Introduction aux réseaux de neurones profonds

Machine Learning & Big Data

Claude Barras (LIMSI, U. Paris-Sud)

Nov. 2016

Introduction

Le cerveau humain

- plus de 80 milliards de neurones
- en moyenne 10.000 synapses par neurone
- traitement massivement parallèle
- 20-30 W de consommation

Système visuel des mammifères

- Temps de traitement $\sim 1/10$ sec

Copier le cerveau ?

- plutôt une source d'inspiration

Machines linéaires

Perceptron (Rosenblatt, 1957)

- somme pondérée + seuillage

$$y = \text{sign}\left(\sum_{i=1}^{p} w_i.x_i + b\right) = \text{sign}(\bar{\mathbf{w}}^t.\bar{\mathbf{x}})$$

avec $\bar{\mathbf{x}} = [1, x_1 \cdots x_p]^t$ et $\bar{\mathbf{w}} = [b, w_1 \ldots w_p]^t$

Optimisation des paramètres

- descente du gradient stochastique (estimation bruitée de la direction à suivre après chaque échantillon d'apprentissage)

$$w' \leftarrow w - \eta \frac{\partial E(\bar{\mathbf{w}}^t.\bar{\mathbf{x}})}{\partial w}$$

Machines linéaires

Regression linéaire $y = \bar{\mathbf{w}}^t.\bar{\mathbf{x}}$

- idem au perceptron mais sans le seuillage
- fonction de coût à minimiser = critère des moindres carrés

$$L = \frac{1}{2}\left(y^{(i)} - \bar{\mathbf{w}}^t.\bar{\mathbf{x}}^{(i)}\right)^2$$

Régression logistique $y = F(\bar{\mathbf{w}}^t.\bar{\mathbf{x}})$

- avec F une fonction sigmoïde (en forme de S)
  - fonction logistique entre 0 et 1 ou tanh entre -1 et 1
- optimisation d'un critère basé sur la log-vraisemblance

Machines linéaires

Limitation

- séparation des données uniquement par des hyper-plans
- Th. de Cover (1966) : la probabilité que N échantillons de dim. d soient linéairement séparables tend rapidement vers 0 quand $N \gg d$

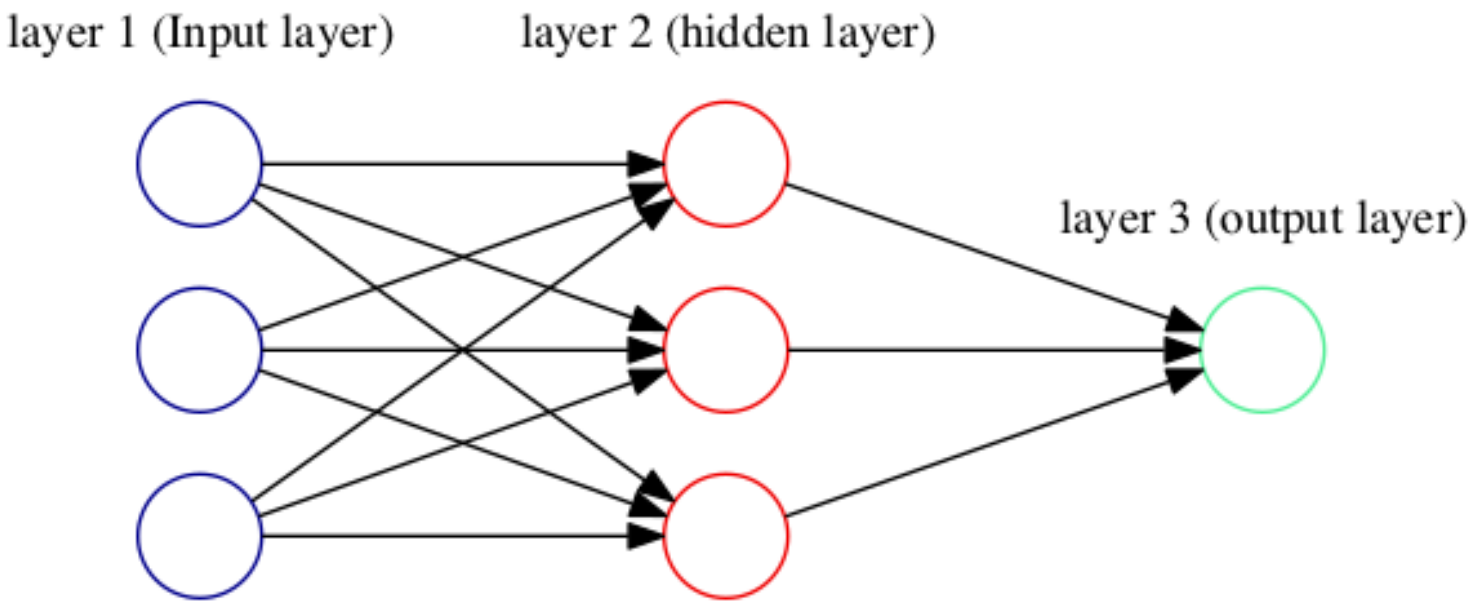
Solution

- augmenter la dimension de l'espace par une fonction $\Phi(x)$ **non linéaire**
- ex: expansion polynomiale $=>$ séparation par coniques
- fonctions à noyaux et SVM
- réseau de neurones avec (au moins) 1 couche cachée

Perceptron multi-couches

Réseau à une couche cachée

- fonction d'activation **non-linéaire** entre les couches intermédiaires
- permet (théoriquement) d'apprendre toute fonction de $\mathbb{R}^p \rightarrow \mathbb{R}^q$
- exemple de classifieur avec une sortie



Perceptron multi-couches

Reconnaissance

- propagation de l'entrée vers la sortie
  - $y_i = f^{(1)}\left(\sum_j w_{ij}^{(1)} x_j\right)$
  - $z_i = f^{(2)}\left(\sum_j w_{ij}^{(2)} y_j\right)$

Apprentissage

- base d'exemples d'apprentissage avec la sortie attendue
- fonction de coût (erreur entre la sortie attendue et réalisée)
- rétro-propagation du gradient de l'erreur depuis la couche de sortie vers la couche d'entrée

Fonctions d'activations

Sigmoïde

- fonction logistique (de 0 à 1) $f(z) = 1/(1 + e^{-z})$
- tangente hyperbolique (entre -1 et 1)

$$f(z) = \frac{e^{2z} - 1}{e^{2z} + 1}$$

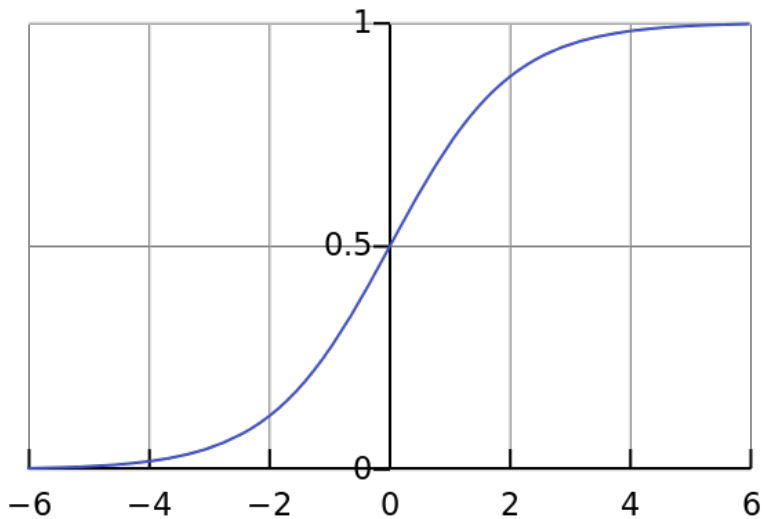


Figure 1: fonction logistique (wikimedia.com)

Fonctions d'activations

Rectified linear unit (ReLU)

- fonction $f(z) = \max(z, O)$
- très simple d'emploi (malgré la non-différenciation au point zéro)
- très efficace avec les réseaux profonds (évite la saturation du gradient)

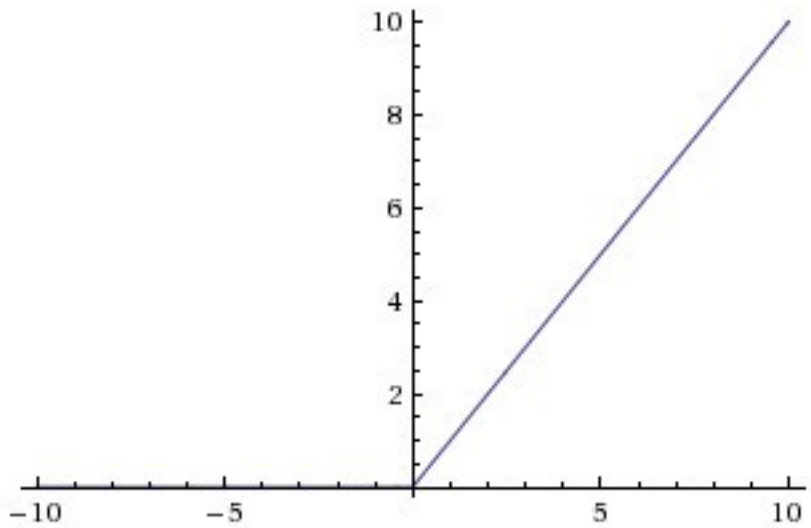


Figure 2: ReLU (cs231n.github.io)

Fonctions d'activations

Softmax

- $f(z_i) \in [0, 1]$ et t.q. $\sum_i f(z_i) = 1$

$$f(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{J} e^{z_j}}$$

- transforme un ensemble de scores en probabilités
- utilisé typiquement en sortie d'un classifieur multi-classes

LogSoftMax

- idem mais travaille dans le domaine logarithmique (évite les problèmes de résolution numérique)

$$f(z_i) = z_i - \log\left(\sum_{j=0}^{J} e^{z_j}\right)$$

Apprentissage profond (*deep learning*)

Modèle traditionnel en reconnaissance des formes

- Extraction de features (complexe, fondée sur l'expertise)
- Classifieur simple

Modèle par apprentissage profond

- Cascade de modules eux-mêmes appris automatiquement
- Chaque couche avec des transformations non linéaires des paramètres
- Hiérarchie de niveaux de représentation avec un degré croissant d'abstraction

Réseau 2 couches vs. multi-couches

- Théoriquement équivalents
- En pratique, augmentation exponentielle du nombre de paramètres dans une architecture 2 couches
- L'architecture multi-couches est beaucoup plus efficace en mémoire (sans être beaucoup plus lente)

Deep Learning = Learning Hierarchical Representations

Y LeCun

- **Traditional Pattern Recognition:** Fixed/Handcrafted Feature Extractor

Feature Extractor → Trainable Classifier →

- **Mainstream Modern Pattern Recognition:** Unsupervised mid-level features

Feature Extractor → Mid-Level Features → Trainable Classifier →

- **Deep Learning:** Representations are hierarchical and trained

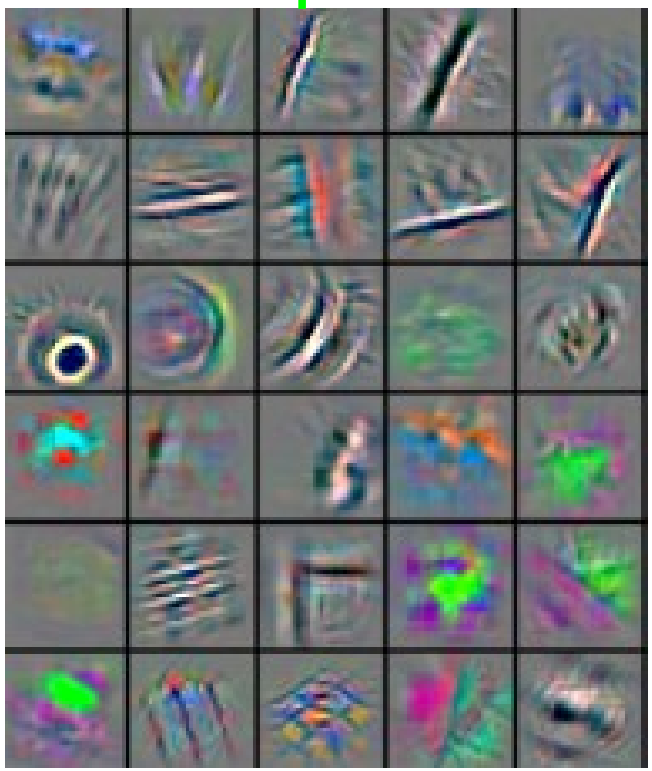
Low-Level Features → Mid-Level Features → High-Level Features → Trainable Classifier →

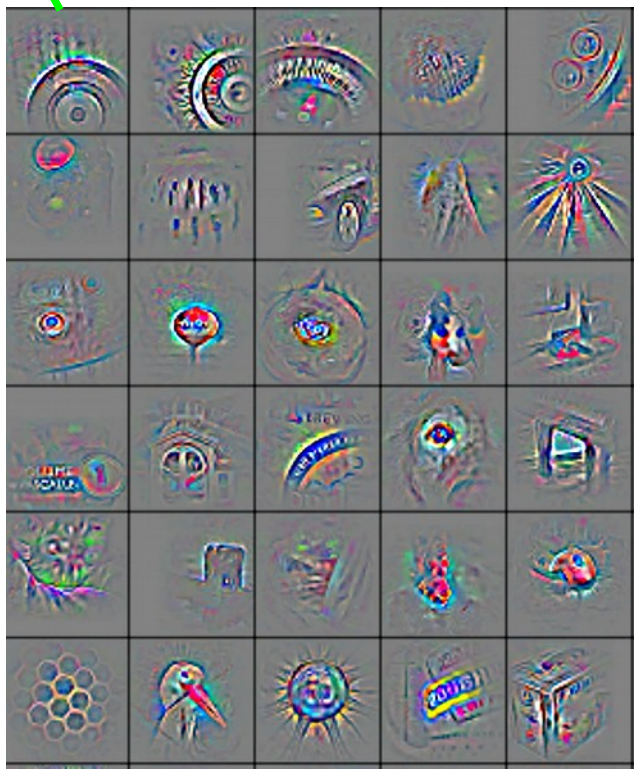
Deep Learning = Learning Hierarchical Representations

Y LeCun

- It's deep if it has more than one stage of non-linear feature transformation

Low-Level Feature → Mid-Level Feature → High-Level Feature → Trainable Classifier





Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

Intérêt des architectures neuronales

Architecture en blocs

- simplicité algorithmique
- principalement du calcul matriciel
- gros apport des GPU en vitesse de traitement

Nombreuses librairies logicielles

- Torch `http://torch.ch/` et `https://github.com/torch/torch7/wiki/Cheatsheet`
- TensorFlow `https://www.tensorflow.org/`
- Theano `http://www.deeplearning.net/software/theano/`

Difficultés

Gain par rapport aux approches traditionnelles

- le *deep learning* ne se justifie vraiment qu'avec de (très) grosses quantités de données d'apprentissage (*big data*)

Apprentissage stochastique

- l'ordre de présentation des exemple d'apprentissage est critique
- stratégies de sélection des exemples (mal classés, à la frontière. . .)
- la convergence est souvent longue à atteindre

Difficultés

Nombreuses architectures possibles

- nombre de couches cachées, nombre de neurones par couche
- choix des fonctions d'activation (ReLU plus stable que sigmoïde)
- partage des paramètres

Expérience empirique

- bien mélanger les exemples
- normaliser les entrées (moyenne et variance)
- régularisation de l'apprentissage (dropout)

Réseaux convolutifs (ConvNet ou CNN)

Architecture proposée par Y. Le Cun (1989)

- inspiré du cortex visuel
- noyau de convolution (somme pondérée d'un voisinage)
- utilisé partout sur l'image (partage des paramètres)

Animation (cours de Stanford, A. Karpathy)

- `http://cs231n.github.io/convolutional-networks/`

Premières applications (dès les années 90)

- reconnaissance de chiffres, lecture de chèques, reconnaissance de visages

Limites liées à l'époque

- peu de données d'apprentissage, machines lentes
- gain peu évident par rapport aux approches classiques

Réseaux convolutifs

Reconnaissance d'objets

- campagne ImageNet `http://www.image-net.org/` - 1000 catégories et 1,2M exemples
- supériorité écrasante des réseaux convolutifs en 2012
- réduction de 15% à 5% d'erreur entre 2012 et 2015

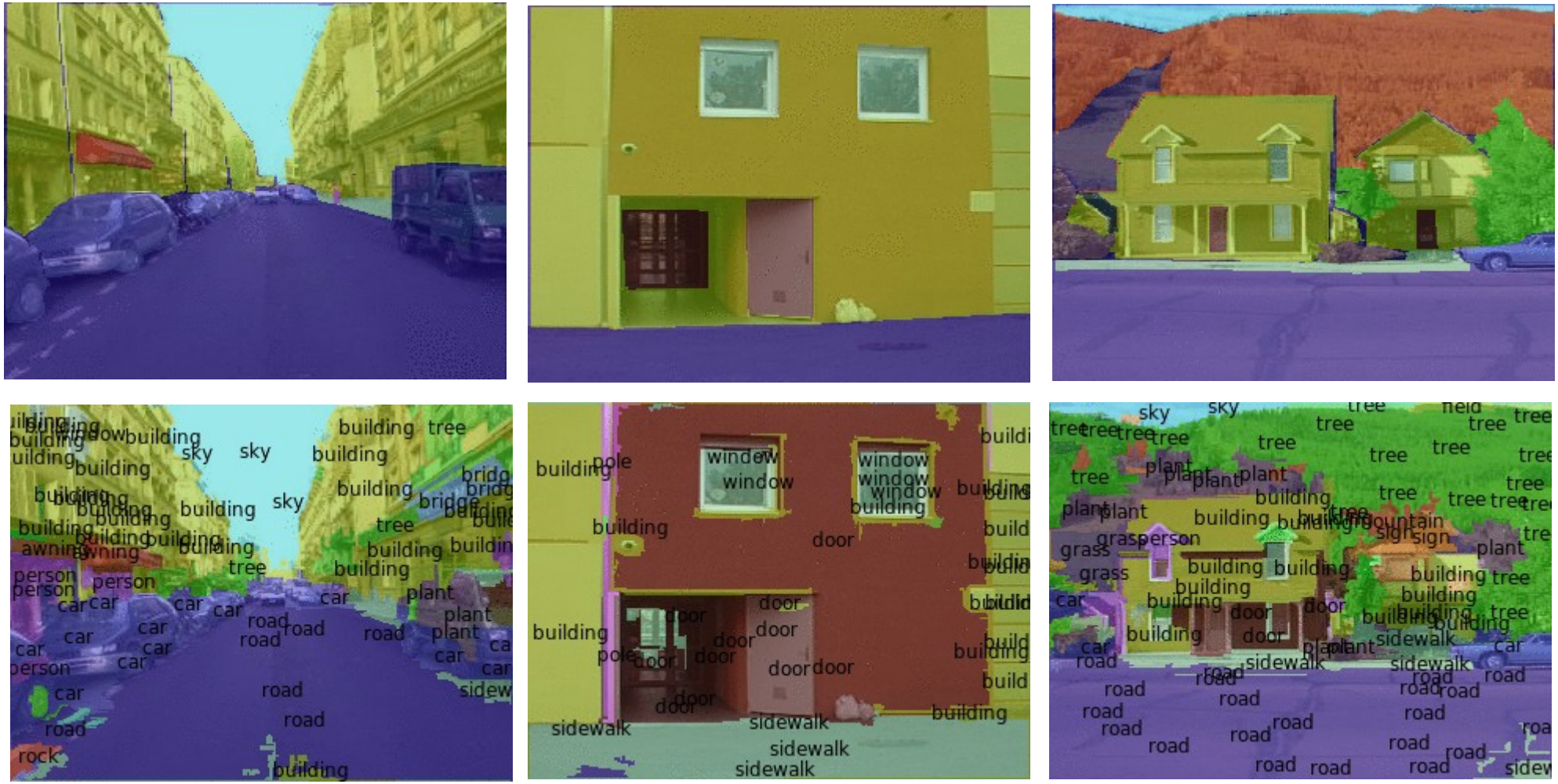
Nouvelles applications

- étiquetage de zones d'images `https://research.facebook.com/blog/learning-to-segment/`
- pilotage de voiture
- détection de piétons

Adapté aux signaux multi-dimensionnels

- corrélations locales fortes
- objets invariants aux translations ou distorsions
  - 1D - signal audio ou texte
  - 2D - images ou spectrogrammes
  - 3D - vidéos - identification d'actions (sport...)

Scene Parsing/Labeling



[Farabet et al. ICML 2012, PAMI 2013]

Autres architectures importantes

Réseaux récurrents

- prise en compte des sorties des échantillons précédents
- permet une mémorisation à plus ou moins long terme du passé
- nombreuses variantes (en particuliers les LSTM)
- l'exemple de WaveNet pour la synthèse vocale `https://deepmind.com/blog/wavenet-generative-model-raw-audio/`

Réseau auto-encodeur

- la sortie est identique à l'entrée
- peut permettre de **débruiter** un signal
- sert aussi à apprendre une représentation **compacte** des données (couche intermédiaire de taille inférieure aux données)

Perspectives

Regain massif d'intérêt pour les réseaux de neurones

- après "l'hiver de l'IA" des années 90
- victoire récente d'AlphaGo de DeepMind contre le champion du monde de Go
- domaine investi par des acteurs majeurs (Google, Facebook, Apple...)
- retombées importantes attendues dans le domaine de la santé

Questions théoriques encore ouvertes

- fondements mathématiques du deep learning
- intégration du raisonnement, attention, mémorisation et planification (memoire épisodique)
- unification de l'apprentissage supervisé, non-supervisé et par renforcement
- méthode efficace d'apprentissage non-supervisé

Sources

Sources et références

- Yann Le Cun (dir. Facebook AI Research)
  - `http://yann.lecun.com`
  - cours du Collège de France
- Cours de Stanford sur les réseaux convolutifs
  - `http://cs231n.github.io/`
  - `http://cs231n.github.io/convolutional-networks/`
- Andrew Ng - apprentissage automatique à Coursera
  - `https://www.coursera.org/course/ml`
- Thiago G. Martins - notes sur le cours précédent avec figures
  - `https://tgmstat.wordpress.com/`