Mohamed Aymen BOUYAHIA
mohamed-aymen.bouyahia@ensta-paris.fr

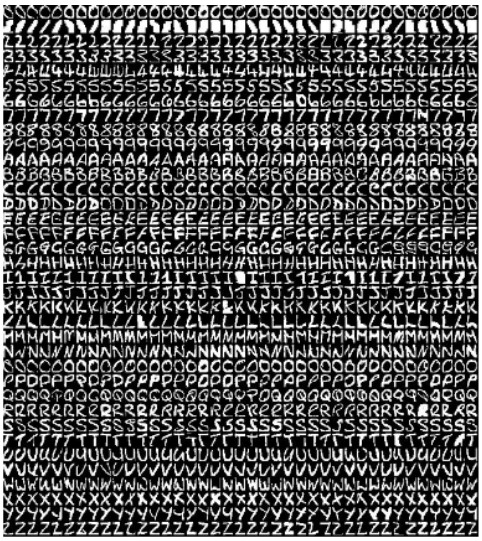
1 Introduction:

En matière d'apprentissage automatique, la machine de Boltzmann restreinte (RBM) représente un type de réseau de neurones artificiels destiné à l'apprentissage non supervisé. Son utilisation fréquente réside dans l'estimation de la distribution probabiliste d'un ensemble de données. Dans le cadre de ce travail pratique, nous allons mettre en œuvre une classe RBM afin de tester ce modèle sur divers ensembles de données. Nous procéderons à une analyse des résultats en tenant compte des hyperparamètres et de l'erreur de reconstruction.

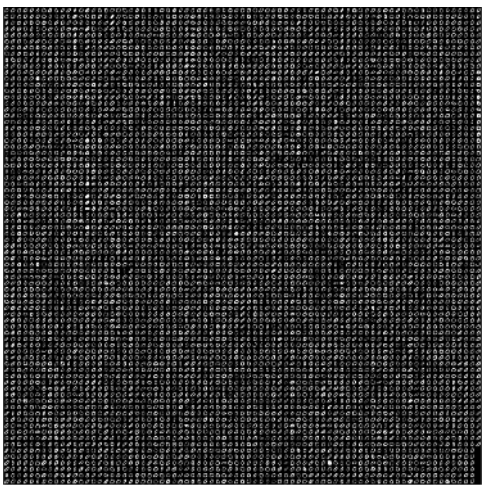
2 Description des bases de données:

Trois Bases de données ont été utilisées pour ce TP:
- Binary Alphadigits
- MNIST
- USPS

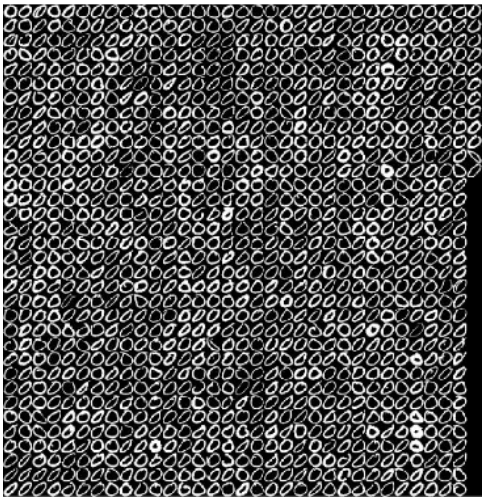
Binary Alphadigits: Ce jeu de données est téléchargeable du lien: "//www.cs.nyu.edu/ roweis/data.html". Il contient des images binaires de tailles 20×16 représentant les caractères de 0 à 9 et de A à Z.



MNIST: Ce jeu de données contient des images 8-bits en nuance de gris de tailles 28×28 représentant les caractères de 0 à 9.

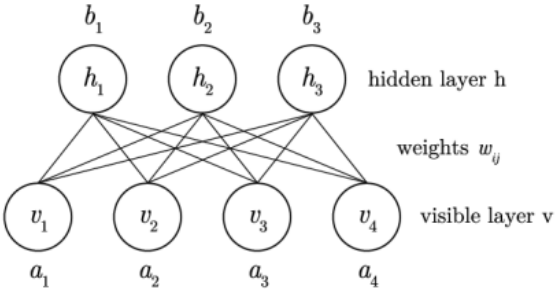


USPS: Ce jeu de données contient des images 8-bits en nuance de gris de tailles 16×16 représentant les caractères de 0 à 9.



3 Architecture de RBM:

La construction d'un RBM (Restricted Boltzmann Machine) se fait à partir de la classe RBM munie des fonctions : entree_sortie_RBM, sortie_entree_RBM et train_RBM



- a : biais des unités d'entréé
- b : biais des unités de sortie
- W : la matrice représentant les poids de notre réseau reliant les variables visibles aux variables cachées.
- p : dimension de a
- q : dimension de b

Pour mieux comprendre le code du projet, on détaillera ci-dessous le rôle de chaque méthode et fonction utilisée :

- entrée_sortie_RBM(RBM,V) renvoie $\sigma(V \times W + b)$ avec $\sigma(x) = \frac{1}{1+e^{-x}}$.

- sortie_entrée_RBM(RBM,H) renvoie $\sigma(H \times W + a)$ avec $\sigma(x) = \frac{1}{1+e^{-x}}$.

- train_RBM(X, nb_iter, lr, batch_size) entraîne le RBM avec les données X.

- lire_alpha_digits(caracteres) permet de lire les données de la base de donnée Binary Alpha Digits. En fait, lire_alpha_digits recupère les données des caractères choisis et les applatit. Elle transforme alors chaque échantillon de taille 20×16 en un tableau de taille 1×320.

- generer_image_RBM(RBM, iter_gibbs, nb_img) est la fonction responsable de la généreration d'images à partir d'un RBM par un échantillonneur de Gibbs

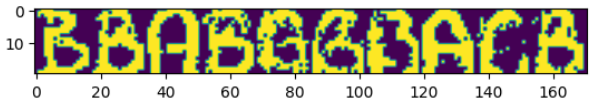
4 Evaluation des performances en fonction des hyperparamètres:

4.1 Influence du nombre d'unités cachées:

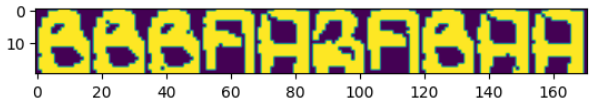
Le tableau suivant représente les performances du modèle RBM sur la dataset Binary Alphadigits en fonction du nombre d'unités cachées. On constate que l'erreur de reconstruction diminue lorsque le nombre d'unités

cachées augmente.

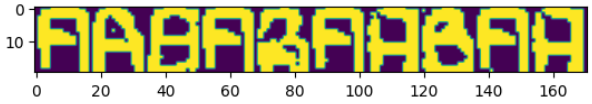
Nombre d'unités cachées	100	200	300	500	1000
Erreur de reconstruction	0.150	0.012	0.004	0.001	0.0002



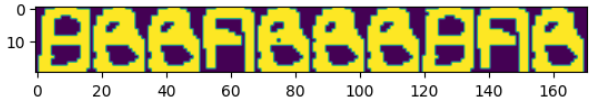
(a) RBM avec 100 unités cachées



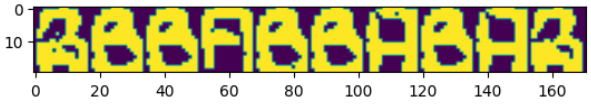
(b) RBM avec 200 unités cachées



(c) RBM avec 300 unités cachées



(d) RBM avec 500 unités cachées



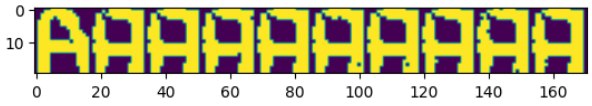
(e) RBM avec 1000 unités cachées

On remarque qu'en augmentant le nombre d'unités cachées, on améliore la qualité de la génération. Ceci est du au fait qu'on complexifie le modèle pour lui permettre de capturer certains détails dans la data. Cependant, il faut aussi le controler pour éviter la Overfit qui pourrait inciter le modèle à recopier la input en output.

4.2 Influence du nombre de caractères à apprendre:

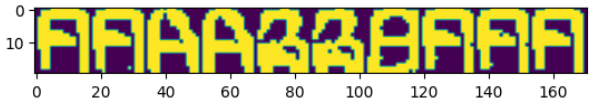
Le tableau suivant représente les performances du modèle RBM sur la dataset Binary Alphadigits en fonction du nombre de caractères à apprendre. On constate que l'erreur de reconstruction augmente lorsque le nombre de caractères à apprendre augmente. (Ce test est fait en utilisant un RBM avec 200 unités cachées)

Nombre de caractères à apprendres	1	2	3	5	10
Erreur de reconstruction	0.007	0.010	0.012	0.014	0.0235



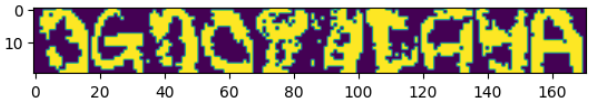
(a) RBM avec 1 caractère à apprendre

(b) RBM avec 2 caractères à apprendre



(c) RBM avec 3 caractères à apprendre

(d) RBM avec 5 caractères à apprendre



(e) RBM avec 10 caractères à apprendre

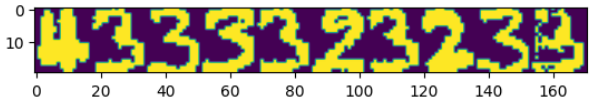
On remarque qu'en augmentant le nombre caractères à apprendre, on diminue la qualité de la génération. Ceci signifie que le modèle peut avoir des difficultés à capturer les caractéristiques distinctives de chaque caractère et à les reproduire avec précision. La solution pourrait etre de complexifier d'avantage le modèle et donc d'ajouter des unités cachées.

5 Evaluation du modèle sur la dataset MNIST et USPS:

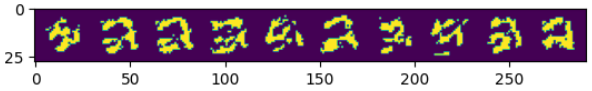
On va comparer le modèle sur les ensembles de chiffres entre 0 et 9 et on ne considère pas les caractères de l'alphabet qui sont uniquement présents sur la base de données Binary Alphadigits.

Pour comparer le modèle sur MNIST ou sur USPS avec Binary Alphadigits nous avons fixés le nombres d'epochs à 1000 et le nombre de caractères à 3.

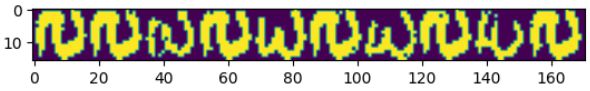
Erreur sur Binary Alphadigits	0.010
Erreur sur MNIST	0.043
Erreur sur USPS	0.002



(a) RBM sur Binary Alphadigits



(b) RBM sur MNIST



(c) RBM sur USPS

On peut constater que l'erreur de reconstruction est plus élevée sur MNIST par rapport à Binary Alphadigits. Ceci peut aussi etre constaté à travers la figure générée avec MNIST. Une autre remarque peut etre de remarquer que le modèle RBM prend plus de temps dans l'entrainement sur les données MNIST que sur Alphadigits. Ceci est du au fait que la taille des images en MNIST est de 28×28 alors qu'en Binary Alphadigits est de 20×16. Une conclusion pourrait etre de dire que le modèle RBM nécessite plus d'unités cachées et beaucoup plus de temps de calcul pour avoir le meme résultat qu'avec la Binary Alphadigits. Ceci montre l'instabilité de la performance du modèle.

En terme de temps d'exécution, l'entrainement avec USPS est le plus rapide. Ceci est du à la taille des images qui est de 16×16. Cependant, la reconstruction se fait d'une mauvaise manière malgrès la faible valeur de l'erreur meme en augmentant le nombre d'unités cachées. Ceci renforce l'idée que les RBM peuvent avoir une performance médiocre!

6 Comparaison entre RBM et TorchGAN:

En utilisant le modèle TorchGAN avec un nombre d'epochs égale à 10, on remarque que les GAN sont capables de générer des images fausses proches de la réalité. Les RBM sont incapables de produire de tels résultats d'où l'utilisation des GAN et des VAE en industrie.

7 Conclusion:

Suite à notre étude réalisée dans le cadre de ce TP, on constate que les Restricted Boltzmann Machines nous permettent de reconstruire des images assez rapidement. Cependant, les performances ne sont pas toujours bonnes, notamment sur un jeu de données couramment utilisé tel que MNIST pour évaluer les performances d'un algorithme d'apprentissage. Ainsi, les RBM sont peu à peu abandonnées par les industriels en faveur d'autres modèles tels que les GAN ou les VAE.