ÉCOLE CENTRALE D'ÉLECTRONIQUE (ECE) PARIS
Promotion 2025 - Semestre 1

═══

MACHINE LEARNING - SÉANCE 3
Réseaux de Neurones et Deep Learning

═══

Enseignant : Prof. Jean Dupont
Date : 15 Octobre 2025
Département : Intelligence Artificielle

CHAPITRE 1 : Introduction aux Réseaux de Neurones

1.1 Le Perceptron

Le perceptron est l'unité de base des réseaux de neurones artificiels. Inventé par Frank Rosenblatt en 1958, il s'inspire du fonctionnement des neurones biologiques.

Définition mathématique :
Un perceptron calcule une somme pondérée de ses entrées $x_1, x_2, ..., x_n$ avec des poids $w_1, w_2, ..., w_n$, ajoute un biais b, puis applique une fonction d'activation :

$$y = f(w_1x_1 + w_2x_2 + ... + w_nx_n + b)$$

où f est typiquement une fonction de Heaviside, sigmoïde, ou ReLU.

1.2 Fonctions d'Activation

Les fonctions d'activation introduisent la non-linéarité dans le réseau :

• Sigmoïde : $\sigma(x) = 1/(1 + e^{-x})$
  - Plage de sortie : [0, 1]
  - Problème : vanishing gradient pour x très grand ou très petit

• ReLU (Rectified Linear Unit) : $f(x) = \max(0, x)$
  - Avantage : calcul rapide, pas de saturation pour $x > 0$
  - Inconvénient : neurones morts si $x < 0$ constamment

• Tanh : $f(x) = (e^x - e^{-x})/(e^x + e^{-x})$
  - Plage de sortie : [-1, 1]
  - Centré sur 0, meilleur que sigmoïde pour hidden layers

CHAPITRE 2 : Algorithme de Backpropagation

2.1 Principe de la Rétropropagation

La backpropagation permet de calculer les gradients de la fonction de coût par rapport aux poids du réseau. C'est l'algorithme clé pour l'entraînement des réseaux de neurones profonds.

Étapes de l'algorithme :

1. Forward Pass : calculer les activations de toutes les couches
2. Calcul de l'erreur : comparer la sortie avec la vraie valeur (loss)
3. Backward Pass : propager l'erreur en arrière pour calculer $\partial L/\partial w$
4. Mise à jour des poids : $w_{nouveau} = w_{ancien} - \eta \times \partial L/\partial w$

où η est le learning rate (taux d'apprentissage).

2.2 Fonction de Coût

Pour la régression : Mean Squared Error (MSE)
$L = (1/n)\ \Sigma(y_{pred} - y_{true})^2$

Pour la classification binaire : Binary Cross-Entropy
$L = -[y \times \log(\hat{y}) + (1-y) \times \log(1-\hat{y})]$

Pour la classification multiclasse : Categorical Cross-Entropy
$L = -\Sigma\ y_i \times \log(\hat{y}_i)$

CHAPITRE 3 : Architecture des Réseaux Profonds

3.1 Multilayer Perceptron (MLP)

Un MLP est composé de :
• Une couche d'entrée (input layer)
• Une ou plusieurs couches cachées (hidden layers)

• Une couche de sortie (output layer)

Exemple d'architecture pour classification d'images MNIST :
- Input : 784 neurones (28×28 pixels)
- Hidden 1 : 128 neurones + ReLU
- Hidden 2 : 64 neurones + ReLU
- Output : 10 neurones + Softmax (chiffres 0-9)

3.2 Régularisation

Pour éviter l'overfitting (surapprentissage) :

• Dropout : désactiver aléatoirement 20-50% des neurones pendant l'entraînement. Proposé par Srivastava et al. en 2014.

• L2 Regularization (Weight Decay) : ajouter $\lambda||w||^2$ à la fonction de coût pour pénaliser les poids trop grands.

• Early Stopping : arrêter l'entraînement quand la validation loss commence à augmenter.

3.3 Batch Normalization

Technique introduite par Ioffe & Szegedy (2015) pour accélérer l'entraînement et stabiliser l'apprentissage. Normalise les activations de chaque mini-batch :

$$x\_norm = (x - \mu\_batch) / \sqrt{(\sigma^2\_batch + \varepsilon)}$$

Avantages :
- Permet d'utiliser des learning rates plus élevés
- Réduit la sensibilité à l'initialisation
- Agit comme régularisateur

CHAPITRE 4 : Optimiseurs Avancés
═══════════════════════════════

4.1 Gradient Descent Classique

$$w\_t+1 = w\_t - \eta \times \nabla L(w\_t)$$

Problème : peut osciller dans les vallées étroites, convergence lente.

4.2 Momentum

Accumule une "vitesse" dans les directions du gradient :

$$v_t = \beta \times v_{t-1} + (1-\beta) \times \nabla L(w_t)$$
$$w_{t+1} = w_t - \eta \times v_t$$

Typiquement $\beta = 0.9$. Accélère la convergence dans les ravins.

4.3 Adam (Adaptive Moment Estimation)

Combine momentum et RMSprop. L'optimiseur le plus utilisé en 2025 :

$$m_t = \beta_1 \times m_{t-1} + (1-\beta_1) \times \nabla L(w_t)$$
$$v_t = \beta_2 \times v_{t-1} + (1-\beta_2) \times (\nabla L(w_t))^2$$

$$w_{t+1} = w_t - \eta \times m_t / (\sqrt{v_t} + \varepsilon)$$

Hyperparamètres par défaut : $\beta_1=0.9$, $\beta_2=0.999$, $\varepsilon=10^{-8}$, $\eta=0.001$

EXERCICES PRATIQUES

Exercice 1 : Implémenter un perceptron en NumPy
Créer une classe Perceptron avec méthodes fit() et predict().
Tester sur le dataset Iris (classification binaire).

Exercice 2 : Backpropagation manuelle
Calculer à la main les gradients pour un réseau 2-2-1 avec MSE loss.

Exercice 3 : Comparaison d'optimiseurs
Entraîner un MLP sur MNIST avec SGD, Momentum, et Adam.
Comparer les courbes de loss et accuracy.

RESSOURCES COMPLÉMENTAIRES

- Goodfellow et al., "Deep Learning" (2016) - Chapitres 6-8
- Stanford CS231n : Convolutional Neural Networks
- Documentation Keras/PyTorch pour implémentations pratiques
- Article original Adam : Kingma & Ba (2014)

Prochain cours : CNN et Vision par Ordinateur
Séance 4 - 22 Octobre 2025