

# **Gradient Stochastique**

**Séance 3/5**

**Florentin Goyens 12 octobre 2022**

Construire un algorithme qui décide s'il y a un feu vert dans une image :



Oui

Non

Construire un algorithme qui décide s'il y a un feu vert dans une image :



Oui

Construire un algorithme qui décide s'il y a un feu vert dans une image :



Oui

Construire un algorithme qui décide s'il y a un feu vert dans une image :



**Non**

Construire un algorithme qui décide s'il y a un feu vert dans une image :



Oui

# Construire un algorithme qui décide s'il y a un feu vert dans une image :

Data:



⋮ ⋮

$a_1$

$a_2$

$a_3$

$a_4$

Labels:  $b_1 = 1$        $b_2 = -1$        $b_3 = 1$        $b_4 = 1$       ⋮ ⋮

On dispose de  $n$  images  $a_i$ , chaque image est un vecteur de dimension  $m$  = "nombre de pixels"

Chaque image  $a_i$  a un label  $b_i = 1$  s'il y a un feu vert,  $-1$  sinon

Construire un algorithme qui décide s'il y a un feu vert dans une image :



Oui

Donnée/Input  $a_i \in \mathbb{R}^m$

Label  $b_i$

Trouver un modèle qui assigne le bon label à chaque input

$$h: \mathbb{R}^m \rightarrow \mathbb{R}: h(a) = \pm 1$$

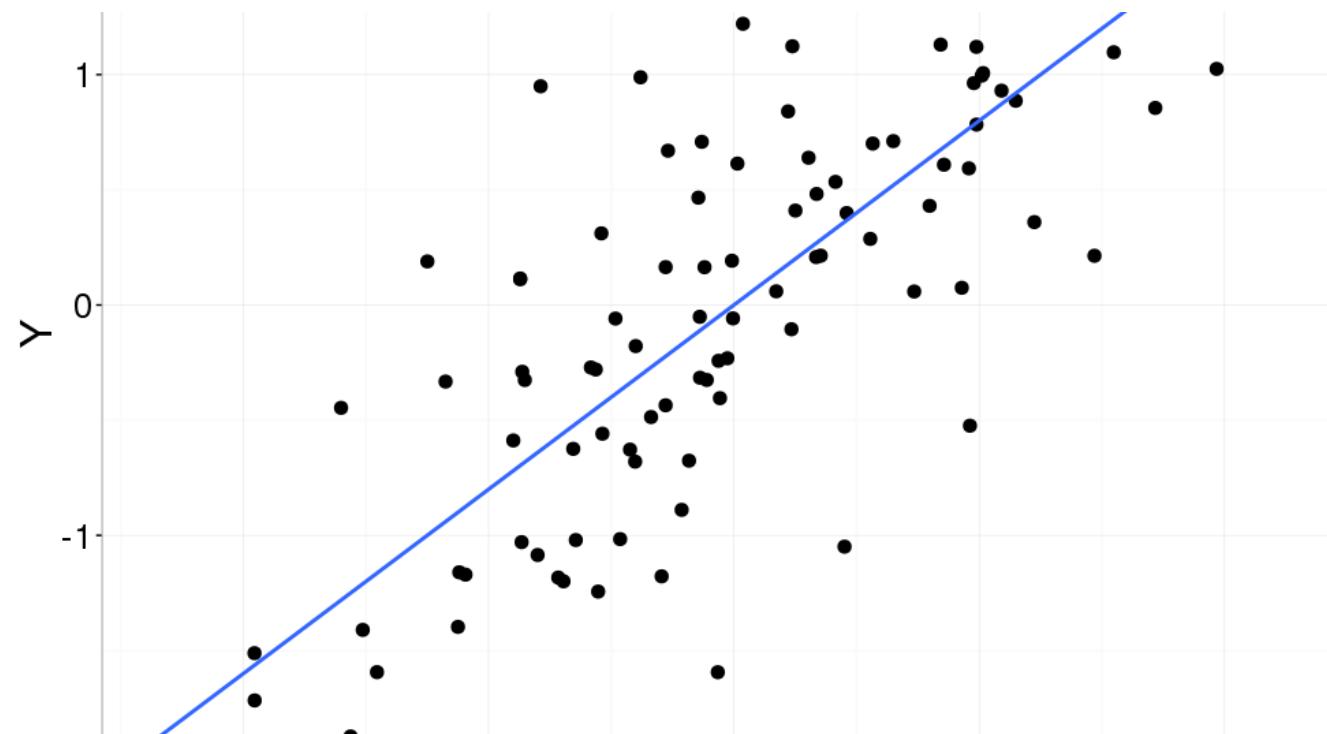
# Comment choisir le modèle $h$ ?

$$h: a \in \mathbb{R}^m \rightarrow b$$

$h$  est un modèle qui dépend de paramètres  $x \in \mathbb{R}^d$

**Exemples:**

Linéaire :  $h_x(a) = a^\top x = a_1x_1 + a_2x_2 + \dots + a_mx_m$



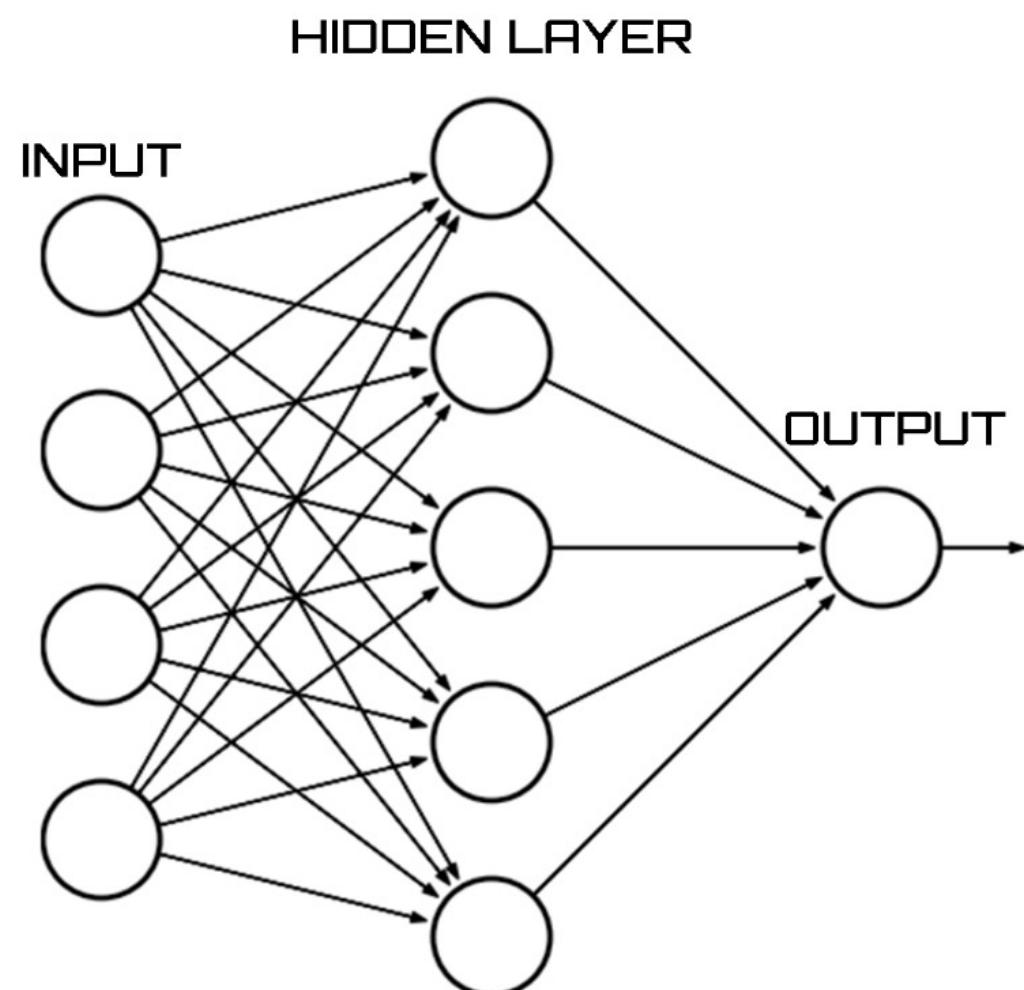
# Comment choisir le modèle $h$ ?

$$h: a \in \mathbb{R}^m \rightarrow b$$

$h$  est un modèle qui dépend de paramètres  $x \in \mathbb{R}^d$

Exemples:

Réseau de neurones :  $h(a) = w_2^\top \sigma(w_1 a)$



$$w_1 \in \mathbb{R}^{q \times m} \text{ et } w_2 \in \mathbb{R}^q$$

$\sigma$  est une transformation non linéaire

# Le modèle $h$ est choisi par un problème d'optimisation

**Goal:** Trouver  $x$  tel que pour les données  $(a_i, b_i)$  pour  $i = 1, \dots, n$ , on a

$$h_x(a_i) \approx b_i \text{ pour } i = 1, \dots, n$$

On veut que notre modèle donne le bon label pour les données disponibles, dans l'espoir qu'il pourra correctement prédire le label de nouvelles données (une image jamais observée )

C'est ce qu'on appelle l'entraînement du modèle ou "learning"

**Formulation mathématique:** Trouver  $x$  qui minimize une fonction de coût

$$\min_{x \in \mathbb{R}^d} F(x) = \frac{1}{n} \sum_{i=1}^n (h_x(a_i) - y_i)^2 \leftarrow$$

Autant de termes que de données !

# Problème d'optimisation

Goal:

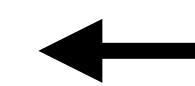
Trouver  $x$  tel que pour les données  $(a_i, b_i)$ , on a

$$h_x(a_i) \approx b_i \text{ pour } i = 1, \dots, n$$

Formulation mathématique:

Trouver  $x$  qui minimize une fonction de coût

$$\min_{x \in \mathbb{R}^d} F(x) = \frac{1}{n} \sum_{i=1}^n \ell(h_x(a_i), y_i)$$



On peut utiliser n'importe quelle fonction de coût

# Problème d'optimisation

Goal:

Trouver  $x$  tel que pour les données  $(a_i, b_i)$ , on a

$$h_x(a_i) \approx b_i \text{ pour } i = 1, \dots, n$$

Formulation mathématique:

Trouver  $x$  qui minimize une fonction de coût

$$\min_{x \in \mathbb{R}^d} F(x) = \frac{1}{n} \sum_{i=1}^n \ell(h_x(a_i), y_i) \quad \leftarrow \text{On peut utiliser n'importe quelle fonction de coût}$$

$$= \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# Minimisation d'une somme finite

$$\min_{x \in \mathbb{R}^d} F(x) = -\frac{1}{n} \sum_{i=1}^n f_i(x)$$

Méthode classique: descent de gradient

$$\nabla F(x) = \nabla \left( \frac{1}{n} \sum_{i=1}^n f_i(x) \right) = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x)$$

---

## Gradient Descent Algorithm

Set  $w^0 = 0$ , choose  $\alpha > 0$ .  
for  $t = 0, 1, 2, \dots, T-1$   
     $w^{t+1} = w^t - \frac{\alpha}{n} \sum_{i=1}^n \nabla f_i(w^t)$   
Output  $w^T$

---

Cela nécessite de calculer un gradient pour chaque image dans le data set, cela peut être énorme !

# Minimisation d'une somme finite

$$\min_{x \in \mathbb{R}^d} F(x) = -\frac{1}{n} \sum_{i=1}^n f_i(x)$$

Typiquement, on a  $n$  beaucoup plus grand que  $d$

C'est à dire, beaucoup plus d'images que de paramètres dans notre modèle.

Comment peut-on utiliser cette structure ? Idée : trouver une méthode qui utilise le gradient d'une seule fonction  $f_i(x)$  à chaque itération

utiliser  $\nabla f_j(x) \approx \nabla F(x)$

Estimation non biaisée: sélectionner au hasard un indice  $j \in \{1, \dots, n\}$  donne le bon gradient en moyenne

$$\mathbb{E}[\nabla f_j(x)] = \frac{1}{n} \sum_{i=1}^n \nabla f_i(x) = \nabla F(x)$$

# Méthode gradient stochastique

## SGD 0.0 Constant stepsize

Set  $w^0 = 0$ , choose  $\alpha > 0$

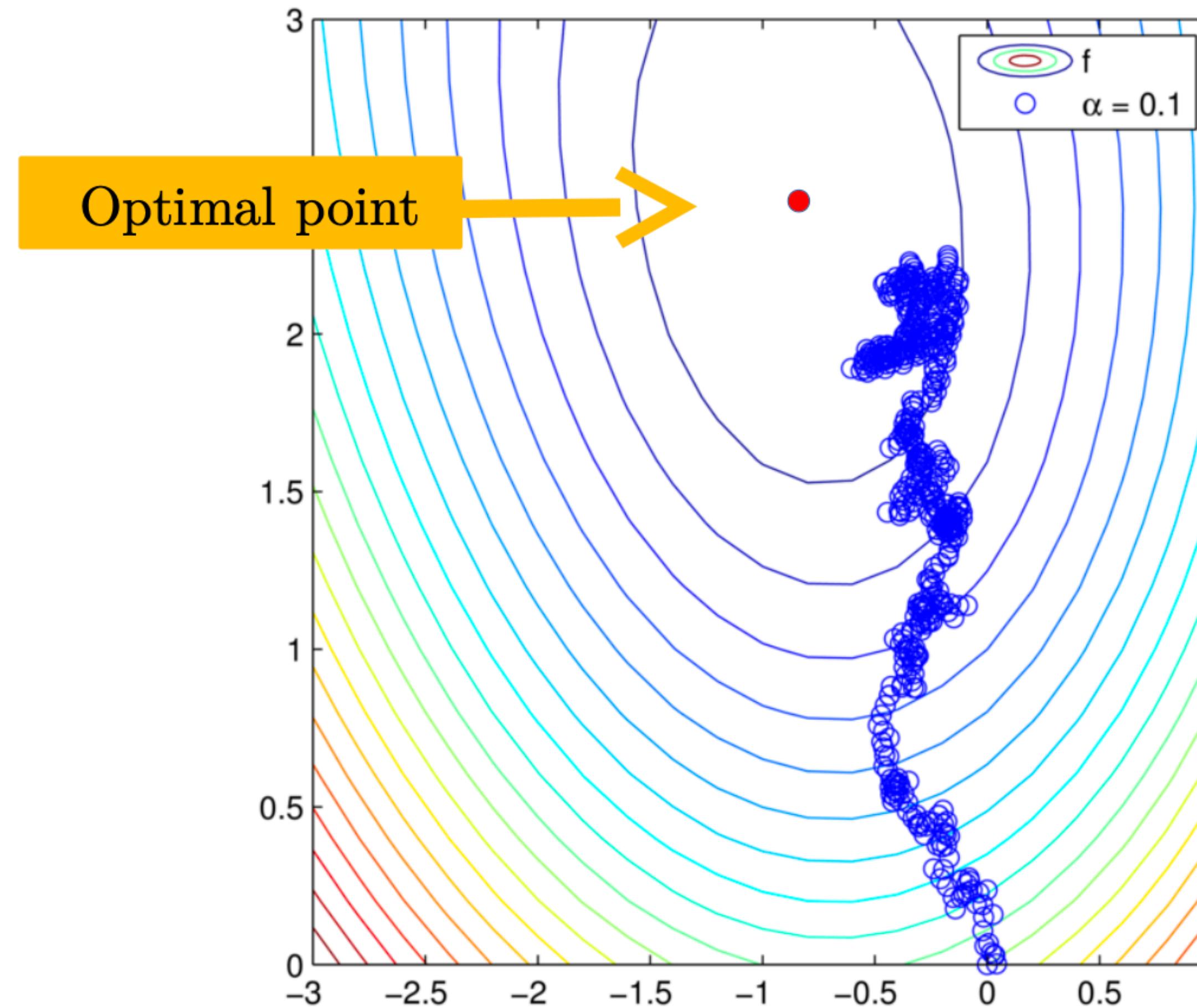
for  $t = 0, 1, 2, \dots, T - 1$

sample  $j \in \{1, \dots, n\}$

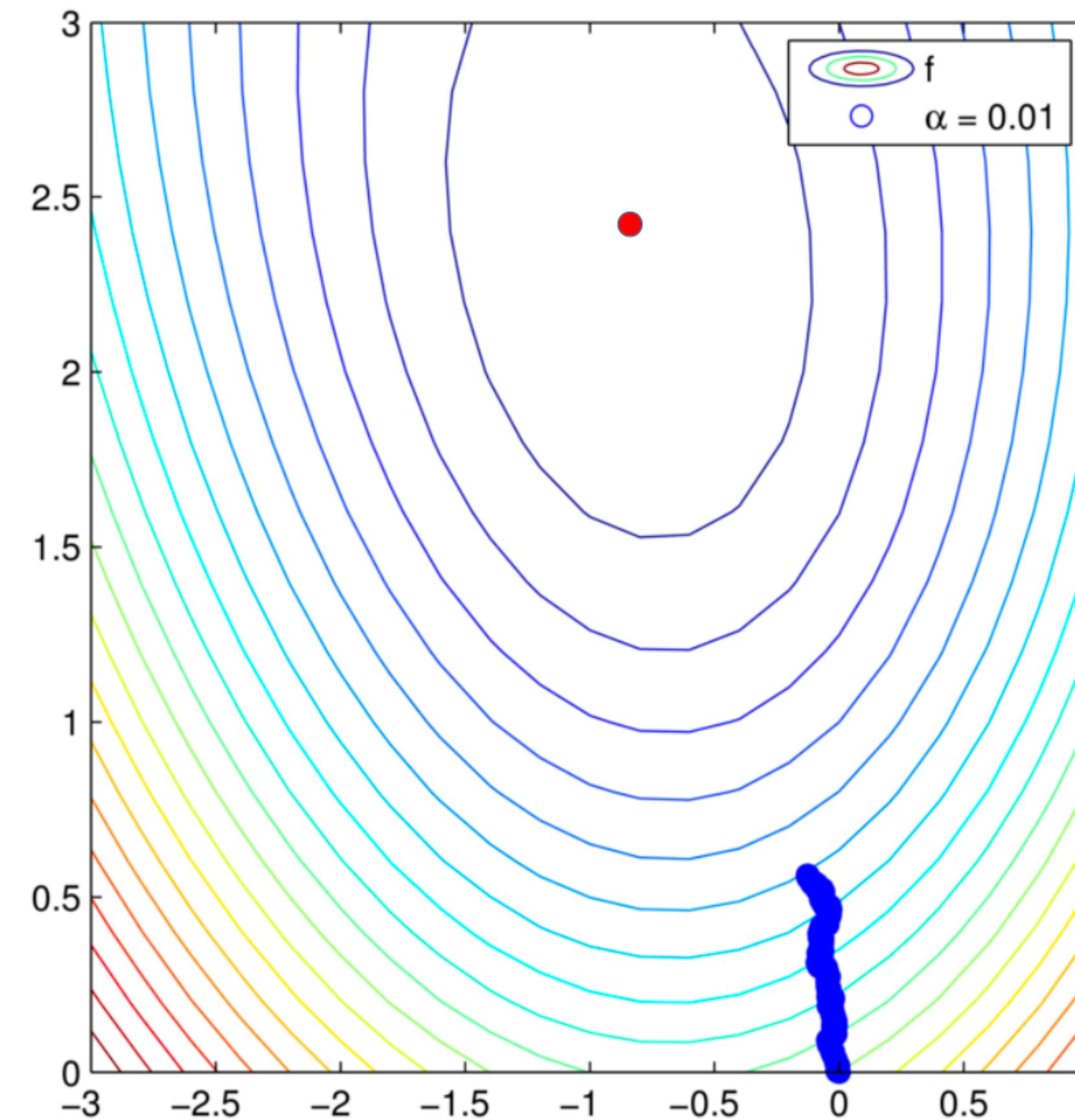
$$w^{t+1} = w^t - \alpha \nabla f_j(w^t)$$

Output  $w^T$

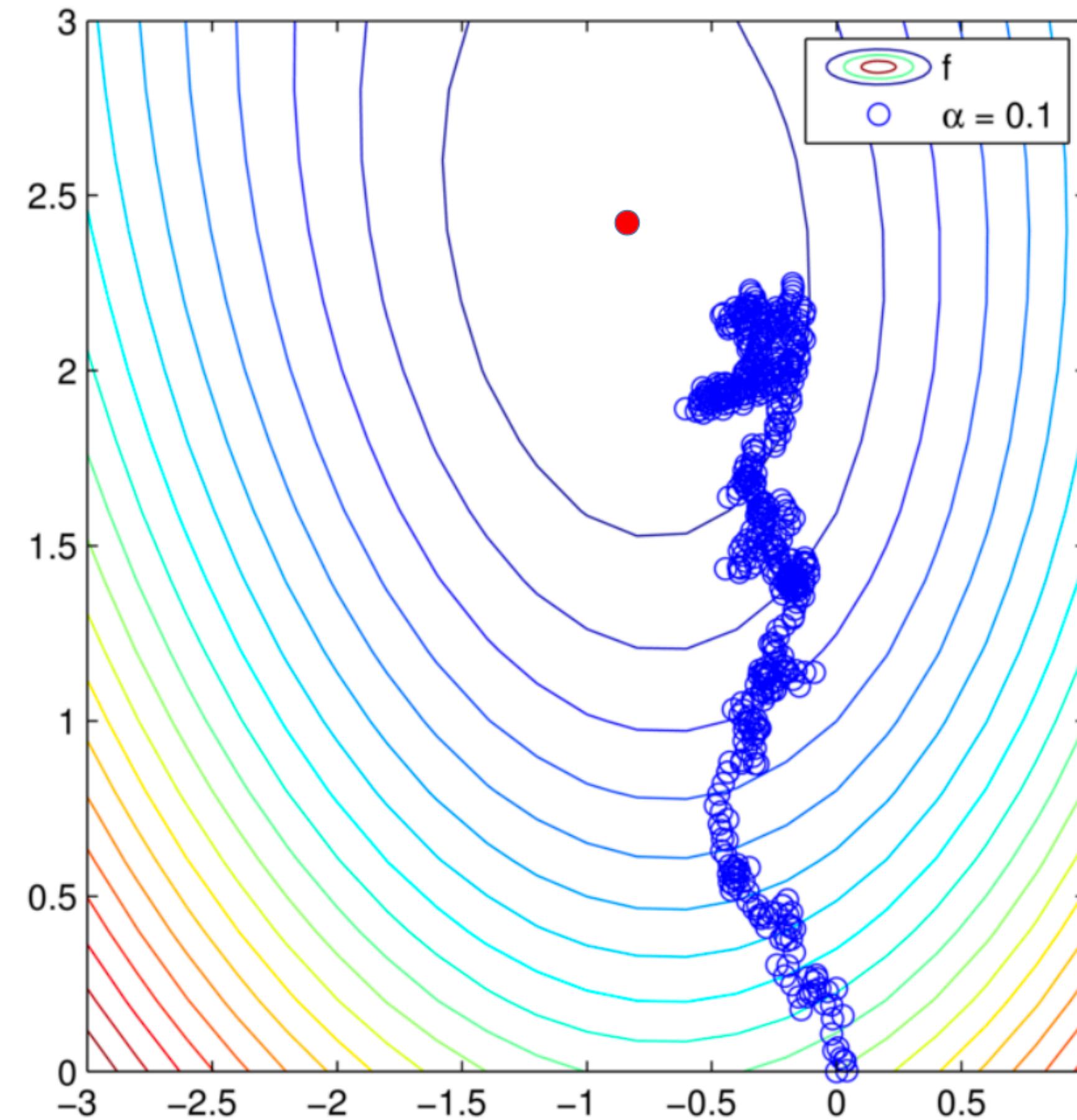
# Choix du learning rate



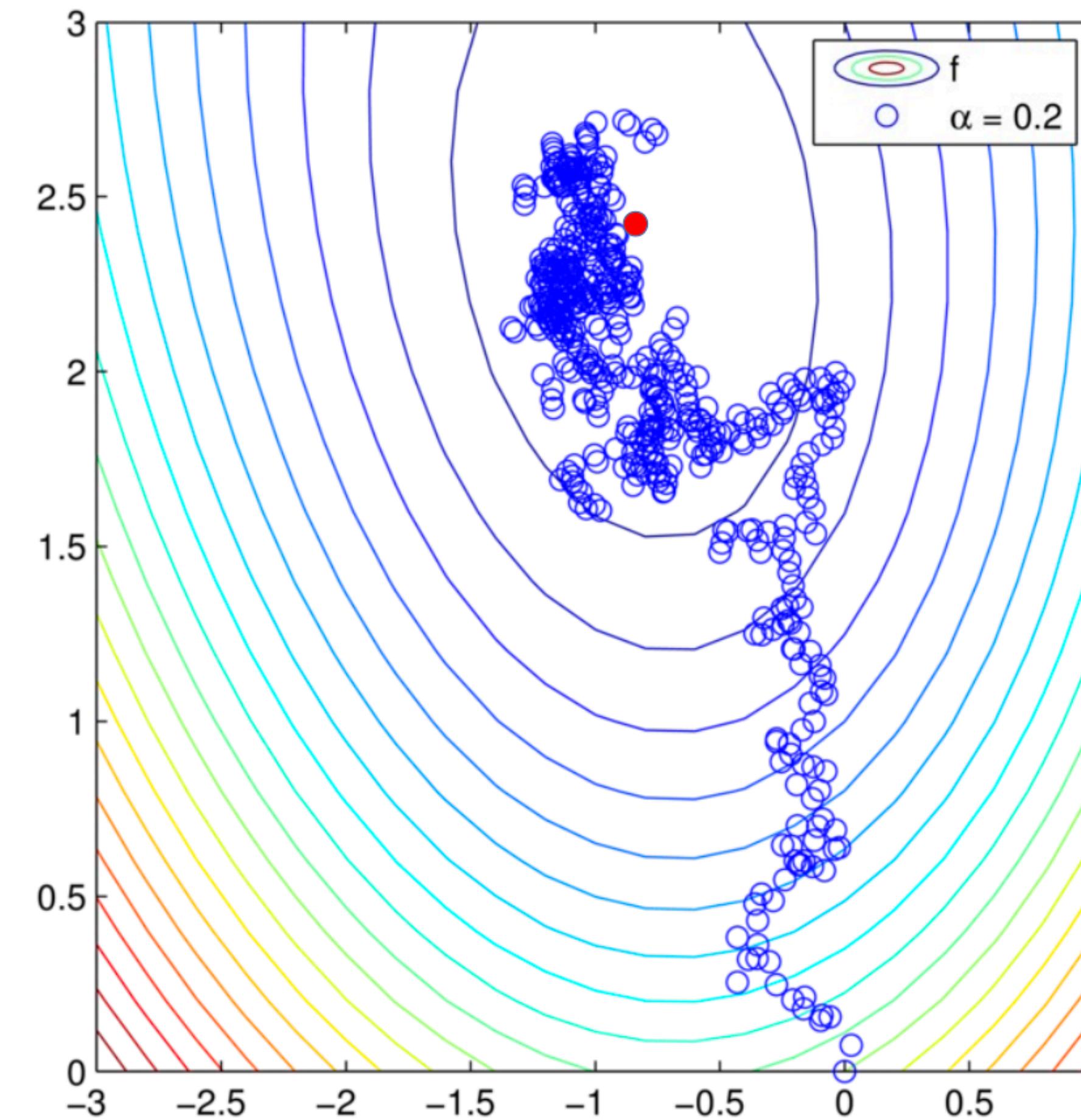
# Choix du learning rate



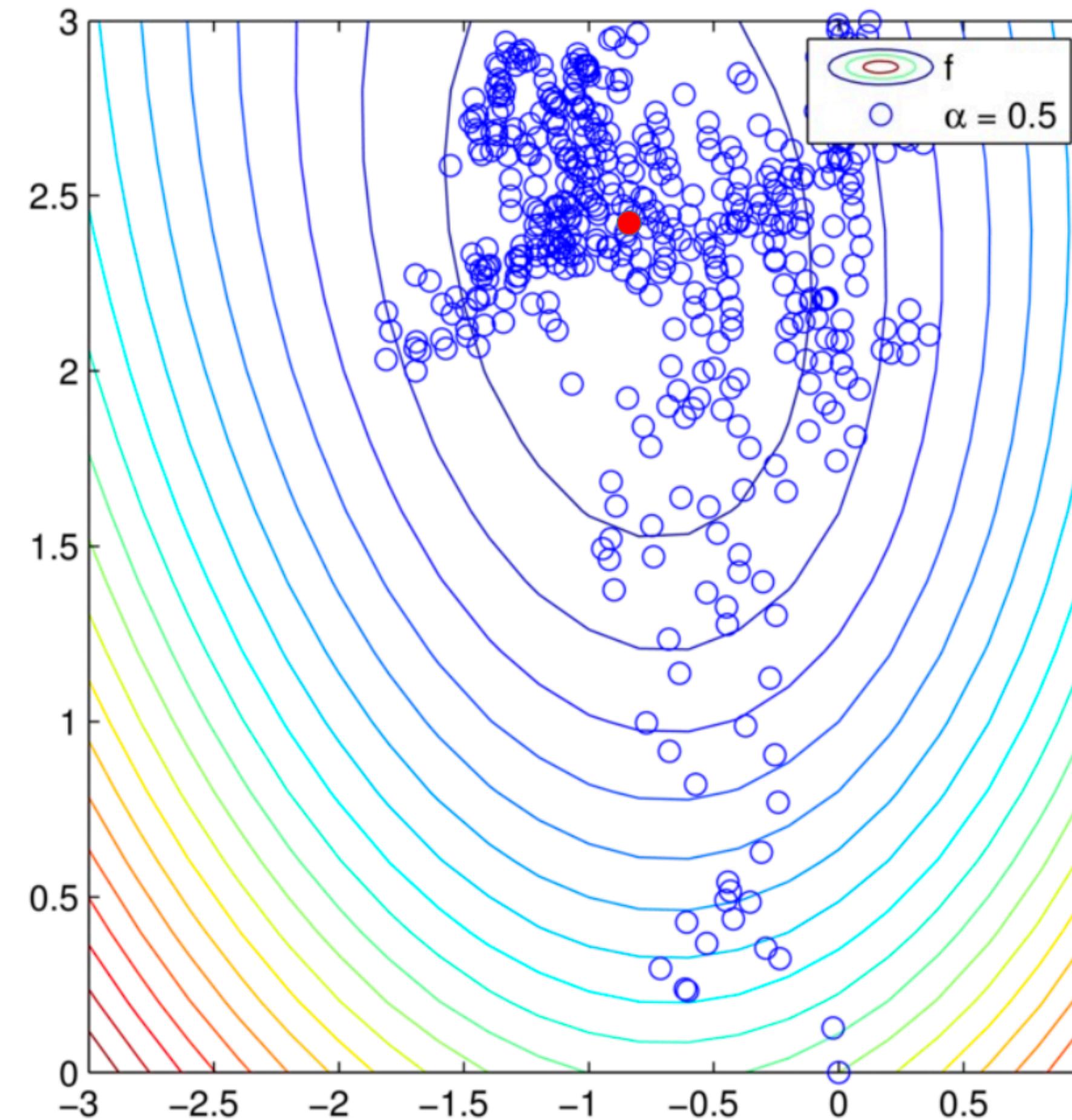
# Choix du learning rate



# Choix du learning rate

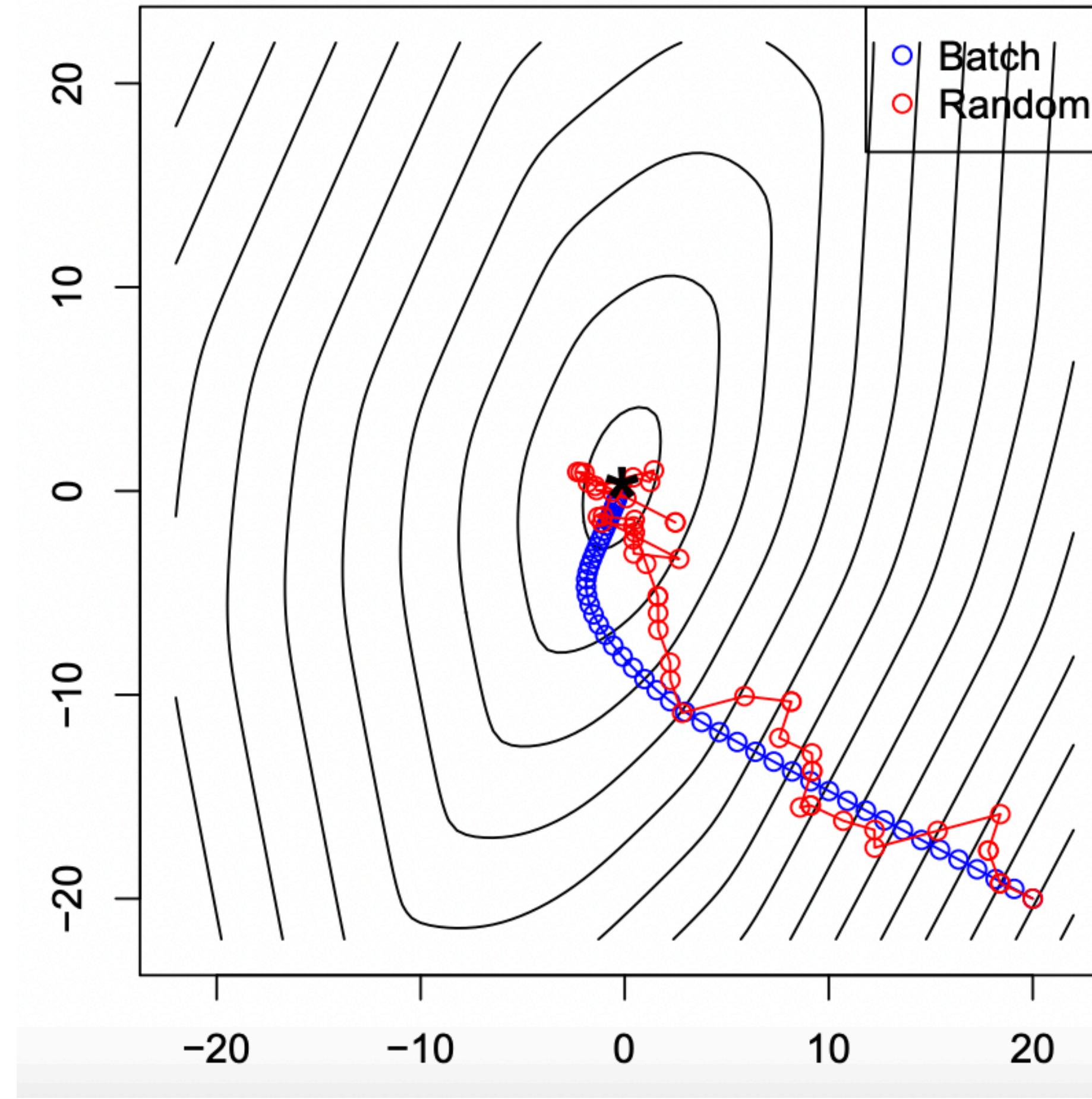


# Choix du learning rate



# Descent de gradient

## Stochastique et classique



La descente de gradient converge beaucoup mieux, en bien moins d'itérations

Chaque itération de la méthode du gradient est beaucoup plus couteuse

Nous sommes gênés par le comportement oscillatoire de la descente de gradient stochastique autour de la solution

# Difficulté de choisir la taille du pas

**Theorem** If  $f$  is  $\mu$  – strongly convex and  $\mathbb{E}[\|\nabla f_i(w)\|^2] \leq B^2$

If  $0 < \alpha \leq \frac{1}{\mu}$  then the iterates of the SGD method satisfy

$$\mathbb{E} [\|w^t - w^*\|_2^2] \leq (1 - \alpha\mu)^t \|w^0 - w^*\|_2^2 + \frac{\alpha}{\mu} B^2$$

Shows that  $\alpha \approx \frac{1}{\mu}$

Shows that  $\alpha \approx 0$

# SGD avec des pas décroissants

Idée: On peut décroître la taille des pas pour diminuer l'effet d'oscillation près de la solution

## SGD 1.0: Descreasing stepsize

Set  $w^0 = 0$

Choose  $\alpha_t > 0$ ,  $\alpha_t \rightarrow 0$ ,  $\sum_{t=0}^{\infty} \alpha_t = \infty$   
for  $t = 0, 1, 2, \dots, T - 1$

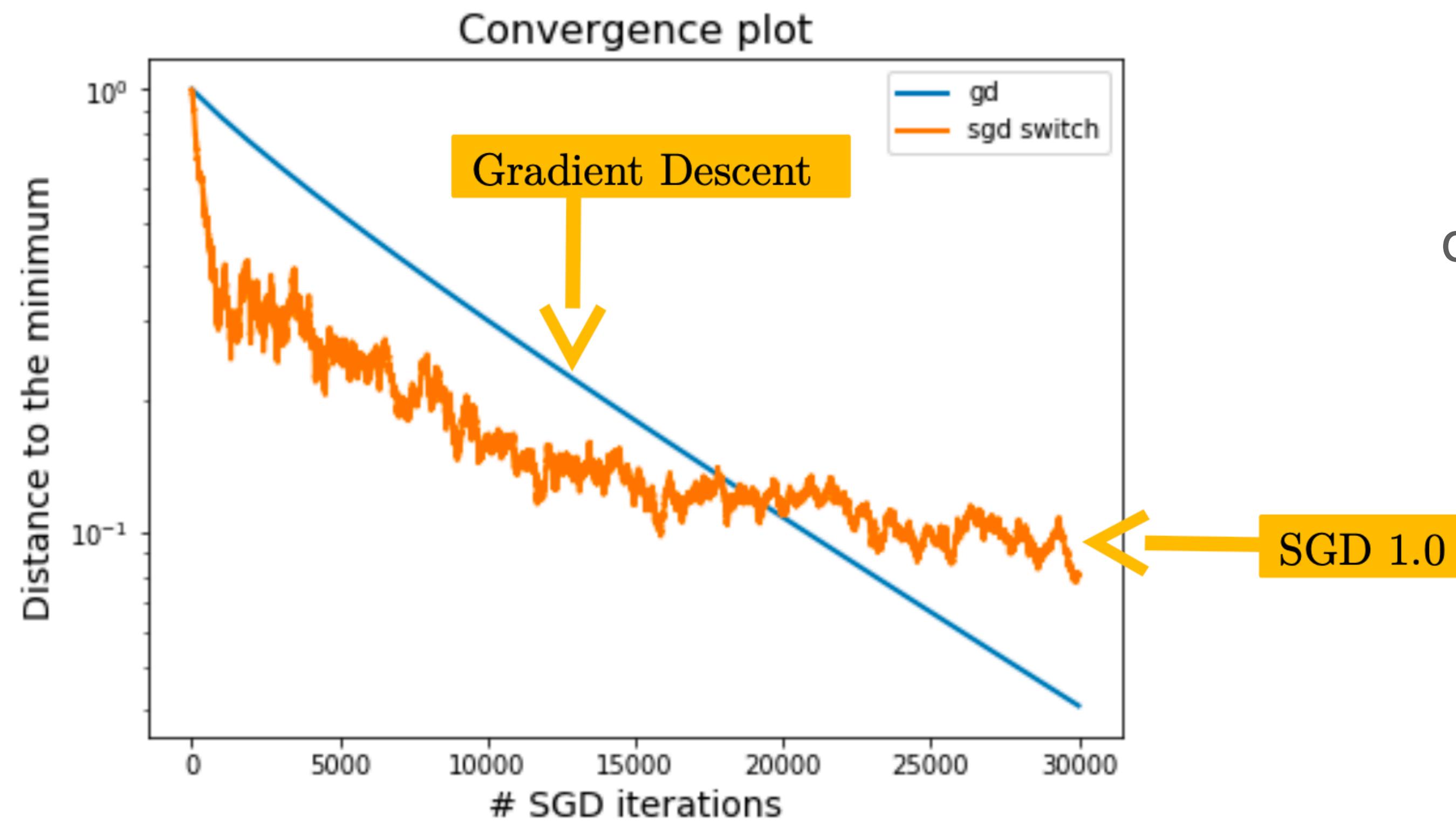
sample  $j \in \{1, \dots, n\}$

$$w^{t+1} = w^t - \alpha_t \nabla f_j(w^t)$$

Output  $w^T$

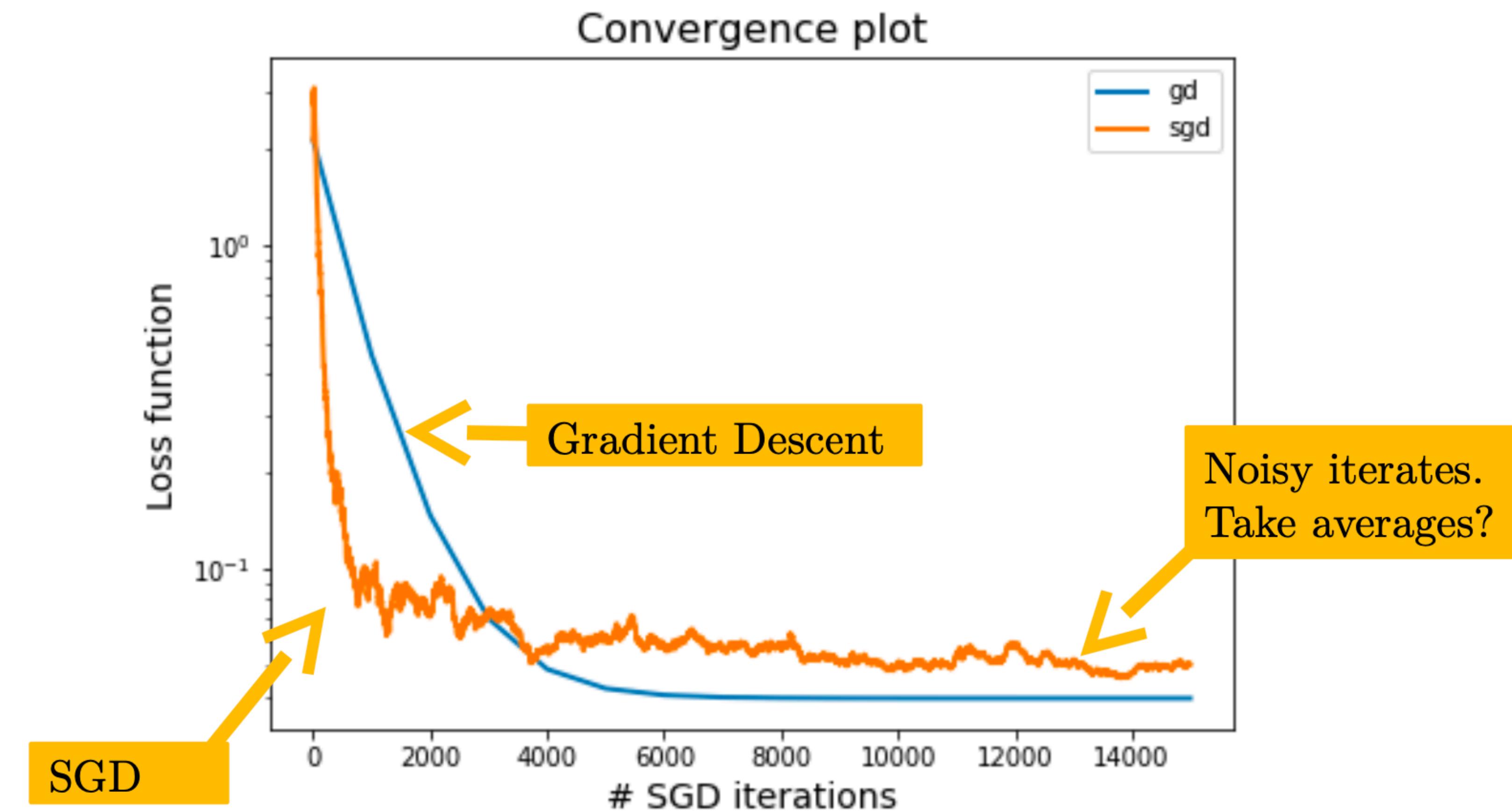
Shrinking  
Stepsize

# SGD avec pas décroissants

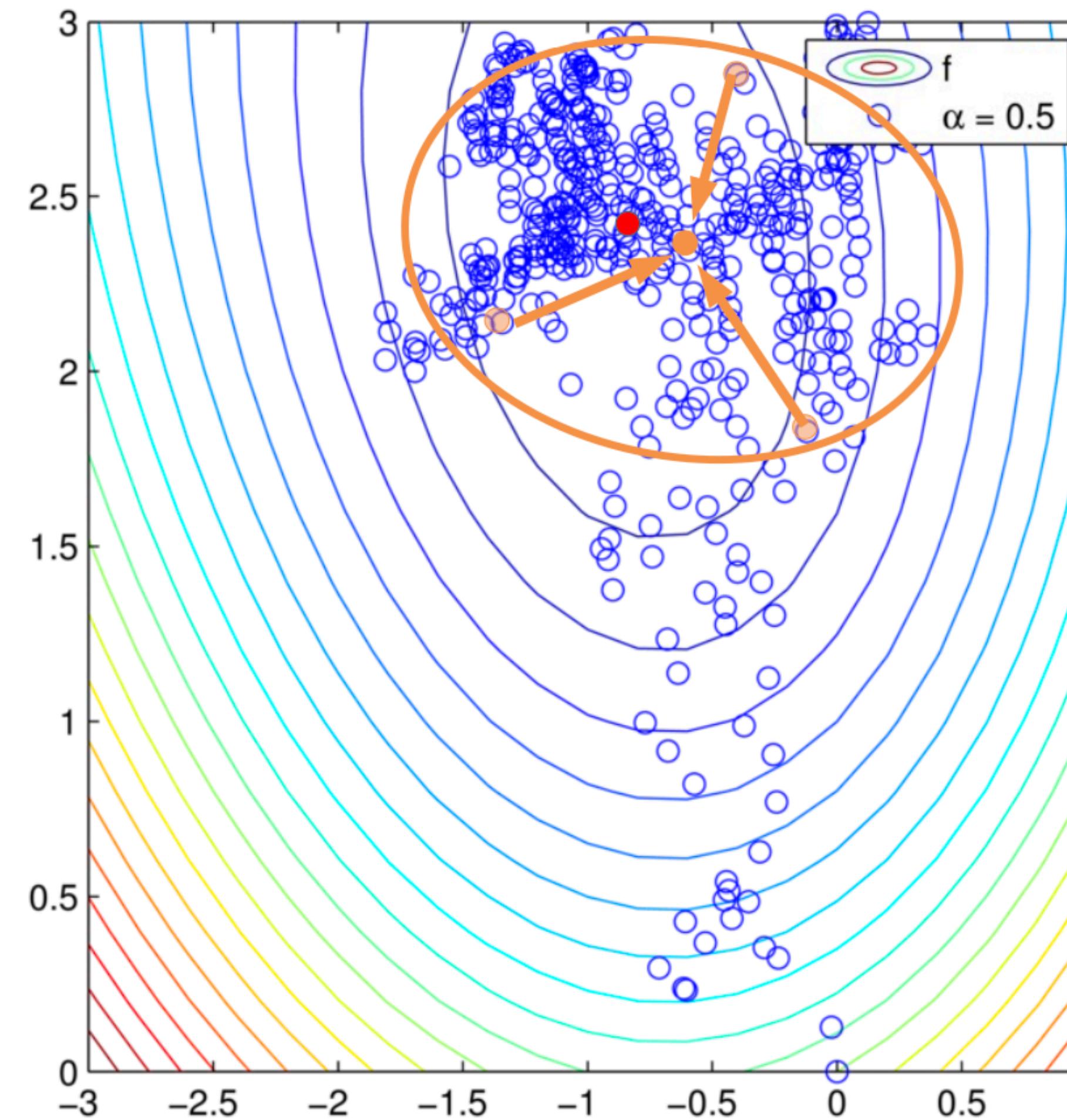


Malheureusement,  
décroître les pas ralenti fort  
SGD

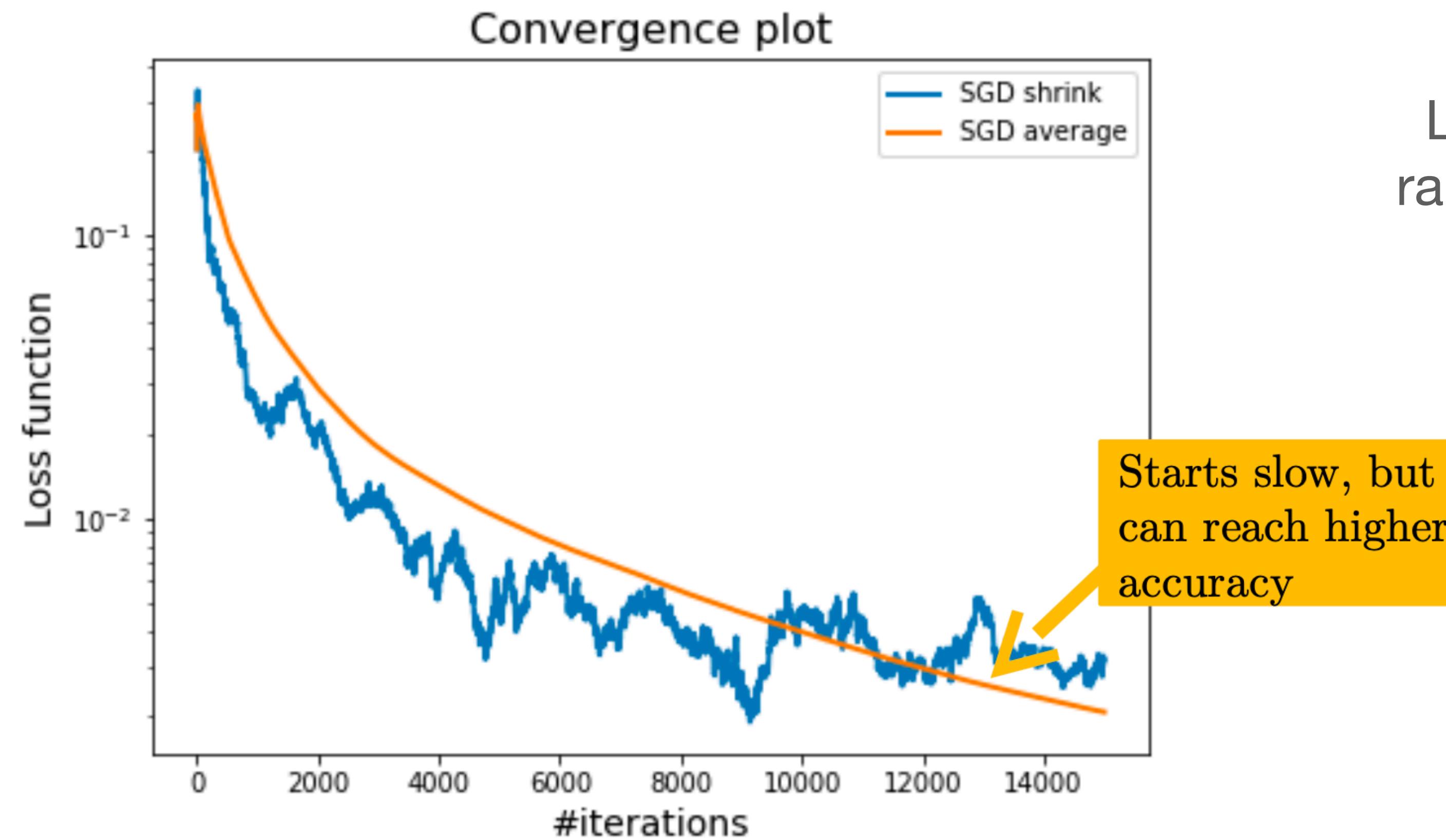
# Comment éviter d'osciller autour de la solution avec SGD ?



# Iterate averaging ?



# SGD avec pas décroissant vs averaging



L'averaging d'itérées n'est pas très rapide, mais il permet d'atteindre une meilleure précision

Idée: faire de l'averaging uniquement à la fin

# SGD avec pas décroissant et late-averaging

## SGDA 1.1

Set  $w^0 = 0$

Choose  $\alpha_t > 0$ ,  $\alpha_t \rightarrow 0$ ,  $\sum_{t=0}^{\infty} \alpha_t = \infty$

Choose averaging start  $s_0 \in \mathbb{N}$

for  $t = 0, 1, 2, \dots, T - 1$

    sample  $j \in \{1, \dots, n\}$

$w^{t+1} = w^t - \alpha_t \nabla f_j(w^t)$

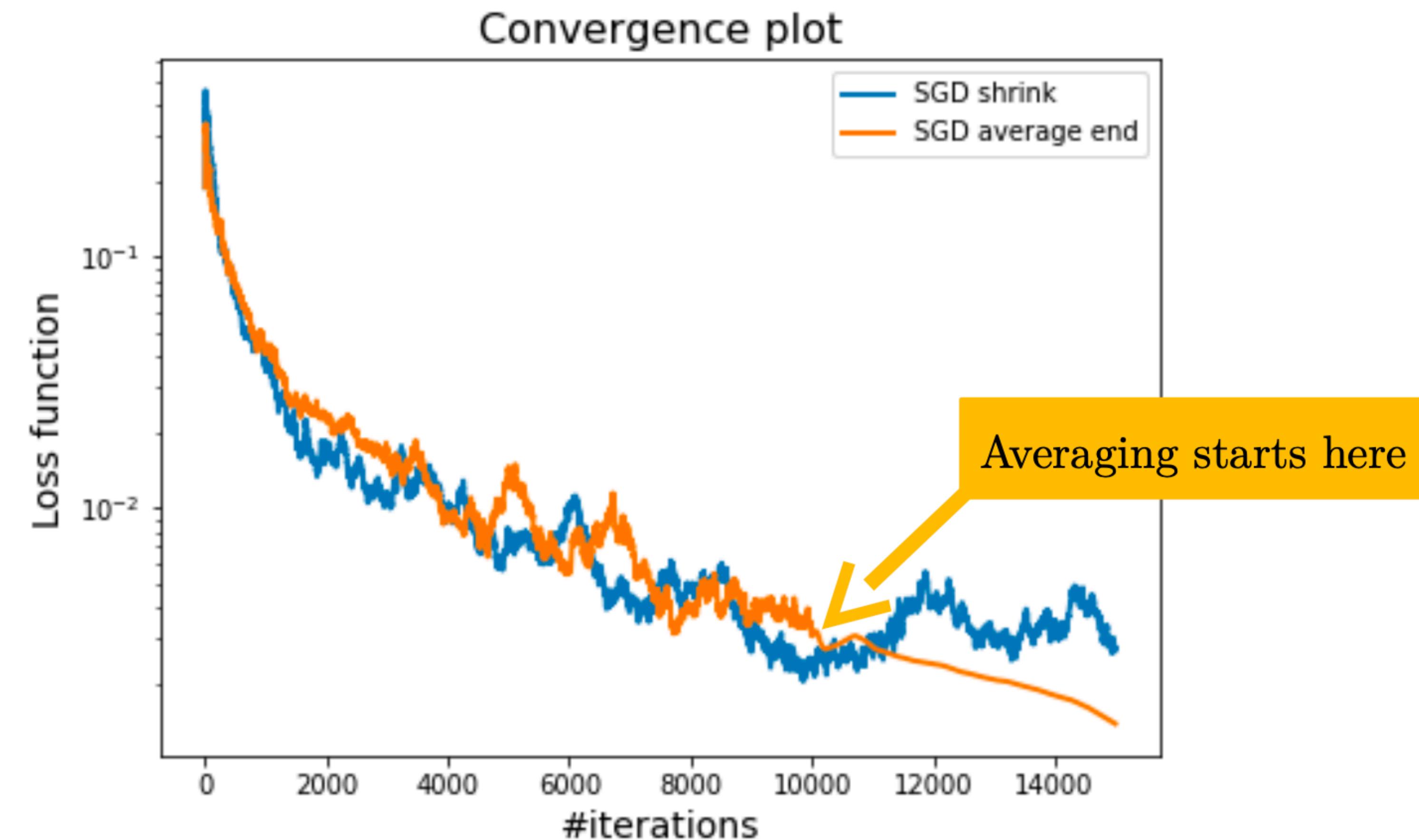
    if  $t > s_0$

$\bar{w} = \frac{1}{t-s_0} \sum_{i=s_0}^t w^i$

    else:  $\bar{w} = w$

Output  $\bar{w}$

# Late-averaging



# Guarantes théoriques

## Pour des fonctions fortement convexes

Combien d'itérations sont nécessaires pour obtenir  $\|x_k - x_*\| \leq \varepsilon$  ?

	SGD	GD
Nombre d'itérations	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(\log\left(\frac{1}{\epsilon}\right)\right)$
Cout d'une itération	$\mathcal{O}(1)$	$\mathcal{O}(d)$
Complexité totale	$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	$\mathcal{O}\left(n \log\left(\frac{1}{\epsilon}\right)\right)$

Interprétation pour petit  $\epsilon$  et grand  $n$

**Pourquoi le gradient stochastic est-il si populaire en machine learning ?**

# Un autre avantage du gradient stochastique

Though we solve:

$$\min_{w \in \mathbf{R}^d} \frac{1}{n} \sum_{i=1}^n \ell(h_w(x^i), y^i) + \lambda R(w)$$

We want to solve:

**The statistical learning problem:**

Minimize the expected loss over an *unknown* expectation

$$\min_{w \in \mathbf{R}^d} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(h_w(x), y)]$$

SGD can solve the  
statistical learning problem!

# Training set vs test set

Soit  $\mathcal{D}$ , la "vraie" distribution des données. C'est à dire, toutes les images de feu de circulation qui existent

Nous avons accès à un nombre fini de réalisations/samples de cette distribution, que nous appelons le Training set :

$(a_1, b_1), \dots, (a_n, b_n)$  = notre collection d'image et les labels

Le learning c'est n'est pas juste un problème d'optimisation, c'est étudier la généralisation du modèle obtenu

Risk attendu (Expected risk)

$$R(x) = \mathbb{E}_{(a,b) \in \mathcal{D}}[f(x; a, b)]$$

Risk Empirique (Empirical risk)

$$R_n(x) = \frac{1}{n} \sum_{i=1}^n \ell(x; a_i, b_i) = \frac{1}{n} \sum_{i=1}^n f_i(x)$$

# SGD for expected risk minimisation

**The statistical learning problem:**

Minimize the expected loss over an *unknown* expectation

$$\min_{w \in \mathbf{R}^d} \mathbb{E}_{(x,y) \sim \mathcal{D}} [\ell(h_w(x), y)]$$

**SGD  $\infty.0$  for learning**

Set  $w^0 = 0, \alpha > 0$

for  $t = 0, 1, 2, \dots, T - 1$

sample  $(x, y) \sim \mathcal{D}$

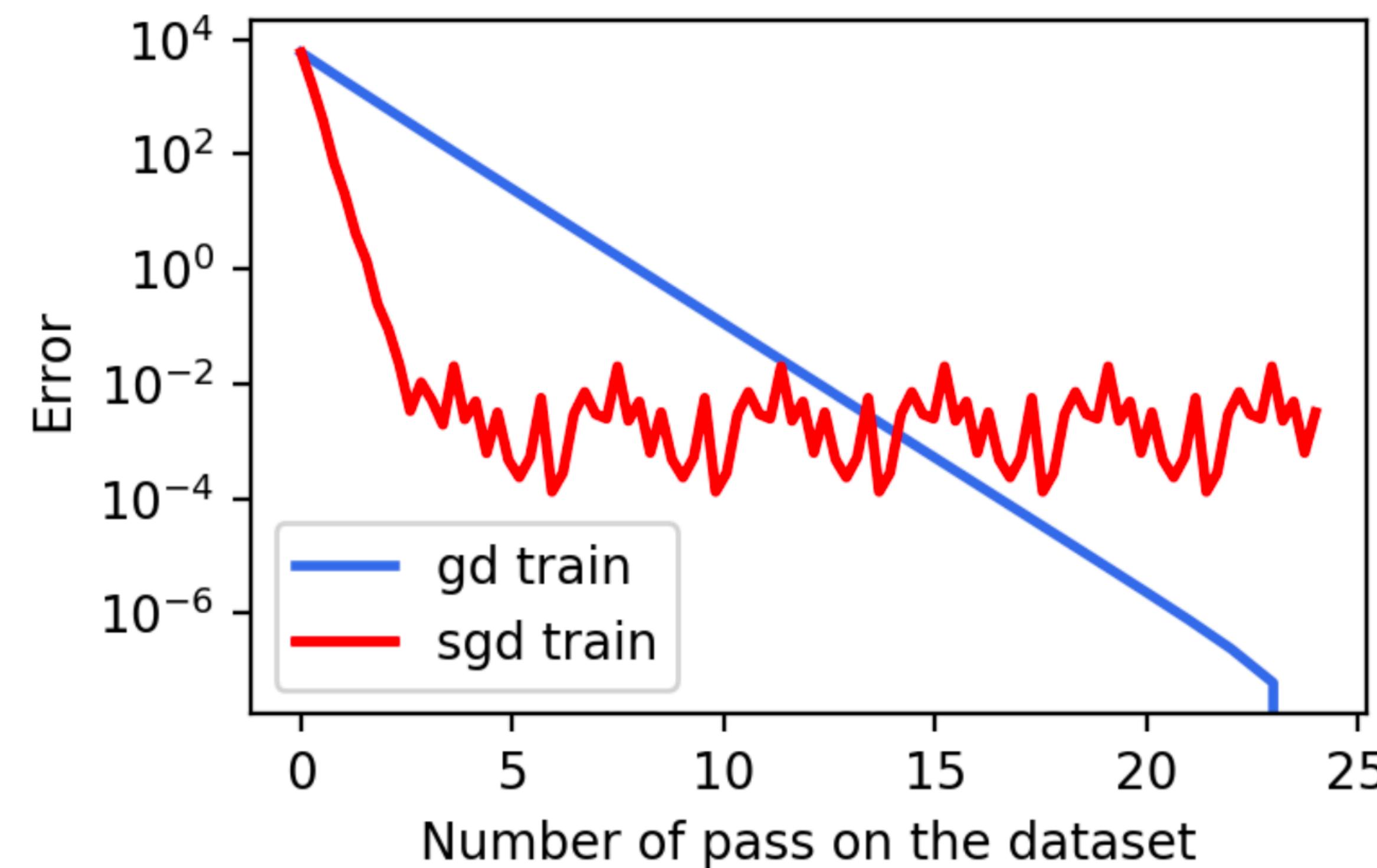
calculate  $v_t \in \partial \ell(h_{w^t}(x), y)$

$$w^{t+1} = w^t - \alpha v_t$$

$$\text{Output } \bar{w}^T = \frac{1}{T} \sum_{t=1}^T w^t$$

La méthode de descente de gradient classique ne peut pas être appliquée à la minimisation de l'expected risk. On ne peut pas calculer le gradient de toute une distribution, uniquement une réalisation à la fois

# Erreur sur le training set



# Erreur sur le Test Set !

## (Un autre avantage du gradient stochastique)



SGD est plus efficace pour réduire l'erreur sur le test set en pratique

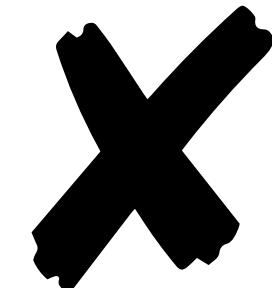
# Complexité pour l'expected risk

SGD garde une complexité théorique de  $1/\epsilon$  pour le expected risk, alors que GD ne peut pas minimiser l'expected risk

Total complexity pour  $\mathbb{E}[R_n(x_k) - R_n(x^*)] \leq \epsilon$

<p>SGD</p> $\mathcal{O}\left(\frac{1}{\epsilon}\right)$	<p>GD</p> $\mathcal{O}\left(n \log\left(\frac{1}{\epsilon}\right)\right)$
---	---

Total complexity pour  $\mathbb{E}[R(x_k) - R(x^*)] \leq \epsilon$

$\mathcal{O}\left(\frac{1}{\epsilon}\right)$	
--	---

# Methodes en batch

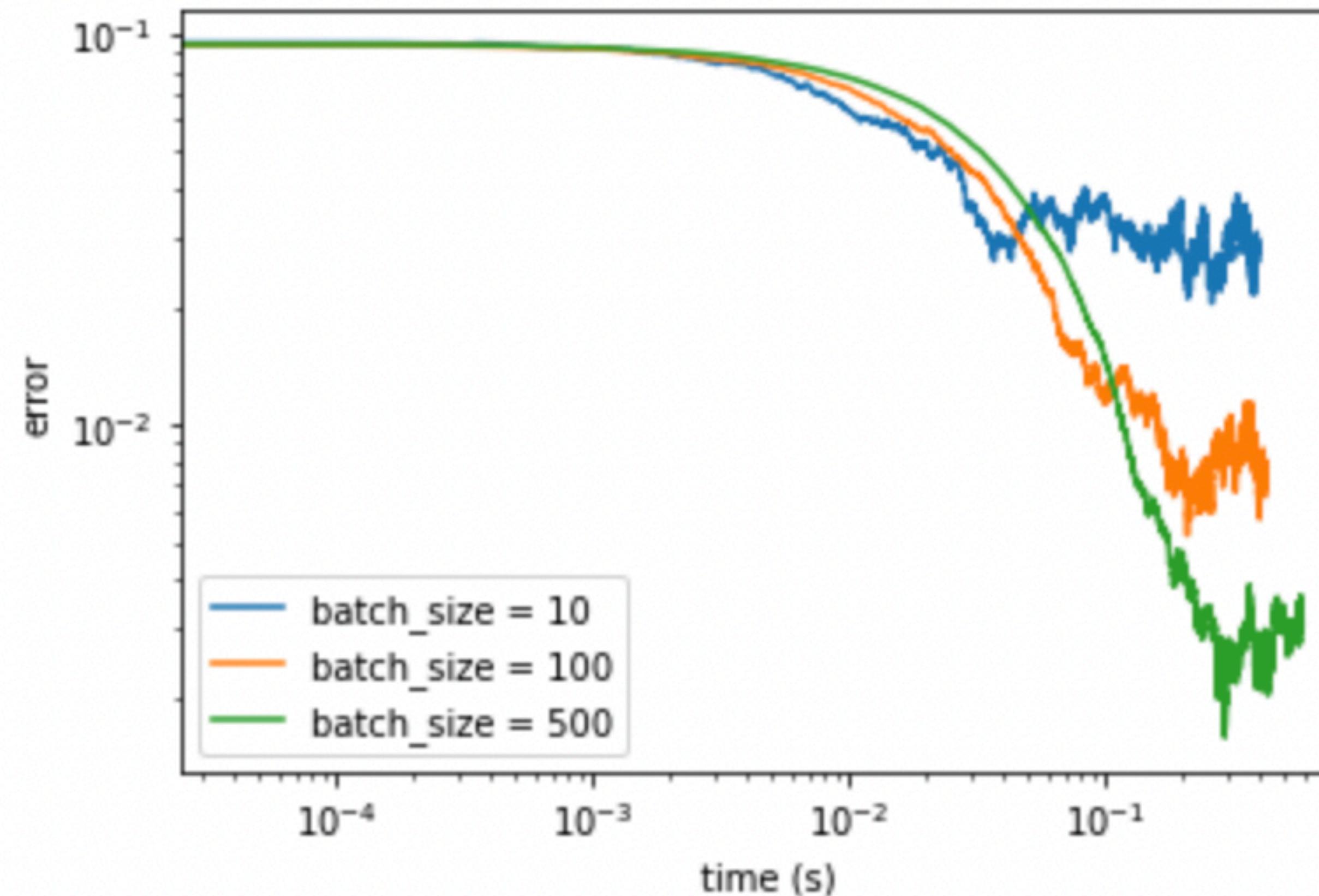
Idée: On peut bénéficier d'opérations en parallèle sur un ordinateur avec plusieurs coeurs pour calculer plus de 1 gradient à la fois

A chaque itération, générer aléatoirement un ensemble d'indices  $\mathcal{S}_k \subset \{1, \dots, n\}$

$$g(w_k, \xi_k) = \frac{1}{|\mathcal{S}_k|} \sum_{i \in \mathcal{S}_k} \nabla f_i(w_k).$$

$$w_{k+1} = w_k - \alpha g(w_k, \xi_k)$$

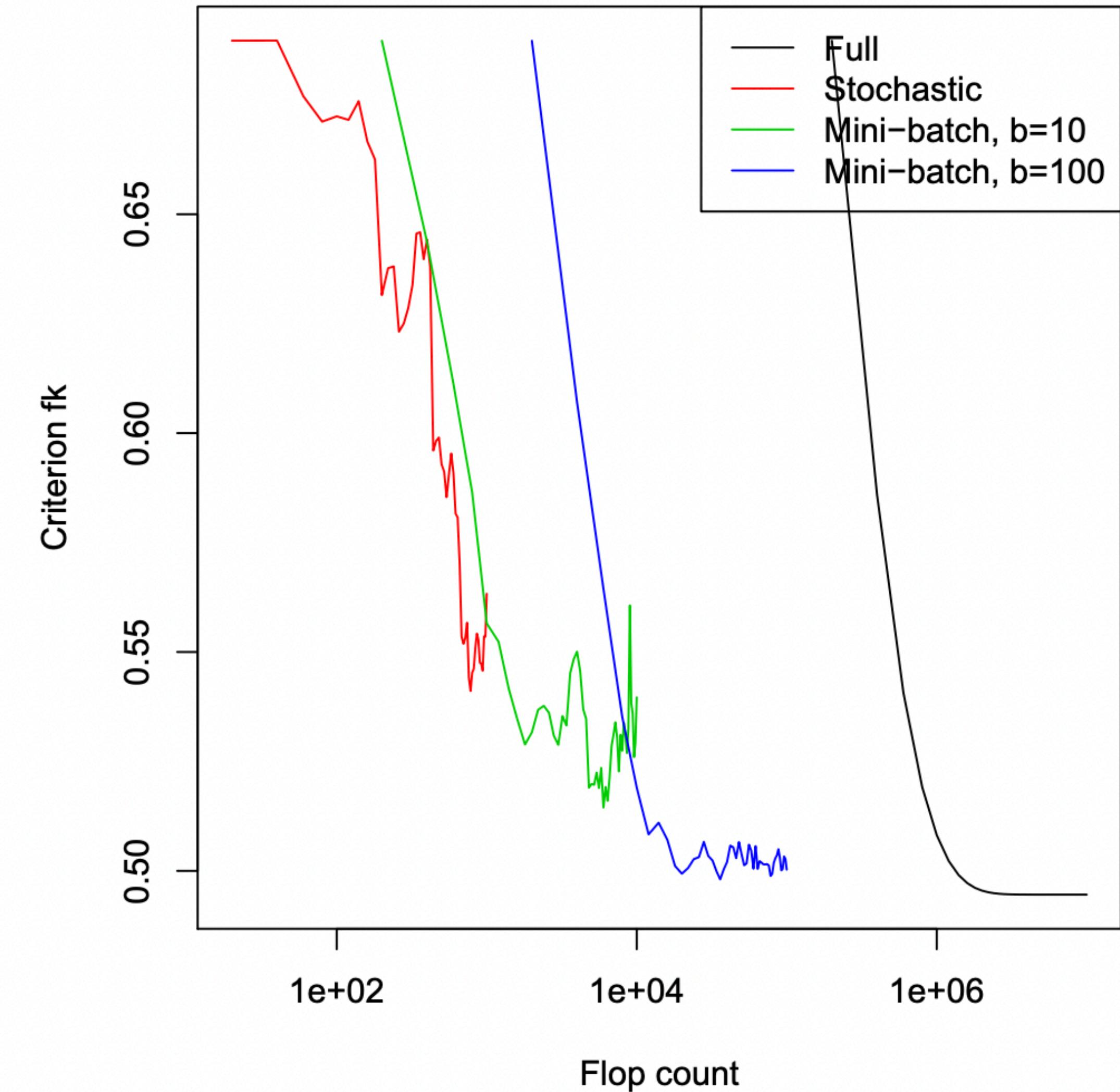
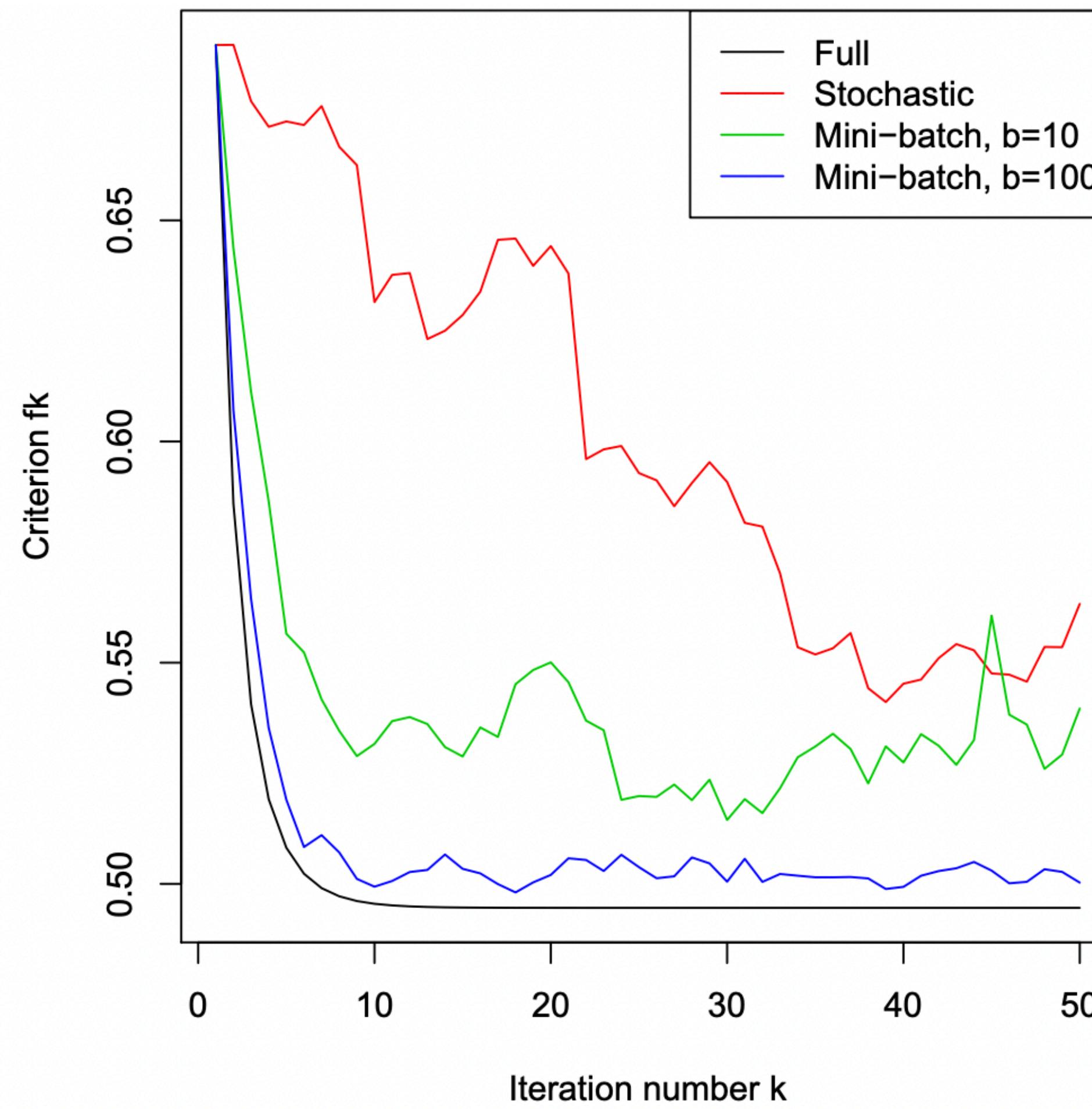
# Batch sizes



L'utilisation d'un batch plus grand permet de diminuer la variance et converger avec plus de précision

La performance des méthodes en Batch est directement liée à l'architecture de l'ordinateur (CPU vs GPU, calcul en parallèle etc.)

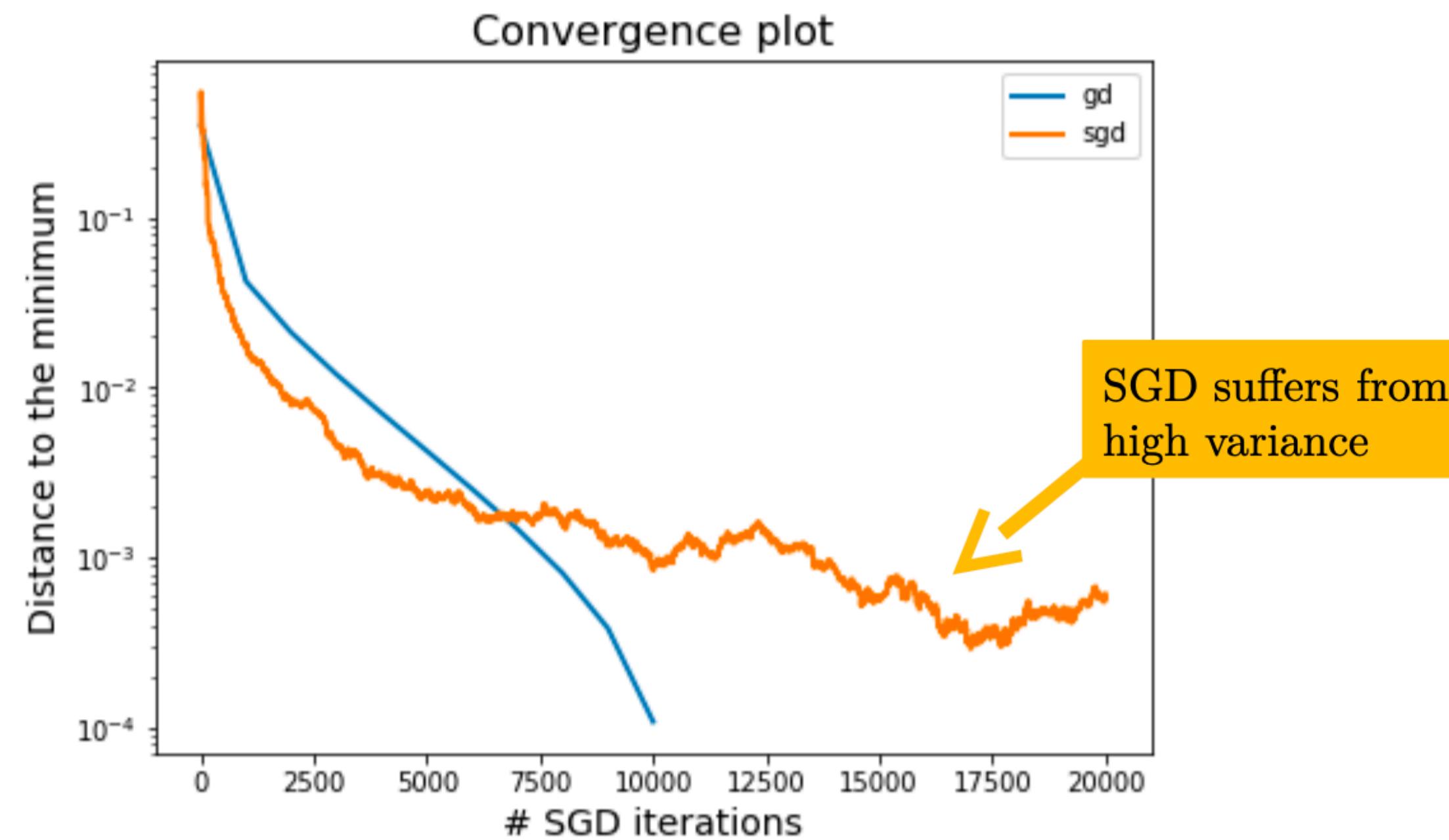
# Batch methods vs GD vs SGD



Plus la taille du batch est grande, plus les itérations sont couteuses et efficaces !

# Noise reduction method

SGD souffre de la variance des estimations des gradients

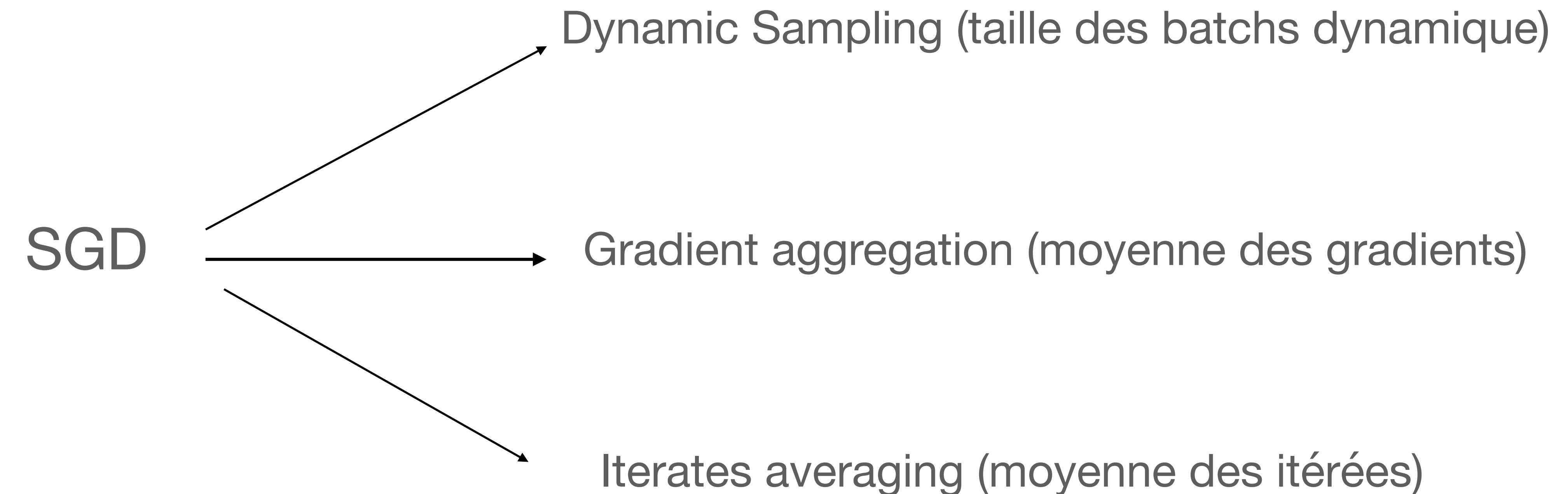


SGD avec des pas constants oscille autour de la solution

SGD avec des pas décroissants converge vers la solution mais est bien plus lent

Comment peut-on réduire la variance/ le bruit introduit par la stochasticité de l'algorithme ?

# Méthodes de reduction de bruit



# Dynamic sampling method

Idée: augmenter progressivement la taille des batchs pour construire une bonne estimation du gradient

$$g(x_k) = \frac{1}{n_k} \sum_{i \in S_k} \nabla f_i(x_k)$$

avec  $n_k = |S_k| = \lceil 1.5^k \rceil$

Progression géométrique de la taille des batchs

$$x_{k+1} = x_k - \alpha g(x_k)$$

Fonctionne avec une taille de pas constants  
(plus rapide que décroissant)

$$\mathbb{E}[F(x_k) - F(x_*)] \leq \epsilon$$

pour  $k = \mathcal{O}(\log(1/\epsilon))$



Coup par itération plus élevé que SGD, trade-off optimal selon l'architecture de calcul

# Gradient aggregation

## Moyenne des gradients

A la place d'utiliser directement  $\nabla f_i(x_k) \approx \nabla F(x_k)$ ,

utiliser  $\nabla f_i(x_k)$  pour update une estimation du gradient  $g_k \approx \nabla F(x_k)$

$$x_{k+1} = x_k - \alpha g_k$$

On voudrait satisfaire:  $\mathbb{E}[g_k] = \nabla F(x_k)$  (estimation non biaisée)

$$\mathbb{E}\|g_k\| \rightarrow 0 \text{ quand } x_k \rightarrow x_*$$

# SVRG

## (Stochastic Variance Reduced Gradient method)

$$w^{t+1} = w^t - \gamma g_{\textcolor{red}{i}}(w^t)$$

Reference point

$$\tilde{w} \in \mathbb{R}^d$$

Sample

$$\nabla f_{\textcolor{red}{i}}(w^t), \quad \text{i.i.d sample with prob } \frac{1}{n}$$

Grad. estimate

$$g_{\textcolor{red}{i}}(w^t) = \nabla f_{\textcolor{red}{i}}(w^t) - \nabla f_{\textcolor{red}{i}}(\tilde{w}) + \nabla f(\tilde{w})$$

It's unbiased  
because:

$$\begin{aligned}\mathbb{E}[g_{\textcolor{red}{i}}(w)] &= \mathbb{E}[\nabla f_{\textcolor{red}{i}}(w)] - \mathbb{E}[\nabla f_{\textcolor{red}{i}}(\tilde{w})] + \nabla f(\tilde{w}) \\ &= \nabla f(w) - \cancel{\nabla f(\tilde{w})} + \cancel{\nabla f(\tilde{w})}\end{aligned}$$

Un point de référence  $\tilde{w}$  est mis à jour occasionnellement

Nécessite de calculer occasionnellement le  
gradient complet

$\nabla F(\tilde{w})$  au point de référence

# SAGA (Stochastic Average of gradients)

$$w^{t+1} = w^t - \gamma g_{\textcolor{red}{i}}(w^t)$$

Sample

$\nabla f_{\textcolor{red}{i}}(w^t)$ , i.i.d sample with prob  $\frac{1}{n}$

Grad. estimate

$$g_{\textcolor{red}{i}}(w^t) = \nabla f_{\textcolor{red}{i}}(w^t) - \nabla f_{\textcolor{red}{i}}(w^{t_{\textcolor{red}{i}}}) + \frac{1}{n} \sum_{j=1}^n \nabla f_j(w^{t_j})$$

$$z_{\textcolor{red}{i}}(w) = f_{\textcolor{red}{i}}(w^{t_{\textcolor{red}{i}}}) + \langle \nabla f_{\textcolor{red}{i}}(w^{t_{\textcolor{red}{i}}}), w - w^{t_{\textcolor{red}{i}}} \rangle$$

$$\nabla z_{\textcolor{red}{i}}(w^t) = \nabla f_{\textcolor{red}{i}}(w^{t_i}) \quad \mathbb{E}[\nabla z_{\textcolor{red}{i}}(w^t)]$$

Store grad.

$$\nabla f_i(w^{t_{\textcolor{red}{i}}}) = \nabla f_i(w^t)$$

Nécessite de garder en mémoire une matrice avec les gradients précédents

# SAGA

## Memorise les gradients de chaque fonction

Set  $w^0 = 0, g_i = \nabla f_i(w^0)$ , for  $i = 1 \dots, n$

Choose  $\gamma > 0$

for  $t = 0, 1, 2, \dots, T - 1$

sample  $i \in \{1, \dots, n\}$

$$g^t = \nabla f_i(w^t) - g_i + \frac{1}{n} \sum_{j=1}^n g_j$$

$$w^{t+1} = w^t - \gamma g^t$$

$$g_i = \nabla f_i(w^t)$$

Output  $w^T$

# Hypothèses de convergence

**Strong Convexity**

$$f(w) \geq f(y) + \langle \nabla f(y), w - y \rangle + \frac{\mu}{2} \|w - y\|_2^2$$

**Smoothness + convexity**

$$f_i(w) \leq f_i(y) + \langle \nabla f_i(y), w - y \rangle + \frac{L_i}{2} \|w - y\|_2^2$$

$$f_i(w) \geq f_i(y) + \langle \nabla f_i(y), w - y \rangle \quad \text{for } i = 1, \dots, n$$

$$L_{\max} := \max_{i=1, \dots, n} L_i$$

# Convergence results for gradient aggregation methods

**Approximate solution**

$$\mathbb{E}[f(w^T)] - f(w^*) \leq \epsilon \quad \text{or} \quad \mathbb{E}\|w^t - w^*\|^2 \leq \epsilon$$

**SGD**

$$O\left(\frac{1}{\epsilon}\right)$$

**Gradient descent**

$$O\left(\frac{nL}{\mu} \log\left(\frac{1}{\epsilon}\right)\right)$$

**SVRG/SAGA/SAG**

$$O\left(\left(n + \frac{L_{\max}}{\mu}\right) \log\left(\frac{1}{\epsilon}\right)\right)$$

Variance reduction faster than GD when

$$L \geq \mu + L_{\max}/n$$

# Variance reduction methods

## (Conclusion)

Elles utilisent l'évaluation d'un seul gradient par itération et convergent en  $\log(1/\epsilon)$  pour des fonctions fortement convexes

Elles utilisent un pas de taille constante

SVRG ne doit stocker qu'un seul gradient, mais occasionnellement doit calculer tous les gradients  $\nabla F(x_k)$

SAGA a pour seul défaut qu'elle nécessite de conserver des gradients en mémoire

# Iterates averaging methods

Sequence classique :

$$x_{k+1} = x_k - \alpha g(x_k)$$

Sequence moyenne :

$$\tilde{x}_{k+1} = \frac{1}{k} \sum_{j=1}^k x_j$$

**Questions ?**