Le rôle du constructeur de la classe MLP est de créer le réseau de neurones en fonction des paramètres. Cela inclut le nombre de couches et de neurones par couche, le taux d'apprentissage et la fonction de transfert.

Les entrées pour le constructeurs sont :

- Un tableau d'entier "layers" qui indique le nombre de neurone par couche
- Un réel qui représente le taux d'apprentissage. Ce paramètre détermine l'ampleur des ajustements des poids lors de l'apprentissage.
- Une fonction de transfert, qui est utilisée par chaque neurone pour transformer sa somme pondérée en une valeur de sortie.

Le rôle de la méthode execute est de calculer les différentes couches de neurones depuis les données fournies grâce à la fonction de transfert qui est appliquée sur chaque neurones.

L'entrée est un tableau de réels qui représente les données initiales du problème.
La sortie est un tableau de réels qui correspond au résultat calculé par le réseau

Le rôle de la méthode backPropagate est de réduire l'erreur du réseau en ajustant les poids et biais des neurones. Cela se fait en remontant à travers les couches, en utilisant les différences entre les résultats calculés et les résultats attendus.

Les entrées pour la méthode sont :

- Un tableau de réels représentant les données d'entrée, qui sera utilisé dans la méthode "execute" pour calculer les sorties actuelles du réseau.
- un tableau de réels représentant les résultats souhaités. (Apprentissage supervisé)

La sortie est un double qui représente la différence moyenne entre les résultats calculés par le réseau et les résultats souhaités.

Pseudo-code :

```
entree = new tab
sortie = new tab

mlp = new MLP(tableauReseau, ta, Sigmoid)


POUR  chaque itération de 1 à iterationMax
        erreurTotal = 0
        POUR chaque i dans entree
            input = entree[i]
            output = sortie[i]
            erreur = backPropagate( MLP, input,output)
            erreurTotal = erreurTotal + erreur
        FIN POUR
        SI erreurTotal est inférieure à un seuil X
              sortir de la boucle
        FIN SI
FIN POUR

testEntree = new tab
POUR chaque exemple dans testEntree
        res = execute(mlp, exemple)
        Afficher "Entrée : ", exemple, "  Résultat prédit : ", RÉSULTAT
FIN POUR
```

new MLP(tableauReseau, ta, fun)
Appel du constructeur pour initialiser le perceptron
Définit l'architecture du réseau de neurones avec les entrées
Entrées :

- TableauReseau : Un tableau avec le nombre de neurones par couche
- ta : Fixe le taux d'apprentissage.
- fun : La fonction de transfert pour calculer les neurones.

Sortie :

- Un objet mlp représentant le réseau de neurones initialisé.

backPropagate(mlp, input, output)
Utilise la méthode backPropagate pour entraîner le perceptron. Elle permet de calculer et d'appliquer les corrections sur les poids des neurones afin de minimiser l'erreur entre la sortie calculée et la sortie attendue.
Entrées :

- input : L'exemple d'entrée
- output : La sortie attendue pour l'exemple d'entrée

Sortie :

- Un réel représentant l'erreur entre la sortie calculée par le réseau et la sortie attendue. Cette erreur est utilisée pour ajuster les poids.

execute(mlp, exemple)

La méthode execute permet d'utiliser le réseau entraîné pour faire des prédictions sur de nouvelles données.

Entrées :

- exemple : Les données d'entrée pour lesquelles on veut obtenir une prédiction.

Sortie :

- Un tableau de réels représentant la sortie prédite par le MLP pour cet exemple.
- Résumé des entrées et sorties pour chaque appel

Lors de l'entraînement :

- Les entrées (input) et sorties (output) sont utilisées dans la méthode backPropagate pour ajuster les poids des neurones.
- Le calcul des erreurs et des ajustements des poids se fait à travers les couches du réseau.

Lors du test :

- Les données de test (testEntree) sont utilisées avec la méthode execute pour faire des prédictions.
- La sortie générée par le MLP est comparée à la sortie attendue pour évaluer la performance du modèle.