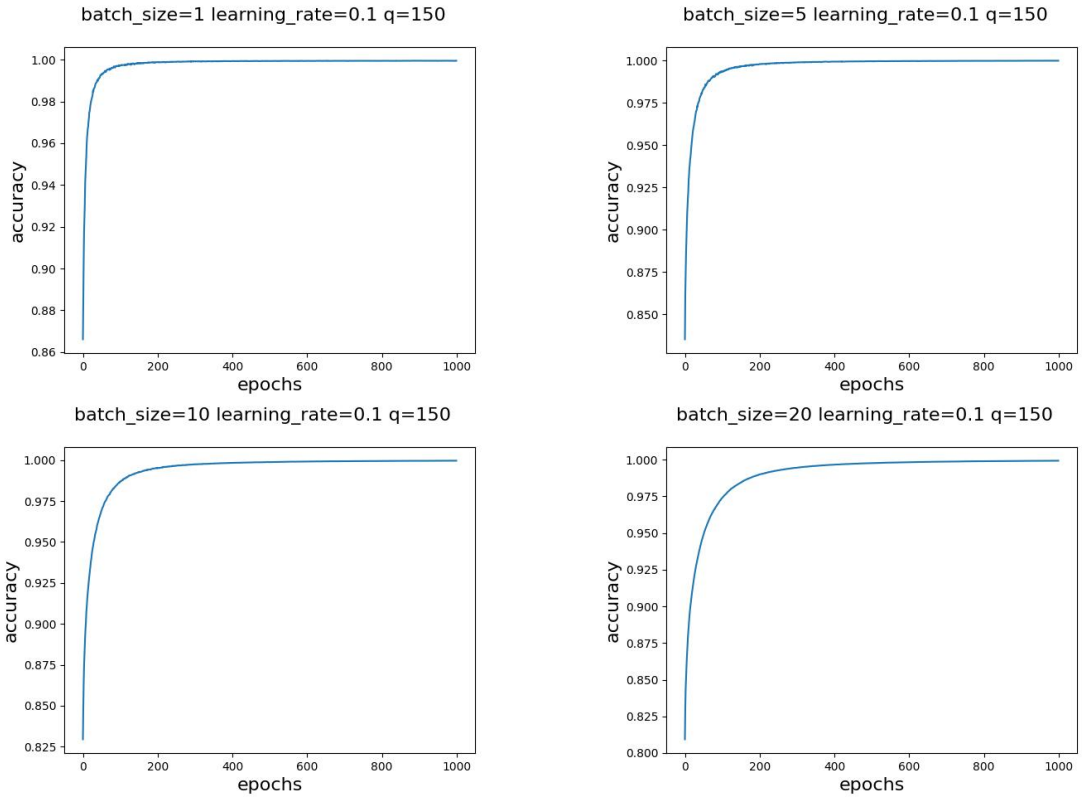


FIGURE 2 – Variation du batch_size

D'après les figures, on remarque que l'effet du batch_size est négligeable vu que la courbe reste presque la même.
On choisit alors un batch_size grand afin de gagner du temps en calcul, soit un batch_size = 20.

2. Learning rate EPSILON : On varie EPSILON en gardant BATCH_SIZE = 1 et q = 150

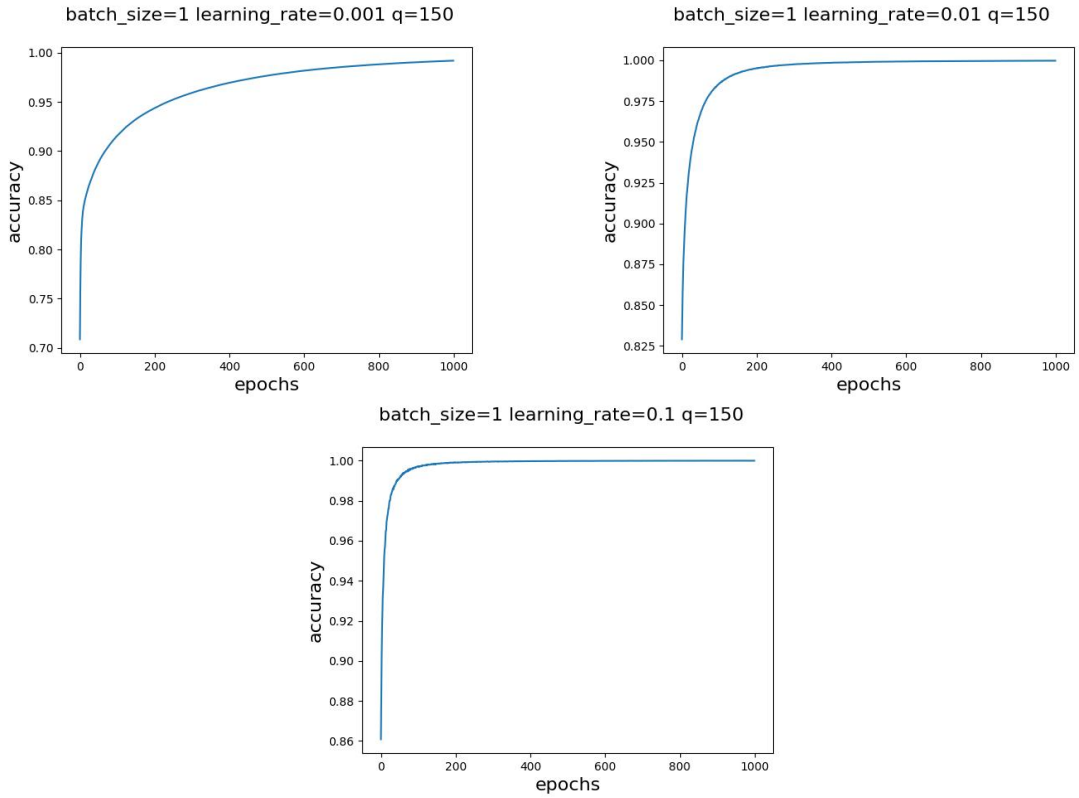


FIGURE 3 – Variation du taux d'apprentissage

On remarque que plus le taux d'apprentissage diminue, plus la courbe devient plus lente.

3. Unités cachées q : On varie q en gardant BATCH_SIZE = 1 et EPSILON = 0.1

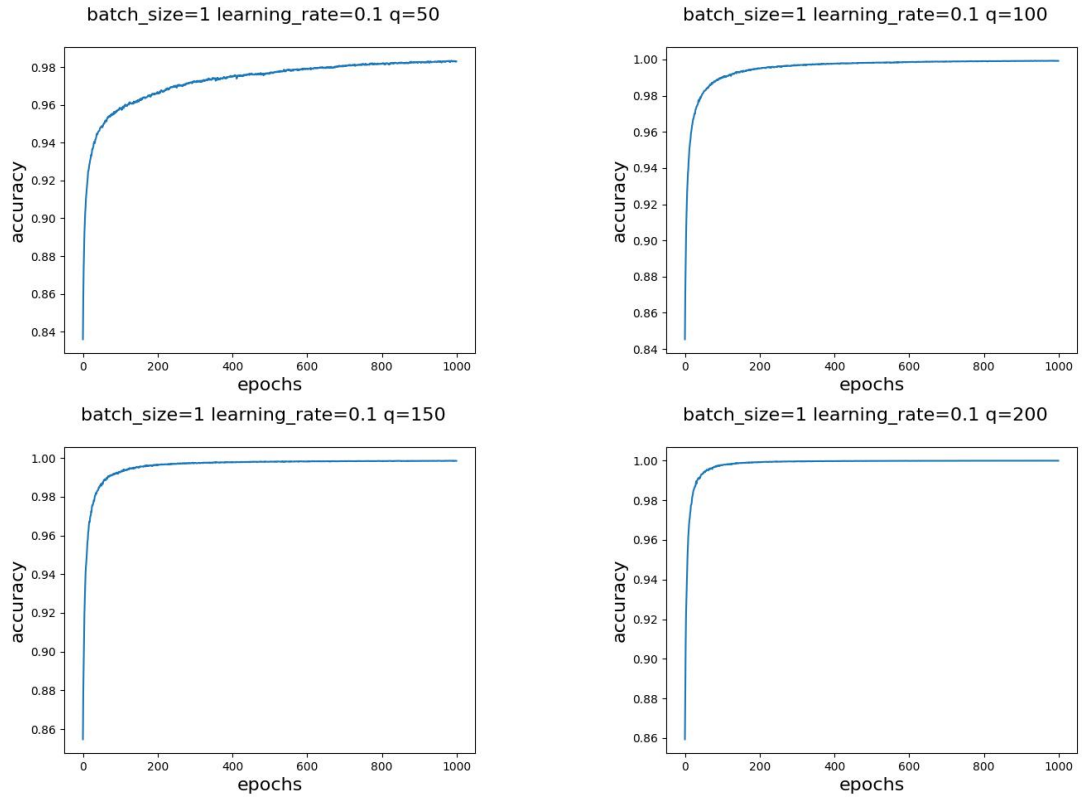
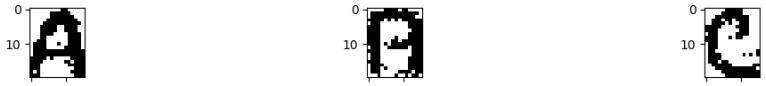


FIGURE 4 – Variation du nombre des unités cachées

On remarque que plus le nombre des unités cachées croît, plus la courbe de précision devient meilleure.

On choisit ainsi : BATCH_SIZE = 20 ,EPSILON = 0.1 et q = 200.

1.2.2 Images générées



(a) (b) (c)

FIGURE 5 – Génération d'images à partir de 3 classes (A, B, C)

Les images générées (figure 5) en entraînant le modèle sur les classes A, B et C sont claires et visibles.

Les images générées (figure 6) en entraînant toutes les lettres (26 classes) sont trop bruitées à part quelques unes dans la figure 6 comme (g) qui ressemble à "C", (b) à "e" et (f) à "B".

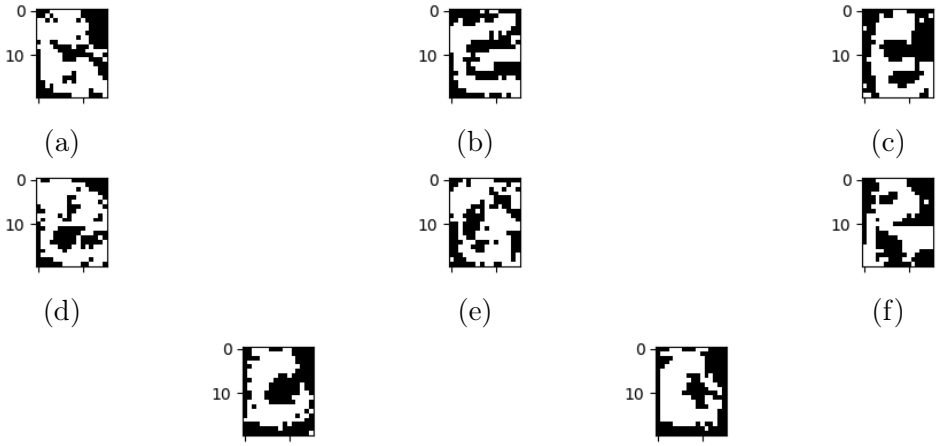


FIGURE 6 – Génération d'images à partir de l'entraînement de toutes les lettres

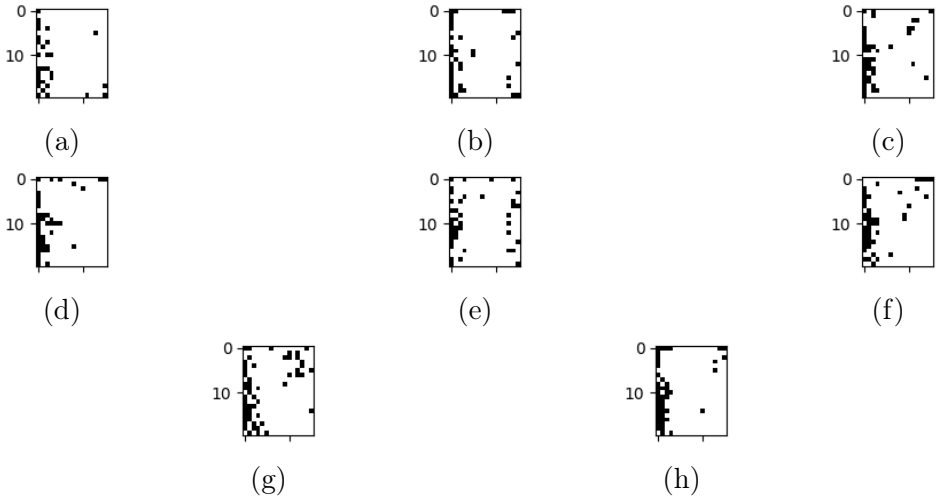


FIGURE 7 – Génération d'images à partir de l'entraînement de tous les chiffres

FIGURE 8 – Génération d'images à partir de l'entraînement des classes 0 et 1

(a) (b)

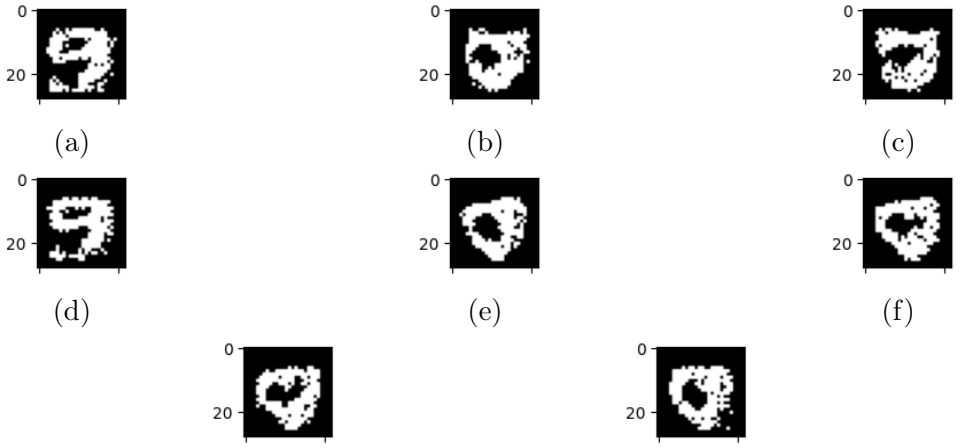
FIGURE 9 – Génération d'images à partir de l'entraînement des classes 2 et 3

C'est le même cas pour les chiffres. On peut voir dans la figure 9 que les images sont claires et que le "2" et "3" sont visibles, alors que dans les figures 7 et 8 on ne peut pas distinguer les chiffres.

On remarque alors, que plus on augmente le nombre des classes utilisées pour entraîner le modèle plus les images deviennent bruitées.

2 MNIST

2.1 RBM



(a) (b) (c)

(d) (e) (f)

(g) (h)

FIGURE 10 – Génération d'images à partir de l'entraînement de tous les chiffres

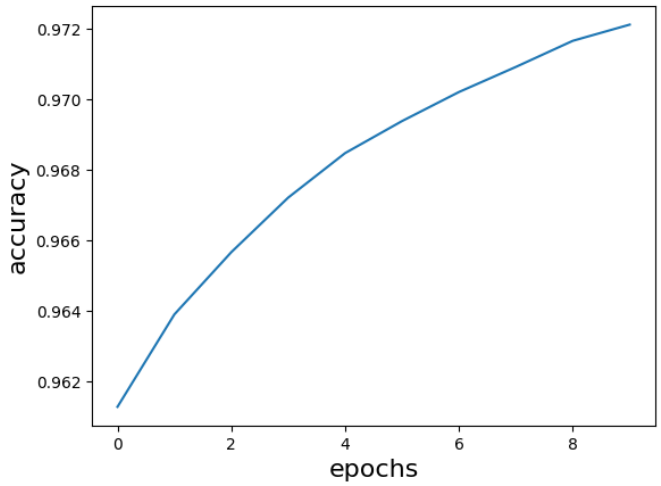


FIGURE 11 – Courbe de précision de MNIST en entraînant le RBM sur toutes les classes

(a) (b)

(c) (d)

FIGURE 12 – Génération d'images à partir de l'entraînement des classes 2, 3 et 4

(a) (b)

(c) (d)

FIGURE 13 – Génération d'images à partir de l'entraînement des classes 2 et 3

On remarque que les images sont toujours très bruitées sauf dans le cas de l'entraînement sur toutes les classes où on peut distinguer dans la figure 10 les chiffre "9" et "0".

2.2 GANs

En utilisant les GANs (Generative adversarial networks), on remarque que les images du dataset MNIST sont beaucoup plus claires qu'en utilisant les RBMs.

FIGURE 14 – Imges générées en appliquant les GANs

Cette amélioration peut être due au fait que MNIST est un dataset de 60K images alors que AlphaDigits contient 39,6K (1100*36) images ce qui causera un sur-apprentissage en entraînant un RBM sur MNIST contrairement à l'AlphaDigits.

Les GANs conviennent beaucoup plus à ce type de dataset car ils sont constitués d'un Générateur qui va générer des images à partir de celles fournies pour ensuite passer par le Discriminateur qui va les classifier en images "fausses" (0) et images "vraies" (1).

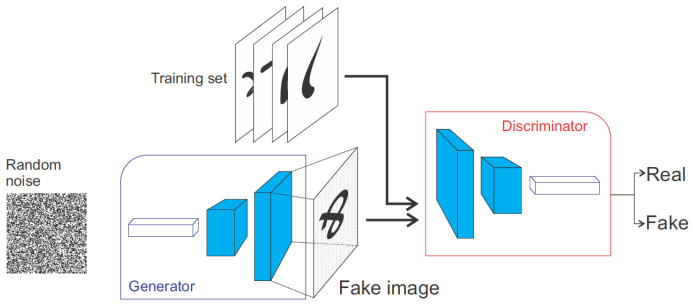


FIGURE 15 – Architecture des GANs