Du Deep Learning aux Transformers

Les équations utilent pour l'Intelligence Artificielle

Georf MIGUIAMA BAMBA

Janvier 2026

Table des matières

Introduction

Ce mini-book est né d'une volonté de démystifier l'Intelligence Artificielle. Derrière les termes complexes de *Deep Learning* ou de *Transformers*, se cachent des concepts mathématiques élégants et souvent très proches de notre quotidien. Que vous soyez un ingénieur ou un curieux, ce guide fait le pont entre le code Python, l'équation formelle et la vie réelle.

Chapitre 1

La Fondation : Le Gradient et la Dérivée

Toute Intelligence Artificielle apprend en mesurant son erreur. La dérivée est l'instrument qui permet de savoir dans quelle direction s'améliorer.

1.1 L'Équation

$$f'(x) = \lim_{h \to 0} \frac{f(x+h) - f(x)}{h} \tag{1.1}$$

1.2 Code Python

```
def gradient(f, x, h=0.0001):
    return (f(x + h) - f(x)) / h
```

1.3 Analogie : L'accélérateur

C'est comme appuyer sur la pédale de votre voiture : le gradient vous dit exactement de combien votre vitesse augmente pour chaque millimètre de pression supplémentaire sur la pédale.

Si vous augmentez le chauffage de 1 degré, à quelle vitesse la sensation de confort change-t-elle ?

Chapitre 2

Le Neurone et l'Activation

Le neurone est l'unité de décision de base.

2.1 La Somme Pondérée

$$z = \sum_{i=1}^{n} (w_i \cdot x_i) + b \tag{2.1}$$

2.2 L'Activation (ReLU)

$$f(z) = \max(0, z) \tag{2.2}$$

2.3 Code Python

```
inputs = [2.0, 3.0]
weights = [0.8, -0.5]
bias = 1.2

z = sum(w*x for w, x in zip(weights, inputs)) + bias
print(f"Activation du neurone : {z}")
```

2.4 Analogie : Le Recruteur

Un recruteur donne des points à votre diplôme (x_1) et votre expérience (x_2). Si le score total dépasse son seuil d'exigence (biais b), il vous appelle. Sinon (si le score est négatif), l'activation ReLU transforme cela en 0 : vous n'êtes pas retenu.

La Recette de Cuisine : Chaque ingrédient a un poids. Si vous mettez trop de sel (w négatif), la note globale du plat va descendre.

Chapitre 3

La Fonctoin d'Activation (ReLU/Tanh)

Elles permettent au réseau d'apprendre des relations complexes (non-linéaires).

3.1 Les Équations

ReLU : $f(x) = \max(0, x)$
Tanh : $f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

3.2 Code Python

```
def relu(x):
    return max(0, x)

def tanh(x):
    return (np.exp(x) - np.exp(-x)) / (np.exp(x) + np.exp(-x))
```

3.3 Analogie : Le Seuil de Douleur

Si on vous touche la main très doucement, vous n'allez rien ressentir (0). Dès que la pression dépasse un certain seuil, le signal de "douleur" s'active et monte proportionnellement.

Chapitre 4

La Règle de la Chaîne (Backpropagation)

C'est le mécanisme qui permet de propager l'erreur à travers toutes les étapes du calcul.

4.1 L'Équation

$$\frac{dz}{dx} = \frac{dz}{dy}.\frac{dy}{dx} \tag{4.1}$$

4.2 Code Python

```
# Dans micrograd, cela se traduit par l'accumulation des gradients
def backward(self):
    self.grad += out.grad * other.data
```

4.3 Analogie : L'Effet Papillon

Une petite erreur dans la livraison des pneus à l'usine (dy/dx) entraîne un retard dans la fabrication de la voiture (dz/dy). La règle de la chaîne permet de savoir de combien l'erreur initiale a impacté la date de livraison finale.

Chapitre 5

La Fonction de Perte (Loss Function)

Elle mesure à quel point l'Intelligence Artificielle se trompe.

$$L'quation(MSE) : L = \frac{1}{n}\sum(y_{pred} - y_{target})^2 \tag{5.1}$$

5.1 Code Python

```
target = 10.0
prediction = 8.5
loss = (prediction - target)**2
print(f"L'erreur au carré est : {loss}")
```

5.2 Analogie : Le Tir à l'Arc

La distance entre votre flèche et le centre de la cible est votre "Loss". Plus vous êtes loin, plus le score est mauvais. On utilise le carré (^{2}) pour que les grandes erreurs soient beaucoup plus punies que les petites.

Chapitre 6

Probabilités et Bigrammes

Avant d'écrire des essais, l'IA a appris à prédire une lettre après l'autre.

6.1 L'Équation

$$P(w_t|w_{t-1}) = \frac{C(w_{t-1}, w_t)}{C(w_{t-1})} \tag{6.1}$$

6.2 Code Python

```
# Tableau de comptage des paires de caractères
probs = torch.randn(27, 27).exp()
probs /= probs.sum(1, keepdim=True)
# Prédire le suivant
next_char = torch.multinomial(probs[current_char], 1).item()
```

6.3 Analogie : Le Clavier Prédictif

C'est la statistique pure. Si vous écrivez "Bonjour", votre téléphone sait que le mot "ça" a une forte probabilité de suivre, simplement parce qu'il l'a vu des millions de fois auparavant.

Chapitre 7

Le MLP (Perceptron Multicouche)

L'organisation des neurones en couches successives.

7.1 L'Équation

$$L'quation : y = f(W_2.f(W_1.x + b_1) + b_2) \tag{7.1}$$

7.2 Code Python

```
h = (x @ W1 + b1).tanh() # Couche cachée
logits = h @ W2 + b2     # Sortie
```

7.3 Analogie : Le Conseil d'Administration

Une première équipe d'experts analyse les données brutes (Couche 1). Ils transmettent leurs avis à un comité de direction (Couche 2) qui prend la décision finale. Plusieurs couches permettent de traiter des problèmes plus subtils qu'un simple oui/non.

Chapitre 8

Batch Normalization (BN)

Stabiliser l'apprentissage en recentrant les données.

8.1 L'Équation

$$L'quation : \hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \tag{8.1}$$

8.2 Code Python

```
# Stabiliser les activations pour qu'elles ne soient pas trop
   grandes
x_norm = (x - x.mean()) / (x.std() + 1e-5)
out = gamma * x_norm + beta
```

8.3 Analogie : L'Égaliseur Audio

Dans un orchestre, si les trompettes jouent trop fort et les violons trop doucement, on ne comprend rien. La BN "normalise" le volume de chaque instrument pour que le chef d'orchestre (le neurone suivant) reçoive un signal équilibré et stable.

Chapitre 9

La Backpropagation (Rétropropagation)

L'algorithme qui permet au réseau d'apprendre de ses erreurs.

9.1 L'Équation

$$L'quation : \delta^L = \nabla_a C \odot \sigma'(z^L) \tag{9.1}$$

9.2 Code Python

```
# Remonter le temps pour corriger les erreurs
loss.backward()
optimizer.step()
```

9.3 Analogie : Le Débriefing Militaire ou Sportif

Après un match perdu (Loss), on regarde la vidéo à l'envers. "Le but a été encaissé à cause du défenseur, qui était mal placé à cause d'une mauvaise passe du milieu de terrain". On identifie le responsable de l'erreur à chaque étape pour corriger lors de l'entraînement.

Chapitre 10

La Vision : La Convolution (CNN : Réseaux de Neurones Convolutifs)

Pour voir, l'Intelligence Artificielle balaie l'image à la recherche de motifs.

10.1 L'Équation de Convolution

$$(f * g)(t) = \int_{-\infty}^{\infty} f(\tau)g(t-\tau)d\tau \tag{10.1}$$

10.2 Code Python

```
# Un filtre qui passe sur l'image pour détecter des bords
conv = nn.Conv2d(in_channels=3, out_channels=16, kernel_size=3)
features = conv(image)
```

10.3 Analogie : La Lampe de Poche

C'est comme explorer une pièce sombre avec une lampe de poche étroite. Vous ne voyez pas tout d'un coup, mais en balayant, vous identifiez des bords, puis des formes, et enfin des objets.

Chapitre 11

Stabilité : Batch Normalisation

Pour apprendre vite, le réseau doit rester stable.

11.1 L'Équation

$$\hat{x} = \frac{x - E[x]}{\sqrt{Var[x] + \epsilon}} \tag{11.1}$$

11.2 Code Python

```
# Stabiliser les activations pour qu'elles ne soient pas trop
   grandes
x_norm = (x - x.mean()) / (x.std() + 1e-5)
out = gamma * x_norm + beta
```

11.3 Analogie : L'Égaliseur Audio

Dans un orchestre, si les percussions couvrent les violons, on ne comprend plus la musique. La Batch Norm ajuste le "volume" de chaque neurone pour que le signal reste clair tout au long du réseau.

Chapitre 12

L'Apogée : Le Transformer et l'Attention (tout ce dont on a besoin)

L'architecture derrière ChatGPT.

12.1 Self-Attention

$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{12.1}$$

12.2 Code Python

```
# Le mécanisme qui permet au modèle de se concentrer sur les mots
   importants
attn_weights = (query @ key.transpose(-2, -1)) * (d_k**-0.5)
attn_weights = F.softmax(attn_weights, dim=-1)
output = attn_weights @ value
```

12.3 Analogie : La Lecture Sélective

Quand vous lisez une phrase, votre cerveau ne donne pas la même importance à chaque mot. Dans "Le chat mange la souris", votre cerveau lie "mange" au "chat". Le Transformer calcule mathématiquement quels mots dans une phrase sont les plus importants les uns pour les autres.

Chapitre 13

L'Apogée : Le Transformer et l'Attention

L'architecture derrière ChatGPT.

$$P_i = \frac{\exp(y_i/T)}{\sum \exp(y_j/T)} \tag{13.1}$$

Une température élevée ($T > 1$) rend l'IA plus audacieuse et imprévisible, comme un barman qui improvise un cocktail.

13.1 Code Python

```
# Ajuster la créativité de l'IA
logits = model(x) / temperature
probs = F.softmax(logits, dim=-1)
next_token = torch.multinomial(probs, num_samples=1)
```

13.2 Analogie : Le Barman Créatif

Si la température (T) est basse, le barman vous sert toujours la boisson la plus populaire (choix sûr, mais répétitif). Si la température est haute, il devient audacieux et choisit des ingrédients au hasard (plus créatif, mais risque de faire n'importe quoi).
Rappel : Une température élevée ($T > 1$) rend l'IA plus audacieuse et imprévisible, comme un barman qui improvise un cocktail.

Conclusion

L'Intelligence Artificielle n'est pas une boîte noire magique. C'est une suite logique d'équations, de la simple pente d'une courbe au mécanisme d'attention complexe. Comprendre ces bases, c'est reprendre le contrôle sur les outils de demain. Ce voyage, du pixel à la pensée, ne fait que commencer.