Des réseaux de neurones peu profonds aux réseaux profonds

ARN

7 - Deep Neural Networks

Abstract

Table des matières

1. Des Réseaux de Neurones Peu Profonds aux Réseaux Profonds

1.1. Évolution Historique des Réseaux de Neurones

1.1.1. L'Ère des Réseaux Peu Profonds (Avant 2006)

Théorème d'approximation universelle (Cybenko, 1989) : Les réseaux de neurones avec une **seule couche cachée** peuvent représenter une grande variété de fonctions intéressantes avec des paramètres appropriés.

Problème majeur : Avant 2006, l'entraînement de réseaux profonds donnait de **moins bons résultats** que les réseaux peu profonds (1-2 couches cachées), tant sur les données d'entraînement que de test.

Conséquence : La communauté scientifique privilégiait les architectures peu profondes par pragmatisme.

1.1.2. Stratégie "Large mais Peu Profond"

Face aux limitations de profondeur, les chercheurs ont développé une approche alternative :

Modèles d'ensemble : Combinaison de multiples réseaux peu profonds pour améliorer les performances.

Trois approches principales :
- **Instances multiples** : même modèle entraîné plusieurs fois sur les mêmes données
- **Sous-ensembles de données** : même modèle sur différents échantillons d'entraînement
- **Architectures variées** : différentes structures de réseaux peu profonds

1.1.3. Réseaux d'Ensemble : Principe Mathématique

Formulation : Pour N modèles individuels produisant des sorties $y_1, y_2, ..., y_N$:

Classification :

$$y_{\text{ensemble}} = \text{argmax}_c \sum_{i=1}^{N} \delta(y_i = c)$$

Régression :

$$y_{\text{ensemble}} = \frac{1}{N} \sum_{i=1}^{N} y_i$$

Avantage fondamental : Les erreurs individuelles des modèles ont tendance à se "compenser" mutuellement, réduisant l'erreur globale.

1.2. Le Problème du Gradient Qui Disparaît

1.2.1. Analyse Mathématique du Problème

Pour un réseau à L couches, le gradient de l'erreur par rapport aux poids des premières couches :

$$\frac{\partial E}{\partial W_{L-2}} = \frac{\partial E}{\partial y_L} \frac{\partial y_L}{\partial y_{L-1}} \frac{\partial y_{L-1}}{\partial y_{L-2}} \frac{\partial y_{L-2}}{\partial W_{L-2}}$$

Décomposition avec fonctions d'activation :

$$\frac{\partial E}{\partial W_{L-2}} = \frac{\partial E}{\partial y_L} f'_{L()} \frac{\partial a_L}{\partial y_{L-1}} f'_{L-1}() \frac{\partial a_{L-1}}{\partial y_{L-2}} f'_{L-2}() \frac{\partial a_{L-2}}{\partial W_{L-2}}$$

1.2.2. Impact de la Fonction Sigmoïde

Propriété critique : Pour la sigmoïde, $0 < f'(x) < 0.25$

Calcul du produit maximal :

$$\max\Big(f'_{L()} \times f'_{L-1}() \times f'_{L-2}()\Big) = 0.25^3 \approx 0.016$$

Conséquence dramatique : Le gradient devient exponentiellement petit, ne fournissant plus d'information utile pour ajuster les poids des premières couches.

1.3. Révolution des Fonctions d'Activation

1.3.1. Fonction ReLU : La Solution

Définition mathématique :

$$\text{ReLU}(x) = \max(0, x) = \begin{cases} x \text{ si } x > 0 \\ 0 \text{ si } x \leq 0 \end{cases}$$

Dérivée :

$$\text{ReLU}'(x) = \begin{cases} 1 \text{ si } x > 0 \\ 0 \text{ si } x \leq 0 \\ \text{indéfini en } x = 0 \end{cases}$$

Avantages décisifs :
- **Gradient non-saturant** : la dérivée vaut 1 pour $x > 0$
- **Calcul ultra-rapide** : simple comparaison et sélection
- **Parcimonie naturelle** : de nombreuses activations sont nulles
- **Inspiration biologique** : les neurones ne transmettent que des signaux positifs

1.3.2. Leaky ReLU : Amélioration

Formulation :

$$\text{LeakyReLU}(x) = \max(\alpha x, x)$$

où α est un petit coefficient (ex: 0.01)

Dérivée :

$$\text{LeakyReLU}'(x) = \begin{cases} 1 \text{ si } x > 0 \\ \alpha \text{ si } x < 0 \end{cases}$$

Avantage : Évite les "neurones morts" en permettant un gradient non-nul pour les valeurs négatives.

1.4. L'Émergence des Réseaux Profonds

1.4.1. Les Pionniers du Deep Learning

Figures clés : Geoffrey Hinton (Toronto & Google), Yann LeCun (NYU & Facebook), Yoshua Bengio (Montréal), Jürgen Schmidhuber (IDSIA & USI, Suisse)

Contexte historique : Les idées pionnières ont été publiées dans les années 1980-1990, mais leur application pratique a dû attendre les années 2000.

Performance : Ces techniques permettent désormais les **meilleures performances actuelles** sur les benchmarks de traitement d'images et bien d'autres problèmes.

1.4.2. Deux Approches Initiales

Deep Belief Networks (DBN) :
- Basés sur des machines de Boltzmann restreintes (RBM)
- Connexions bidirectionnelles et symétriques
- Apprentissage non-supervisé récursif de détecteurs de caractéristiques
- **Verdict** : Finalement moins performants que prévu

Convolutional Neural Networks (CNN) :
- Couches multiples de convolutions (filtres spatiaux)
- Couches de sous-échantillonnage
- Couches entièrement connectées
- **Succès** : Base de nombreuses solutions état de l'art

1.5. Technique d'Entraînement : Auto-encodeurs Empilés

1.5.1. Méthode de Pré-entraînement Couche par Couche

Étape 1 : Entraîner un auto-encodeur sur les deux premières couches
- **Objectif** : reproduire l'entrée à la sortie
- **Architecture** : entrée → couche cachée → reconstruction de l'entrée

Étape 2 : Fixer les poids appris et répéter pour la couche suivante

Étape 3 : Affinage final par rétropropagation classique sur l'ensemble du réseau

Avantage : Cette approche permet d'initialiser intelligemment les poids avant l'entraînement supervisé final.

1.6. CNN : De Peu Profond à Profond

1.6.1. Comparaison Architecturale

Réseau peu profond traditionnel :

$$\text{Image} \rightarrow \text{Aplatissement} \rightarrow \text{MLP} \rightarrow \text{Classification}$$

CNN profond moderne :

$$\text{Image} \rightarrow [\text{Conv} + \text{ReLU} + \text{Pool}] \times N \rightarrow \text{Flatten} \rightarrow \text{Dense} \rightarrow \text{Softmax}$$

Innovation clé : L'extraction automatique de caractéristiques hiérarchiques remplace l'ingénierie manuelle de features.

1.6.2. Traitement Spatial et Partage de Poids

Principe fondamental : Chaque neurone traite séquentiellement de petites régions de l'image avec les **mêmes poids**.

Avantages du partage de poids :
- **Réduction drastique des paramètres** : de millions à quelques milliers
- **Invariance par translation** : détection indépendante de la position
- **Généralisation** : ce qui est appris localement s'applique globalement

Formulation mathématique : Pour un filtre W et une position (x, y) :

$$O(x, y) = \sum_{i,j} I(x + i, y + j) \times W(i, j) + b$$

1.7. Paramètres de Convolution

1.7.1. Stride (Pas) et Padding (Rembourrage)

Stride = 1 avec zero-padding :
- Conservation de la taille de l'image d'entrée
- Exploration exhaustive de tous les pixels

Stride = 2 sans padding :
- Réduction de taille par facteur 2
- Calcul plus rapide, moins de redondance

Formule de taille de sortie :

$$O_{\text{size}} = \left\lfloor \frac{I_{\text{size}} + 2P - K}{S} \right\rfloor + 1$$

Où :
- I_{size} : taille d'entrée
- P : padding
- K : taille du noyau
- S : stride

1.7.2. Convolutions Multiples et Détection de Caractéristiques

Principe : Chaque "neurone" applique un filtre particulier à l'image complète pour détecter une caractéristique spécifique.

Pour images RGB : Chaque neurone additionne les sorties des filtres appliqués sur chaque canal :

$$O(x,y) = \sum_{c \in \{R,G,B\}} I_{c(x,y)} \star W_c + b$$

Objectif d'apprentissage : Le réseau apprend automatiquement les filtres optimaux pour la tâche (ex: détection d'yeux, oreilles, museau pour reconnaître des chats).

1.8. Détection Hiérarchique de Caractéristiques

1.8.1. Progression des Niveaux d'Abstraction

Première couche : Détection de caractéristiques de bas niveau
- Contours, edges, coins
- Textures basiques

Couches intermédiaires : Combinaisons de caractéristiques de bas niveau
- Formes géométriques simples
- Motifs plus complexes

Couches supérieures : Caractéristiques de haut niveau
- Parties d'objets (yeux, oreilles pour les animaux)
- Objets complets

Formulation : Pour la couche l avec N_{l-1} entrées de la couche précédente :

$$O^{l(x,y)} = f\left(\sum_{k=1}^{N_{l-1}} I^{l-1}_{k(x,y)} \star W^l_k + b^l\right)$$

1.8.2. Convolutions dans les Couches Intermédiaires

Processus : Si la couche précédente produit 10 cartes de caractéristiques, chaque neurone de la couche suivante :
1. Applique un filtre sur chaque carte d'entrée
2. Additionne toutes les sorties filtrées
3. Ajoute un biais
4. Applique une fonction d'activation
5. Produit une nouvelle carte de caractéristiques

1.9. Fonctions d'Activation Modernes

1.9.1. Fonction Softmax pour la Classification

Définition mathématique :

$$\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{K} e^{x_j}} \quad \text{pour } j = 1, ..., K$$

Propriétés :

$$\sum_{i=1}^{K} \text{Softmax}(x_i) = 1 \quad \text{et} \quad 0 \leq \text{Softmax}(x_i) \leq 1$$

Interprétation probabiliste :

$$P(\text{classe} = i \mid \text{entrées}, \text{poids}, \text{biais})$$

Usage : Couche de sortie pour problèmes de classification multi-classes.

1.10. Régularisation et Prévention du Surapprentissage

1.10.1. Technique du Dropout

Principe (Hinton et al., 2012) : Désactiver aléatoirement des neurones pendant l'entraînement pour prévenir le surapprentissage.

Mécanisme : Avec probabilité p (ex: 0.5), un neurone est temporairement "supprimé" du réseau.

Effet : Les autres neurones doivent apprendre à compenser l'absence du neurone désactivé, forçant une représentation plus robuste et distribuée.

Implémentation : Pendant l'entraînement seulement ; à l'inférence, tous les neurones sont actifs.

1.10.2. Stratégies Anti-Surapprentissage

Liste des techniques efficaces :

1. **Augmentation des données** : créer des variations artificielles des données d'entraînement
2. **Augmentation de données** : rotation, translation, zoom, recadrage, retournement, perturbation colorimétrique
3. **Architectures robustes** : utiliser des modèles connus pour bien généraliser
4. **Régularisation L1/L2** : pénaliser les poids trop importants
5. **Normalisation par batch** : stabiliser l'entraînement
6. **Réduction de complexité** : simplifier l'architecture si nécessaire

1.10.3. Augmentation de Données (Data Augmentation)

Transformations géométriques :

- **Rotation** :

$$[x', y'] = [x\cos\theta - y\sin\theta, x\sin\theta + y\cos\theta]$$

- **Translation** :

$$[x', y'] = [x + \Delta x, y + \Delta y]$$

- **Zoom/Recadrage** : modification d'échelle
- **Retournement** : symétrie horizontale/verticale

Transformations colorimétriques :

- Perturbation des canaux RGB
- Modification de luminosité/contraste
- Dé-texturation et amélioration des contours

1.11. Le Défi de la Sélection de Modèle

1.11.1. Questions Architecturales

Choix structurels critiques :

- **Nombre de couches convolutionnelles** : profondeur du réseau
- **Nombre de filtres par couche** : largeur du réseau
- **Tailles de filtres** : 3×3, 5×5, 7×7, etc.
- **Placement des couches de pooling** : où réduire la dimensionnalité
- **Fonctions d'activation** : ReLU, Leaky ReLU, autres
- **Configuration des couches denses finales** : nombre et taille

1.11.2. Paramètres d'Apprentissage

Optimisation :

- **Optimiseur** : SGD, Adam, RMSprop, AdaGrad
- **Fonction de perte** : entropie croisée, MSE, autres
- **Taux d'apprentissage** : souvent le paramètre le plus critique
- **Taille de batch** : compromis vitesse/qualité de convergence
- **Nombre d'époques** : équilibre entre sous et sur-apprentissage

- **Régularisation** : L1, L2, dropout, normalisation par batch

1.12. Impact et Applications

1.12.1. Succès Médiatiques du Deep Learning

Moments historiques de l'IA :

1. **Deep Blue vs Kasparov** : victoire de l'IA aux échecs
2. **Watson à Jeopardy** : compréhension du langage naturel
3. **Challenge Kaggle Merck** (nov. 2012) : étudiants utilisant le deep learning remportent le défi d'activité moléculaire

Démocratisation : Outils interactifs permettant d'expérimenter le deep learning directement dans le navigateur.

1.12.2. Hiérarchies de Caractéristiques Apprises

Reconnaissance d'images :

$$\text{pixel} \rightarrow \text{edge} \rightarrow \text{texton} \rightarrow \text{motif} \rightarrow \text{partie} \rightarrow \text{objet}$$

Traitement de texte :

$$\text{caractère} \rightarrow \text{mot} \rightarrow \text{groupe de mots} \rightarrow \text{clause} \rightarrow \text{phrase} \rightarrow \text{histoire}$$

Traitement de la parole :

$$\text{échantillon} \rightarrow \text{bande spectrale} \rightarrow \text{son} \rightarrow \text{phonème} \rightarrow \text{mot} \rightarrow ...$$

Principe de Jeff Dean (Google) : "Tout ce que les humains peuvent faire en 0.1 seconde, un bon réseau à 10 couches peut aussi le faire."

1.13. Conclusion

La transition des réseaux peu profonds aux réseaux profonds représente une révolution majeure en intelligence artificielle. Cette évolution s'appuie sur plusieurs innovations clés :

Innovations techniques décisives :

- Nouvelles fonctions d'activation (ReLU) résolvant le problème du gradient qui disparaît
- Techniques de régularisation (dropout) prévenant le surapprentissage
- Méthodes d'entraînement sophistiquées (auto-encodeurs empilés)
- Augmentation de données pour améliorer la généralisation

Impact transformateur : Les réseaux profonds, particulièrement les CNN, ont révolutionné la vision par ordinateur et établi les fondations de l'IA moderne, ouvrant la voie aux succès actuels dans de nombreux domaines d'application.