Traitement et Analyse d'Images

Réseaux de neurones

Stratégies d'optimisation

olivier.bernard@insa-lyon.fr

Stratégies d’optimisation

Descente de gradient par mini-batch

Mini-batch gradient descent

Descente de gradient par mini-batch

- La taille de la base de données peut dépasser plusieurs millions
- Dans ce cas, il est impossible de charger la matrice X en mémoire CPU ou GPU
- On organise l'ensemble de la base de données en plusieurs « mini-batch » de taille plus petite (que l'on peut charger en mémoire)

Création de mini-batch $\{X^{\{t\}}, y^{\{t\}}\}$

$X = [x^{(1)}\ x^{(2)}\ \ x^{(3)}\ \ \cdots\ \ x^{(5.000.000)}]$ avec $X \in \mathbb{R}^{[n_x \times n_y \times m]}$

$y = [y^{(1)}\ y^{(2)}\ \ y^{(3)}\ \ \cdots\ \ y^{(5.000.000)}]$ avec $y \in \mathbb{R}^{[1 \times m]}$

$$X = [\underbrace{x^{(1)}\ x^{(2)}\ \ x^{(3)} \dots x^{(1000)}}_{\mathrm{X}^{\{1\}} \in \mathbb{R}^{[n_x \times n_y \times 1000]}} | \cdots | \cdots | \underbrace{\cdots\ x^{(5.000.000)}}_{\mathrm{X}^{\{5000\}} \in \mathbb{R}^{[n_x \times n_y \times 1000]}}]$$

$$y = [\underbrace{y^{(1)}\ y^{(2)}\ \ y^{(3)} \dots y^{(1000)}}_{\mathrm{y}^{\{1\}} \in \mathbb{R}^{[1 \times 1000]}} | \cdots | \cdots | \underbrace{\cdots\ y^{(5.000.000)}}_{\mathrm{y}^{\{5000\}} \in \mathbb{R}^{[1 \times 1000]}}]$$

Descente de gradient par mini-batch

Répéter jusqu'à convergence {

Pour $t = 1, \cdots, 5000$ {

propagation avant à partir de $X^{\{t\}}$

calcul de l'énergie $J^{\{t\}} = \frac{1}{1000}\sum_{i=1}^{1000} \mathcal{L}\left(\hat{y}^{(i)}, y^{(i)}\right)$

propagation arrière à partir de $J^{\{t\}}$ *(via* $\left(X^{\{t\}}, y^{\{t\}}\right)$*)*

$W^{[l]} := W^{[l]} - \alpha\, dW^{[l]}$

$b^{[l]} := b^{[l]} - \alpha\, db^{[l]}$

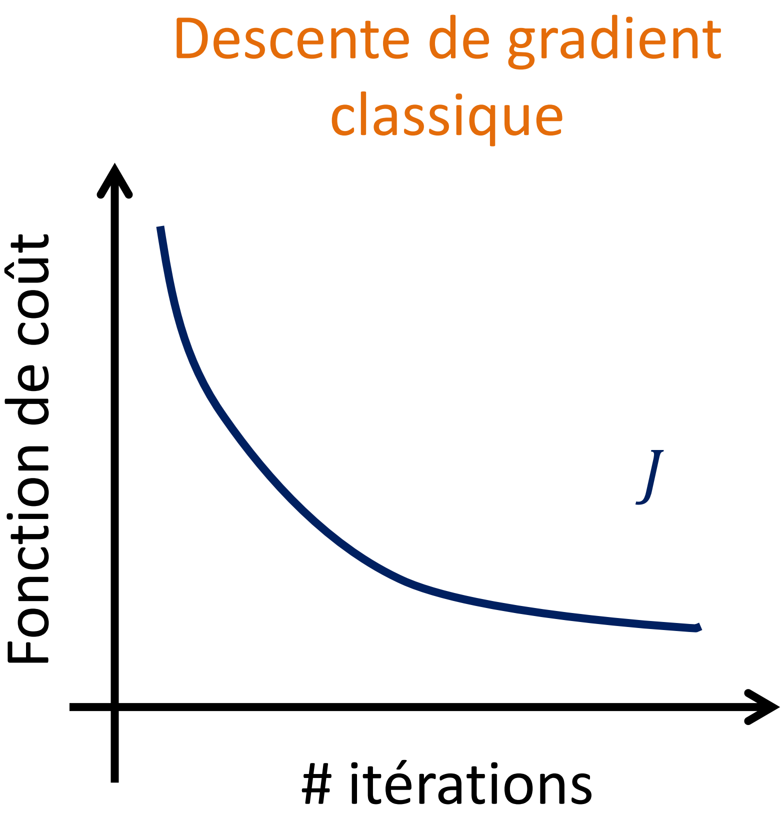
}

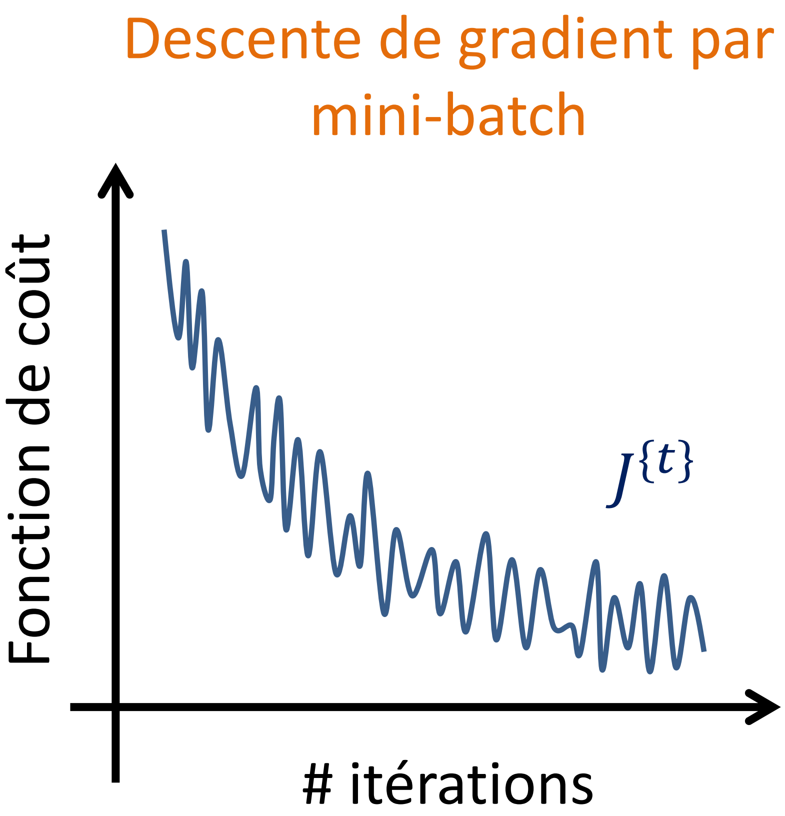
}

Descente de gradient par mini-batch

- 1 epoch = une passe au travers de l'ensemble de la base de données
- Dans l'exemple précédent, 1 epoch = 5000 itérations
- Lorsque l'on utilise une grande base de données, l'entrainement sur des mini-batch est beaucoup plus rapide

Entrainement par mini-batch





Choix de la taille des mini-batch

- Taille = m ⇨ descente de gradient classique

$$\left(X^{\{1\}}, y^{\{1\}}\right) = (X, y)$$

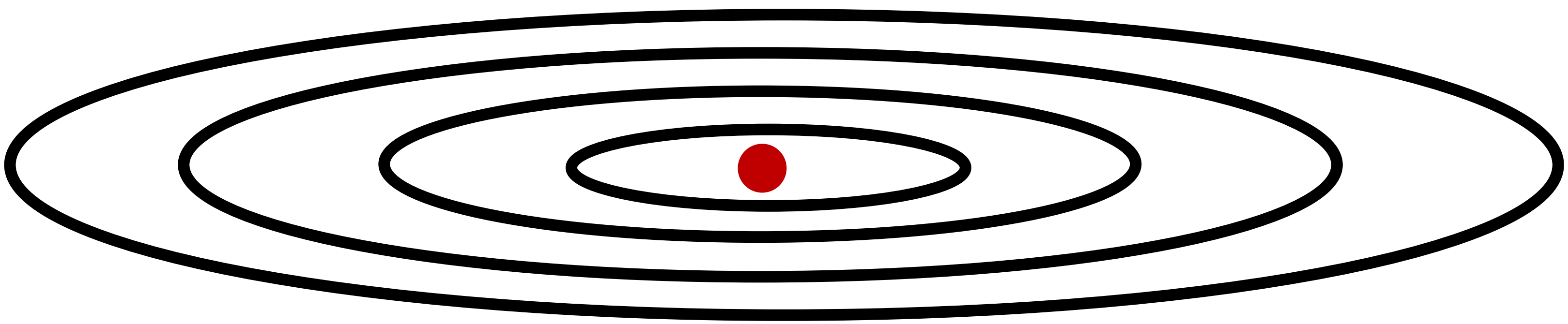
- Taille = 1 ⇨ descente de gradient stochastique

$$\left(X^{\{1\}}, y^{\{1\}}\right) = \left(x^{(1)}, y^{(1)}\right)$$

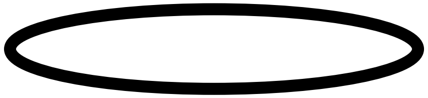
- Taille = s avec $1 < s < m$ ⇨ utilisée en pratique

$$\left(X^{\{1\}}, y^{\{1\}}\right) = \left(\left[x^{(1)} \dots x^{(s)}\right], \left[y^{(1)} \dots y^{(s)}\right]\right)$$

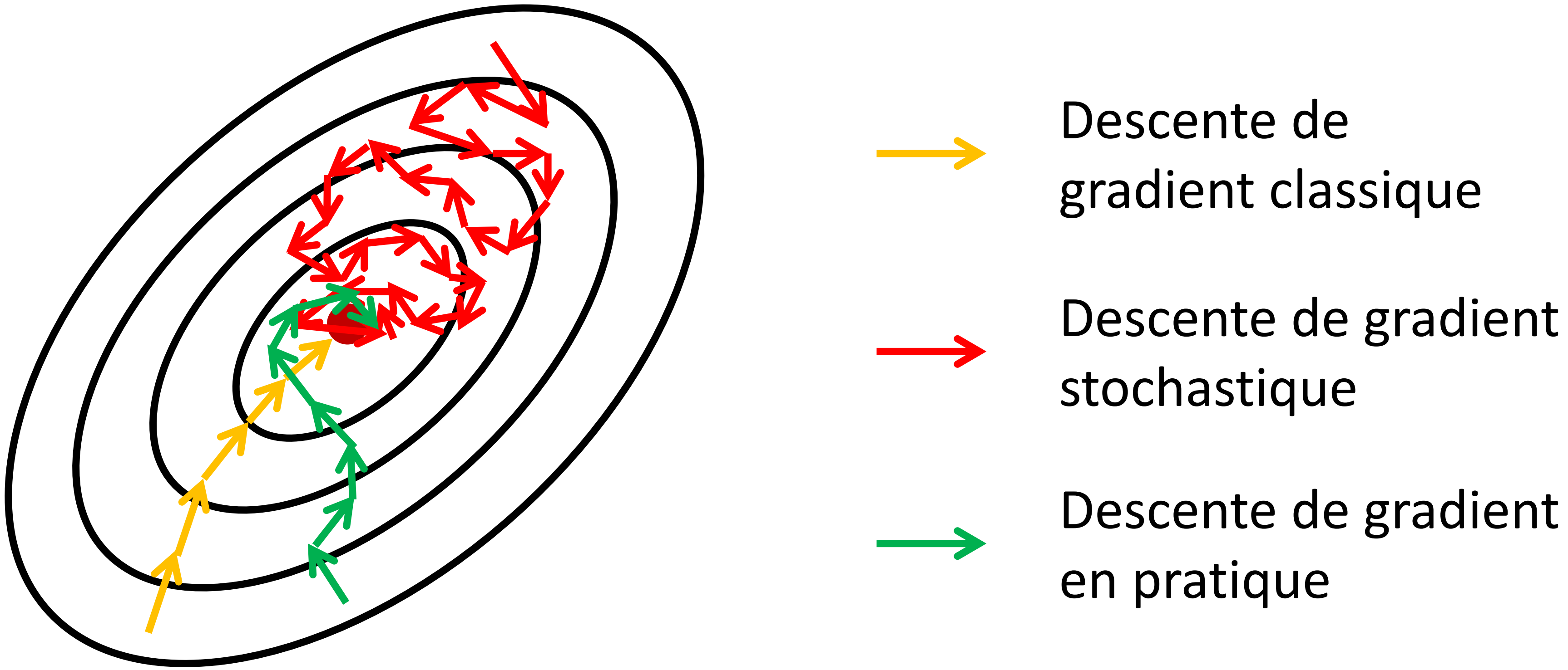
Choix de la taille des mini-batch



 Minimum de la fonction d’énergie

 Ligne de niveau de la fonction d’énergie

Choix de la taille des mini-batch



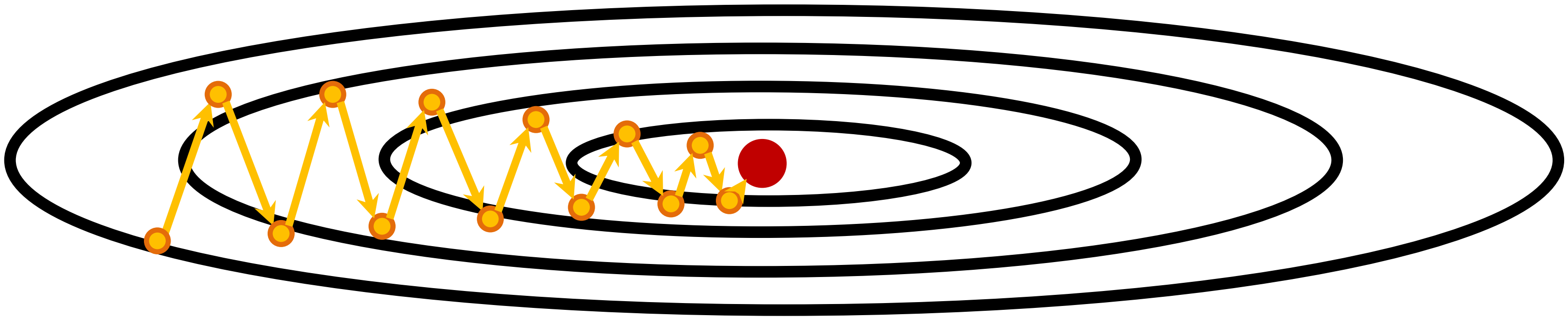
Choix de la taille des mini-batch

- Taille de la base de données est relativement petite
  - ⇒ implémenter une descente de gradient classique
- Sinon, taille typique de mini-batch
  - ⇒ 16, 32, 64, 128, 256
  - ⇒ Vérifier que la mémoire CPU/GPU n’est pas saturée

Optimisation de l'algorithme de descente de gradient

Motivations

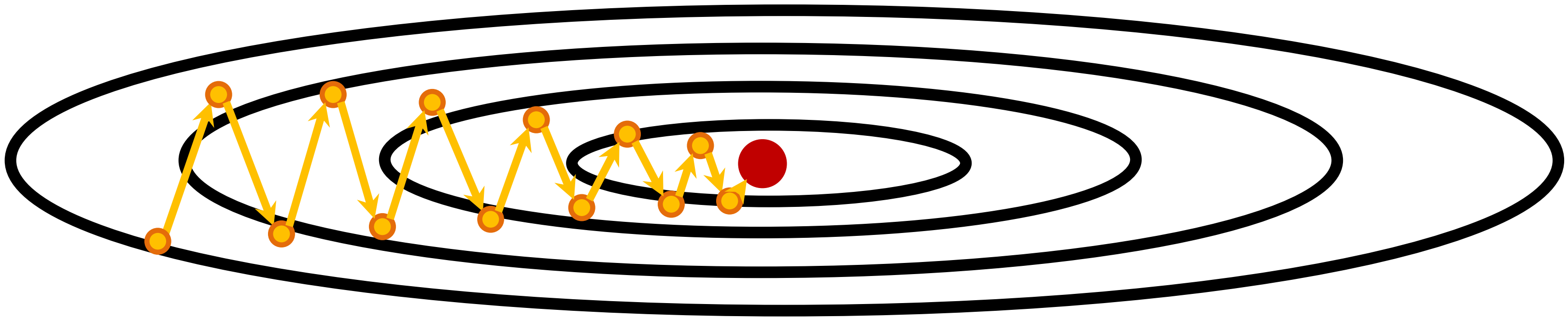
Motivations



- Oscillations de haut en bas ralentissent la convergence de l'algorithme de descente de gradient

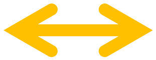
- Force l'utilisation d'un taux d'apprentissage α petit pour ne pas diverger ⇒ apprentissage plus lent

Motivations



- Idéalement nous voulons

Apprentissage plus faible



Apprentissage plus rapide

L'introduction d'inertie permettrait de converger plus rapidement !

Moyenne pondérée exponentielle

- Outil permettant d'exploiter des propriétés d'inertie pour le suivi de mesures

Mesures

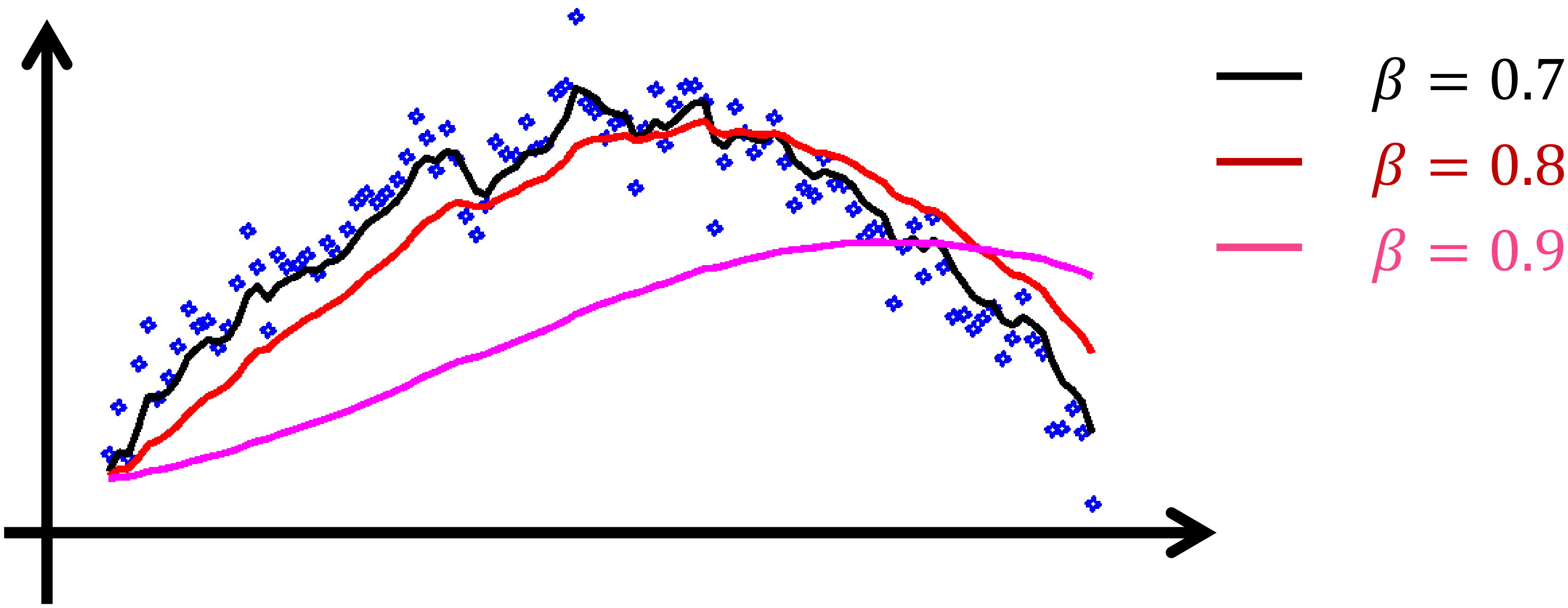
Valeurs associées calculées

$$y_1, y_2, y_3, \cdots, y_t \quad \Rightarrow \quad \begin{cases} v_0 = 0 \\ v_t = \beta\, v_{t-1} + (1-\beta)\, y_t \end{cases}$$

- Moyenne les mesures autour de $\approx \frac{1}{1-\beta}$

Moyenne pondérée exponentielle

- Outil permettant d'exploiter des propriétés d'inertie pour le suivi de mesures



Descente de gradient avec élan

- Gradient descent with momentum

- Momentum

Descente de gradient classique

Répéter jusqu'à convergence
{
propagation avant
propagation arrière

$$W^{[l]} := W^{[l]} - \alpha\, dW$$
$$b^{[l]} := b^{[l]} - \alpha\, db$$
}

Descente de gradient avec élan

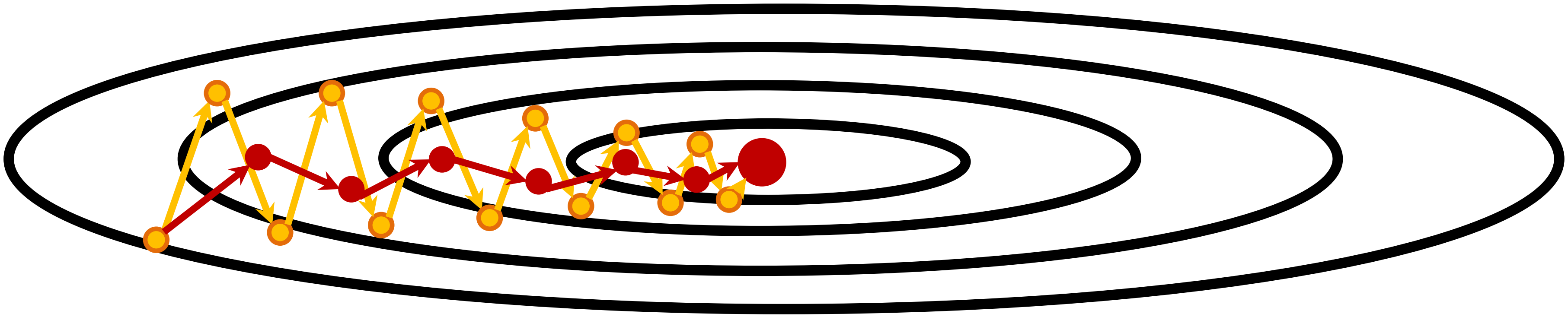
$V_{dw} = 0$ et $V_{db} = 0$

Répéter jusqu'à convergence
{
propagation avant
propagation arrière

$$V_{dw} := \beta V_{dw} + (1-\beta)dW$$
$$V_{db} := \beta V_{db} + (1-\beta)db$$

$$W^{[l]} := W^{[l]} - \alpha V_{dw}$$
$$b^{[l]} := b^{[l]} - \alpha V_{db}$$
}

Descente de gradient avec élan



Descente de gradient classique

Descente de gradient classique avec élan

- Classiquement on choisit $\beta = 0.9$

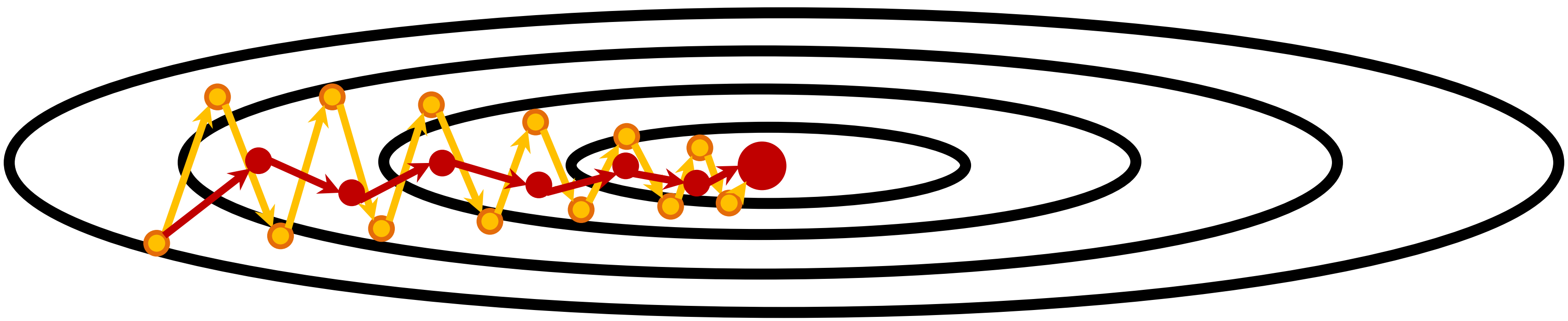
Moyenne avec ≈ 10 dernières valeurs de gradient

Descente de gradient RMSprop

RMSprop

Descente de gradient « RMSprop »

- Même idée que pour la descente de gradient avec élan
- Introduction de phénomène d'inertie lors de la descente de gradient



Descente de gradient classique

Répéter jusqu'à convergence
{
propagation avant
propagation arrière

$$W^{[l]} := W^{[l]} - \alpha\, dW$$
$$b^{[l]} := b^{[l]} - \alpha\, db$$
}

Descente de gradient « RMSprop »

$S_{dw} = 0$ et $S_{db} = 0$

Répéter jusqu'à convergence
{
propagation avant
propagation arrière

$$S_{dw} := \beta S_{dW} + (1-\beta) dW^2$$
$$S_{db} := \beta S_{db} + (1-\beta) db^2$$

$$W^{[l]} := W^{[l]} - \alpha\, dw/\sqrt{S_{dw} + \varepsilon}$$
$$b^{[l]} := b^{[l]} - \alpha\, db/\sqrt{S_{db} + \varepsilon}$$
}

Optimisation ADAM

ADAptive Moment estimation

Optimisation ADAM

- Utilisation de l'algorithme Momentum et RMSprop

$V_{dw} = 0, V_{db} = 0, S_{dW} = 0$ et $S_{db} = 0$

Répéter jusqu'à convergence

{

propagation avant

propagation arrière

$$V_{dw} := \beta V_{dW} + (1-\beta)\, dW \;,\; V_{db} := \beta V_{db} + (1-\beta)\, db$$
$$S_{dw} := \beta S_{dW} + (1-\beta)\, dW^2 \,,\; S_{db} := \beta S_{db} + (1-\beta)\, db^2$$

$$W^{[l]} := W^{[l]} - \alpha \frac{V_{dW}}{\sqrt{S_{dw} + \epsilon}} \;,\; b^{[l]} := b^{[l]} - \alpha \frac{V_{db}}{\sqrt{S_{db} + \epsilon}}$$

}

Optimisation ADAM

- Hyperparamètres

α: Nécessité de le fixer à la main

β_1: 0.9 $\rightarrow$ (dw)

β_2: 0.98 $\rightarrow$ (dw^2)

ε: 10^{-8}

Normalisation par batch

Batch normalization

Normalisation par batch

- Normalisation de la sortie d'une couche via la moyenne et l'écart type de chaque batch
- Déplacement β et échelle γ optimaux appris par le réseau

Stabilise le processus d'apprentissage

Permet une convergence plus rapide

Normalisation par batch

Entrées — Valeurs de $x^{(i)}$ d'un mini-batch $X^{\{t\}} = \left\{x^{(1)} \cdots x^{(T)}\right\}$

Paramètres γ et β à apprendre

Sortie — $\left\{\widetilde{x^{(i)}} = BN_{\gamma,\beta}\left(x^{(i)}\right)\right\}$

$$\mu_{X^{\{t\}}} = \frac{1}{T}\sum_{i=1}^{T} x^{(i)} \qquad \sigma^2_{X^{(t)}} = \frac{1}{T}\sum_{i=1}^{T}\left(x^{(i)} - \mu_{X^{\{t\}}}\right)^2$$

Mini-batch mean and variance

$$\widehat{x^{(i)}} = \frac{x^{(i)} - \mu_{X^{\{t\}}}}{\sqrt{\sigma^2_{X^{(t)}} + \epsilon}}$$

Normalize

$$\widetilde{x^{(i)}} = \gamma\, \widehat{x^{(i)}} + \beta \equiv BN_{\gamma,\beta}\left(x^{(i)}\right)$$

Scale and shift

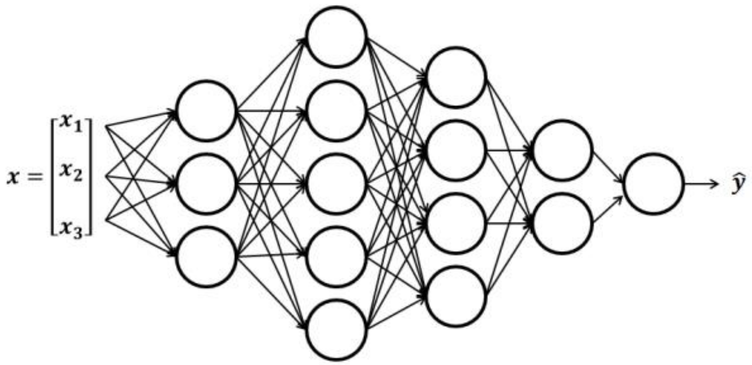
Dropout

Dropout

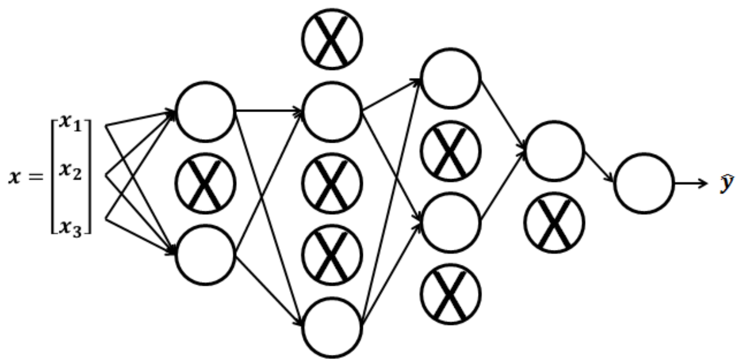
- Pendant l'entrainement, éteindre des neurones de façon aléatoire avec une probabilité *p*

Meilleure distribution de l'information extraite de la couche précédente

- Apprentissage implicite d'un ensemble de réseaux



Réseau de neurones profond standard



Réseau de neurones profond avec dropout

Gestion d'une base de données

Entrainement / validation / test

Tout est question de distribution !

▶ Motivation

L'entrainement doit permettre la généralisation de l'algorithme sur des nouvelles bases de données

▶ Comment caractériser la distribution d'une base de données ?

$$p_\theta(x) = ?$$

Tout est question de distribution !

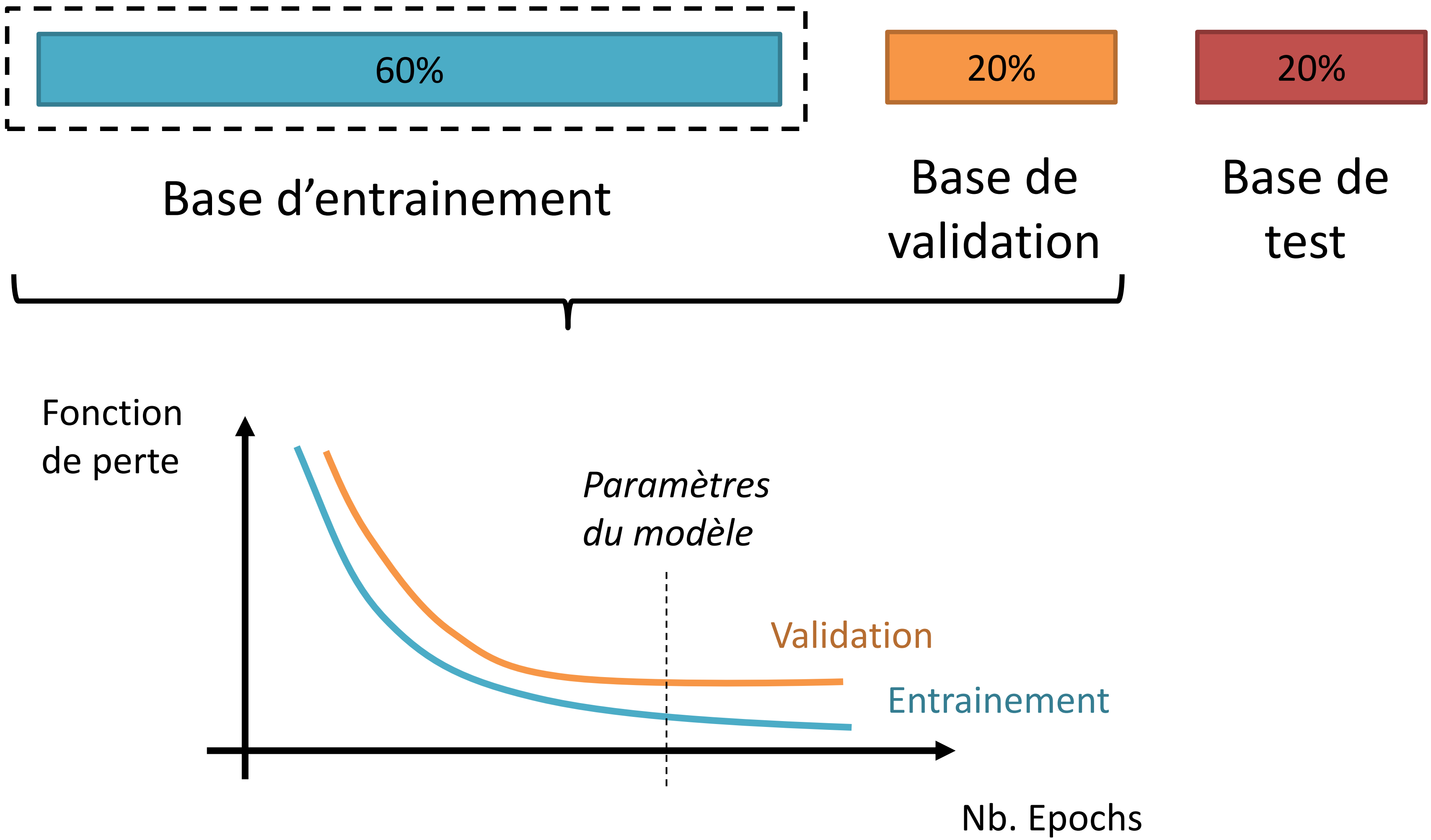
- Estimation de la distribution au travers de critères plus simples
  - Age, genre, couleur de peau, …

- Création de 3 bases de données en respectant la(les) distribution(s)

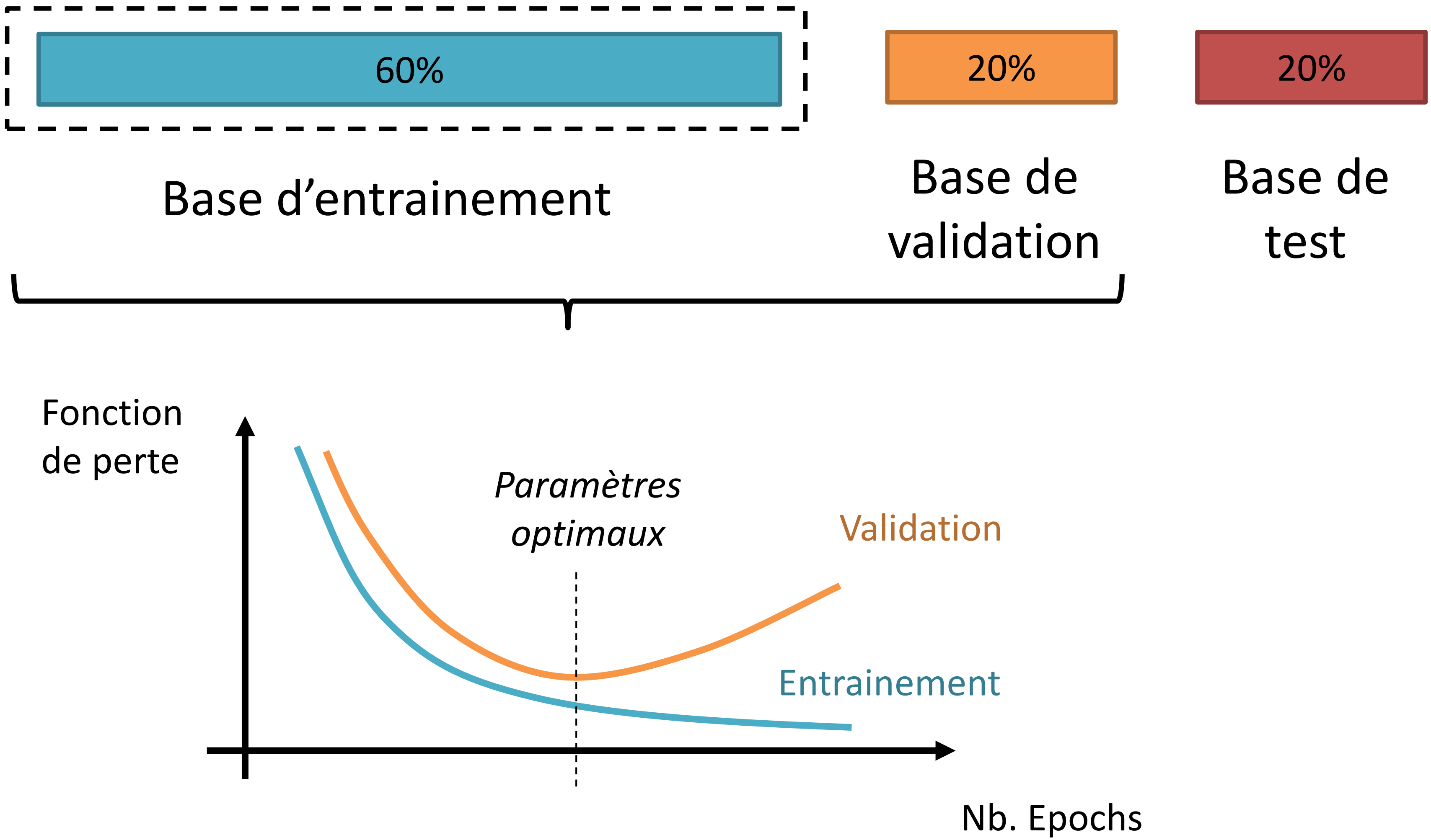
Base d'entrainement	60%
Base de validation	20%
Base de test	20%

Entrainement Validation Test

Apprentissage d’un modèle

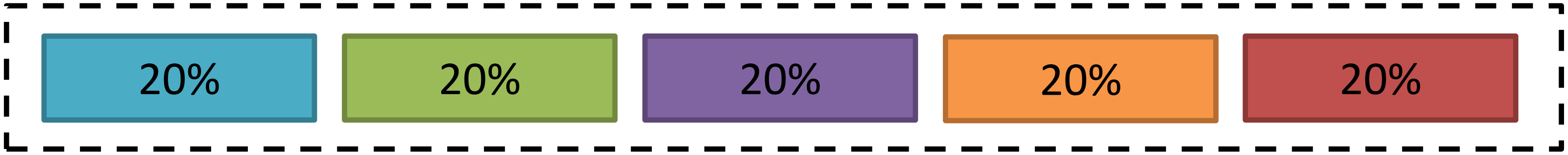


Sur-apprentissage



Validation croisée

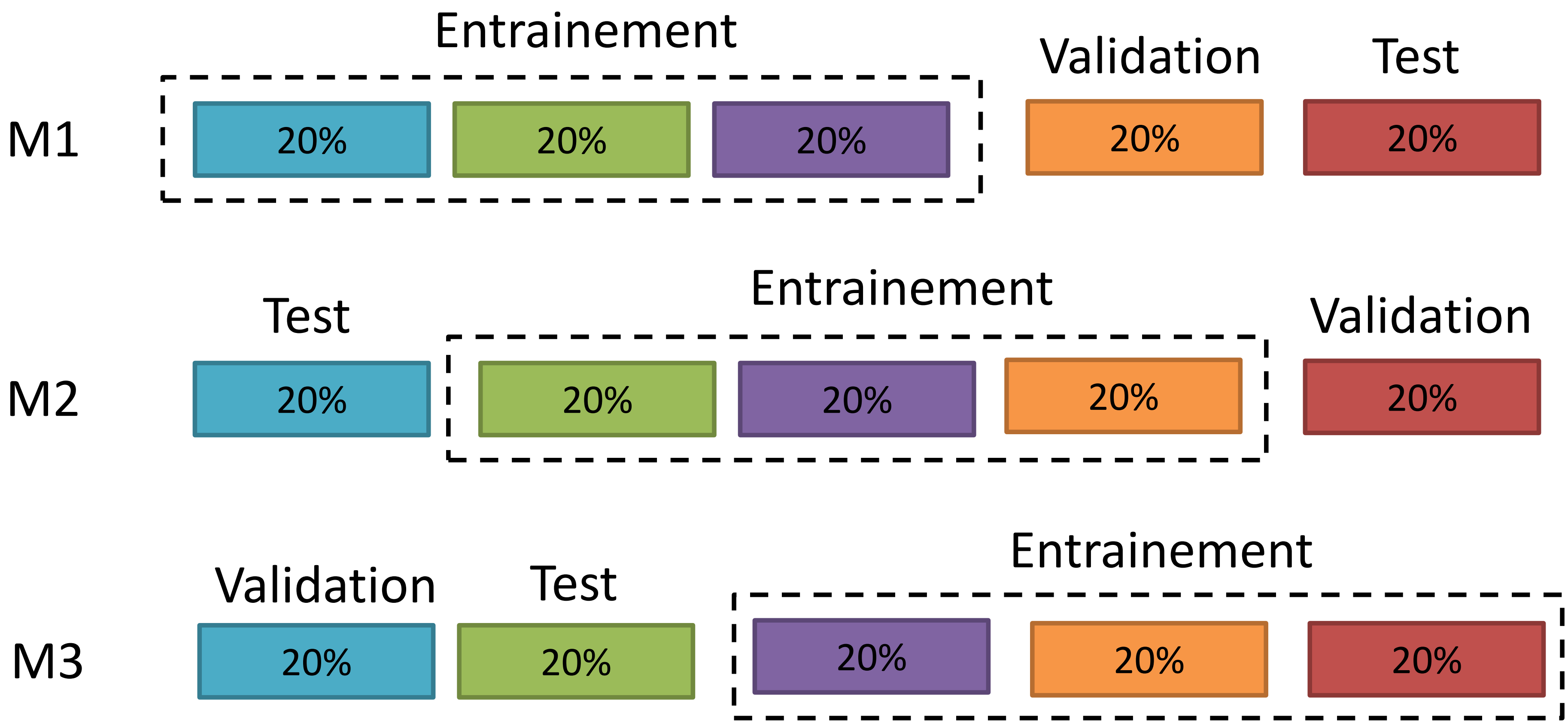
- Découpage de l'ensemble de la base de données en (5) sous-groupes ayant les même distributions



Ensemble de la base de données

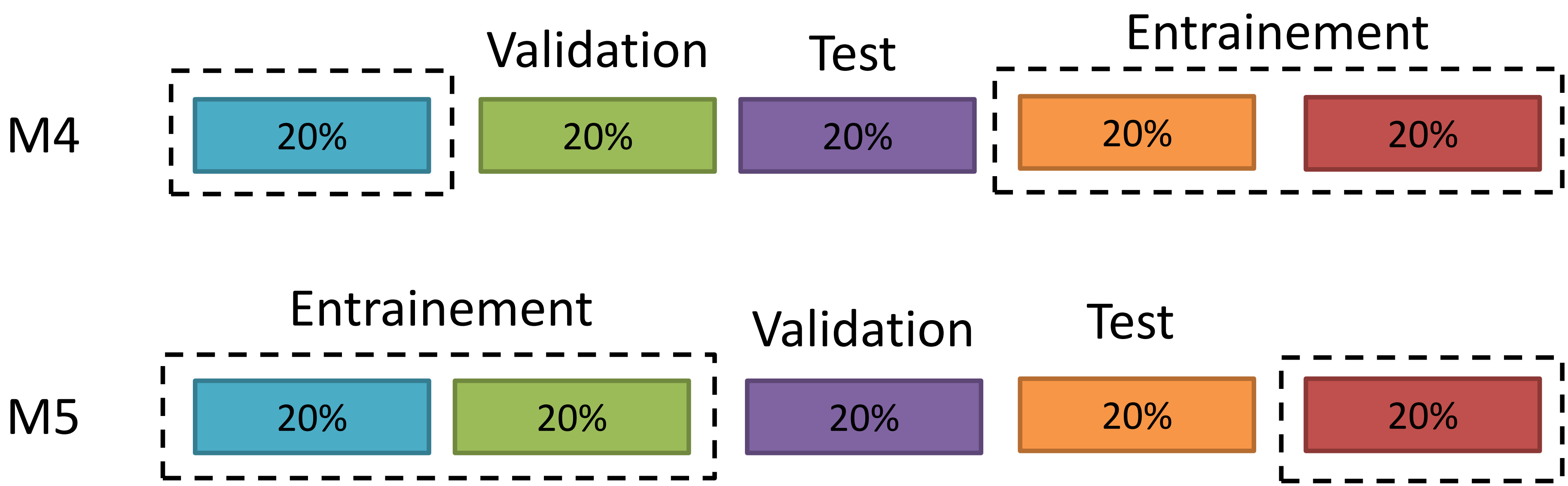
Validation croisée

- Apprentissage de (5) modèles: l'ensemble de la bd sera testée !



Validation croisée

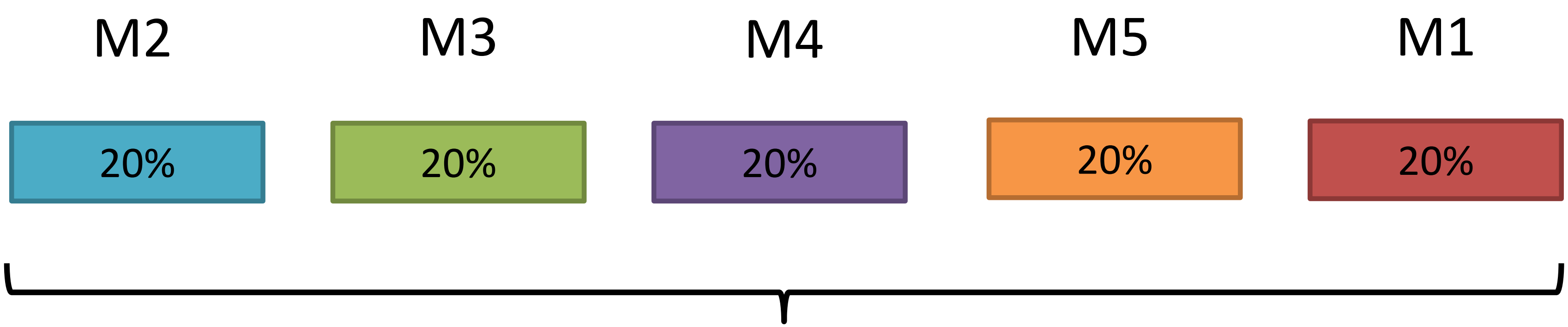
- Apprentissage de (5) modèles: l'ensemble de la bd sera testée !



Validation croisée

- Apprentissage de (5) modèles: l’ensemble de la bd sera testée !

L’ensemble de la base de données est vue en tant que base de test au travers des (5) modèles



Calcul des performances moyennes

That’s all folks