Introduction au Deep Learning

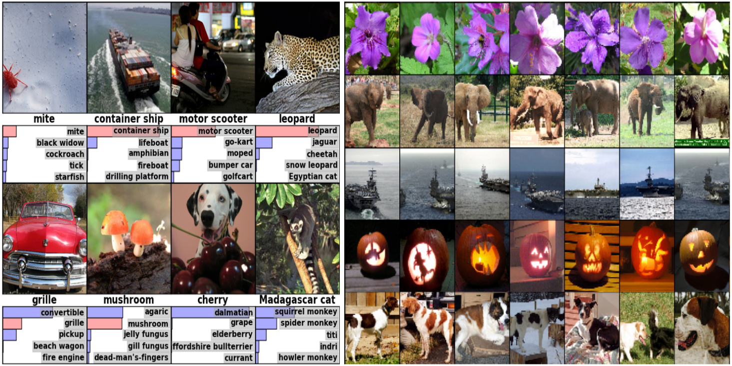
Jérémie Bigot

UFMI, Institut de Mathématiques de Bordeaux - Université de Bordeaux

Master MAS-MSS & CMI ISI
Université de Bordeaux

Classification d'images par Deep Learning ILSVRC Challenge (2010) [1]

- apprentissage : 1.2 million d'images labellisées (1000 classes)
- test : 150 000 images



1. Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012)

Ce cours : un bref aperçu du Deep Learning

- Introduction aux principes des réseaux de neurones (tels que formulés dès les années 50)
- Réseaux de neurones profonds (après 2012) : définition et méthodes d'inférence
- Mise en oeuvre pratique sur des **exemples très simples** : facilité d'utilisation et résultats spectaculaires comme raisons du succès ?

Apprentissage statistique

Problème générique : étant donné une **base d'apprentissage** $(X_i, Y_i)_{1 \leq i \leq n}$ où

- $X_i \in \mathbb{R}^d$
- $Y_i \in \mathbb{R}$ (**régression**) ou $Y_i \in \{1; 2; \ldots; K\}$ (**classification**)

On souhaite :

- déterminer un modèle qui permet de lier l'entrée X_i à la sortie Y_i pour tout $1 \leq i \leq n$
- pour $i_0 \notin \{1, \ldots, n\}$ on veut déterminer $\hat{Y}_{i_0}$ prédiction de Y_{i_0} (**non-observé**) au vu de l'observation de X_{i_0}

Le couple (X_{i_0}, Y_{i_0}) est un élément de l'**ensemble test**.

Remarque : en classification on peut aussi considérer que

$$Y_i \in \Sigma_K = \left\{ (p_1, \ldots, p_K) : p_k \geq 0 \text{ et } \sum_{k=1}^{K} p_k = 1 \right\}$$

Choix d'une classe de modèles

Definition : une classe de modèles est un ensemble de fonctions $f_\theta : \mathbb{R}^d \to \mathbb{R}$ (**régression**) ou $f_\theta : \mathbb{R}^d \to \Sigma_K$ (**classification**) indexées par un paramètre

$$\theta \in \Theta \subset \mathbb{R}^p$$

Apprentissage d'un modèle : minimisation du risque empirique

$$\hat{\theta} \in \underset{\theta \in \Theta}{\arg\min}\, M_n(\theta) \quad \text{avec} \quad M_n(\theta) := \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f_\theta(X_i))$$

où L est une fonction de perte e.g.

$$L(y, z) = \|y - z\|^2$$

ou bien cross-entropy en classification i.e.

$$L(y, z) = -\sum_{k=1}^{K} y_k \log(z_k)$$

Prédiction : $\hat{Y}_{i_0} = f_{\hat{\theta}}(X_{i_0})$

Choix d'une classe de modèles + pénalisation

Definition : une classe de modèles est un ensemble de fonctions $f_\theta : \mathbb{R}^d \to \mathbb{R}$ (**régression**) ou $f_\theta : \mathbb{R}^d \to \Sigma_K$ (**classification**) indexées par un paramètre

$$\theta \in \mathbb{R}^p$$

Apprentissage d'un modèle : minimisation du risque empirique pénalisé

$$\hat{\theta} \in \arg\min_{\theta \in \mathbb{R}^p} M_n(\theta) \quad \text{avec} \quad M_n(\theta) := \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\theta(X_i)) + \lambda \, \mathrm{pen}(\theta)$$

où $\mathrm{pen}(\theta)$ est une fonction de pénalisation qui augmente avec la complexité du modèle paramètré par θ, et $\lambda \geq 0$ est un paramètre de régularisation (**Exemple :** $\mathrm{pen}(\theta) = \|\theta\|_2^2$ ou $\mathrm{pen}(\theta) = \|\theta\|_1$)

Prédiction : $\hat{Y}_{i_0} = f_{\hat{\theta}}(X_{i_0})$

Choix d'une méthode d'optimisation

Remarque : pour les réseaux de neurones $\theta \mapsto M_n(\theta)$ est non-convexe et la dimension de θ est très grande !

Calcul de $\hat{\theta}$ par **descente de gradient** : (e.g. package `nnet` de R basée sur la méthode BFGS)

$$\hat{\theta}_{j+1} = \hat{\theta}_j - \gamma_j \nabla M_n(\hat{\theta}_j) \quad \text{et} \quad \hat{\theta} = \hat{\theta}_J,$$

pour J assez grand.

Choix d'une méthode d'optimisation

Deep learning : le nombre n d'exemples est très grand, coût élevé de l'évaluation

$$M_n(\theta) := \frac{1}{n}\sum_{i=1}^{n} L(Y_i, f_\theta(X_i))$$

Calcul de $\hat{\theta}$ par **descente de gradient stochastique** : (e.g. librairie `Tensorflow` de Python)

A chaque itération j, choix aléatoire d'un sous-ensemble de données $X_{i_1}, \dots X_{i_q}$ (**batch**) de taille $q \ll n$

$$\hat{\theta}_{j+1} = \hat{\theta}_j - \gamma_j \nabla m_q(\hat{\theta}_j) \quad \text{où} \quad m_q(\theta) = \frac{1}{q}\sum_{\ell=1}^{q} L(Y_{i_\ell}, f_\theta(X_{i_\ell}))$$

En pratique : partition des données en un ensemble de **batch (disjoints)** de taille m

Epoch : nombre de passage sur l'ensemble des données

Eléments bibliographiques

Exposé basé sur le livre en ligne de Michael Nielsen

`http://neuralnetworksanddeeplearning.com/index.html`

avec codes en Python :

`https://github.com/mnielsen/neural-networks-and-deep-learning.git`

et exemples d'utilisation de la librairie `Theano` de Python

Eléments bibliographiques

Exemple illustratif : base de données MNIST [1]

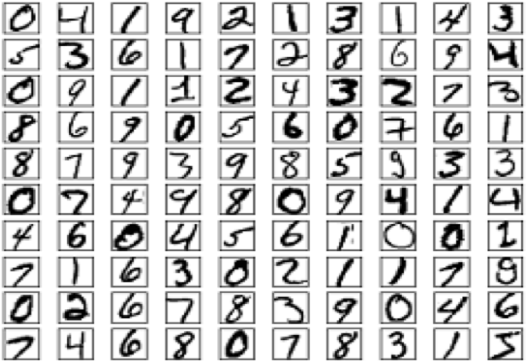
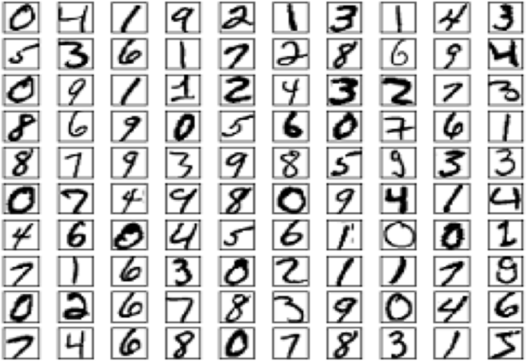


Image de taille $d = 28 \times 28 = 784$ pixels, $K = 10$ classes

1. http://neuralnetworksanddeeplearning.com/index.html

Eléments bibliographiques

Classification par réseaux de neurones convolutionnels

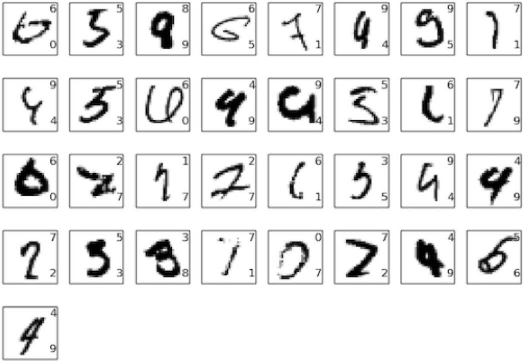


Ensemble d'apprentissage de 50000 images
Taux de classification $> 99\%$

Ensemble test de 10000 images : taux de classification $> 99\%$

Eléments bibliographiques

33 images mal-classées sur l'ensemble test [1]



Vraie classe : coin supérieur droit

Classe prédite : coin inférier droit

1. http://neuralnetworksanddeeplearning.com/index.html

Eléments bibliographiques

Pour ceux qui n'aiment pas la lecture...

Vidéos en ligne sur le site du Collège de France :

- Cours de Yann LeCun, "L'apprentissage profond : théorie et pratique", Collège de France (2015-2016), Informatique et Sciences du Numérique.

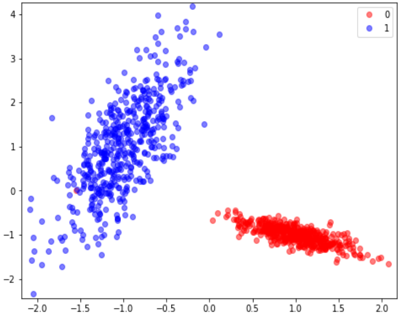
  www.college-de-france.fr/site/yann-lecun/course-2015-2016.htm

- Cours de Stéphane Mallat, "Mystères mathématiques des réseaux de neurones convolutionnels", Collège de France (2015-2016), Informatique et Sciences du Numérique.

  www.college-de-france.fr/site/yann-lecun/seminar-2016-02-19-15h30.

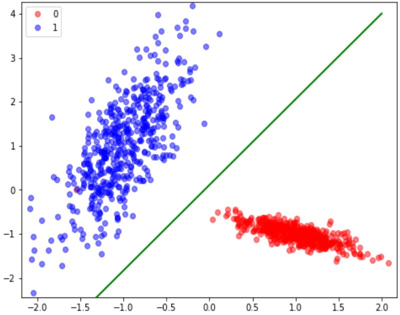
Méthode d'apprentissage linéaire

Choix d'une méthode d'apprentissage - séparation de classes par un hyperplan



Méthode d'apprentissage linéaire

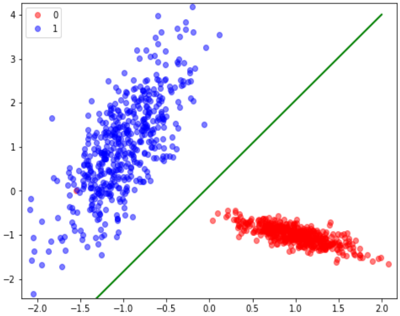
Choix d'une méthode d'apprentissage - séparation de classes par un hyperplan



Equation d'un hyperplan $\{x \in \mathbb{R}^d \ : \ \langle x, w \rangle + b = 0\}$ où $\theta = (w, b) \in \mathbb{R}^d \times \mathbb{R}$ sont les paramètres de l'hyperplan (ici $d = 2$)

Méthode d'apprentissage linéaire

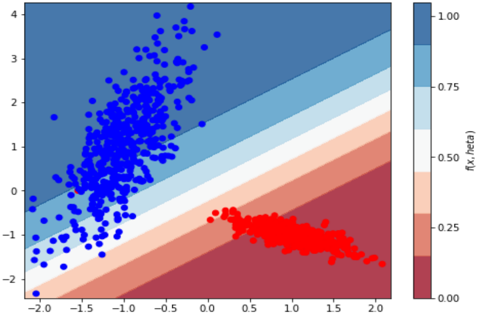
Choix d'une méthode d'apprentissage - séparation de classes par un hyperplan



- Points au-dessus de l'hyperplan $\{x \in \mathbb{R}^d \; : \; \langle x, w \rangle + b > 0\}$
- Points au-dessous de l'hyperplan $\{x \in \mathbb{R}^d \; : \; \langle x, w \rangle + b < 0\}$

Méthode d’apprentissage linéaire

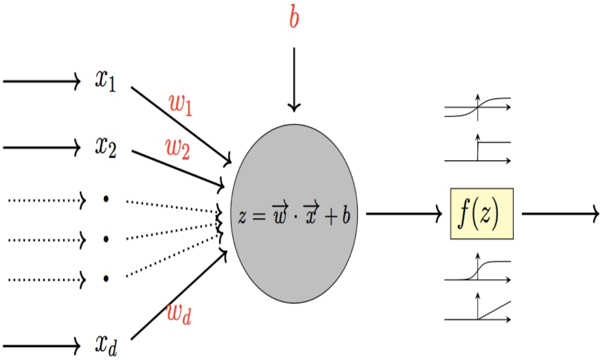
Choix d’une méthode d’apprentissage - séparation de classes par un hyperplan



Règle de classification $f(x, \theta) = \sigma\left(\langle x, w \rangle + b\right)$ avec $\sigma(z) = \mathbb{1}_{\mathbb{R}+}(z)$ ou $\sigma(z) = \frac{1}{1+\exp(-z)}$ avec $\theta = (w, b) \in \mathbb{R}^d \times \mathbb{R}$

Construction d'un réseau de neurones

Neurone de base : modèle du **Perceptron** (Rosenblatt, 1957)



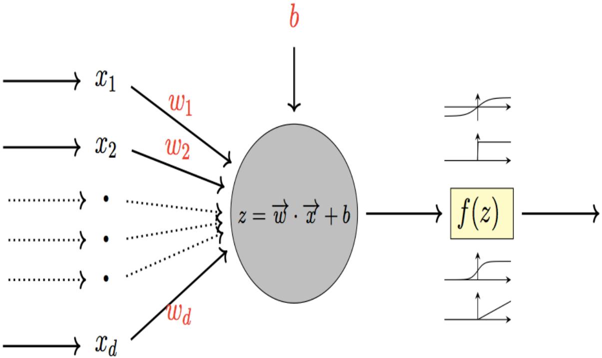
Source : `https://stats385.github.io/`

Combinaison linéaire de $x \in \mathbb{R}^d$ avec les poids $\omega_1, \ldots, \omega_d$ et un biais b

Fonction non-linéaire d'activation $f(z) = \sigma(z) = \mathbb{1}_{\{z \geq 0\}}$

Construction d'un réseau de neurones

Neurone de base : modèle du **Perceptron** (Rosenblatt, 1957)



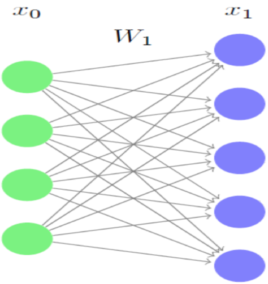
Source : `https://stats385.github.io/`

Combinaison linéaire de $x \in \mathbb{R}^d$ avec les poids $\omega_1, \ldots, \omega_d$ et un biais b

Autre choix $\sigma(z) = \frac{1}{1+\exp(-z)}$ (sigmoïde) ou $\sigma(z) = \max(0, z)$ (ReLU)

Construction d'un réseau de neurones

Single layer Perceptron



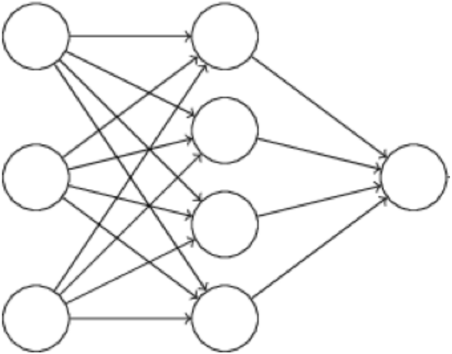
Source : `https://stats385.github.io/`

Ecriture condensée : $f_\theta(\mathbf{x}_0) = \sigma_1\left(W_1\mathbf{x}_0 + b_1\right)$, où

- $\sigma_1 : \mathbb{R}^{d_1} \to \mathbb{R}^{d_1}$ est une **fonction non-linéaire entrée par entrée**
- $W_1 \in \mathbb{R}^{d \times d_1}$ (poids) $b_1 \in \mathbb{R}^{d_1}$ (biais)
- $\theta = (W_1, b_1)$: paramètres du réseau

Construction d'un réseau de neurones

Single layer Perceptron - Régression



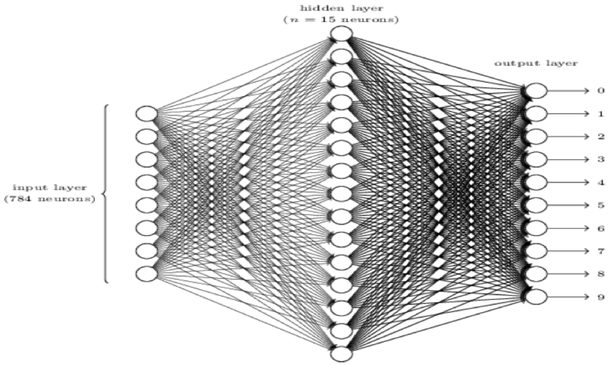
Source : `http://neuralnetworksanddeeplearning.com/index.html`

Ecriture condensée : $f_\theta(\mathbf{x}_0) = W_2\sigma_1\left(W_1\mathbf{x}_0 + b_1\right) + b_2$,

avec $W_1 \in \mathbb{R}^{d\times d_1}$, $b_1 \in \mathbb{R}^{d_1}$, $W_2 \in \mathbb{R}^{d_1\times 1}$, $b_2 \in \mathbb{R}$ et $\theta = (W_1, b_1, W_2, b_2)$

Construction d'un réseau de neurones

Single layer Perceptron - Classification



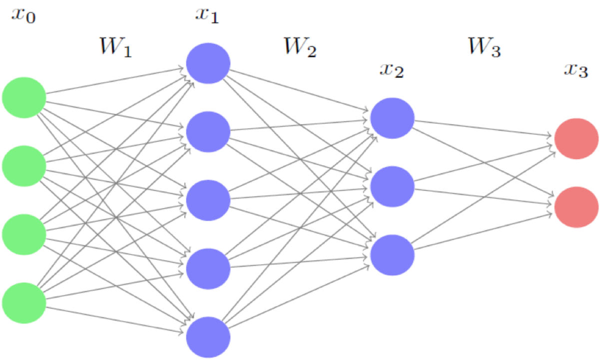
Source : `http://neuralnetworksanddeeplearning.com/index.html`

Ecriture condensée : $f_\theta(\mathbf{x}_0) = \sigma_{\text{softmax}}\left(W_2\sigma_1\left(W_1\mathbf{x}_0 + b_1\right) + b_2\right)$,

avec $W_1 \in \mathbb{R}^{d\times d_1}$, $b_1 \in \mathbb{R}^{d_1}$, $W_2 \in \mathbb{R}^{d_1\times K}$, $b_2 \in \mathbb{R}^K$ et $\theta = (W_1, b_1, W_2, b_2)$

Construction d'un réseau de neurones

Multi-layer Perceptron

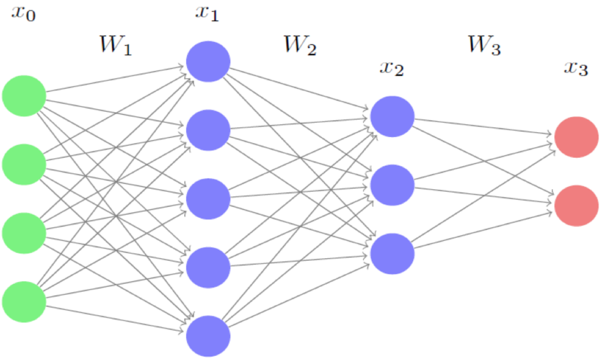


Source : `https://stats385.github.io/`

Ecriture condensée - entrée $\mathbf{x}_0 \in \mathbb{R}^d$, sortie $\mathbf{x}_L$, puis, pour $\ell = 1, \ldots, L$, faire $\mathbf{x}_\ell = \sigma_\ell \left(W_\ell \mathbf{x}_{\ell-1} + b_\ell \right)$ avec $\sigma_L = Id$ ou bien $\sigma_L = \sigma_{\text{softmax}}$

Construction d'un réseau de neurones

Multi-layer Perceptron

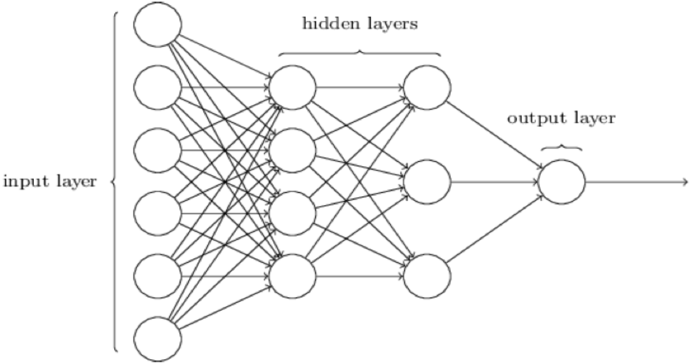


Source : `https://stats385.github.io/`

Si $\mathbf{x}_{L-1} = (x^{(1)}, \dots, x^{(K)})$ alors $[\sigma_{\text{softmax}}(\mathbf{x}_{L-1})]_k = \frac{\exp(x^{(k)})}{\sum_{\ell=1}^{K} \exp(x^{(\ell)})}$ pout tout $1 \leq k \leq K$.

Construction d’un réseau de neurones

Multi-layer Perceptron

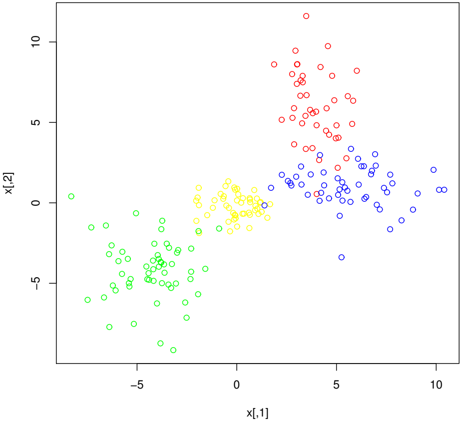


Source : `http://neuralnetworksanddeeplearning.com/index.html`

Réseau de neurones profonds : “nombreuses” couches cachées

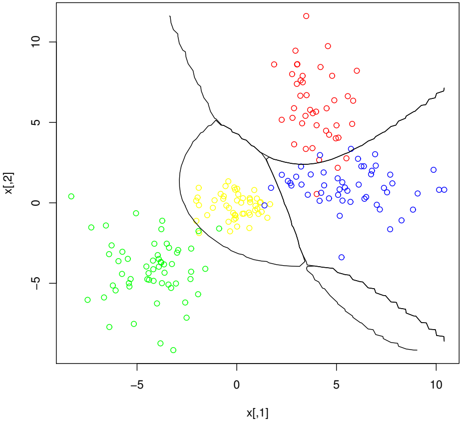
Exemple : classification dans $\mathbb{R}^2$ avec $K = 4$ classes

Visualisation des données - Mélange gaussien



Exemple : classification dans $\mathbb{R}^2$ avec $K = 4$ classes

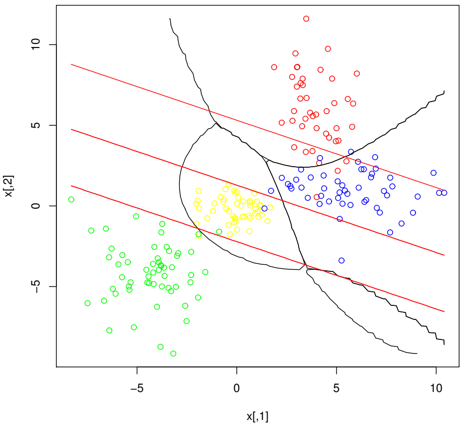
Classificiation optimale : règle de Bayes (indépendante des données)



Exemple : classification dans $\mathbb{R}^2$ avec $K = 4$ classes

Single-layer Perceptron avec 1 neurone dans la couche cachée

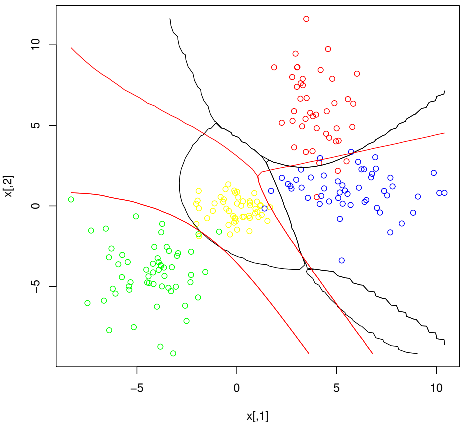
Optimisation par descente de gradient (BFGS, librairie `nnet` de R)



Exemple : classification dans $\mathbb{R}^2$ avec $K = 4$ classes

Single-layer Perceptron avec 2 neurones dans la couche cachée

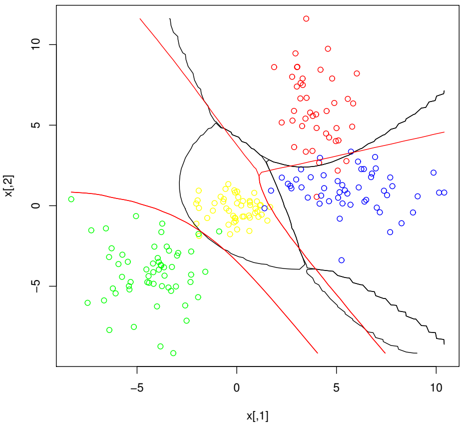
Optimisation par descente de gradient (BFGS, librairie `nnet` de R)



Exemple : classification dans $\mathbb{R}^2$ avec $K = 4$ classes

Single-layer Perceptron avec 10 neurones dans la couche cachée

Optimisation par descente de gradient (BFGS, librairie `nnet` de R)



Calcul du gradient par rétro-propagation de l'erreur

Soit f_θ un réseau de neurones - entrée $\mathbf{x}_0 \in \mathbb{R}^d$, sortie $\mathbf{x}_L = f_\theta(\mathbf{x}_0)$,
puis pour $\ell = 1, \ldots, L$, faire $\mathbf{x}_\ell = \sigma_\ell\left(W_\ell \mathbf{x}_{\ell-1} + b_\ell\right)$.

Paramètres : $\theta = (W_\ell, b_\ell)_{1 \leq \ell \leq L}$

Algorithme de back-propagation (Rumelhart et al. 1986 ; LeCun 1988) : calcul efficace du gradient de $\theta \mapsto f_\theta(\mathbf{x}_0)$ (avec $\mathbf{x}_0$ fixé) c'est à dire de

$$\frac{\partial}{\partial W_\ell} f_\theta(\mathbf{x}_0) \quad \text{et} \quad \frac{\partial}{\partial b_\ell} f_\theta(\mathbf{x}_0)$$

Calcul du gradient par rétro-propagation de l'erreur

Formulation par multiplicateur de Lagrange (LeCun 1988) pour le calcul du gradient de f_θ

- Optimisation (écriture sans le biais)

$$\min_{W,\mathbf{x}} \|\mathbf{x}_L - \mathbf{y}\|^2$$

sous-contraintes $\mathbf{x}_\ell = \sigma_\ell\left(W_\ell \mathbf{x}_{\ell-1}\right)$ pour $\ell = 1, \ldots, L$.

- Formulation lagrangienne (sans contrainte)

$$\min_{W,\mathbf{x},B} \mathcal{L}(W, \mathbf{x}, B)$$

avec

$$\mathcal{L}(W, \mathbf{x}, B) = \|\mathbf{x}_L - \mathbf{y}\|^2 + \sum_{\ell=1}^{L} B_\ell^T \left(\mathbf{x}_\ell - \sigma_\ell\left(W_\ell \mathbf{x}_{\ell-1}\right)\right)$$

Calcul du gradient par rétro-propagation de l'erreur

Différentiation du terme lagrangien

- $\frac{\partial}{\partial B}\mathcal{L} = 0$ implique que (forward pass) pour $\ell = 1, \ldots, L$

$$\mathbf{x}_\ell = \sigma_\ell\big(\underbrace{W_\ell \mathbf{x}_{\ell-1}}_{\mathbf{a}_\ell}\big)$$

- $\frac{\partial}{\partial \mathbf{x}}\mathcal{L} = 0$ implique que (backward pass)

$$\mathbf{z}_L = \underbrace{2\nabla\sigma_L\big(\mathbf{a}_L\big)(\mathbf{y} - \mathbf{x}_L)}_{\text{Erreur initiale}} \quad \text{et} \quad \underbrace{\mathbf{z}_\ell = \nabla\sigma_\ell\big(\mathbf{a}_\ell\big) W_{\ell+1}^T \mathbf{z}_{\ell+1}}_{\text{Rétro-propagation}}$$

et ainsi

$$\frac{\partial}{\partial W_\ell} f_\theta(\mathbf{x}_0) = \mathbf{z}_\ell \mathbf{x}_{\ell-1}^T$$

peut être calculé de façon récursive.

Remarque : $\nabla\sigma_\ell\big(\mathbf{a}_\ell\big)$ est une matrice diagonale !

Réseaux de neurones profond pour le traitement d'images

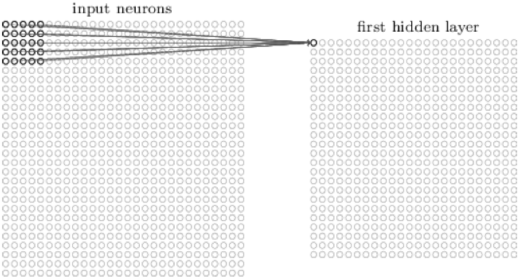
- Proposés par LeCun, Bottou, Bengio and Haffner (1998)... mais plusieurs semaines pour l'optimisation d'un réseau de neurones convolutionnel sur la base de données MNIST

- Révolution récente (après 2012) : possibilité d'entrainer un réseau de neurones profond sur des bases de données massives (ImageNet - 16 millions d'images couleur - $K = 20000$ classes, ou bien sur un sous-ensemble ILSVRC Challenge)

- Raisons du succès :
  - Moyens de calculs
  - Taille des bases d'apprentissage
  - Raffinement des méthodes d'optimisation (activation ReLU, régularisation par dropout,...)
  - Popularisation par librairies de calcul facilement utilisables

Couche cachée par convolutions de l'entrée

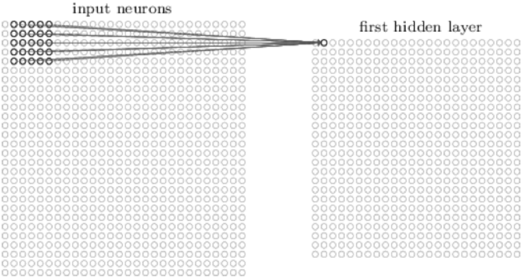
- Choix d'un filtre discret (ex de taille 5×5)
- La couche cachée est le résultat de la convolution de l'image en entrée par ce filtre (suivie par une fonction d'activation non-linéaire)
- Ecriture condensée : $f_\theta(\mathbf{x}_0) = \sigma_1 \left(W_1 \mathbf{x}_0 + b_1 \right)$, où W_1 est une matrice de Toeplitz très creuse



Source : `http://neuralnetworksanddeeplearning.com/index.html`

Couche cachée par convolutions de l'entrée

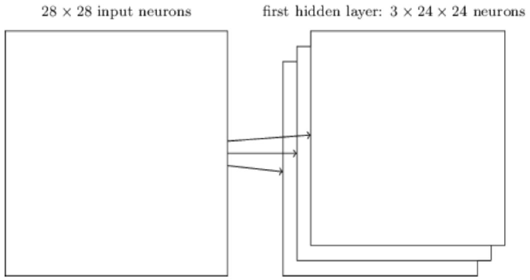
- Choix d'un filtre discret (ex de taille 5×5)
- La couche cachée est le résultat de la convolution de l'image en entrée par ce filtre (suivie par une fonction d'activation non-linéaire)
- Ecriture condensée : $f_\theta(\mathbf{x}_0) = \sigma_1\left(W_1\mathbf{x}_0 + b_1\right)$, où W_1 est une matrice de Toeplitz très creuse



Source : `http://neuralnetworksanddeeplearning.com/index.html`

Couche cachée par convolutions de l'entrée

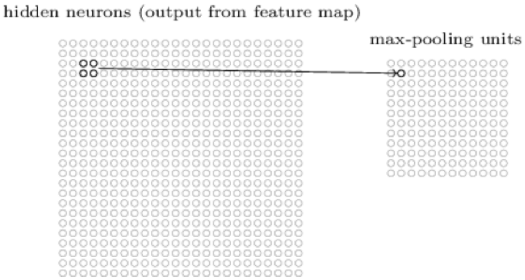
- Choix de plusieurs filtres discret (ex trois de de taille 5×5)
- La couche cachée est le résultat de la convolution de l'image en entrée par ces trois filtres (suivie par une fonction d'activation non-linéaire)
- Ecriture condensée : $f_\theta(\mathbf{x}_0) = \sigma_1 \left(W_1 \mathbf{x}_0 + b_1 \right)$, où W_1 est une matrice de Toeplitz par blocs



Source : `http://neuralnetworksanddeeplearning.com/index.html`

Couche cachée par pooling

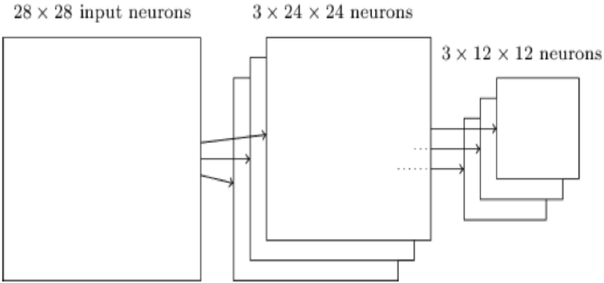
- Ajout d'une couche caché par max-pooling ou L_2-pooling (réduction de dimension et des effets de déformation locale)



Source : `http://neuralnetworksanddeeplearning.com/index.html`

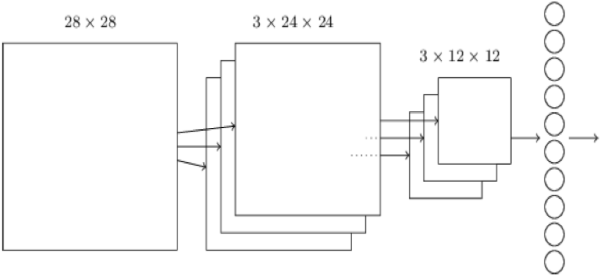
Couche cachée par pooling

- Ajout d'une couche caché par max-pooling ou L_2-pooling (réduction de dimension et des effets de déformation locale)



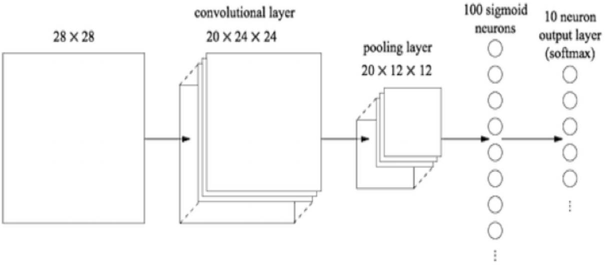
Source : `http://neuralnetworksanddeeplearning.com/index.html`

Couche finale fully-connected + softmax



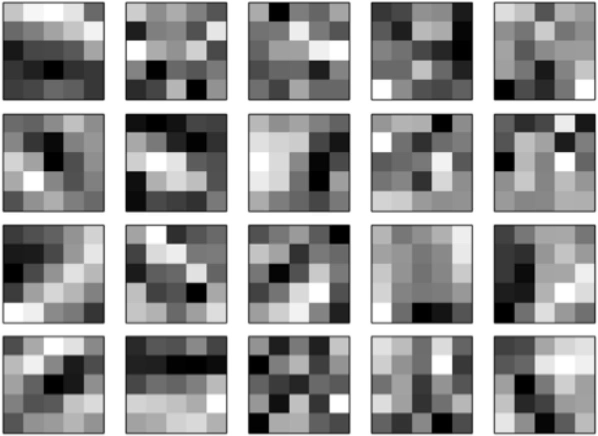
Source : `http://neuralnetworksanddeeplearning.com/index.html`

Couche finale fully-connected + perceptron + softmax



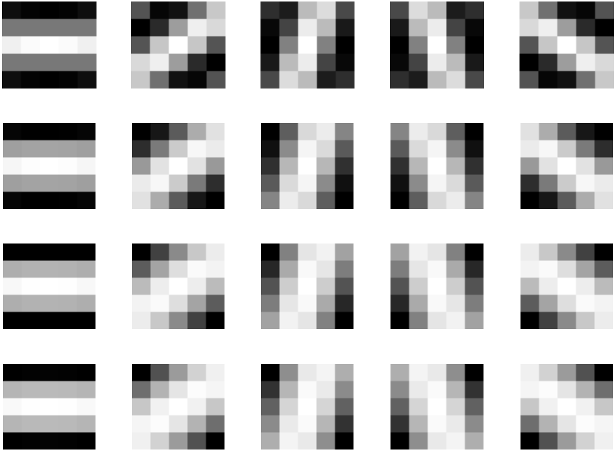
Source : `http://neuralnetworksanddeeplearning.com/index.html`

Exemple de filtres appris avec MNIST



Source : `http://neuralnetworksanddeeplearning.com/index.html`

Filtres de Gabor



Source : `MathWorks`

33 images mal-classées sur l'ensemble test ?

Attention : nombreux paramètres à choisir !

- choix des fonctions d'activation : sigmoïde, tanh, ReLU...
- choix du nombre de couches, nombre de neurones et du type d'opérations (convolution, pooling, fully-connected)
- taille des batch, nombre d'epoch
- taux d'apprentissage γ_j dans la descente de gradient stochastique
- régularisation explicite

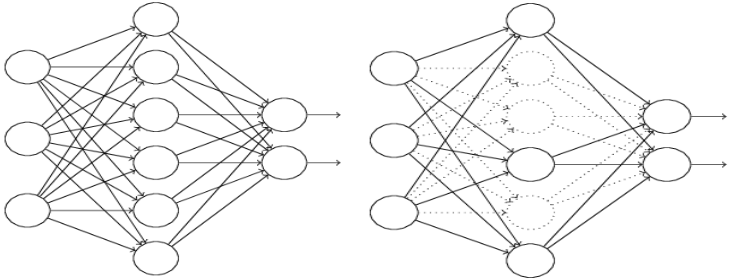
$$\min_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^{n} L(Y_i, f_\theta(X_i)) + \lambda \|\theta\|^2$$

- **et bien d'autres** : agrégation de réseaux de neurones, augmentation artificielle de la base de données d'apprentissage...

Régularisation par Dropout

Principe : suppression aléatoire de certains poids dans le réseau de neurones à chaque itération de la descente de gradient stochastique i.e répéter pour $j = 1, 2, \ldots$

1. choix d'un mini-batch à l'étape j
2. suppression aléatoire de certains poids (i.e. connexions) dans le réseau de neurones complet

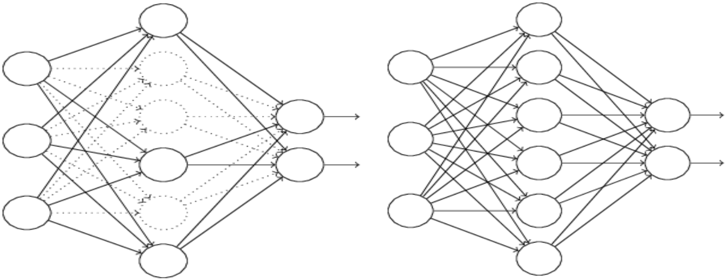


Source : `http://neuralnetworksanddeeplearning.com/index.html`

Régularisation par Dropout

Principe : suppression aléatoire de certains poids dans le réseau de neurones à chaque itération de la descente de gradient stochastique i.e répéter pour $j = 1, 2, \ldots$

3. pour chaque donnée du mini-batch propager l'entrée x dans le sous-réseau et rétropropager la sortie y pour calculer le gradient
4. revenir au réseau de neurones complet initial



Source : `http://neuralnetworksanddeeplearning.com/index.html`

33 images mal-classées sur l'ensemble test ?

Quel sont les paramètres finalement choisis ?

Rendez-vous sur le chapitre 6 du livre en ligne de Michael Nielsen ! !