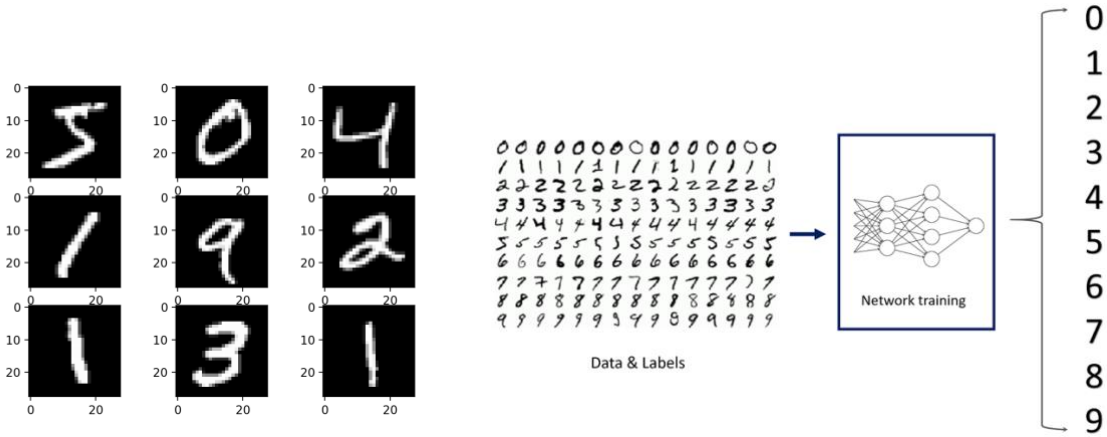
Semaine 5: Réseaux de Neurones artificiels

Les Chiffres Manuscrits

La base de données MNIST est un vaste ensemble de données de 70 000 chiffres manuscrits qui sont principalement utilisés pour la formation de plusieurs systèmes de traitement d'images, en particulier dans le domaine de l'apprentissage profond. Le jeu de données est composé des chiffres de 0 à 9 dans un format basé sur 28x28 pixels.

Les valeurs sont stockées dans un fichier séparé par des virgules dans lequel chaque ligne contient 785 valeurs (dont la 1ère représentant le chiffre et les 784 autres les pixels) (28x28 = 784 pour les pixels), chaque ligne représentant un nombre différent qui est initialement affiché dans la première valeur de la ligne (784 + 1). Si vous êtes intéressé par le MNIST-Dataset, vous trouverez des informations détaillées sur le site Web de Yann LeCun ici.

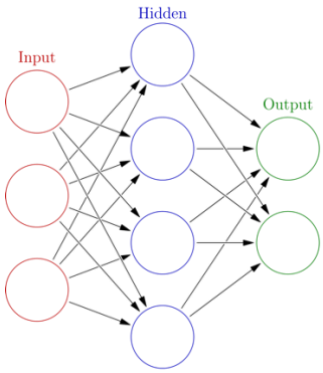


La base de données est disponible ici : https://www.kaggle.com/oddrationale/mnist-in-csv

I. Pré-traitement

a. Préparer votre environnement de travail et importer les ensembles de données d'apprentissage et de test.
b. Remodeler vos entrées (Matrices de tailles 28x28) et visualiser des exemples des chiffres.

II. Réseaux de Neurones - Détection des Chiffres Manuscrits



Dans cette étape, on va définir la classe « reseaux de neurones » qui a pour but d'entrainer notre modèle.

a. Définir un constructeur qui initialise les attributs suivants :
   - Paramètres d'entrée
     i. Le nombre de nodes en entrée, en sortie et cachés
     ii. Taux d'apprentissage (Learning Rate)
   - Fonction d'activation sigmoïde.
   - Matrices de Poids générées par prélèvement d'échantillons aléatoires à partir d'une distribution normale (gaussienne):
     i. Hidden Weights Input
     ii. Hidden Weights Output

Pointeur : Utiliser la fonction random.normal(moyenne_de la_distribution=0, ecart_type, taille_de_matrice) de numpy pour générer vos vecteurs
Ecart_Type Distribution normale = pow(hidden_nodes, -0.5)

b. Définir une fonction « prédiction » qui prend en paramètre une liste d'entrée et calcule la sortie du réseau de neurones :
   - Convertir la liste d'entrée en matrice 2D
   - Calculer l'entrée et la sortie cachée
   - Calculer et retourner la sortie finale.

c. Définir une fonction « Train » qui prend en paramètres une liste d'entrée et une liste de labels observées:
   - Notre fonction d'erreur sera l'erreur quadratique moyenne.
   - Calculer la sortie du réseau (se baser sur la fonction *prédiction*)
   - Étape BackPropagation
     i. Calculer l'erreur de la sortie finale et de la couche cachée.
     ii. Mettre à jour vos poids grâce au taux d'apprentissage. (Étape d'apprentissage)

$$\theta^{t+1} = \theta^t - \alpha \frac{\partial E(X, \theta^t)}{\partial \theta}$$

III. Apprentissage

Dans cette partie on va apprendre notre modèle via plusieurs exécutions ou époques.

a. Étapes à suivre :
   i. Splitter votre record (‘,’)
   ii. Mettre à l'échelle et décaler les entrées (Rappel : Un pixel prend une valeur entre 0 et 255)
   iii. Créer les valeurs de sortie cibles (un tableau d'éléments nœuds de sortie de valeurs 0.01) sauf la valeur observée qui sera mise à 0,99 ici (1ère colonne)
   iv. Appeler la fonction Train

IV. Test

a. Visualiser le 1er élément de la base test.
b. Développer un tableau de bord pour évaluer votre modèle de façon à ce que si la valeur prédite est correcte, une valeur de 1 est rajoutée au score. Sinon, 0
   i. Le score de performance est donc la moyenne.
c. Visualiser un chiffre prédit