FACULTÉ DES SCIENCES
DÉPARTEMENT D'INFORMATIQUE

UNIVERSITÉ LIBRE DE BRUXELLES ULB

INFO-F-311: Intelligence Artificielle

Projet 5: Réseaux de neurones

Pascal Tribel **Yannick Molinghen** **Axel Abels**

Tom Lenaerts

1 Préambule

Ce projet vise à vous familiariser avec la notion de **réseaux de neurones** et leur utilisation. Nous vous demandons d'implémenter un *auto-encoder* de chiffres écrits à la main, issus du jeu de données MNIST.

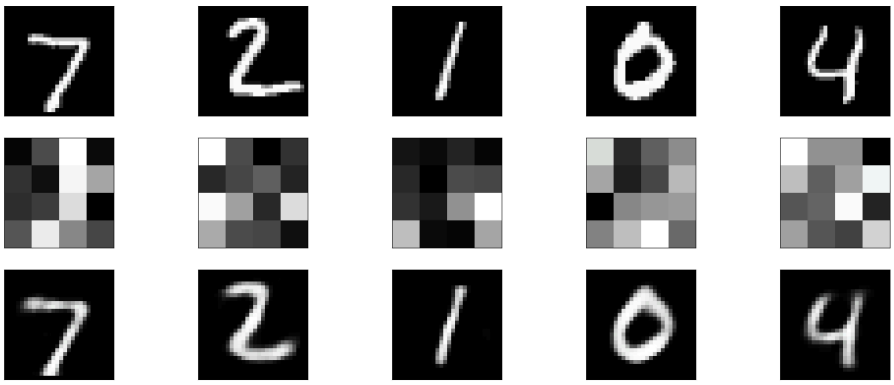

Figure 1: Des chiffres écrits à la mains, puis encodés, puis décodés grâce à un *auto-encoder*.

2 Introduction

Le jeu de données https://www.kaggle.com/datasets/oddrationale/mnist-in-csv contient deux fichiers *csv*:

- Un fichier d'entraînement, de 60000 éléments,
- Un fichier de test, de 10000 éléments.

Chaque ligne commence par le chiffre représenté, suivi de 28×28 nombres représentant chacun un pixel de l'image. Chaque pixel correspond à un niveau de gris représenté par un entier entre 0 et 255. Vous pouvez partir du principe que ces fichiers sont corrects et bien encodés.

3 Auto-encoders

Les réseaux de neurones composent une grande famille d'algorithmes de Machine Learning. Dans ce projet, vous vous concentrerez sur une classe élémentaire: les *auto-encoders*. Vous allez implémenter un tel réseau, encodant une image vers un vecteur de taille inférieure à l'aide d'une couche intermédiaire, et décodant un vecteur encodé à l'aide d'une autre couche intermédiaire. La strucutre du réseau est montrée dans la figure suivante:

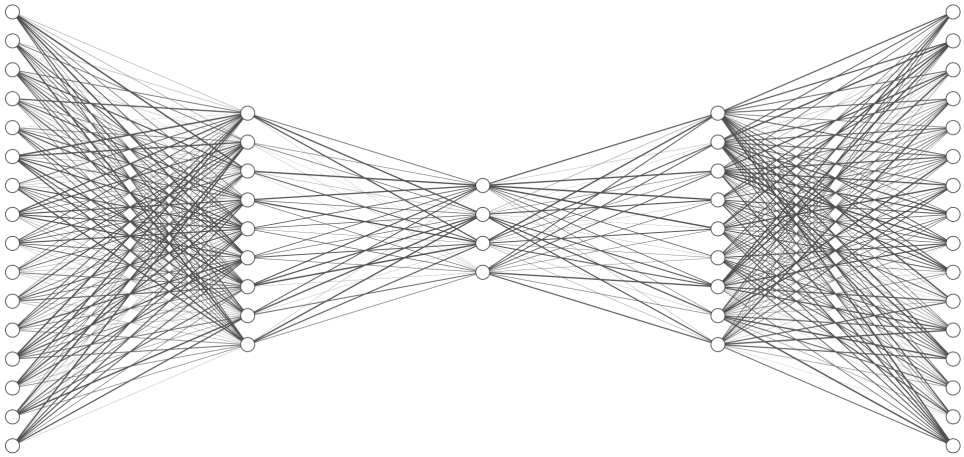

Figure 2: Exemple de structure d'*auto-encoder*.

L'objectif d'un tel réseau est d'appendre une méthode d'encodage et de décodage vers un espace de dimension plus faible que celui d'entrée, tout en maintenant une erreur de reconstruction minimale.

3.1 Entrées et sorties

L'entrée du réseau de neurones est un tableau `numpy` de dimension 784 (sous forme d'un vecteur, pas d'une matrice 28×28). La sortie est un autre tableau, aussi de taille 784. La méthode `get_train_test` doit permettre de récupérer les données d'entrée et de sortie stockées dans un fichier *csv* spécifié en paramètre. Veillez à normaliser les pixels dans l'intervalle $[0, 1]$.

3.2 Algorithme d'apprentissage

L'apprentissage du réseau se fait en plusieurs étapes:

- Présenter une image au réseau, en calculer la sortie (`forward`)
- Calculer l'erreur de prédiction pour évaluer les performances du modèle
- Rétropropager l'erreur de prédiction dans le réseau, pour mieux approcher la sortie espérée (`backward`)

Les calculs correspondants à ces étapes sont spécifiés dans les sections suivantes. Vous devez les implémenter. Veillez à utiliser les fonctions de `numpy` pour permettre une exécution suffisamment rapide de votre code.

La fonction d'entraînement vous est fournie. Elle présente un exemple au réseau, calcule les sorties de chaque couche, et rétropropage l'erreur dans le réseau. Ce processus est répété pour toutes les images du jeu d'entraînement, présentées en lots, `epochs` fois. A chaque *epoch*, la MSE (Mean Squared Error) est calculée pour donner une estimation de l'apprentissage du réseau.

3.3 Fonction d'activation

On vous fournit une fonction d'activation, la *sigmoïde*: $\text{activation}(x) = \frac{1}{1+e^{-x}}$, et sa dérivée: $\text{derivative}(x) = x(1-x)$.

Passe avant (`forward`)

Considérez que le réseau est composé de quatre couches, chacune reliée par une matrice de poids :

- Une matrice de poids W_1 connecte la couche d'entrée à la première couche intermédiaire.
- Une matrice de poids W_2 connecte la première couche intermédiaire à la couche d'encodage.
- Une matrice de poids W_3 connecte la seconde couche intermédiaire à la seconde couche intermédiaire.
- Une matrice de poids W_4 connecte la seconde couche intermédiaire à la couche de sortie.

Les étapes pour le passage avant sont les suivantes :

1. Calculez la sortie de la première couche intermédiaire : $x_1 = \text{activation}(x \times W_1)$
2. Calculez le vecteur encodé : $\hat{x} = \text{activation}(x_1 \times W_2)$
3. Calculez la sortie de la seconde couche intermédiaire : $x_2 = \text{activation}(\hat{x} \times W_3)$
4. Calculez le vecteur décodé: $y = \text{activation}(x_2 \times W_4)$

Les deux premières étapes forment l'*encodage*, et les deux suivantes forment le *décodage*.

Erreur de prédiction

L'erreur de prédiction entre l'image attendue x et l'image décodée y est la Mean Squared Error: $\text{MSE}(x, y)$. La MSE est définie comme $\text{MSE}(x, y) = \frac{1}{n}\sum(x - y)^2$.

Rétropropagation (`backward`)

La rétropropagation ajuste les poids du réseau en fonction de l'erreur de reconstruction. Les étapes sont les suivantes :

1. Calcul de l'erreur de sortie: $e_4 = y - x$. Le gradient de cette erreur est multiplié par la dérivée de la fonction d'activation sur la sortie pour obtenir : $\delta_4 = e_4 \odot \text{derivative}(y)$.
2. Propagation des erreurs aux couches intermédiaires:
   - Erreur de la seconde couche intermédiaire: $e_3 = \delta_4 \times W_4^T$, et $\delta_3 = e_3 \odot \text{derivative}(x_2)$.
   - Erreur de la couche d'encodage: $e_2 = \delta_3 \times W_3^T$, et $\delta_2 = e_2 \odot \text{derivative}(\hat{x})$.
   - Erreur de la première couche intermédiaire: $e_1 = \delta_2 \times W_2^T$, et $\delta_1 = e_1 \odot \text{derivative}(x_1)$.
3. Mise à jour des poids:
   - $W_4 = W_4 - \mu(x_2^T \times \delta_4)$
   - $W_3 = W_3 - \mu(\hat{x}^T \times \delta_3)$
   - $W_2 = W_2 - \mu(x_1^T \times \delta_2)$
   - $W_1 = W_1 - \mu(x^T \times \delta_1)$

Dans les équations qui précèdent, μ désigne le *learning rate*, $\times$ désigne le produit matriciel et $\odot$ désigne le produit composante par composante.

Fonctions supplémentaires

On vous fournit la fonction `plot_image` pour une visualisation basique de vecteurs représentant des images.

4 Consignes

On vous fournit deux fichiers Python:

- `AutoEncoder.py`, qui contient le code à compléter. Vous aurez besoin des librairies `numpy`, `matplotlib` et `tqdm`.
- `utils.py` qui contient différentes fonctions utiles à la réalisation du projet. Certaines fonctions sont à compléter aussi.

4.1 Rapport

Produisez un rapport *scientifique*, qui suit la structure suivante :

1. Introduction
2. Cadre expérimental
3. Résultats
4. Analyse
5. Conclusion
6. Bibliographie (éventuelle)

et qui ne doit pas excéder **5 pages A4**, en comptant les figures et la bibliographie. Veillez à obtenir un rendu similaire à Latex (mais vous pouvez utiliser un outil similaire, comme http://typst.app). **N'incluez ni page de garde, ni table des matières**. Veillez à être *concis·e*: c'est une qualité.

4.2 Questions

On vous demande:

1. De compléter les fichiers `AutoEncoder.py`, `utils.py` et de créer un scipt pour l'exécution nommé `main.py` (5 points)
2. D'analyser l'impact de la taille du vecteur compressé $\hat{x}$ (2 point)
3. D'analyser l'impact de la valeur du *learning rate* μ (2 point)
4. D'analyser l'impact du nombre d'*epochs* e (1 point)
5. De proposer une raison de l'analyse des performances du réseau sur un fichier `test.csv` plutôt que sur le fichier `train.csv` (1 point)
6. De montrer ce que le réseau décode si on lui présente un vecteur encodé aléatoire, pour plusieurs exemples intéressants (*hallucination des auto-encoders*) (2 points)
7. De suggérer deux modifications à faire au réseau pour améliorer les performances (2 points)

La qualité générale du rapport (structure, introduction, conclusion, orthographe, sources, pertinence des questions et des remarques) compte pour 5 points.

4.3 Utilisation de Chat-GPT, Copilot, ... et autres assistants

Dans la conclusion de votre rapport, détaillez votre utilisation éventuelle d'outils d'assistanat, que ce soit pour l'écriture du code ou du rapport. Toutes vos sources doivent être pertinentes et exister. L'utilisation de tels outils est autorisée **mais déconseillée pour le code**: en effet, l'intérêt de l'écriture du code est la familiarisation avec les opérations qui composent l'apprentissage des réseaux de neurones. Ce travail est *individuel*.

Remise

- Un fichier `.zip` portant votre nom_prénom (exemple: `tribel_pascal.zip`), contenant les fichiers suivants:
  - Les fichiers complétés `AutoEncoder.py` et `utils.py`
  - Un exécutable `main.py`. Vous pouvez considérer qu'il sera exécuté dans un répertoire contenant aussi les fichiers `train.csv` et `test.csv`, mais n'incluez pas ces fichiers `.csv` dans votre remise.
  - Votre rapport, au format `nom_prenom.pdf`
- Tout manquement à une consigne de remise entraînera 0.5 points en moins.

Le travail est **individuel** et doit être rendu sur l'Université Virtuelle pour le 23/12/2024 à 23:59. Vous pouvez envoyer vos questions à l'adresse **pascal.tribel@ulb.be**. Mentionnez l'intitulé du cours (F311) dans l'en-tête de votre mail.

Bon travail !