Université Abdelmalek Essaâdi
Faculté des Sciences et Techniques d'Al-Hoceima
Département de Mathématiques et Informatique

Filière : LST- IDLL
Module : Intelligence artificielle
A .U : 2023/2024
Pr : ZANNOU Abderrahim

Version 1

TP N°7
Énoncé (ANN)

L'objectif de ce TP est d'appliquer :

- ✓ Classification multi-classes en utilisant réseaux de neurones artificiels (ANN)

***Tâche :* Prédire l'étiquette d'une image**

- ✓ **Jeux de données (Dataset) : fashion-mnist_train et fashion-mnist_test**
- ✓ **La variable cible** : *label(0 …9)*
- ✓ **Le fichier associé :** TP7_1_Classification_Multiclass_ANN_MNIST
- ✓ **Les étapes essentielles :**
  - Importer les bibliothèques nécessaires et la bibliothèque *tensorflow*
  - Charger les jeux de données :

    df_train=pd.read_csv("./datasets/fashion-mnist_train.csv")
    df_test=pd.read_csv("./datasets/fashion-mnist_test.csv")
  - Diviser les données sur X et Y :

    X_train = df_train.iloc[:,1:785].values
    Y_train = df_train.iloc[:,0].values
    X_test = df_test.iloc[:,1:785].values
    Y_test = df_test.iloc[:,0].values
  - Normaliser X_train en utilisant MinMaxScaler
  - Préparer les étiquettes cibles (Y_train) pour le modèle de classification multiclasse :

    Y_train_enc = tf.keras.utils.to_categorical(Y_train, num_classes=10)
  - Initialiser l'ANN :

    ann = tf.keras.models.Sequential()
  - Déclarer les couches suivantes :

    ann.add(tf.keras.layers.Dense(units=100, activation='relu'))
    ann.add(tf.keras.layers.Dense(units=100, activation='relu'))
    ann.add(tf.keras.layers.Dense(units=10, activation='softmax'))
  - Compiler l'ANN

    Méthode 1 :

    ann.compile(optimizer = 'adam', loss = 'categorical_crossentropy')

    Méthode 2 :

    Vous pouvez utiliser une autre méthode, en spécifiant la valeur de learning rate (Le taux d'apprentissage est un paramètre de réglage dans un algorithme d'optimisation qui détermine la taille du pas à chaque itération tout en se déplaçant vers un minimum d'une fonction de perte), comme ci-dessous :

```
# Spécifier le taux d'apprentissage
learning_rate = 0.001
# Création de l'optimiseur Adam avec le taux d'apprentissage spécifié
opt = tf.keras.optimizers.Adam(learning_rate=learning_rate)
# Compilation du modèle avec l'optimiseur spécifié et la fonction de perte
ann.compile(optimizer=opt, loss='categorical_crossentropy')
```

- Entraîner l'ANN sur l'ensemble d'entraînement

```
ann.fit(X_train,Y_train_enc, batch_size=128,epochs =60,verbose=1)
```

- Réaliser les prédictions sur X_test qui doit être normalisé.

```
Y_pred = ann.predict(X_test)
# Obtenir la classe prédite pour chaque exemple
Y_pred = np.argmax(Y_pred, axis=1)
print(Y_pred)
```

- Déterminer la matrice de confusion et les métriques d'évaluation
- Prédire la classe de l'observation d'indice 0 dans le training set :

```
# Prédire la classe d'une observation
xObs=np.array(X_train[0, :]).reshape(1, -1)
Y_prevu1 = ann.predict(xObs)
Y_prevu1  = np.argmax(Y_prevu1, axis=1)
print(Y_prevu1 )
```