Apprentissage profond

Introduction à l'apprentissage automatique – GIF-4101 / GIF-7005

Professeur : Christian Gagné

Semaine 8

UNIVERSITÉ LAVAL

8.1 Motivations de l'apprentissage profond

Historique des réseaux de neurones

- 1957 : proposition du perceptron par Rosenblatt
- 1967 : démonstration par Minsky que le perceptron est incapable de traiter des données non linéairement séparables, désintérêt pour les approches neuronales
- 1986 : Rumelhart, Hinton et Williams démontrent l'utilisation de la rétropropagation des gradients pour l'entraînement du perceptron multicouche
- 1995-2005 : développement des SVM, perte d'intérêt pour les réseaux de neurones
- 2006 : premières architectures profondes de réseaux de neurones
- 2012 : résultats en reconnaissance d'objets (Toronto, ImageNet) et de la parole (Microsoft) démontre le potentiel de technologie disruptive de l'apprentissage profond
- 2014 : explosion d'investissements privés en apprentissage automatique, en particulier en apprentissage profond
- 2018 : prix Turing de l'ACM (« Nobel » de l'informatique) à Bengio, Hinton et LeCun pour leurs travaux sur l'apprentissage profond
- 2020-2022 : modèles génératifs d'envergure pour le texte (ChatGPT) et l'image (DALL-E, Midjourney)

Émergence des réseaux profonds

- Conditions ayant permis l'émergence des réseaux profonds :
  1. Disponibilité de très grands jeux de données (*big data*)
  2. Disponibilité d'une capacité de calcul faramineuse (GPU)
  3. Nouveaux modèles d'apprentissage très flexibles, avec des *a priori* permettant de bien gérer la malédiction de la dimensionnalité

Apprentissage de représentations

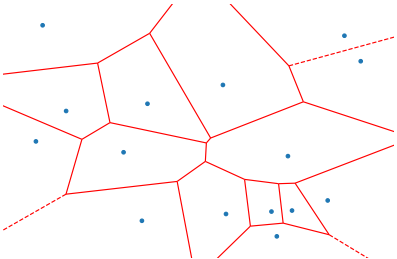
- Motivation des réseaux profonds : apprendre une représentation pour des données faiblement structurées
  - Données faiblement structurées : données dont l'information d'intérêt est présente dans du contenu brut, sans être clairement identifiée (exemple : image, texte, voix)
  - Par opposition à des données tabulaires, où chaque variable est bien identifiée et a souvent été choisie selon la tâche d'intérêt
- L'apprentissage profond permet d'extraire une représentation des données brutes adaptée à la tâche à effectuer
  - Évite de devoir faire l'ingénierie d'une représentation des données par des experts du domaine
  - Requiert cependant une grande quantité de données pour apprendre la représentation à partir de celles-ci

Composition de modèles

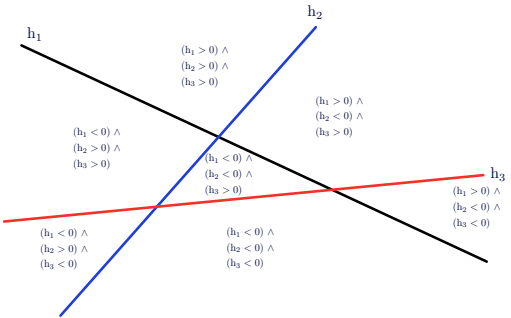
- « Compositionnalité » de modèle est nécessaire en apprentissage automatique
  - Comme en langue naturelle, on compose des éléments simples permettant d'exprimer des notions complexes
- Exploiter la compositionnalité permet un gain exponentiel en puissance de représentation
  - Représentations distribuées, apprentissage de caractéristiques
  - Architectures profondes : plusieurs niveaux d'apprentissage de représentations
- Composition de modèles est utile pour décrire notre monde efficacement

Représentation locale vs distribuée

- Ensemble de discriminants distribués (non mutuellement exclusifs) est exponentiellement plus efficace sur le plan statistique que des représentations locales (k-plus proches voisins, clustering)



Représentation locale



Représentation distribuée

Profondeur des réseaux

- Les réseaux profonds, lorsque bien entraînés, apprennent mieux que les réseaux obèses (*fat networks*)
  - Capacité des réseaux croît linéairement selon la largeur d'une couche, exponentiellement selon la profondeur du réseau

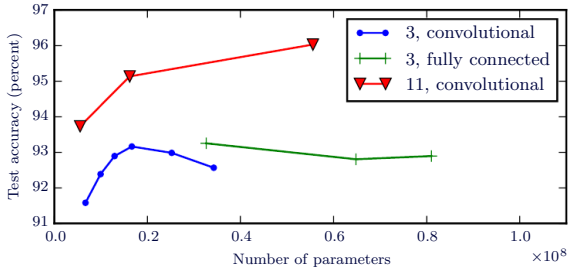


Figure 6.7 de *I. Goodfellow, Y. Bengio et A. Courville, Deep Learning, MIT Press, 2016.* Accédée en ligne le 19 octobre 2020 au `https://www.deeplearningbook.org/contents/mlp.html`.

- Réseau obèse fait du surapprentissage à 20M de poids, réseau profond fonctionne bien avec 60M de poids

8.2 Autoencodeurs

Pré-entraînement non supervisé

- Réseaux profonds avant 2011 : pré-entraînement non supervisé nécessaire
  - Initialisation aléatoire de réseaux profonds génère une grande variété de solutions sous-optimales (minima locaux)
  - Pré-entraînement non supervisé permet de démarrer la rétropropagation dans une « bonne configuration » (bassin d'attraction)

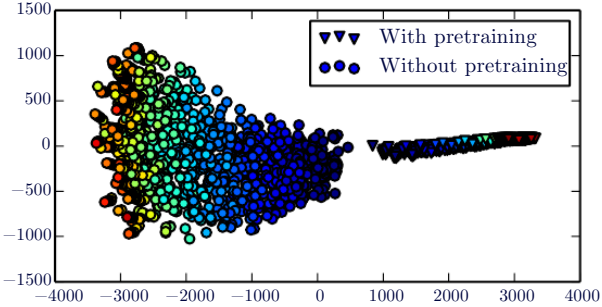
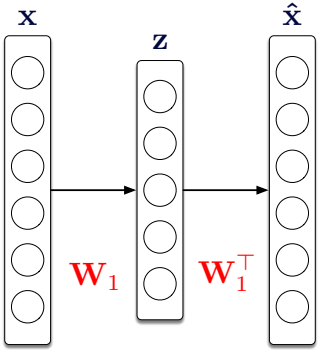


Figure 15.1 de *I. Goodfellow, Y. Bengio et A. Courville, Deep Learning, MIT Press, 2016.* Accédée en ligne le 19 octobre 2020 au `https://www.deeplearningbook.org/contents/representation.html`.

Autoencodeurs

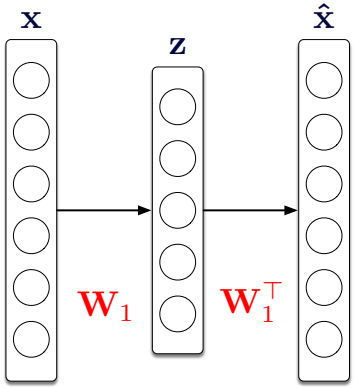
- Autoencodeur : modèle permettant de faire une compression de l'entrée (encodeur) et une décompression de celle-ci (décodeur)
  - Objectif : compresser tout en gardant l'erreur de reconstruction $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$ faible
  - Poids du décodeur liés aux poids de l'encodeur (habituellement, transposé)



Entraînement d'autoencodeurs

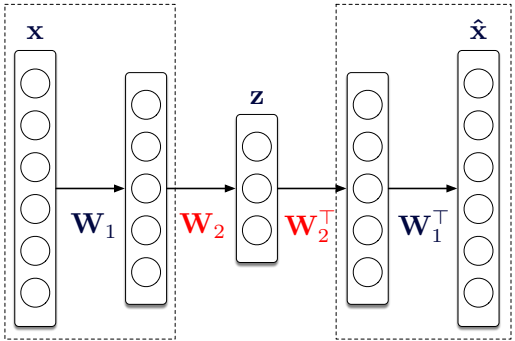
- Autoencodeur entraîné de façon non supervisée, pour apprendre représentation
  - Encodeur utilisé pour extraire une représentation compacte
- Entraînement vorace, une couche à la fois
  - Entraînement de la couche la plus externe
  - Ajout d'une nouvelle couche, qui est entraînée individuellement, couche externe étant fixée, et ainsi de suite
- Fonction de transfert non linéaire entre les couches
  - Nécessaire, sinon plusieurs couches linéaires pourraient se simplifier en une seule couche
  - Apprentissage des poids par descente du gradient
- Couche de sortie ajoutée à l'encodeur, avec entraînement supervisé
  - Entraînement complet de la couche de sortie par rétropropagation
  - Ajustement des poids de l'encodeur par rétropropagation (*fine-tuning*)

Exemple d'entraînement d'un autoencodeur



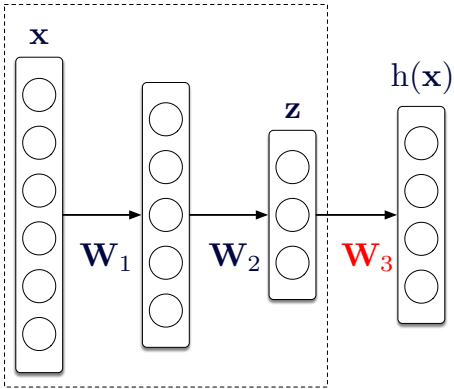
- Entraînement non supervisé du poids $\mathbf{W}_1$, poids $\mathbf{W}_1^\top$ lié
- Minimize erreur $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- Représentation intermédiaire dans valeurs centrales (vecteur latent $\mathbf{z}$)

Exemple d'entraînement d'un autoencodeur



- Ajout de deux nouvelles couches (une dans encodeur et une dans décodeur)
- Entraînement non supervisé du poids $\mathbf{W}_2$, poids $\mathbf{W}_1$ fixés
- Minimise toujours erreur $\|\mathbf{x} - \hat{\mathbf{x}}\|^2$
- Nouvelle représentation intermédiaire
- Peut être répété ainsi sur plusieurs couches

Exemple d'entraînement d'un autoencodeur



- Retrait de la partie décodeur du réseau
- Ajout d'une couche de sortie, avec autant de sorties que de classe
- Entraînement **supervisé** de $\mathbf{W}_3$ par rétropropagation
- Poids $\mathbf{W}_1$ et $\mathbf{W}_2$ souvent également ajustés finement par rétropropagation (*fine-tuning*)

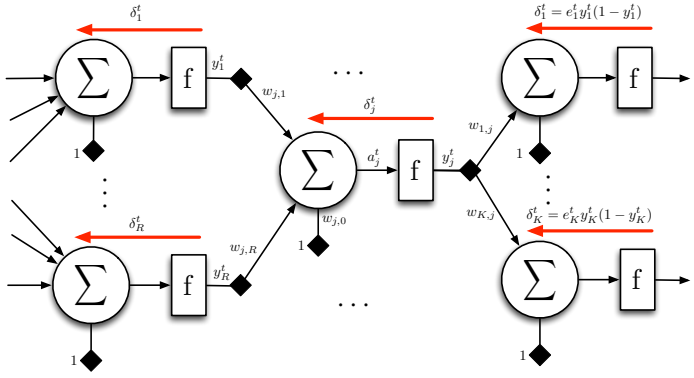
8.3 Éléments d'apprentissage profond

Apprentissage profond sans pré-entraînement non-supervisé

- Le pré-entraînement non-supervisé de réseaux profonds n'est plus requis de façon générale
- Diverses techniques permettent des entraînements directs de réseaux de grande profondeur
  - Nouvelles fonctions de transfert (ex. ReLU) qui atténuent le problème de dilution du gradient
  - Meilleures initialisations des poids du réseau (techniques de Xavier et He)
  - Désactivation aléatoire de poids avec dropout, permettant une meilleure distribution des traitements sur l'ensemble du réseau
  - Batch normalization, pour renormalisation des valeurs entre les couches permettant certaines invariances dans l'apprentissage
  - Liens résiduels, permettant de diffuser l'information de l'entrée plus directement vers des couches profondes

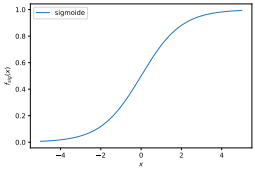
Problème de la dilution du gradient

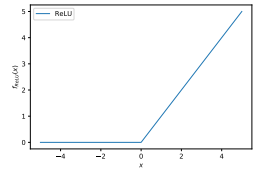
- L'entraînement d'un perceptron multicouche classique de plus de deux couches cachées avec rétropropagation ne fonctionne pas bien de façon générale
  - Neurones saturés, avec gradient très faible
  - Dilution du gradient (*vanishing gradient*) de couche en couche



Fonctions de transfert

- Fonction sigmoïde
  - Interprétation probabiliste
  - Approximation d'une fonction *step* (binaire)
  - Problème de saturation sur le gradient
- Fonctions de transfert doivent inclure des non-linéarités
- Fonction ReLU (*Rectified Linear Unit*), $f_{\text{ReLU}}(a) = \max(0, a)$
  - Modèle simple de fonction de transfert avec non-linéarité
  - Composition de ReLU permet de l'approximation linéaire par morceaux
  - Motivation biologique de réseaux profonds avec ReLU (*leaky integrate-and-fire model*)
  - Apprentissage de réseaux profonds avec ReLU possible sans pré-entraînement non supervisé





Initialisation des réseaux profonds

- Valeurs initiales des poids à un important sur les valeurs de gradients utilisés pour l'apprentissage
  - Poids initiaux trop faibles $\Rightarrow$ implosion du gradient, stagnation de l'apprentissage
  - Poids initiaux trop élevés $\Rightarrow$ explosion du gradient, instabilité de l'apprentissage
  - Faire le bon compromis pour initialiser les poids, en tenant compte des fonctions de transfert utilisées
- Justification mathématique : exemple d'un réseau profond de L couches avec fonction de transfert linéaire

$$y = \mathbf{W}_L \, \mathbf{W}_{L-1} \, \ldots \, \mathbf{W}_2 \, \mathbf{W}_1 \, \mathbf{x}$$

  - Supposons que $L-1$ premières couches identiques et égales à $\mathbf{W}$, $y = \mathbf{W}_L \, (\mathbf{W})^{L-1} \, \mathbf{x}$
  - Avec $\mathbf{W} = c \, \mathbf{I}$ et $c > 1$, explosion de la valeur de sortie, $\lim\limits_{L \to \infty} y = \infty$
  - Avec $c < 1$, implosion de la valeur de sortie, $\lim\limits_{L \to \infty} y = 0$

Méthodes d'initialisation

- Méthode de Xavier
  - Adaptée avec fonction de transfert sigmoïde et tanh
  - Consiste en des valeurs aléatoires uniformes dans $\left[-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}\right]$, où n_{in} est le nombre d'entrées et n_{out} est le nombre de sorties du neurone associé au poids généré

$$w_{j,i} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}, \sqrt{\frac{6}{n_{\text{in}}+n_{\text{out}}}}\right)$$

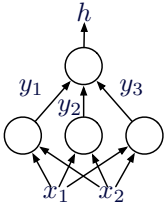
- Méthode de He
  - Pour fonction de transfert asymétrique comme ReLU, la méthode de He est préférable
  - Initialise selon une loi normale qui dépend du nombre d'entrées du neurone

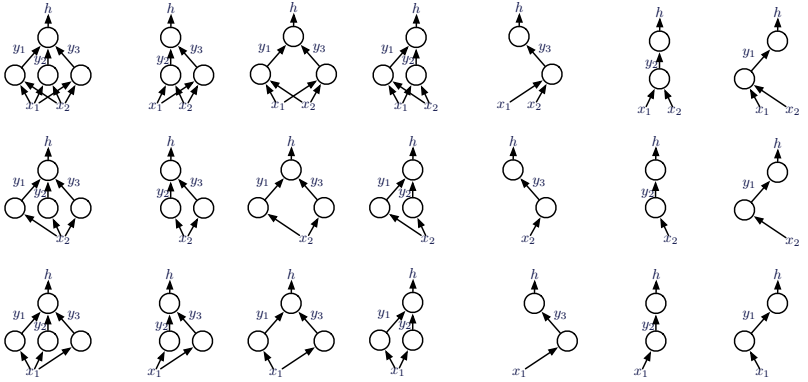
$$w_{j,i} \sim \mathcal{N}\left(0, \frac{2}{n_{\text{in}}}\right)$$

Dropout

- Dropout : méthode d'entraînement par désactivation aléatoire des neurones
  - Typiquement, moitié des neurones des couches cachées (80 % des entrées) sont activés à la présentation de chaque donnée durant l'entraînement
  - Masques aléatoires pour sélectionner neurones actifs, un différent à chaque présentation
- Effectue une régularisation du réseau
  - Force l'apprentissage d'une représentation distribuée dans l'ensemble du réseau
  - Rend difficile l'émergence de « neurones grand-mère »
  - S'est avéré très efficace pour améliorer les performances des réseaux profonds
- Évaluation de nouvelles données en test par moyennage sur plusieurs masques de sélection
  - Analogie avec méthodes par ensemble (vu plus tard dans la session), en particulier bagging

Dropout





Batch normalization

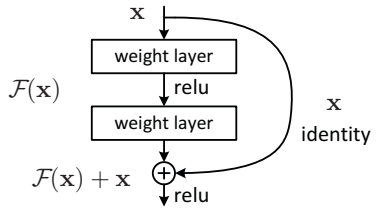
- Modification d'un poids par rétropropagation basée sur gradient local
  - Poids des couches précédentes et suivantes eux aussi modifiés !
- *Batch normalization* : normaliser activation des neurones entre toutes les données d'un mini-lot (*mini-batch*)
  - Mini-lot : petit sous-ensemble d'instances de données de l'ensemble d'entraînement (typiquement quelques centaines)
- Activation des neurones $\mathbf{H}$ normalisées selon

$$\mathbf{H}' = \frac{\mathbf{H} - \boldsymbol{\mu}}{\boldsymbol{\sigma}}, \quad \boldsymbol{\mu} = \frac{1}{m}\sum_i \mathbf{H}_{i,:}, \quad \boldsymbol{\sigma} = \sqrt{\epsilon + \sum_i (\mathbf{H} - \boldsymbol{\mu})_i^2}$$

  - $\mathbf{H}$: activation des neurones (ligne) d'une couche pour les données du mini-lot (colonne)
  - ϵ : petite valeur (typiquement 10^{-8}) pour éviter division par zéro lorsque variance nulle

Liens résiduels

- Liens résiduels : permettre des connexions directes entre couches non adjacentes (*skip links*)



Tiré de *K. He, X. Zhang, S. Ren, et J. Sun, Deep residual learning for image recognition. CVPR, 2016.* Accédé en ligne le 6 novembre 2020 au `https://arxiv.org/abs/1512.03385`.

- Permet des réseaux beaucoup plus profonds et performants (ResNets)
  - ResNets : gagnants de compétition ImageNet 2015 (3,57 % d'erreur top 5)
  - Facilite l'optimisation et la propagation du signal à travers le réseau
  - Bloc résiduel doit faire un traitement produisant une amélioration de la sortie du bloc précédent, sinon on pourrait ignore le bloc courant

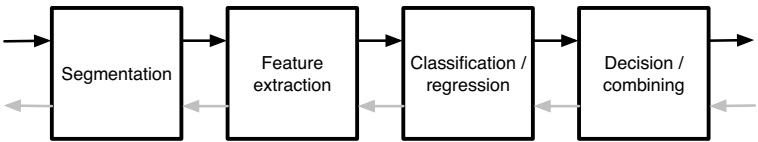
8.4 Apprentissage de données non structurées

Apprendre à partir de données non structurées

- Modèles classiques de réseaux de neurones traitent des vecteurs de taille fixe
  - Suppose que les données sont représentées sur un nombre préétablis de variables
- Données tabulaires
  - Données structurées, avec variables connues et en nombre raisonnable
  - Directement utilisables par réseaux de neurones (ex. PMC) et autres modèles
- Images
  - Matrice de nombres, avec chaque pixel représentés par trois valeurs réelles (RGB)
  - Données peu structurées, grand nombre de variables (millions de pixels par image) et peu de signification des pixels individuels
  - Forte localité des pixels dans les images, opérateurs adaptés (ex. convolution) peuvent en tirer avantage
- Texte
  - Assemblage de mots formant des phrases, séquences de longueur variable
  - Vocabulaire vaste ($\sim 100\,000$ mots en français), avec synonymes, homonymes, mots apparentés

Apprentissage de représentations

- Pipeline classique de la reconnaissance des formes

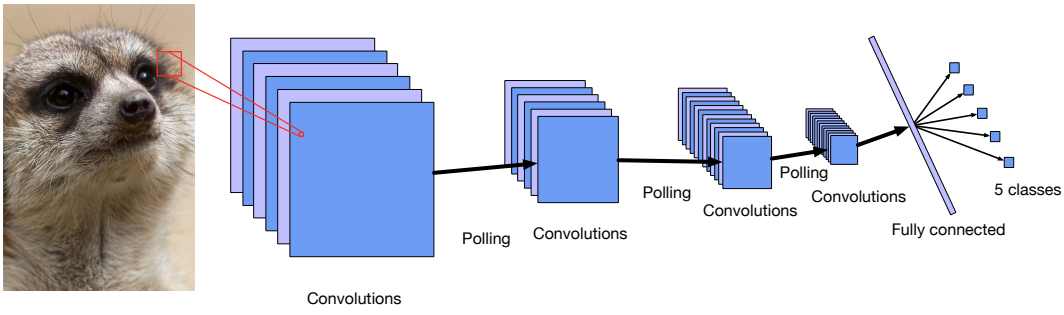


- Dans le passé, chaque module conçu indépendamment
- Apprentissage profond permet l'apprentissage de représentations
  - Apprentissage de tous les modules simultanément
  - Possibilité de récupérer les représentations (segmentation, extraction de caractéristiques) et les utiliser avec d'autres modules de classement et de prise de décision

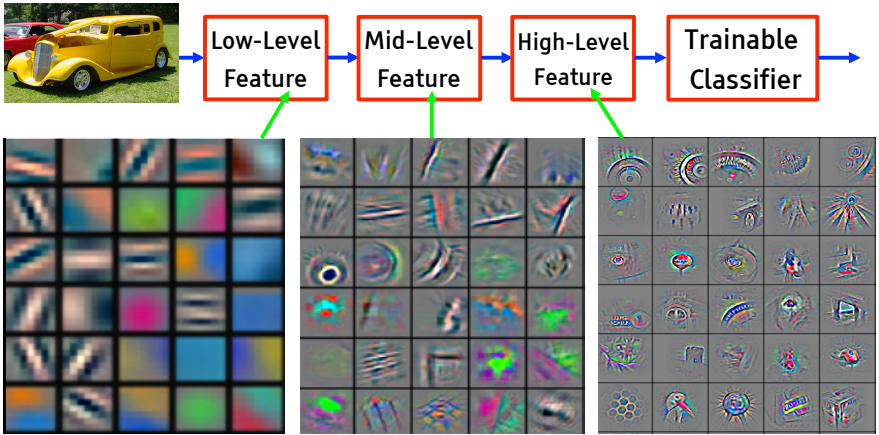
Réseau à convolution

- Réseau à convolution : traiter des signaux temporels ou spatiaux
  - Signal temporel : son et parole
  - Signal spatial : image
- Couche de convolution : filtres convolués sur données temporelles/spatiales
  - Données peuvent être valeurs d'entrée du réseau ou sorties de couches précédentes
  - Convolution sur chaque canal (plusieurs canaux possibles)
  - Apprentissage des filtres par rétropropagation
- Couche de *pooling* : sélection de valeurs (maximum d'une fenêtre)
  - Permet de réduire taille des valeurs, sinon explosion de la taille du modèle en vue !
- Neurones pleinement connectés en sortie pour prise de décision
- Présenté plus en détail dans le prochain module

Réseau à convolution



Composition de filtres



Tiré de *G. Hinton, Y. Bengio et Y. LeCun, Deep Learning NIPS'15 Tutorial, 2015.* Accédé en ligne le 19 octobre 2020 au `https://media.nips.cc/Conferences/2015/tutorialslides/DL-Tutorial-NIPS2015.pdf`.

Reconnaissance d'objets

- *ImageNet Large Scale Visual Recognition Challenge* : reconnaître les objets d'images (1000 classes), en donnant la bonne classe dans un top 5

ILSVRC 2012	
Équipe	% erreur
SuperVision (Toronto)	15,3
ISI (Tokyo)	26,1
VGG (Oxford)	26,9
XRCE / INRIA	27,0
UvA (Amsterdam)	29,6
INRIA / LEAR	33,4

ILSVRC 2013	
Équipe	% erreur
Clarifai	11,7
NUS	12,9
Zeiler-Fergus (NYU)	13,5
A. Howard	13,5
OverFeat (NYU)	14,1
UvA (Amsterdam)	14,2
Adobe	15,2
VGG (Oxford)	15,2
VGG (Oxford)	23,0

ILSVRC 2014	
Équipe	% erreur
GoogLeNet	6,6
VGG (Oxford)	7,3
MSRA	8,0
A. Howard	8,1
DeeperVision	9,5
NUS-BST	9,7
TTIC-ECP	10,2
XYZ	11,2
UvA	12,1

Traitement de texte

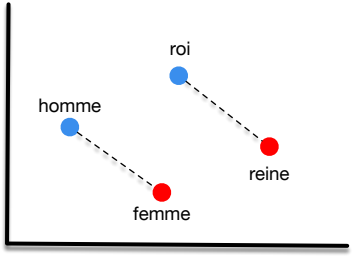
- Comment traiter des documents (séquence de chaînes de caractères) avec un réseau de neurones qui reçoit des vecteurs de réels de taille fixe en entrée ?
- Modèle *Bag-of-Words* (BoW)
  - Identifier dictionnaire de mots les plus fréquents / intéressants
  - Calculer la fréquence de chaque mot pour chaque document traité (vecteur d'entiers de taille fixe)

    $$\mathbf{x}^t = [x_1^t, x_2^t, \ldots, x_v^t]^\top$$

    où x_i^t est le nombre d'occurrences (valeur entière) du $i-$ème mot (selon le dictionnaire) dans le document
- Ne tient pas compte de l'ordre
  - Modèles avec N-gram : mesure fréquence de groupes de mots adjacents
  - Modèle des skip-gram : mots connexes peuvent ne pas être adjacents

Plongement lexical

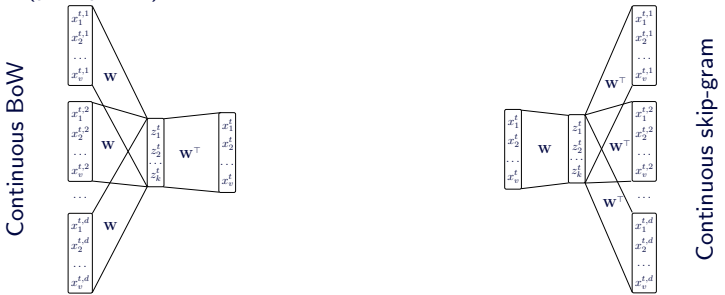
- Plongement lexical (*word embedding* en anglais) : projection des mots dans un espace vectoriel capturant des relations sémantiques
  - Des mots proches dans l'espace vectoriel ont un sens similaire ou apparenté
  - Postulat qu'opérations algébriques dans l'espace respectent une logique sémantique



- Construction de plongements lexicaux généralement faite par des approches d'apprentissage non supervisé ou auto supervisé
- Espace induit est intéressant pour faire des traitements
  - Par exemple, entrée d'un réseau de neurones pour le classement de documents

Construction de plongements lexicaux

- Idée : prédire mots d'une séquence pour encoder le texte
  - *Continuous BoW* : prédire le mot selon ceux qui précèdent et suivent (plus rapide)
  - *Continuous skip-gram* : prédire les mots qui précèdent et suivent selon le mot d'intérêt (plus précis)

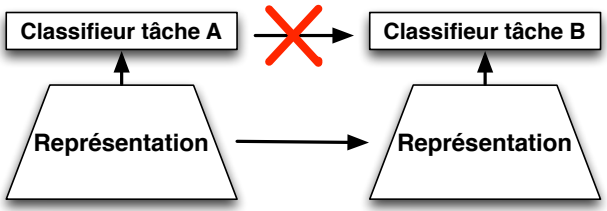


- word2vec : utiliser *Continuous BoW* ou *Continuous skip-gram* pour construire des plongements
  - Réseau PMC, deux couches cachées, plongement de quelques centaines de dimensions

8.5 Transfert et adaptation de représentations

Transfert de représentations

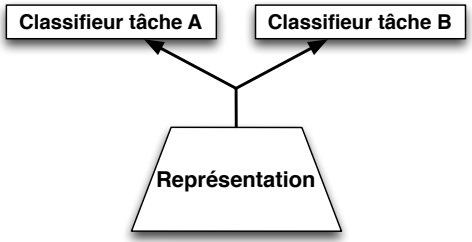
- Apprentissage d'un réseau profond sur tâche A
- Nouvelle tâche B, basée sur données similaires à tâche A
  - Récupérer représentation de tâche A
  - Entraîner nouveau classifieur pour tâche B



  - Permet un transfert de représentation (*transfer learning*)
  - Ajustement fin possible de la représentation sur la nouvelle tâche (*fine-tuning*)
- Approche standard pour apprendre modèle de reconnaissance d'objets, utilisant représentation produite avec ImageNet

Apprentissage multitâche

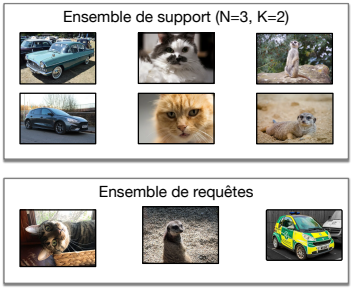
- Apprentissage multitâche : apprendre *simultanément* une représentation pour des opérations distinctes
  - Réseau à deux têtes, une pour chaque tâche



  - Rétropropagation provient d'une tête à la fois
  - Mélange des données et des tâches durant l'apprentissage
  - Performe bien à produire des représentations capturant des concepts généraux

Apprentissage few-shot

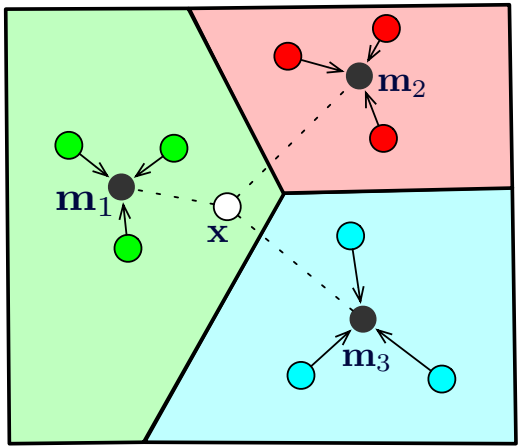
- Comment apprendre avec peu de données de chaque classe ?
  - Problème général : apprendre à apprendre (méta-apprentissage)
- Modèle avec N classes de K instances chacun (N-*way*-K-*shot*)



  - Ensemble de support : K instances pour chacune des N classes à traiter
  - Ensemble de requêtes : nouvelles instances à traiter
- Classes des ensembles de support et de requêtes varient à chaque essai
  - Apprentissage de modèles conçus pour fonctionner avec des classes a priori inconnues

Apprentissage few-shot : réseau prototypique

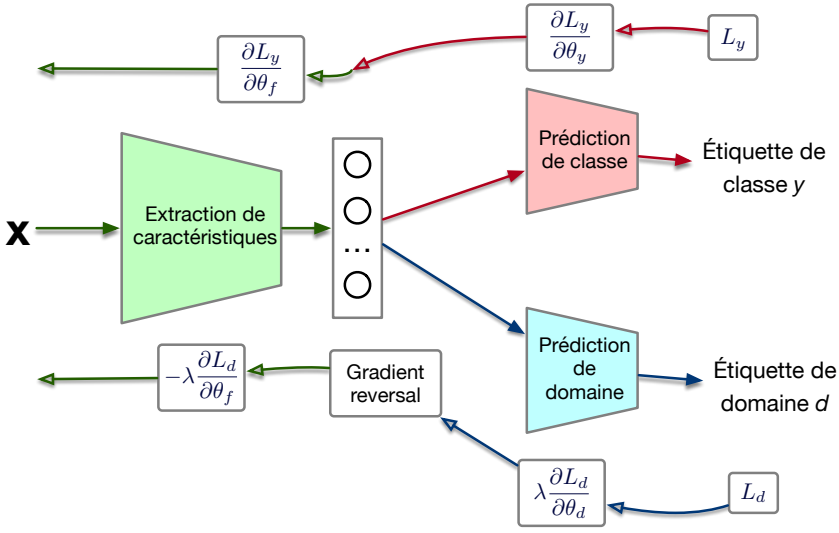
- Réseau prototypique (*prototypical network*)
  - Synthétiser le support d'une classe par une valeur moyenne (prototype)
  - Classer les requêtes selon le plus proche prototype



Adaptation de domaine

- Adaptation de domaine : utiliser un ou plusieurs domaines sources pour mieux traiter domaine cible
  - Les entrées et sorties du modèle sont les mêmes pour tous les domaines
  - Cas particulier de l'apprentissage par transfert, où les entrée/sorties peuvent changer dans le cas général
- Réseau de neurones adversariel au domaine (DANN : *Domain Adversarial Neural Network*)
  - Apprendre pour plusieurs domaines source, selon un modèle d'apprentissage multitâche
  - Une tête associée à la tâche de classement
  - Un deuxième tête vise à discriminer entre les domaines source dans la sortie de la partie partagée
  - Apprentissage de la tête supplémentaire comporte un renversement du gradient (*gradient reversal*) pour forcer une représentation intermédiaire commune et générale

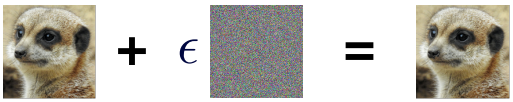
Domain adversarial neural network (DANN)



8.6 Approches adversariales

Données adversariales

- Utiliser génération de données pour déterminer plus petite variation permettant de faire une erreur de classement



- Causé par l'utilisation de représentation distribuée dans un espace à très haute dimensionnalité
- Illustre une difficulté actuelle avec réseaux profonds, robustesse aux données adversariales doit être améliorée

Attaques et défenses

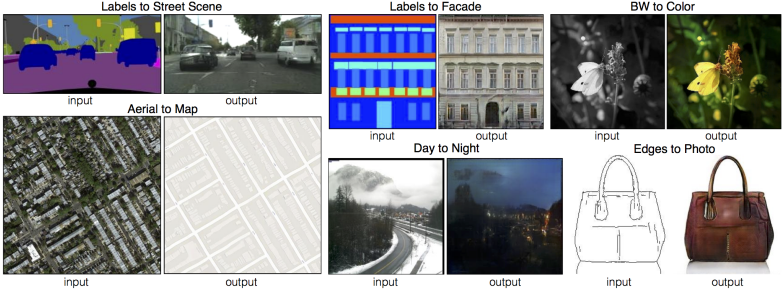
- Attaques typiques : descente du gradient sur la donnée pour tromper le réseau
  - *Fast gradient sign method* (FGSM) :

$$\mathbf{x} = \mathbf{x} + \epsilon \operatorname{sign} \frac{\partial L(\mathbf{x}, y|\theta)}{\partial \mathbf{x}}$$

  - Plusieurs autres variantes proposées pour produire données adversariales
- Mécanisme de défense : entraînement adversariel
  - Augmenter le jeu d'entraînement avec des données adversarielles pour rendre le réseau robuste

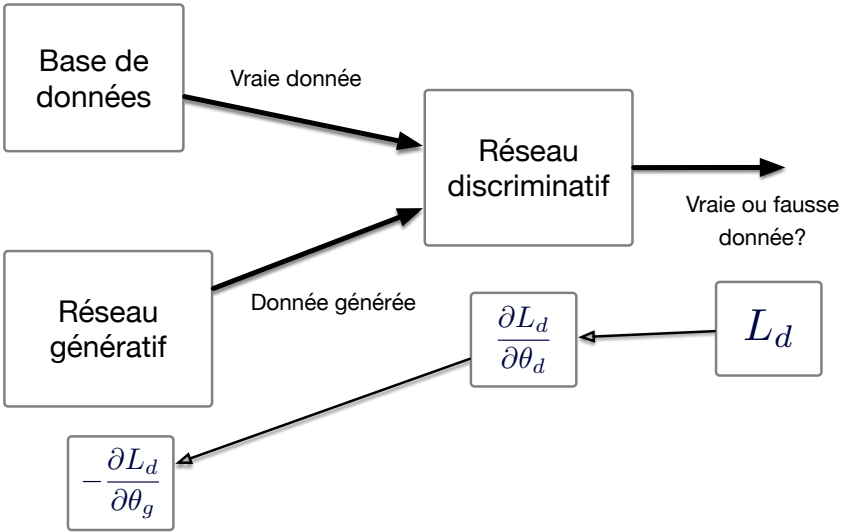
Generative Adversarial Networks (GAN)

- Modèle GAN : mettre en compétition deux réseaux de neurones
  - Réseau discriminatif : distinguer données véritables du problème des données générées
  - Réseau génératif : produire des données ayant l'air authentiques
  - Permet des traitements variés basés sur un apprentissage non supervisé
- Exemple : traduction image à image avec GANs conditionnels



Tiré de *Isola, Zhu, Zhou et Efros, Image-to-Image Translation with Conditional Adversarial Networks, CVPR, 2017.* Accédé en ligne le 19 octobre 2020 au `https://arxiv.org/pdf/1611.07004v3.pdf`.

Generative Adversarial Networks (GAN)



8.7 Implémentations logicielles

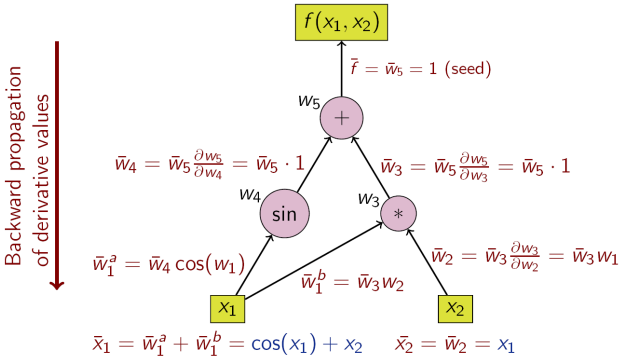
Gradient automatique

- Graphe computationnel : représenter les opérations mathématiques d'un réseau dans graphe
  - Capture l'ordre et la nature des opérations
- Gradient automatique : calculer les gradients **analytiques** sur l'ensemble du réseau automatiquement, via les graphes computationnels
- Permet de définir des topologies complexes et hétérogènes de réseau sans devoir faire les dérivées analytiques manuellement !
- Permet également d'optimiser les traitements sur l'architecture visée (ex. GPU)

Gradient automatique

$$\frac{\partial f(x_1,x_2)}{\partial x_1} = \frac{\partial}{\partial x_1}\left(\sin(x_1) + x_1\, x_2\right) = \cos(x_1) + x_2$$

$$\frac{\partial f(x_1,x_2)}{\partial x_2} = \frac{\partial}{\partial x_2}\left(\sin(x_1) + x_1\, x_2\right) = x_1$$



Par Berland, domaine public, https://en.wikipedia.org/wiki/File:ReverseaccumulationAD.png

Outils pour l'apprentissage profond (open source)

- TensorFlow : `https://www.tensorflow.org/`
  - Code en C++, avec interface d'utilisation en Python
  - Entièrement organisé autour de graphes computationnels
- PyTorch : `https://pytorch.org/`
  - Offre interface de programmation convivial en Python, approche *programmatique*
  - Différentiation automatique faite dynamiquement, plus versatile sur certains aspects que TensorFlow
- Google JAX : `https://jax.readthedocs.io/`
  - Combine gradient automatique et calcul numérique haute performance dans une interface standard (à la NumPy)
  - Pas spécifiquement fait pour apprentissage profond, mais offre une grande flexibilité
- Keras : `https://keras.io/`
  - Interface convivial pour apprentissage profond, au coût d'une flexibilité réduite
  - Surcouche utilisant TensorFlow, PyTorch ou JAX comme environnement sous-jacent d'apprentissage profond

Références

- Yann LeCun, Yoshua Bengio et Geoffrey Hinton. *Deep learning*. Nature, vol. 521, pages 436–444, 2015. https://doi.org/10.1038/nature14539
- Ian Goodfellow, Yoshua Bengio et Aaron Courville. “Deep Learning”, MIT Press, 2016. http://www.deeplearningbook.org/
- Geoffrey Hinton, Yoshua Bengio et Yann LeCun, *Deep Learning NIPS'15 Tutorial*, 2015. https://nips.cc/Conferences/2015/Schedule?showEvent=4891