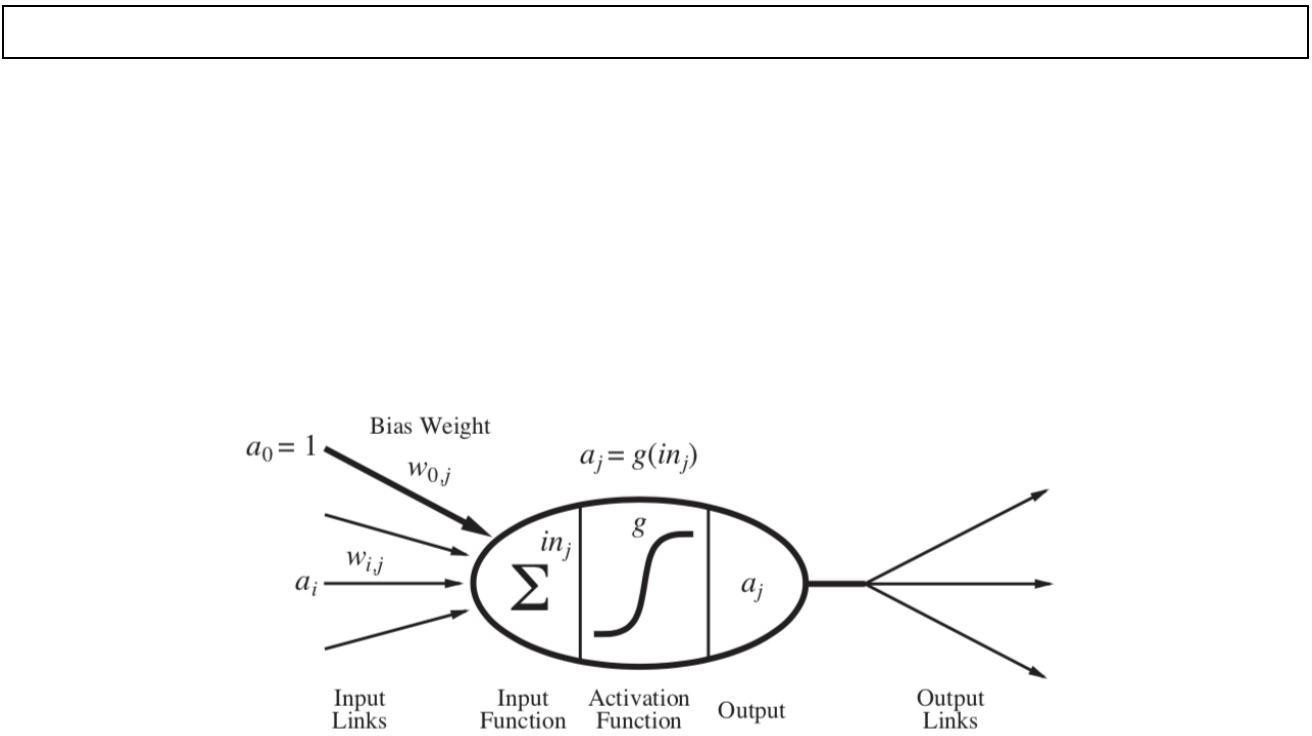
Activité de laboratoire 7 – Réseau de neurones artificiels Travail

Vous devez soumettre vos solutions de ce devo, dans le respect du délai fixé par votre professeur. Vous êtes autorisé à travailler en groupes de deux ou trois, mais une seule soumission par groupe est requise. La tricherie ne sera pas tolérée et sera pénalisée.

Réseaux de neurones artificiels

Les réseaux de neurones sont l’une des formes de systèmes d’apprentissage les plus populaires et les plus efficaces.

### Un réseau de neurones est un ensemble de nœuds ou d'unités reliés par des liens dirigés. Un lien de l'unité i vers l'unité j sert à propager l'activation ai de i vers j. Chaque activation de lien est également associée à un poids numérique wi,j , qui détermine la force et le signe du poids de la connexion.

Figure 1•Un modèle mathématique simple pour un neurone.

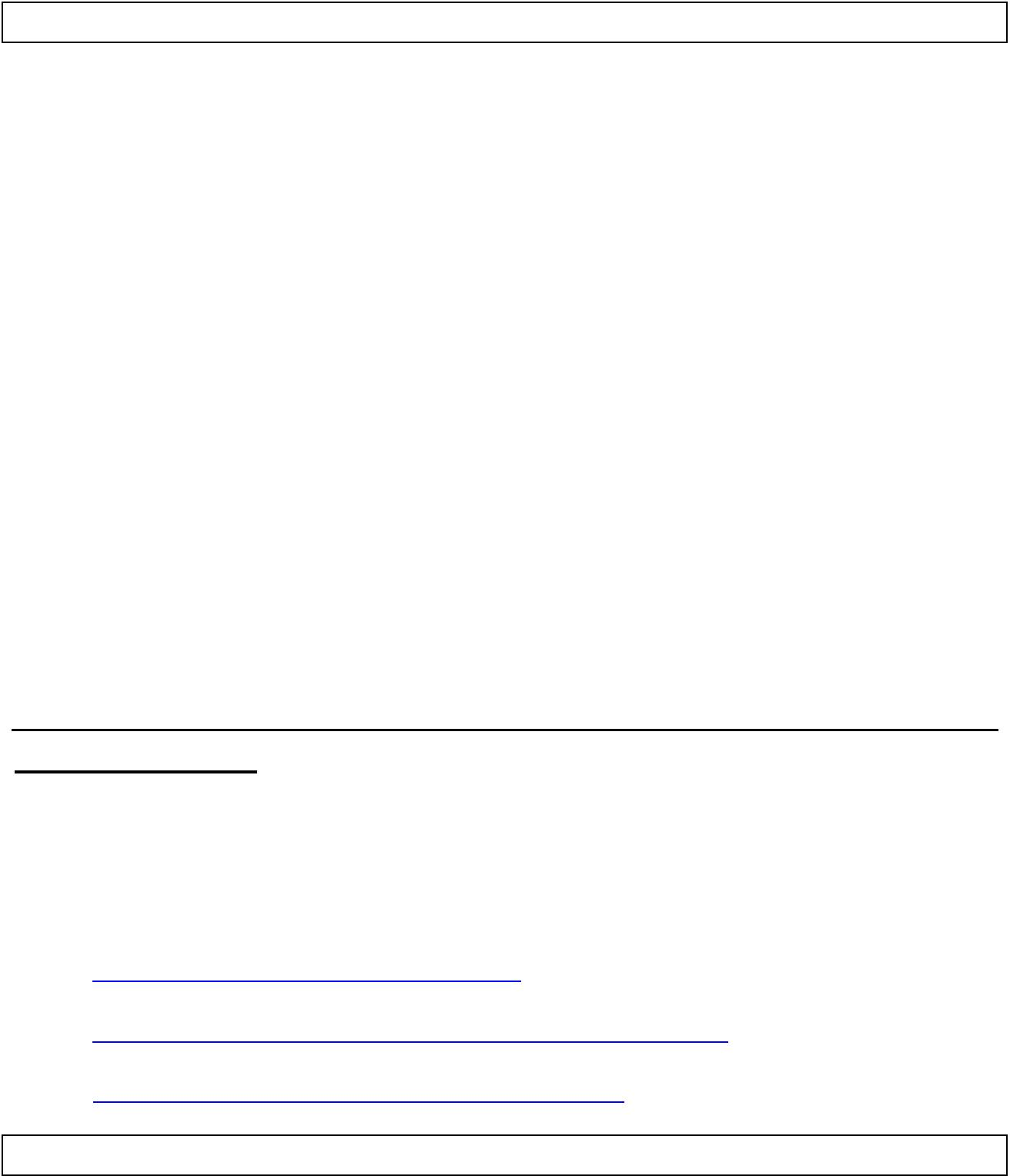
### Il existe plusieurs types de réseaux de neurones artificiels. Les plus courants sont :

* Un réseau à action directe n'a de connexions que dans une seule direction, il n'y a pas de boucles. Un réseau à action directe représente une fonction de son entrée actuelle ; ainsi, il n’a pas d’état interne autre que les poids eux•mêmes.

Les réseaux à action directe sont généralement disposés en couches, de telle sorte que chaque unité reçoive les entrées uniquement des unités de la couche immédiatement précédente. Dans un réseau multicouche, une ou plusieurs couches d' unités cachées qui ne sont pas connectées aux sorties du réseau peuvent exister.

* Un réseau récurrent, en revanche, réinjecte ses sorties dans ses propres entrées.

Les signaux voyagent dans les deux sens en introduisant des boucles dans le réseau. Les calculs dérivés d’entrées antérieures sont réinjectés dans le réseau, ce qui leur confère une sorte de mémoire.

* Un réseau convolutif se compose d'une couche d'entrée et d'une couche de sortie, ainsi que de plusieurs couches cachées. Les couches cachées d'un réseau convolutif sont généralement constituées d'une série de couches convolutives qui convoluent avec une multiplication ou un autre produit scalaire.

Dans cet atelier, vous allez modéliser un simple réseau de neurones artificiels (ANN) à feed•forward.

Keras : la bibliothèque Python Deep Learning

Keras est une API de réseaux neuronaux de haut niveau, écrite en Python et capable de s'exécuter sur TensorFlow, CNTK ou Theano. Il a été développé dans le but de permettre une expérimentation rapide. Être capable de passer de l’idée au résultat dans les plus brefs délais est la clé d’une bonne recherche.

Avantages Keras :

* Permet un prototypage facile et rapide (grâce à la convivialité, à la modularité et à la extensibilité).
* Prend en charge les réseaux convolutifs et les réseaux récurrents, ainsi que les combinaisons des deux.
* Fonctionne de manière transparente sur CPU et GPU.

La structure de données de base de Keras est un modèle, un moyen d'organiser les couches. Le type de modèle le plus simple est le modèle séquentiel , une pile linéaire de couches. Empiler des couches dans votre NN est aussi simple que .add()

Afin d'installer Keras et TensorFlow, référez•vous au document pdf dans votre dossier intitulé

«Instructions d'installation Keras.pdf». Certaines instructions sont également disponibles dans votre ANN.ipynb .

Lecture préliminaire :

Rappel : L'apprentissage se fait avec une pédagogie axée sur les travaux pratiques et l'autonomie.

Avant de commencer à coder et à mettre en œuvre votre modèle d'apprentissage automatique, vous devez commencer par les lectures recommandées :

1. Conférence 4 – Introduction au Deep Learning et aux réseaux de neurones
2. Documentation du modèle séquentiel Keras : https:// keras.io/guides/sequential\_model/
3. Documentation sur les couches denses : https:/[/www.tutorialspoint.com/keras/keras\_dense\_layer.htm](http://www.tutorialspoint.com/keras/keras_dense_layer.htm)
4. Documentation des Wrappers Keras pour Scikit•learn : https://faroit.com/keras•docs/1.0.6/scikit•learn•api/

Le problème des clients de la banque

Dans ce laboratoire, nous allons prédire à l'aide d'ANN si le client d'une banque est susceptible de quitter la banque ou non.

L'ensemble de données avec lequel vous allez travailler est Bank\_customers.csv . Dans ce fichier, vous avez plusieurs colonnes.

Vous devez vérifier ce que chaque colonne représente et voir s'il s'agit d'une information pertinente à utiliser comme fonctionnalité ou non.

L'activité du laboratoire est divisée en plusieurs parties. Il est recommandé de finaliser toutes les parties de cette session de laboratoire. Ces différentes étapes sont les suivantes :

1. Chargez l'ensemble de données.
2. Inspectez et nettoyez les données si nécessaire. Supprimez les fonctionnalités indésirables.
3. Encodez les caractéristiques catégorielles en numérique.
4. Divisez vos données en ensembles d'entraînement et de test.
5. Utilisez StandardScaler pour mettre à l'échelle vos données.(Standarisation)
6. Utilisez le séquentiel de keras.models pour initialiser votre modèle ANN en le définissant comme une séquence de couches.
7. Ajoutez la couche d'entrée et la première couche cachée en utilisant la méthode add() .

classifier.add(Dense(units = 6, kernel\_initializer = 'uniform', activation = 'relu', input\_dim = 11))

Utilisez Dense() pour créer une couche Dense qui est la couche de réseau neuronal profondément connectée habituelle. Il s’agit de la couche la plus courante et la plus fréquemment utilisée.

La méthode Dense prend les arguments suivants :

unités : dimensionnalité de l'espace de sortie.

activation : Fonction d'activation à utiliser. o Dans cette activité, utilisez :

relu : fonction d'activation du redresseur pour les couches cachées. ▪ fonction

sigmoïde pour la couche de sortie utilisable lorsque la sortie est binaire.

kernel\_initializer : Initialiseur pour la matrice de poids du noyau.

De plus, ajoutez l'argument input\_dim : dimension de la couche d'entrée, à transmettre pour la première couche cachée.

1. Ajoutez une deuxième couche cachée composée de 6 unités.
2. Ajoutez la couche de sortie avec une unité binaire. N'oubliez pas, cette fois, l'activation sigmoïde fonction doit être utilisée.
3. Avant de former le modèle ANN, vous devez configurer le processus d'apprentissage, qui est effectué via la méthode de compilation .

classifier.compile (optimizer = 'adam', loss = 'binary\_crossentropy', metrics=['accuracy'])

compile() reçoit trois arguments : un optimiseur (utilisez 'adam'), une fonction de perte (utilisez 'binary\_crossentropy') et une liste de métriques (utilisez la 'précision').

Cela signifie que votre processus de validation sera effectué à l'aide de l'optimiseur Adam qui utilise la descente de gradient stochastique pour optimiser les hyperparamètres du réseau, principalement les pondérations des caractéristiques. L' erreur est calculée en utilisant l'entropie croisée bianaire fonction et les mesures d'évaluation utilisées dans le score de précision.

1. Ajustez votre modèle et appliquez la validation croisée avec une taille de lot de 10. Répétez 50 ou 100 fois (époque = 50 ou 100).

classifier.fit (X\_train, y\_train, batch\_size = 10, époques = 50)

1. Faites des prédictions.
2. Évaluer le modèle à l'aide de la matrice de confusion et du rapport de classification.
3. Prédire si le client avec les informations suivantes quittera la banque ou non : Géographie : France • Pointage de crédit : 600 • Sexe : Homme • Âge : 40 • Ancienneté : 3 • Solde : 60000 • Nombre de produits : 2 • Possède un crédit carte : Oui • Est membre actif : Oui • Salaire estimé : 50000
4. Vous pouvez utiliser des modèles Keras séquentiels dans le cadre de votre flux de travail Scikit•Learn via le wrappers trouvés dans la bibliothèque keras. Suivez les instructions données dans le notebook Python pour comprendre comment utiliser KerasClassifier à partir de keras.wrappers.scikit\_learn.