

Notebook on gradient averaging for variance reduction

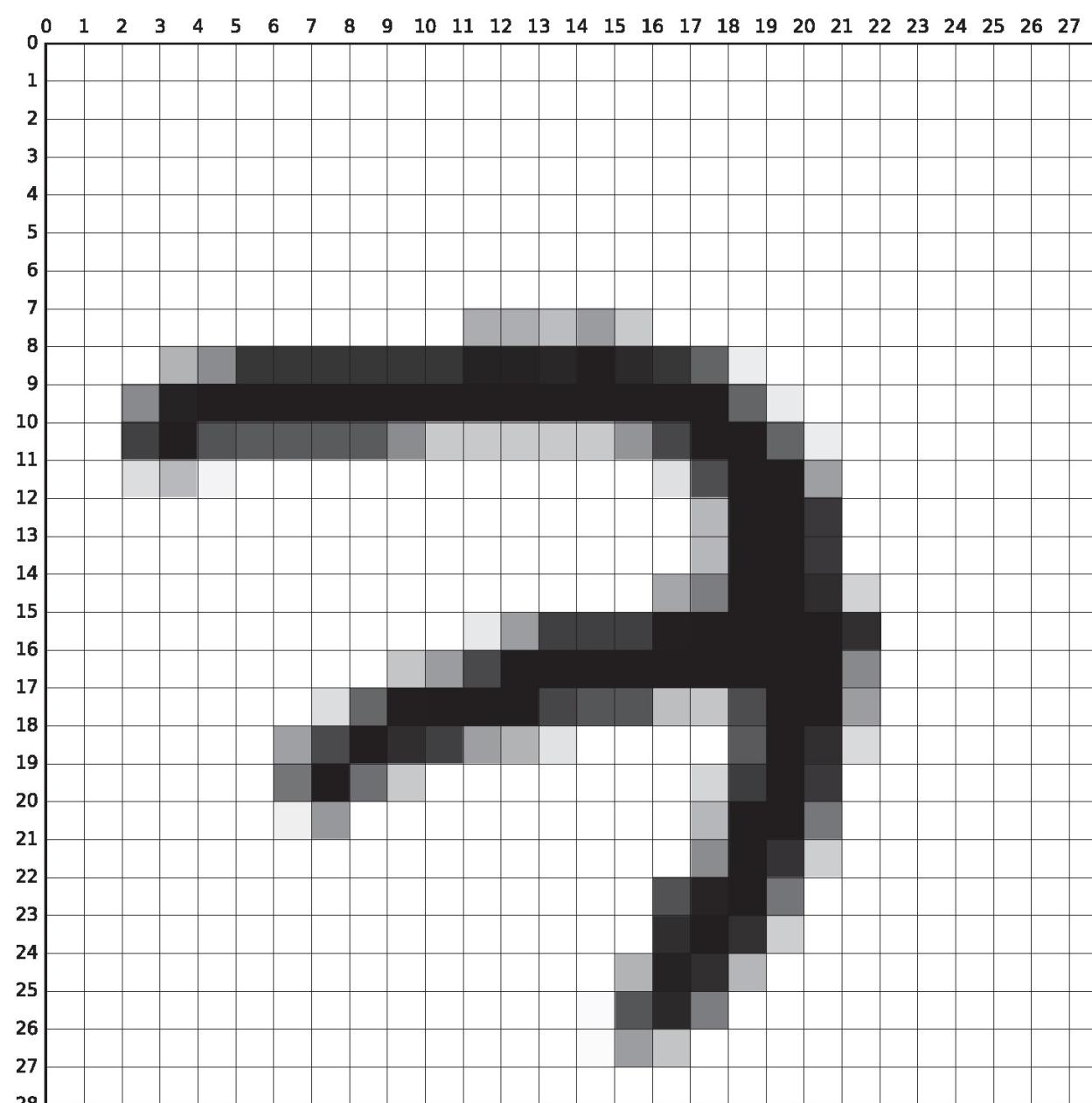
Part 3: SVRG and SAGA

Large scale optimisation

Partie 5/5

MNIST data set

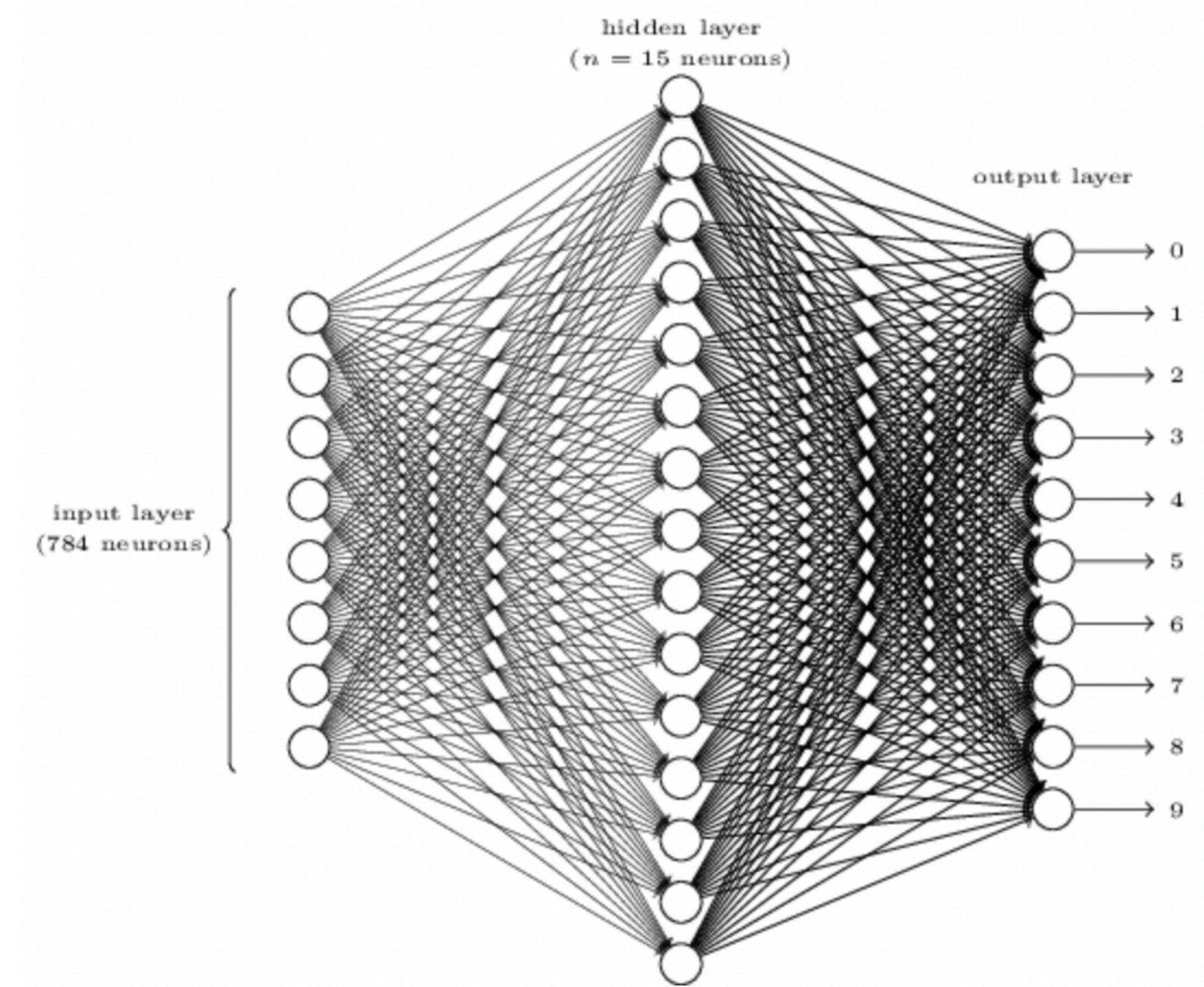
Benchmark for classification algorithms based on neural networks



(a) MNIST sample belonging to the digit '7'.



(b) 100 samples from the MNIST training set.



Méthodes stochastiques pour le deep learning

On veut un modèle qui fit bien les données : $h_x(a_i) \approx b_i$ pour $i = 1, \dots, n$

Minimisation du risque empirique

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

Méthodes stochastiques : on ne calcule pas tous les gradients à la fois !

SGD avec momentum

Choisir un indice i au hasard : $g_k = \nabla f_i(x_k)$

Update avec momentum : $\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha(1 - \beta)\mathbf{g}_k + \alpha\beta (\mathbf{w}_k - \mathbf{w}_{k-1}), \beta \in [0,1]$

Combinaison entre la direction du gradient stochastique et une inertie

Très populaire en pratique pour l'entraînement de réseaux de neurones

AdaGrad

Adaptative Gradient algorithm

Diagonal scaling du gradient : Utilise un pas différent pour chaque coordonnée

$$\mathbf{w}_{k+1} = \mathbf{w}_k - \alpha \mathbf{g}_k \oslash \sqrt{\mathbf{r}_k},$$

Scaling diagonal qui dépend de la magnitude de chaque coordonnée aux itérations précédentes :

$$\forall i = 1, \dots, d, \quad \begin{cases} [\mathbf{r}_{-1}]_i = 0 \\ [\mathbf{r}_k]_i = [\mathbf{r}_{k-1}]_i + [\mathbf{g}_k]_i^2 \quad \forall k \geq 0, \end{cases}$$

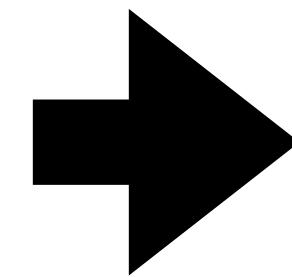
$$\text{Séquence de pas} = \left\{ \left[\frac{\alpha}{\sqrt{[\mathbf{r}_k]_i}} \right]_{i=1}^d \right\}_k.$$

S'adapte à la géométrie locale et enlève des problèmes liés au choix du learning rate
(adaptative method)

RMSProp

Root mean square Propagation

Meme principe de diagonal rescaling
mais on donne plus de poids au dernier
gradient grace au paramètre λ



Bien pour des problèmes non stationnaires

$$\forall i = 1, \dots, d, \quad \begin{cases} [\mathbf{r}_{-1}]_i = 0 \\ [\mathbf{r}_k]_i = (1 - \lambda)[\mathbf{r}_{k-1}]_i + \lambda[\mathbf{g}_k]_i^2 \end{cases} \quad \forall k \geq 0, \quad \lambda \in [0,1]$$

Cette idée permet d'éviter que les stepsizes tendent vers 0 trop rapidement

Très efficace en pratique pour entraîner des réseaux de neurones très profonds

Adam

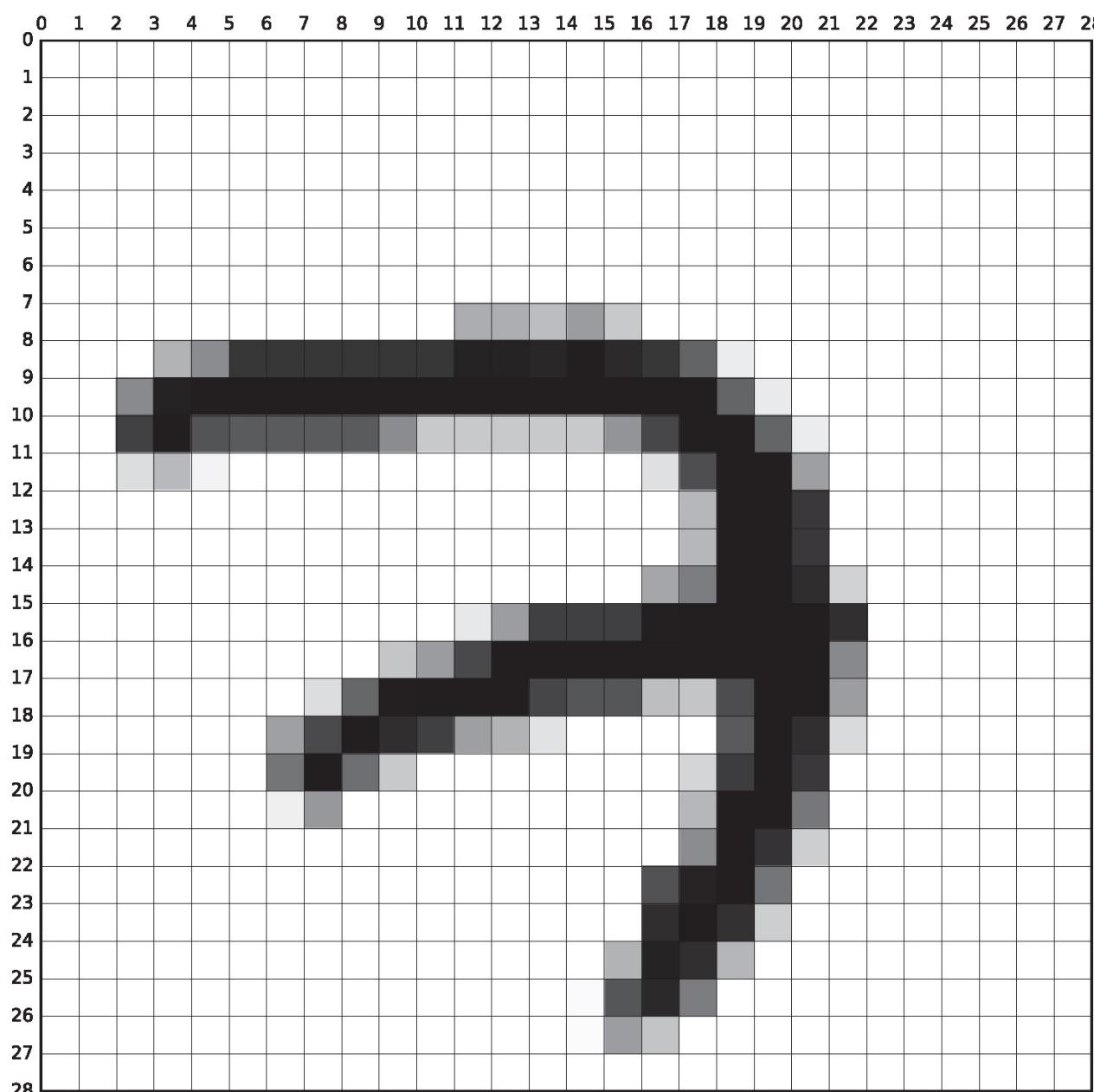
Diagonal scaling of gradients + momentum

```
while  $\theta_t$  not converged do
     $t \leftarrow t + 1$ 
     $g_t \leftarrow \nabla_{\theta} f_t(\theta_{t-1})$  (Get gradients w.r.t. stochastic objective at timestep  $t$ )
     $m_t \leftarrow \beta_1 \cdot m_{t-1} + (1 - \beta_1) \cdot g_t$  (Update biased first moment estimate)
     $v_t \leftarrow \beta_2 \cdot v_{t-1} + (1 - \beta_2) \cdot g_t^2$  (Update biased second raw moment estimate)
     $\hat{m}_t \leftarrow m_t / (1 - \beta_1^t)$  (Compute bias-corrected first moment estimate)
     $\hat{v}_t \leftarrow v_t / (1 - \beta_2^t)$  (Compute bias-corrected second raw moment estimate)
     $\theta_t \leftarrow \theta_{t-1} - \alpha \cdot \hat{m}_t / (\sqrt{\hat{v}_t} + \epsilon)$  (Update parameters)
end while
```

Publié en 2015, actuellement le meilleur algorithme pour entraîner des réseaux de neurones dans une majorité d'applications

MNIST data set

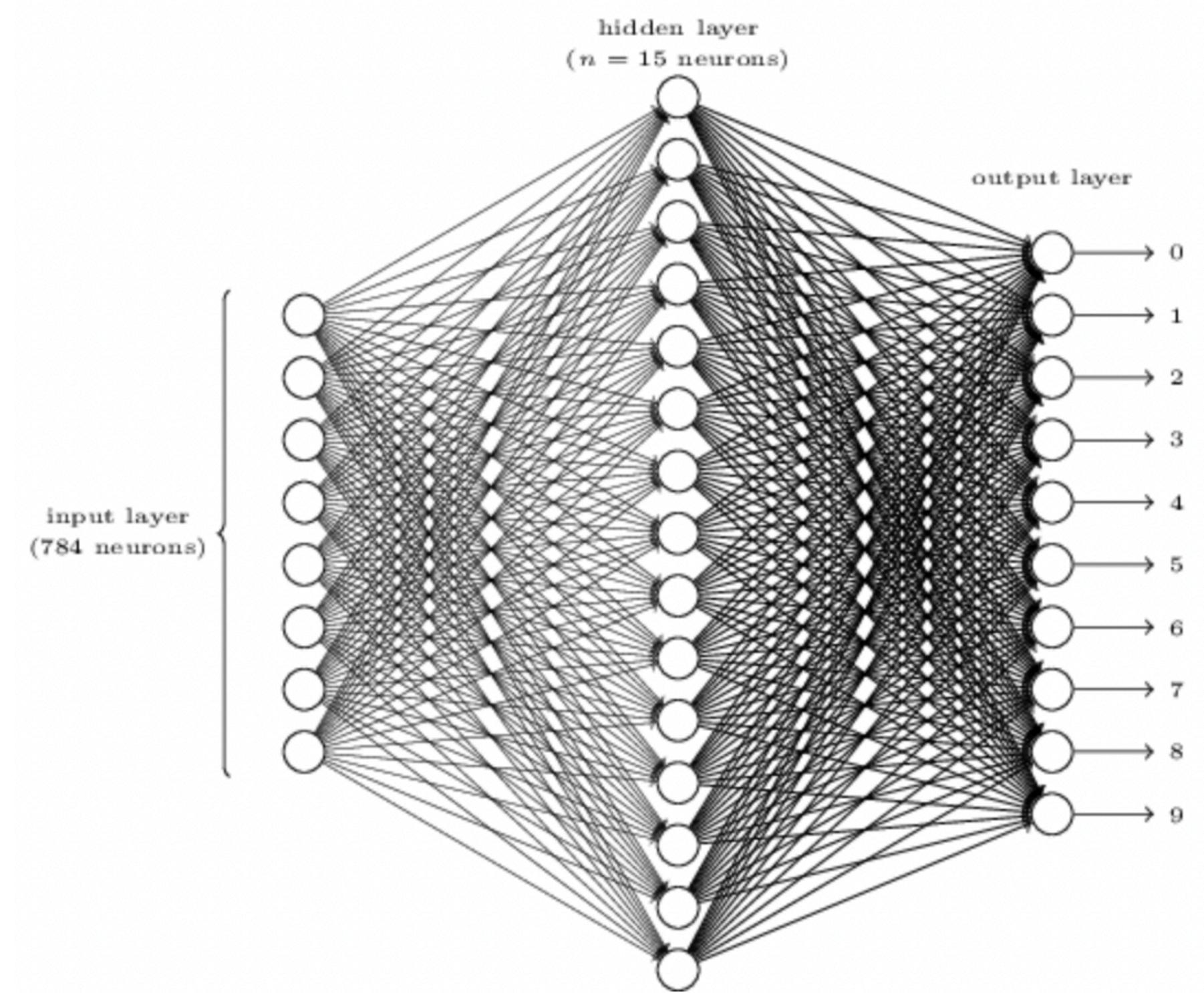
Famous benchmark for classification algorithms based on neural networks



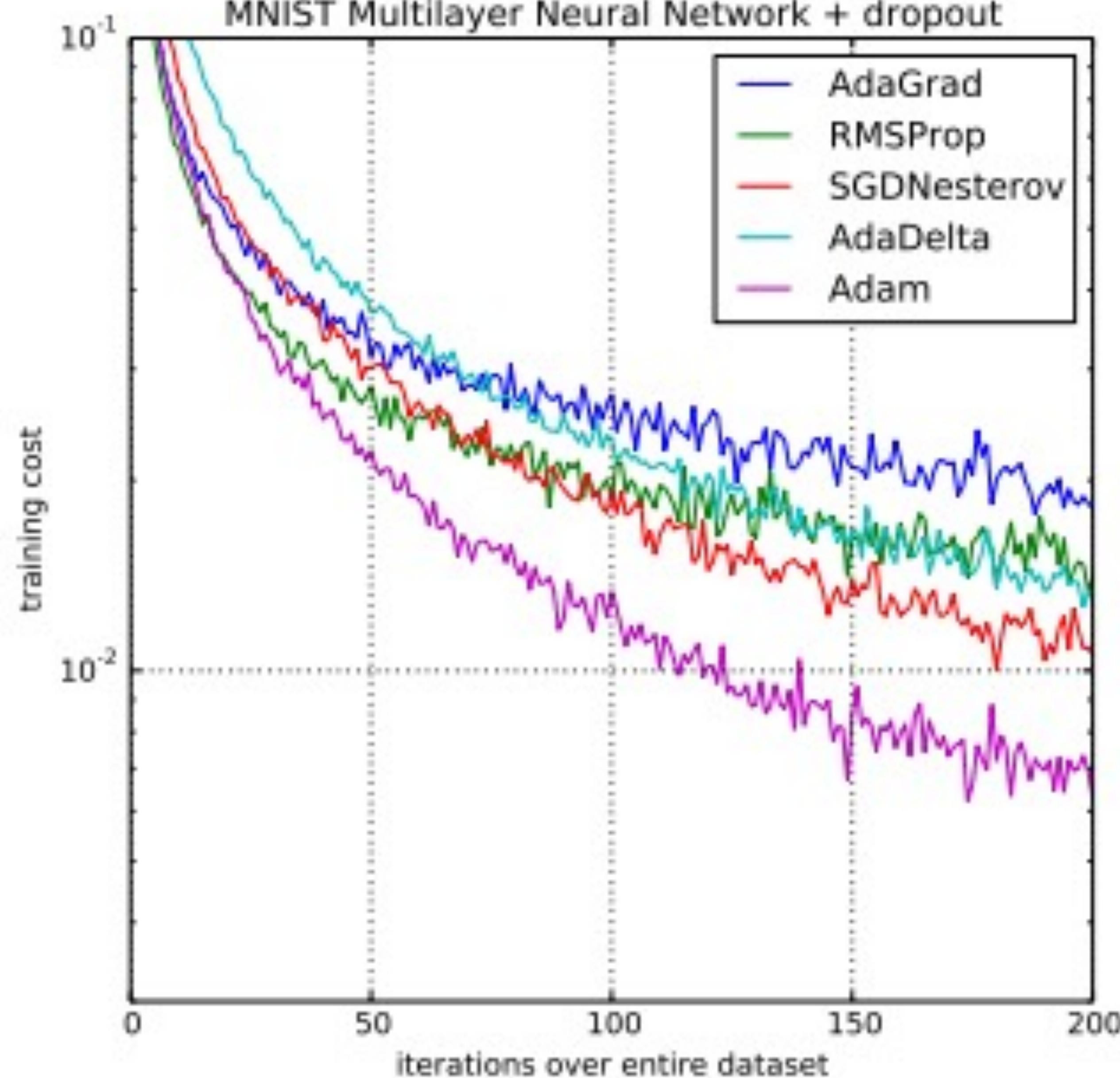
(a) MNIST sample belonging to the digit '7'.



(b) 100 samples from the MNIST training set.



MNIST Multilayer Neural Network + dropout



Descente par coordonnée

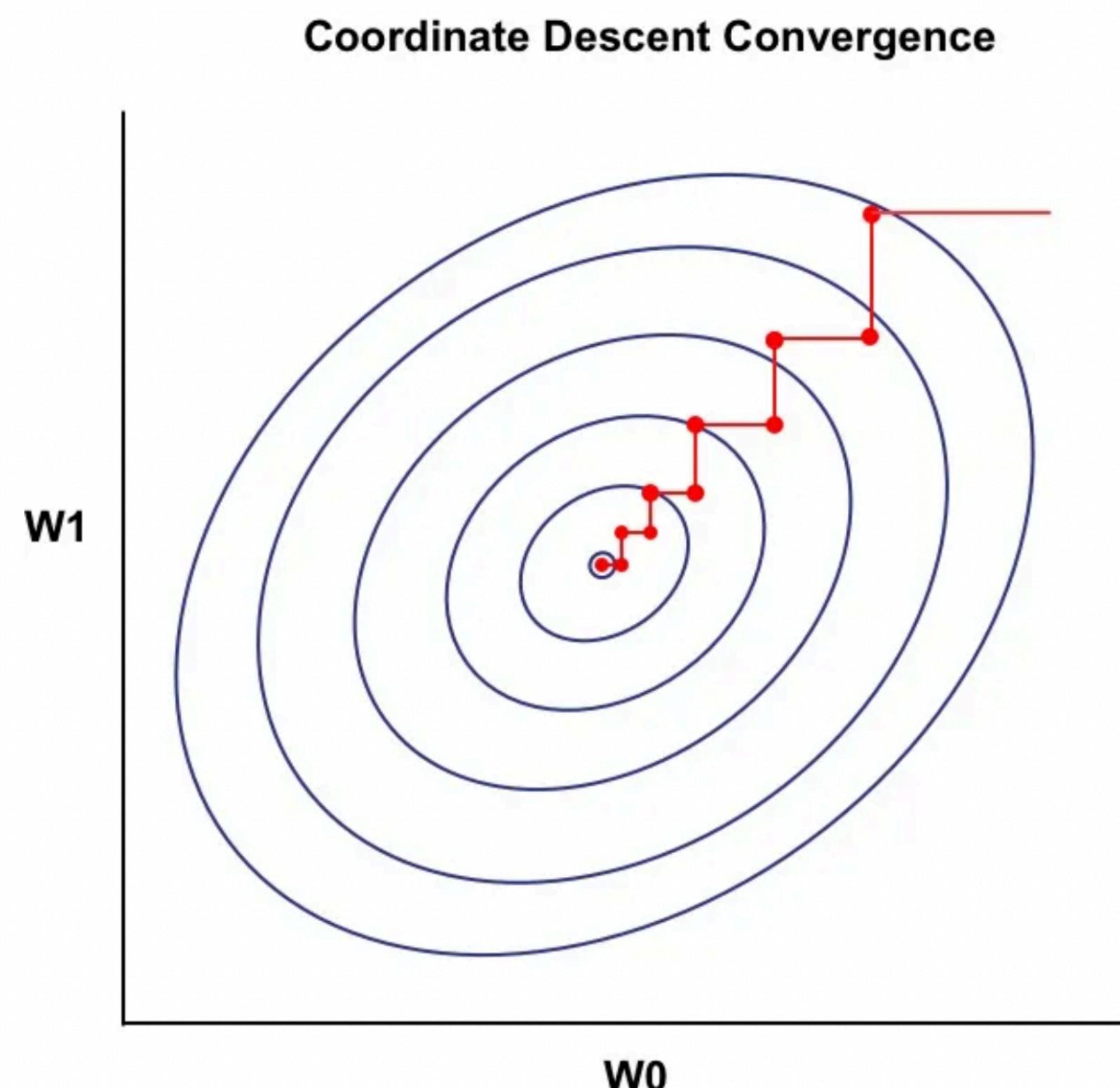
Descente par coordonnée

$$\min_{\omega \in \mathbb{R}^d} f(\omega)$$

Soit un indice $i_k \in \{1, \dots, d\}$,

$$\omega_{k+1} = \omega_k - \gamma_k e_{i_k}$$

où $e_{i_k} = \begin{pmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \leftarrow i_k$



Descente par coordonnée

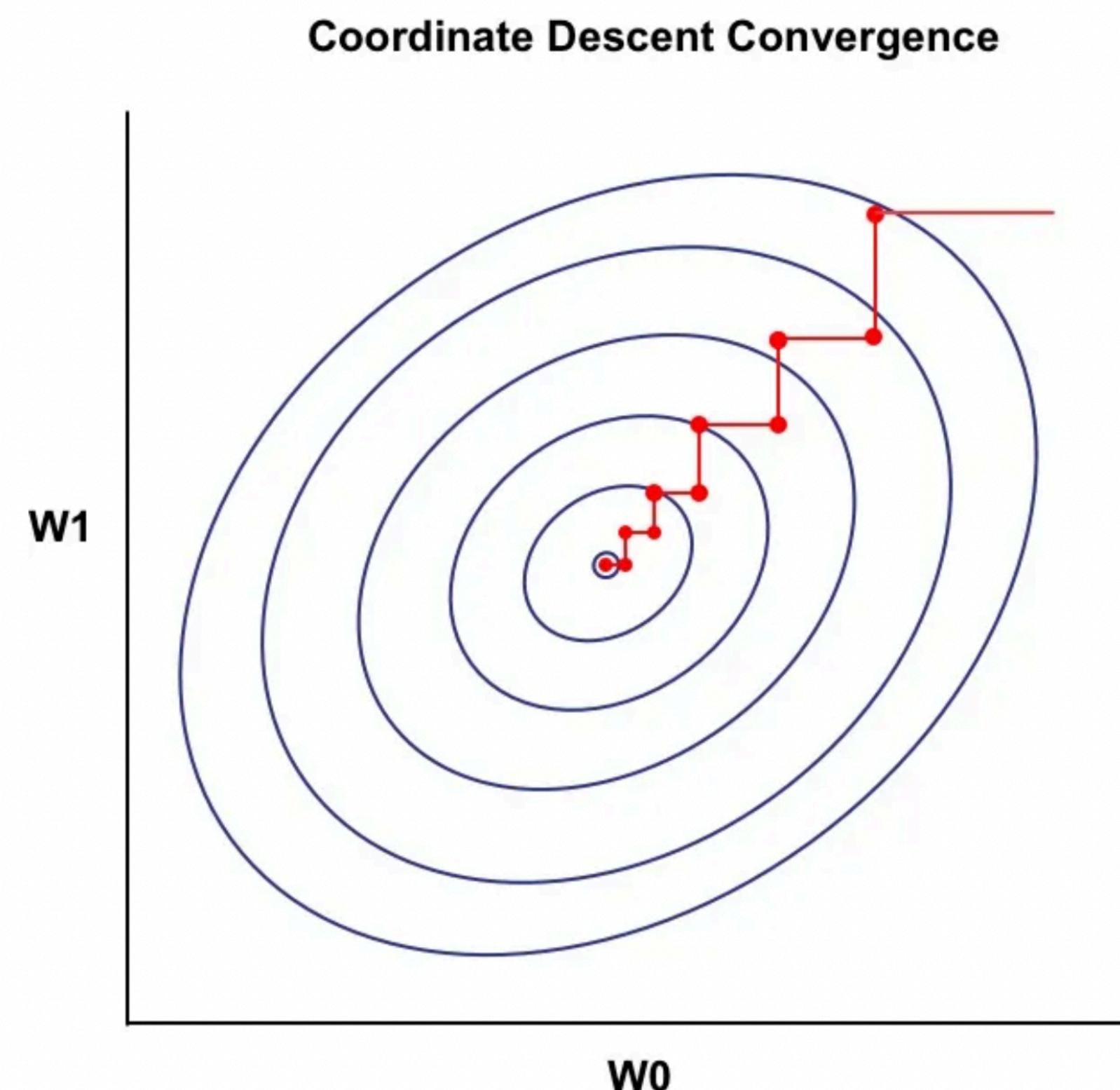
$$\min_{\omega \in \mathbb{R}^d} f(\omega)$$

$$\omega_{k+1} = \omega_k + \gamma_k e_{i_k}$$

Comment choisir γ_k ?

$$\gamma_k := \operatorname{argmin}_\gamma f(\omega_k + \gamma e_{i_k})$$

Idée: minimiser la fonction f dans la direction e_{i_k}



Descente par coordonnée

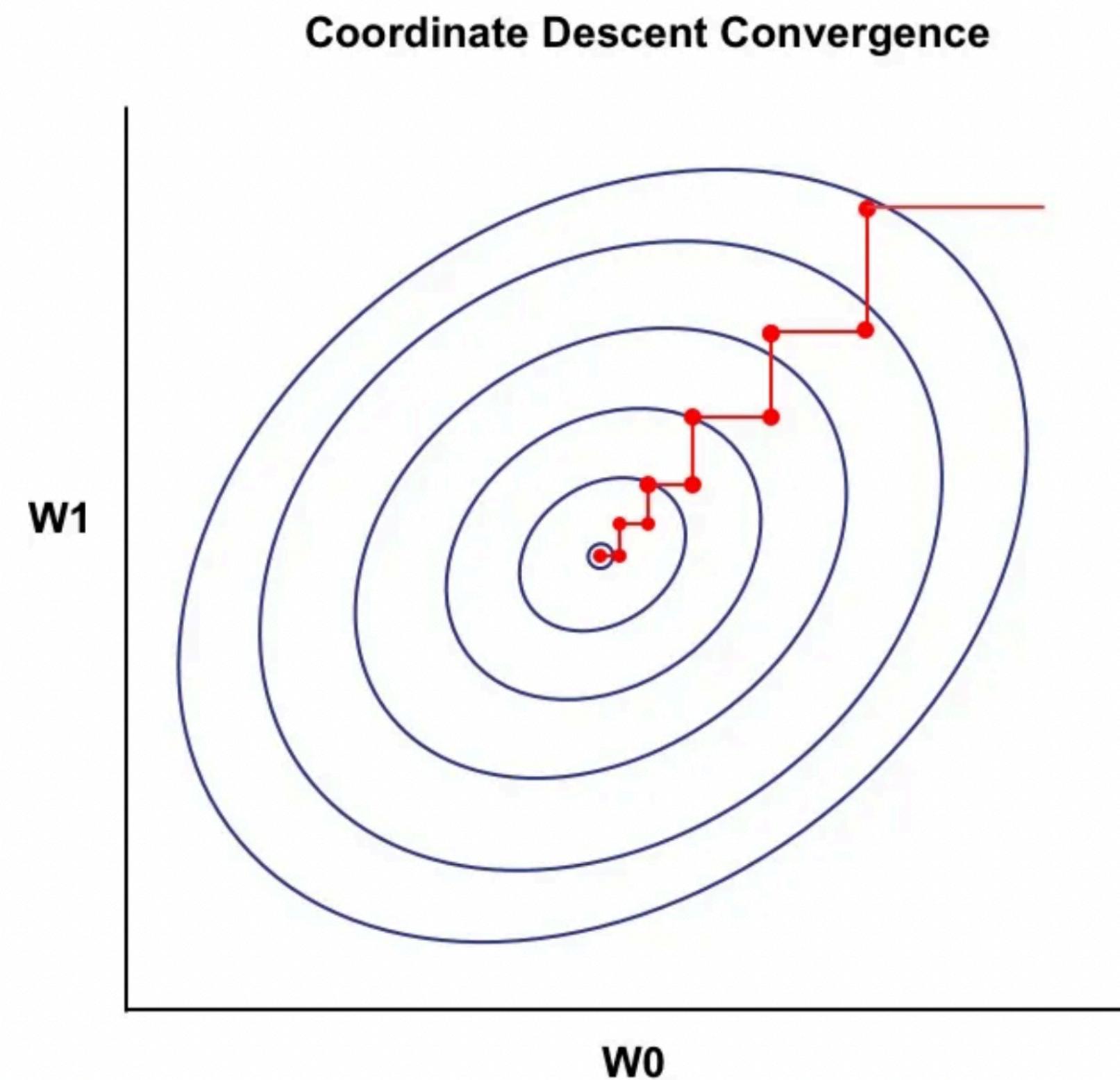
$$\min_{\omega \in \mathbb{R}^d} f(\omega)$$

$$\omega_{k+1} = \omega_k - \alpha_k \nabla_{i_k} f(\omega_k) e_{i_k}$$

où

$$\nabla_{i_k} f(\omega_k) = \frac{\partial f(\omega_k)}{\partial \omega_{i_k}}$$

Questions : - Comment choisir α_k ?
- Comment choisir i_k ?



Descente par coordonnée

Introduites avant les 70's , elles redeviennent populaires suite à leur efficacité pour des problèmes de grande taille en Machine Learning

CD demande généralement plus d'iterations que GD, il faut donc que les itérations soient moins cher à calculer pour que cela vaille la peine

Parfois, la minimisation selon une ou plusieurs des variables a une formule explicite

Algorithm 1 Coordinate Descent for (1)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;

repeat

 Choose index $i_k \in \{1, 2, \dots, n\}$;

$x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;

$k \leftarrow k + 1$;

until termination test satisfied;

Descente par coordonnée

Algorithm 1 Coordinate Descent for (1)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;

repeat

 Choose index $i_k \in \{1, 2, \dots, n\}$;

$x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;

$k \leftarrow k + 1$;

until termination test satisfied;

Stratégie pour la longueur du pas α_k : constant : $\alpha_k = \alpha$ pour tout k ,

minimization exacte dans la direction i_k

recherche linéaire (linesearch), condition de décroissance suffisante

Descente par coordonnée

Algorithm 1 Coordinate Descent for (1)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;

repeat

 Choose index $i_k \in \{1, 2, \dots, n\}$;

$x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;

$k \leftarrow k + 1$;

until termination test satisfied;

Stratégie pour le sampling de la coordonnée i_k :

- cyclique : $1, 2, \dots, n, 1, 2, \dots, n, 1, \dots$
- cyclique randomisé : cybler sur une permutation de $\{1, \dots, n\}$ qui change tous les n pas
- sélection aléatoire d'un indice dans $\{1, \dots, n\}$ à chaque itération

} Meilleures garanties théoriques, impact en pratique moins clair

Descente par coordonnée

Soit f qui est : - μ -fortement convexe
- gradient L -Lipschitz

$\alpha_k = \frac{1}{L}$ et i_k aléatoire , on a

$$\mathbb{E}[f(x_k)] - f(x_*) \leq \left(1 - \frac{\mu}{nL}\right)^k f(x_0) - f(x_*)$$

Algorithm 1 Coordinate Descent for (1)

Set $k \leftarrow 0$ and choose $x^0 \in \mathbb{R}^n$;

repeat

 Choose index $i_k \in \{1, 2, \dots, n\}$;

$x^{k+1} \leftarrow x^k - \alpha_k [\nabla f(x^k)]_{i_k} e_{i_k}$ for some $\alpha_k > 0$;

$k \leftarrow k + 1$;

until termination test satisfied;

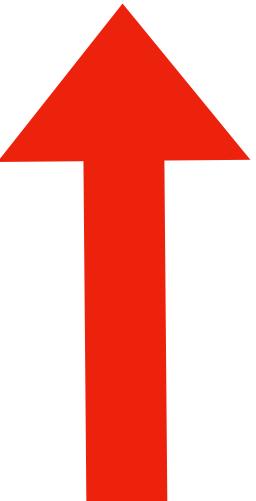
Convergence linéaire

Descente par bloc de coordonnée

Descente par bloc de coordonnée : soit $B_k \subset \{1, \dots, d\}$

$$\min_{\omega \in \mathbb{R}^d} f(\omega)$$

$$\omega_{k+1} = \omega_k - \alpha_k \sum_{i \in B_k} \nabla_i f(\omega_k) e_i \quad \text{où} \quad \nabla_{i_k} f(\omega_k) = \frac{\partial f(\omega_k)}{\partial \omega_{i_k}}$$



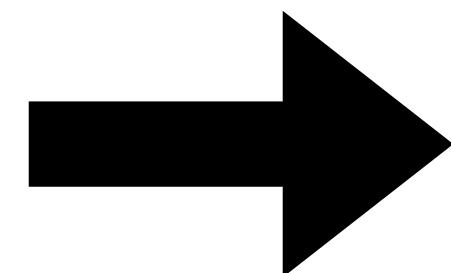
Example: The Netflix Problem

						...
Alice	1			4		
Bob		2	5			
Carol			4	5		
Dave	5				4	
:						

Comment compléter la matrice d'avis pour faire des recommandations ?

The Matrix completion Problem

Hypothèse : la matrice d'avis est de rang faible



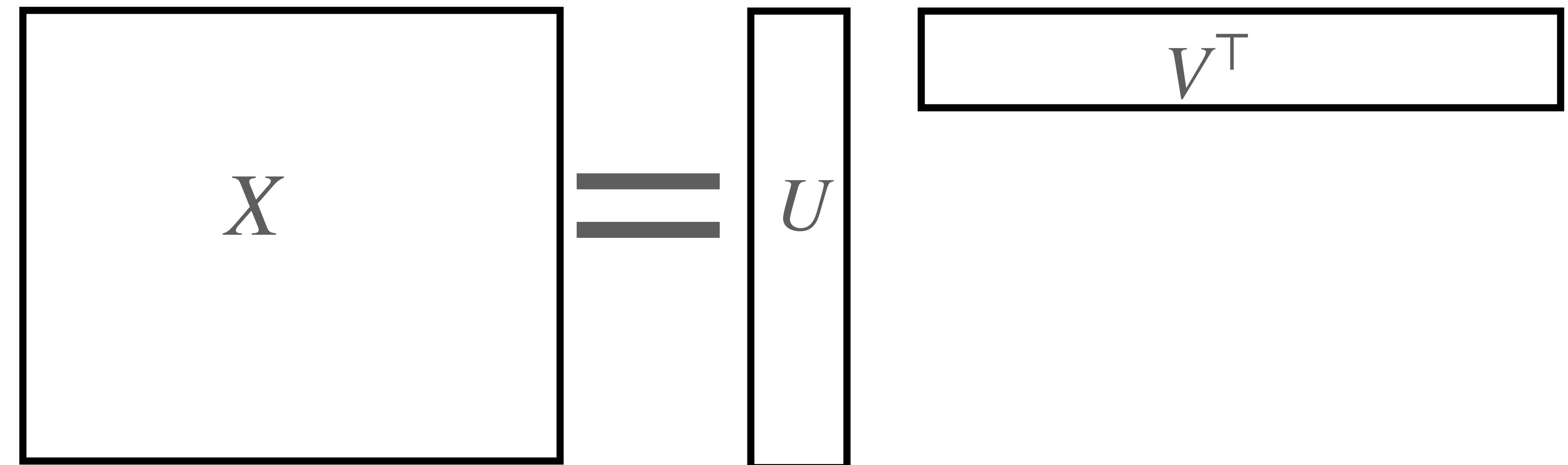
Les films et les consommateurs sont classés dans un nombre limité de catégories

$$X = UV^T$$

$$X \in \mathbb{R}^{m \times n}$$

$$U \in \mathbb{R}^{m \times r}$$

$$V \in \mathbb{R}^{n \times r}$$



The Matrix completion Problem

$$\Omega \subset \{1, \dots, m\} \times \{1, \dots, n\}$$

Représente les entrées (i,j) observées dans la matrice (les films vus)

$$\min_{U,V} = \sum_{i,j \in \Omega} (UV_{ij}^\top - M_{ij})^2 = \|UV^\top - M\|_\Omega^2 \quad \text{Problème nonconvex compliqué en } (U,V)$$

Coordinate descent par bloc (algorithme LMAFIT)

$$\left. \begin{array}{l} U_{k+1} = \operatorname{argmin}_U \|UV_k^\top - M\|_\Omega^2 \\ V_{k+1} = \operatorname{argmin}_V \|U_{k+1}V^\top - M\|_\Omega^2 \end{array} \right\}$$

Chacun des deux sous-problèmes est un système de moindres carrés linéaire (facile à résoudre)

Example : moindres carrés avec sparse data et régularisation

$\min_w f(w) = \|Aw - y\|^2 + \lambda\|w\|^2$, pour matrice $A \in \mathbb{R}^{n \times d}$ sparse (beaucoup d'entrées nulles)

$$\nabla f(w) = 2A^\top(Aw - y) + 2\lambda w$$

$$\nabla_i f(w) = 2 \underbrace{\sum_{j=1}^n A_{ji}(Aw - y)_i}_{\text{Somme sur les éléments de la } i\text{ème colonne } A} + 2\lambda w_i$$



Somme sur les éléments de la i ème colonne A

Cout de calculer $\nabla f(w) \rightarrow \text{nnz}(A)$

Cout de calculer $\nabla_i f(w) \rightarrow \text{nnz}(\text{col}_i(A)) \approx \text{nnz}(A)/n$

Si A est sparse, les itérations de CD sont beaucoup moins chères

Optimisation distribuée

Optimisation distribuée

$$\min_{x \in \mathbb{R}^d} f(x) = \sum_{i=1}^d f_i(x_i)$$

peut être résolu en parallèle par d acteurs indépendants

Peut-on paralléliser le même problème avec une contrainte commune ?

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^d f_i(x_i) \text{ tel que } Ax = b$$

$x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$ et $b \in \mathbb{R}^m$

f convexe

Optimisation distribuée

$$\min_{x \in \mathbb{R}^d} f(x) = \sum_{i=1}^d f_i(x_i)$$

peut être résolu en parallèle par d acteurs indépendants

Peut-on paralléliser le même problème avec une contrainte commune ? **Oui !**

$$\min_{x \in \mathbb{R}^d} \sum_{i=1}^d f_i(x_i) \text{ tel que } Ax = b$$

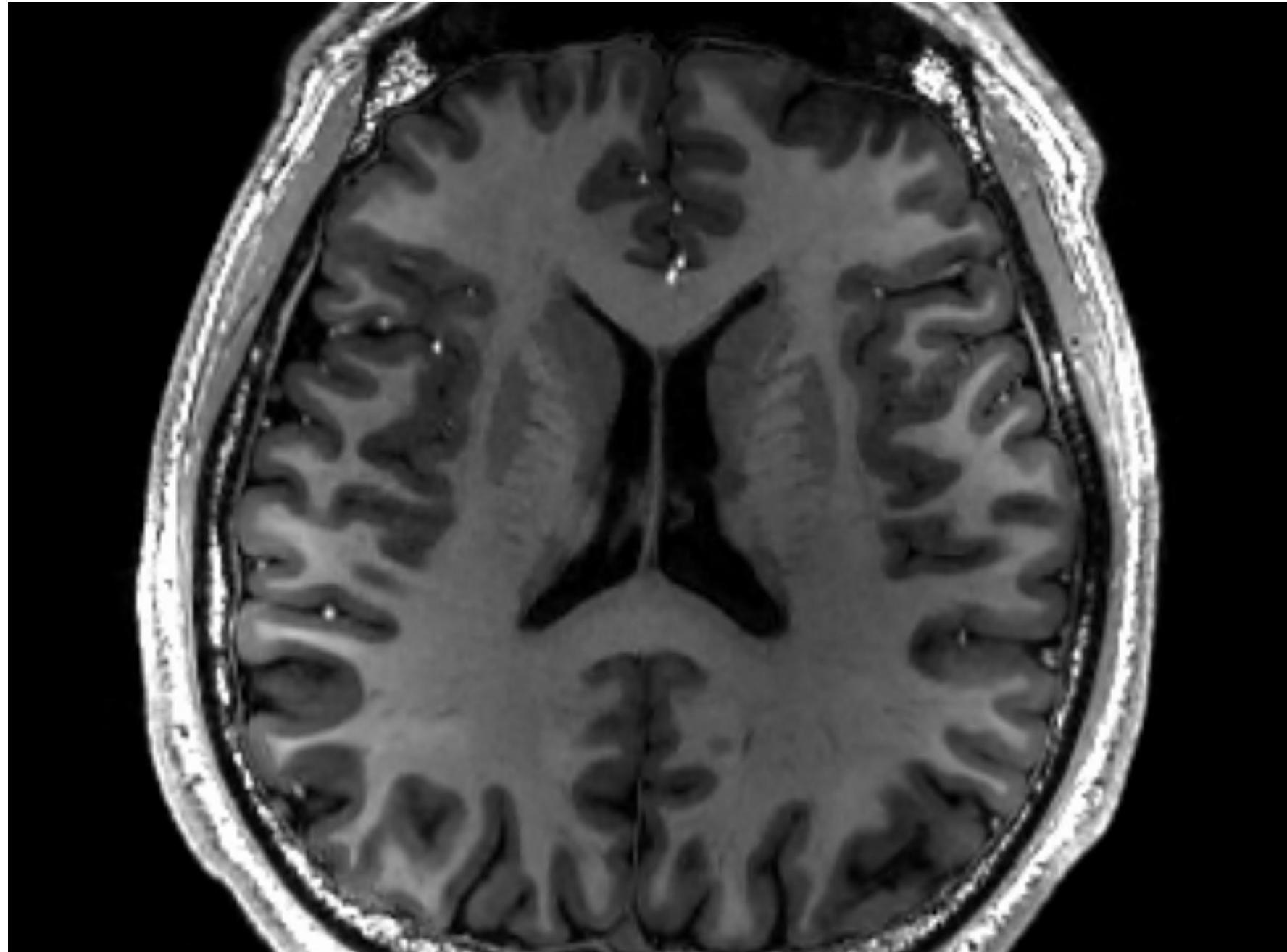
$x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n}$ et $b \in \mathbb{R}^m$

f convexe

Mais pourquoi veut-on faire cela ? Beaucoup d'applications pratiques (compressed sensing,...)

Compressed sensing

Applications en photographie, IRM...



Chaque mesure prend beaucoup de temps, on veut reconstituer le scanner à partir de quelques mesures dans un espace fréquentiel

Compressed sensing

Idée: la majorité des signaux sont sparses quand ils sont représentés dans la bonne base (en fréquence, coefficients de Fourier)

On veut trouver un signal qui satisfait aux mesures, mais qui a un minimum d'entrées non nulles

Les mesures des coefficients de Fourier du signal correspondent à des équations physiques linéaires $Ax=b$

$$\boxed{A} \quad \boxed{x} = \boxed{b}$$

On fait m mesures sur un signal de dimension n avec $m \ll n$.

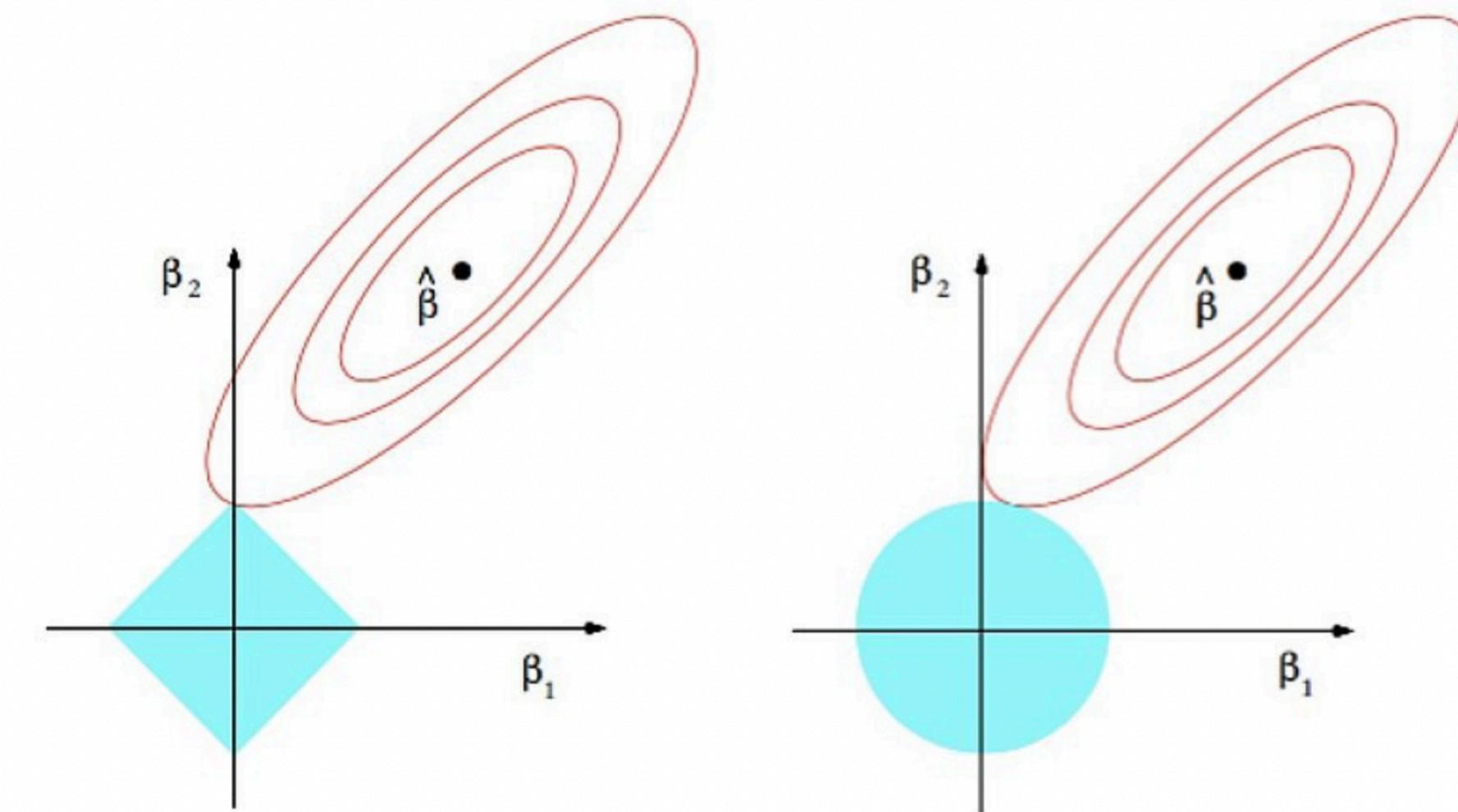
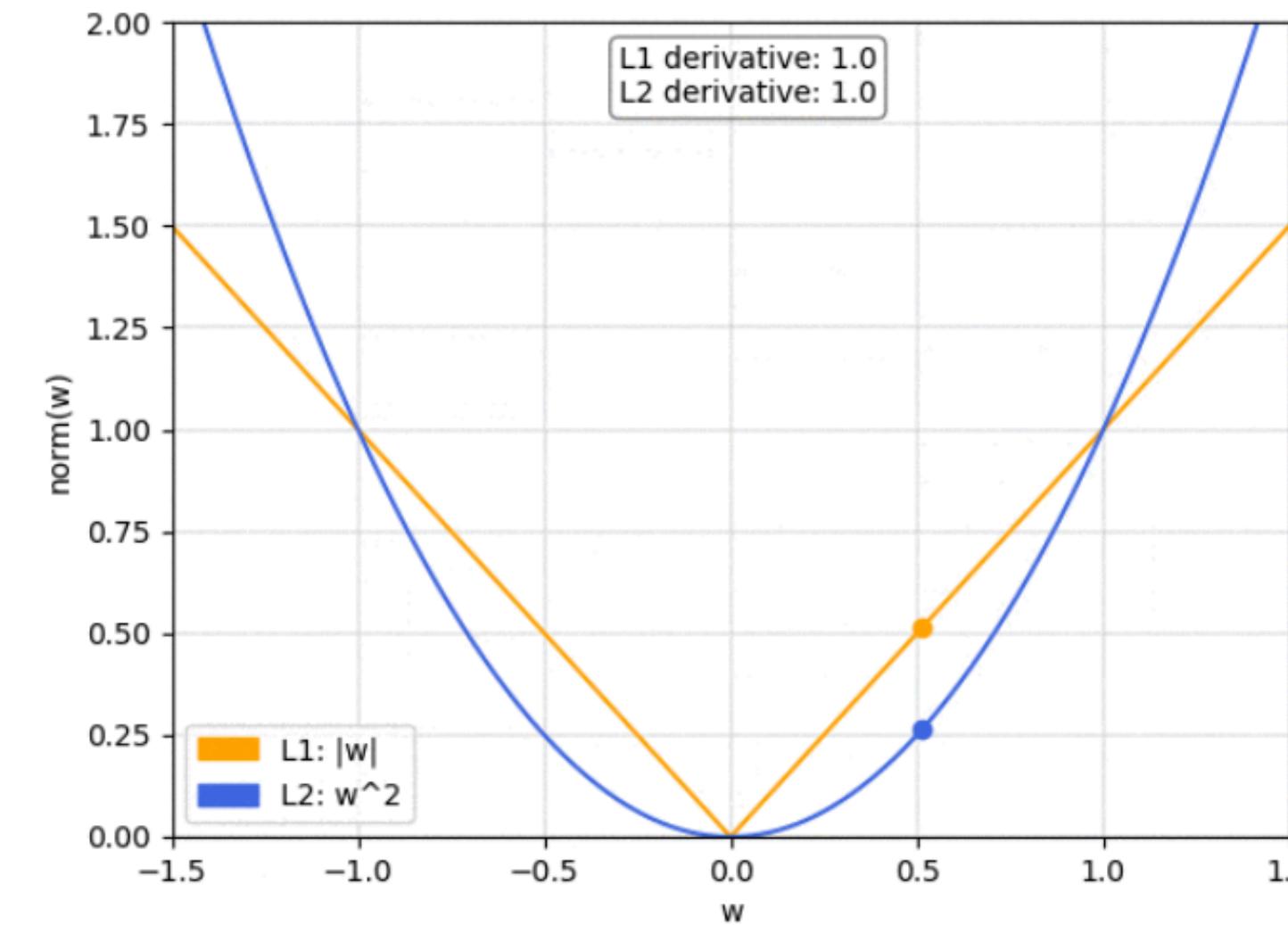
Compressed sensing

$$\min_x \|x\|_0 \text{ tel que } Ax = b \quad \|x\|_0 = \text{ le nombre d'entrées non nulles de } x$$

La fonction $\|x\|_0$ n'est pas continue ! On ne peut pas l'optimiser en pratique

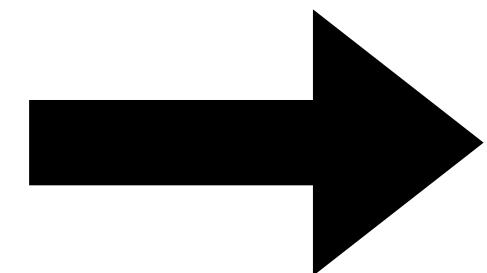
On remplace $\|x\|_0$ par $\|x\|_1 = |x_1| + |x_2| + \dots + |x_d|$

La norme 1 est connue pour promouvoir la sparsité (forcer les entrées à 0)



Compressed sensing

$$\min_x \|x\|_0 \text{ tel que } Ax = b \quad \|x\|_0 = \text{ le nombre d'entrées non nulles de } x$$

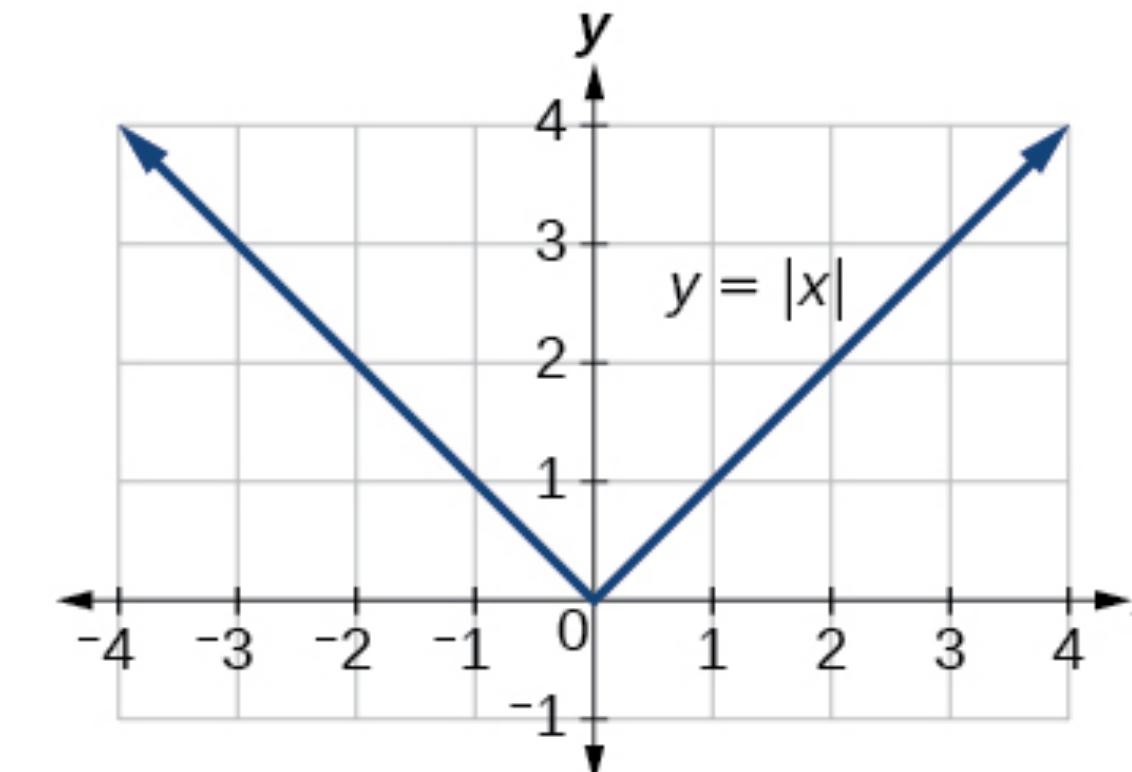


Basis Pursuit :

$$\text{minimiser}_{x \in \mathbb{R}^d} \|x\|_1 \text{ tel que } Ax = b$$

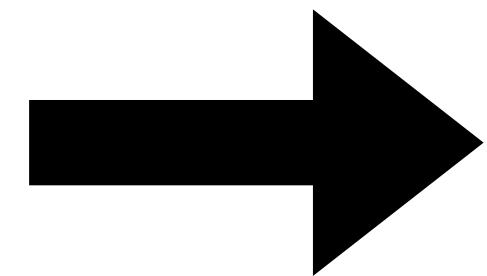
La fonction $\|x\|_1$ n'est pas derivable partout (pas un problème en pratique : sous-gradients)

mais elle est continue (important)



Compressed sensing

$$\min_x \|x\|_0 \text{ tel que } Ax = b \quad \|x\|_0 = \text{ le nombre d'entrées non nulles de } x$$



Basis Pursuit :

$$\text{minimiser}_{x \in \mathbb{R}^d} \|x\|_1 \text{ tel que } Ax = b$$

$$\text{minimiser}_{x \in \mathbb{R}^d} \sum_{i=1}^d f_i(x_i) = |x_1| + |x_2| + \dots + |x_d| \text{ tel que } Ax = b$$

Comment peut-on paralléliser ce problème d'optimisation malgré la contrainte qui joint les variables ?

Construire le dual d'un problème d'optimisation

minimize $f(x)$ $x \in \mathbb{R}^n$ et $A \in \mathbb{R}^{m \times n}$ n variables et m contraintes d'égalité
subject to $Ax = b,$

Les contraintes $Ax = b$ sont ennuyantes.

Idée: on va les mettre dans la fonction objectif !

On construit le Lagrangien: $L(x, y) = f(x) + y^T(Ax - b)$

où $y \in \mathbb{R}^m$ est un vecteur de multiplicateurs de Lagrange

On paye un prix si elles ne sont pas satisfaites

Si x satisfait les contraintes $Ax = b$, alors $L(x, y) = f(x)$

Le Lagrangien peut être minimisé car on a éliminé les contraintes

Construire le dual d'un problème d'optimisation

On minimise le Lagrangien

Fonction duale:

$$g(y) = \inf_x L(x, y)$$

On a la propriété fondamentale que pour tout $y \in \mathbb{R}^m$, $x \in \mathbb{R}^n$ tel que $Ax = b$:

$$g(y) \leq f(x)$$

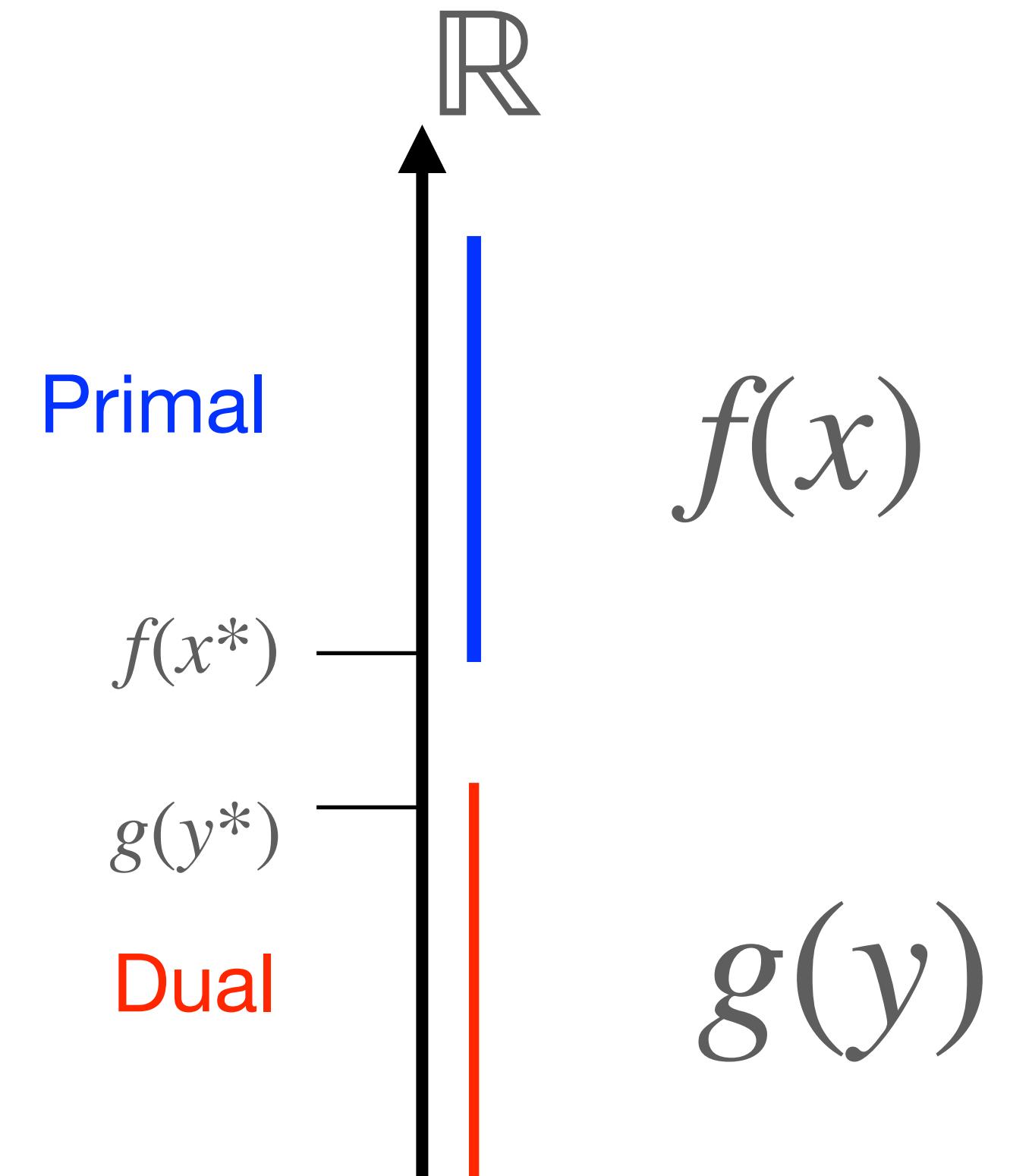
$$g(y) = \inf_x L(x, y) \leq L(x^*, y) = f(x^*) + y^\top (Ax^* - b) = f(x^*)$$

pour x^* la solution du primal

La fonction $g(y)$ est une borne inférieure sur $f(x)$

On veut la maximiser !

Problème dual: $\max_{y \in \mathbb{R}^m} g(y)$



Construire le dual d'un problème d'optimisation

Problème dual:

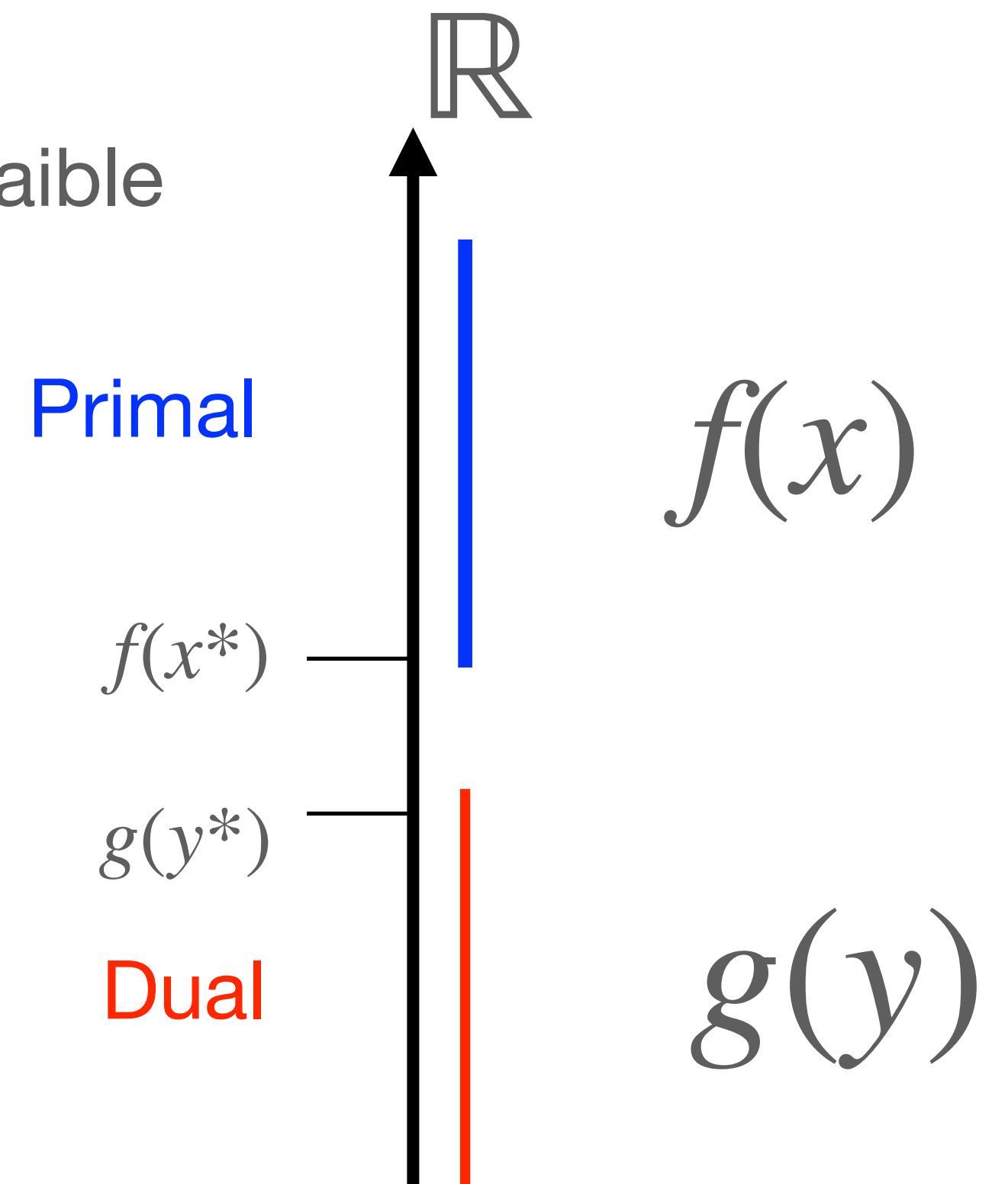
$$\max_{y \in \mathbb{R}^m} g(y)$$

Si $g(y^*) = f(x^*)$, on a dualité forte. Si $g(y^*) < f(x^*)$, on parle de dualité faible

En cas de dualité forte, résoudre les deux problèmes est essentiellement équivalent

$$x^* = \underset{x}{\operatorname{argmin}} L(x, y^*),$$

Si f est fortement convexe, on a dualité forte



Comment maximiser le dual ?

On fait une gradient ascent ! (Au lieu de gradient descent, car c'est un problème de maximisation)

Quel est le gradient de $g(y)$?
$$g(y) = \inf_x L(x, y) \quad L(x, y) = f(x) + y^T(Ax - b)$$

Théorème :

Pour $y \in \mathbb{R}^m$, soit $x^+ = \operatorname{argmin}_x L(x, y)$, alors $\nabla g(y) = Ax^+ - b$

Dual ascent iteration :

$$\begin{aligned} x^{k+1} &:= \operatorname{argmin}_x L(x, y^k) \\ y^{k+1} &:= y^k + \alpha^k(Ax^{k+1} - b), \end{aligned}$$

Optimisation distribuée grace au dual

Peut-on paralléliser le même problème avec une contrainte commune ?

$$\text{minimiser}_{x \in \mathbb{R}^d} \sum_{i=1}^d f_i(x_i) \text{ tel que } Ax = b \quad x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n} \text{ et } b \in \mathbb{R}^m$$

f convexe

Oui, on utilise le dual !

$$\text{On écrit } Ax = [a_1 \ a_2 \ \cdots \ a_d] \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} = \sum_{i=1}^d a_i x_i$$

$$\text{De sorte que } L(x, y) = f(x) + y^\top(Ax - b) = \sum_{i=1}^d f_i(x_i) + y^\top \left(\sum_{i=1}^d a_i x_i - b \right) = \sum_{i=1}^d \left[f_i(x_i) + y^\top a_i x_i - \frac{1}{d} y^\top b \right]$$

Le Lagrangien est séparable !

Optimisation distribuée grâce au dual

Peut-on paralléliser le même problème avec une contrainte commune ?

$$\text{minimiser}_{x \in \mathbb{R}^d} \sum_{i=1}^d f_i(x_i) \text{ tel que } Ax = b \quad x \in \mathbb{R}^n, A \in \mathbb{R}^{m \times n} \text{ et } b \in \mathbb{R}^m$$

Dual ascent iteration :

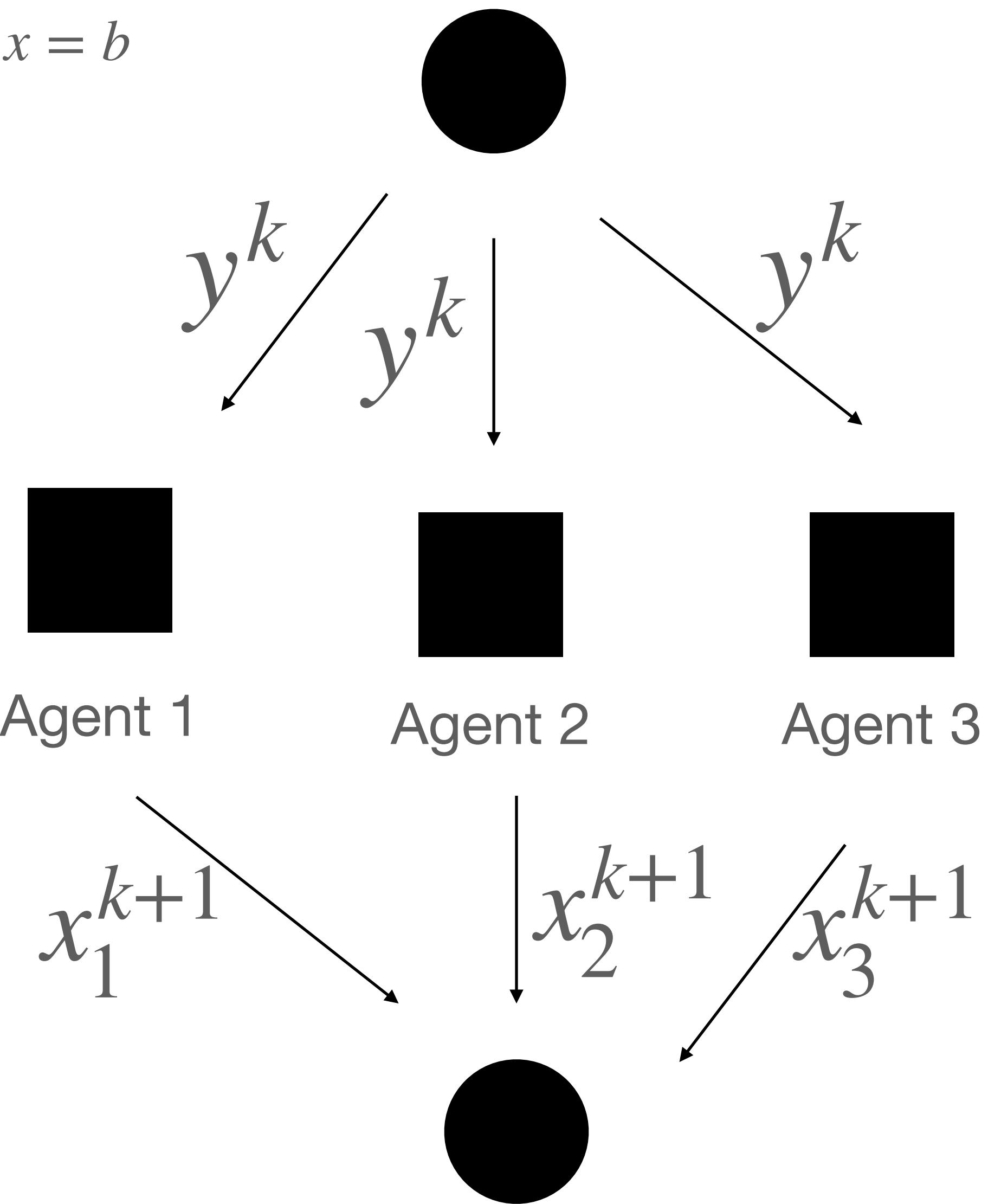
$$x^{k+1} := \operatorname{argmin}_x L(x, y^k) \quad \leftarrow \quad \text{Le Lagrangien est séparable !}$$
$$y^{k+1} := y^k + \alpha^k (Ax^{k+1} - b),$$

Dual ascent iteration en parallèle :

$$x_i^{k+1} := \operatorname{argmin}_{x_i} L_i(x_i, y^k) \quad \leftarrow \quad \text{Optimisation distribuée}$$
$$y^{k+1} := y^k + \alpha^k (Ax^{k+1} - b). \quad \leftarrow \quad \text{Mise en commun}$$

Optimisation distribuée grace au dual

minimiser _{$x \in \mathbb{R}^d$} $\sum_{i=1}^d f_i(x_i)$ tel que $Ax = b$



Dual ascent iteration en parallèle :

$$x_i^{k+1} := \underset{x_i}{\operatorname{argmin}} L_i(x_i, y^k)$$

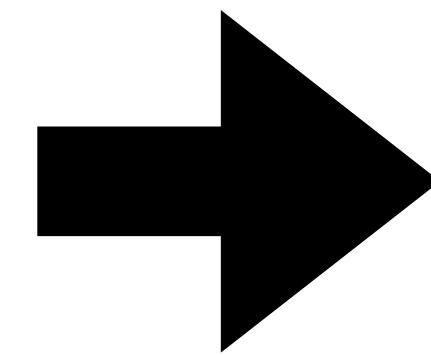
$$y^{k+1} := y^k + \alpha^k (Ax^{k+1} - b).$$

Optimisation distribuée grace au dual

Compressed sensing :

$$\text{minimiser}_{x \in \mathbb{R}^d} \|x\|_1 \text{ tel que } Ax = b \quad \text{où } \|x\|_1 = |x_1| + |x_2| + \dots + |x_d|$$

La fonction de cout est parallelisable mais les contraintes communes empêchent de faire de l'optimisation distribuée.



On passe au dual !

Génial ! MAIS, on peut faire mieux: Dual Ascent a de fortes hypothèses de convergence (convexité forte)

Améliorer Dual Ascent

The method of multipliers

Lagrangien augmenté: $L_\rho(x, y) = f(x) + y^T(Ax - b) + (\rho/2)\|Ax - b\|_2^2,$

Destiné à rendre plus robuste le Lagrangien classique, qui correspond à $\rho = 0$

Le Lagrangien augmenté est en fait le Lagrangien classique de

minimize $f(x) + (\rho/2)\|Ax - b\|_2^2$
subject to $Ax = b.$

qui est équivalent au problème original.

Fonction duale : $g_\rho(y) = \inf_x L_\rho(x, y).$

Avantage: la fonction duale $g_\rho(y)$ est différentiable sans hypothèses de convexité

$$\begin{array}{ll} \text{minimize} & f(x) \\ \text{subject to} & Ax = b, \end{array}$$

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && Ax = b, \end{aligned}$$

The method of multipliers

$$x^{k+1} := \underset{x}{\operatorname{argmin}} L_\rho(x, y^k)$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} - b),$$

Dual ascent :

$$x^{k+1} := \underset{x}{\operatorname{argmin}} L(x, y^k)$$

$$y^{k+1} := y^k + \alpha^k(Ax^{k+1} - b),$$

revient à faire Dual Ascent avec le Lagrangien augmenté au lieu du Lagrangien classique

On choisit aussi le pas $\alpha_k = \rho$

Downside : quand f est séparable, $L_\rho(x, y)$ n'est pas séparable en général

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + \boxed{(\rho/2)\|Ax - b\|_2^2},$$

The method of multipliers

$$x^{k+1} := \underset{x}{\operatorname{argmin}} L_\rho(x, y^k)$$

$$y^{k+1} := y^k + \rho(Ax^{k+1} - b),$$

$$\begin{aligned} & \text{minimize} && f(x) \\ & \text{subject to} && Ax = b, \end{aligned}$$

Dual ascent

$$x^{k+1} := \underset{x}{\operatorname{argmin}} L(x, y^k)$$

$$y^{k+1} := y^k + \alpha^k(Ax^{k+1} - b),$$

revient à faire Dual Ascent avec le Lagrangien augmenté au lieu du Lagrangien classique

On choisit aussi le pas $\alpha_k = \rho$

Downside : quand f est séparable, $L_\rho(x, y)$ n'est pas séparable en général

$$L_\rho(x, y) = f(x) + y^T(Ax - b) + \boxed{(\rho/2)\|Ax - b\|_2^2},$$

Mais on a un autre truc pour utiliser la séparation !

ADMM

(Alternating direction of multiplier method)

Combine séparabilité avec robustesse du Lagrangien augmenté

On suppose que notre fonction f se décompose selon 2 blocs de variables x et z

$$\begin{aligned} & \text{minimize} && f(x) + g(z) \\ & \text{subject to} && Ax + Bz = c \end{aligned}$$

On forme le Lagrangien augmenté :

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2.$$

ADMM

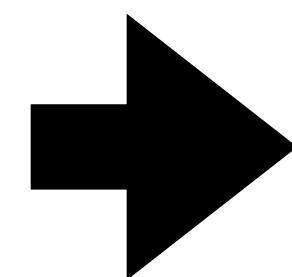
(Alternating direction of multiplier method)

On forme le Lagrangien augmenté (non séparable) :

$$L_\rho(x, z, y) = f(x) + g(z) + y^T(Ax + Bz - c) + (\rho/2)\|Ax + Bz - c\|_2^2.$$

Methode des multiplicateurs :

$$(x^{k+1}, z^{k+1}) := \underset{x, z}{\operatorname{argmin}} L_\rho(x, z, y^k)$$
$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c).$$



ADMM :

$$x^{k+1} := \underset{x}{\operatorname{argmin}} L_\rho(x, z^k, y^k)$$
$$z^{k+1} := \underset{z}{\operatorname{argmin}} L_\rho(x^{k+1}, z, y^k)$$
$$y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c),$$

Minimisation de L_p en descente par coordonnées en 2 blocks

ADMM

(**Alternating direction of multiplier method**)

Les itérations de ADMM sont données par :

$$\left. \begin{array}{l} x^{k+1} := \underset{x}{\operatorname{argmin}} L_\rho(x, z^k, y^k) \\ z^{k+1} := \underset{z}{\operatorname{argmin}} L_\rho(x^{k+1}, z, y^k) \\ y^{k+1} := y^k + \rho(Ax^{k+1} + Bz^{k+1} - c), \end{array} \right\}$$

Dans beaucoup d'applications, ces deux sous-problèmes sont beaucoup plus faciles à résoudre individuellement que en (x, z) ensemble

Compressed sensing using ADMM

$$\begin{aligned} & \text{minimize} && \|x\|_1 \\ & \text{subject to} && Ax = b, \end{aligned}$$

On reformule pour utiliser ADMM (problème équivalent)

$$\begin{aligned} & \text{minimize} && f(x) + \|z\|_1 \\ & \text{subject to} && x - z = 0, \end{aligned} \quad \text{avec} \quad f(x) = \begin{cases} 0 & \text{si } Ax = b \\ 1 & \text{si } Ax \neq b \end{cases}$$

ADMM appliqué à ce problème:

$$x^{k+1} := \Pi(z^k - u^k)$$

où Π est la projection sur $\{x \in \mathbb{R}^d : Ax = b\}$

$$z^{k+1} := S_{1/\rho}(x^{k+1} + u^k)$$

$$u^{k+1} := u^k + x^{k+1} - z^{k+1},$$

Les deux sous-problèmes sont faciles à résoudre !

$$x^{k+1} := (I - A^T(AA^T)^{-1}A)(z^k - u^k) + A^T(AA^T)^{-1}b.$$

$$S_\kappa(a) = \begin{cases} a - \kappa & a > \kappa \\ 0 & |a| \leq \kappa \\ a + \kappa & a < -\kappa, \end{cases} \quad (\text{Soft thresholding})$$

Compressed sensing using ADMM

$$\begin{aligned} & \text{minimize} && \|x\|_1 \\ & \text{subject to} && Ax = b, \end{aligned}$$

On reformule pour utiliser ADMM (problème équivalent)

$$\begin{aligned} & \text{minimize} && f(x) + \|z\|_1 \\ & \text{subject to} && x - z = 0, \end{aligned} \quad \text{avec} \quad f(x) = \begin{cases} 0 & \text{si } Ax = b \\ 1 & \text{si } Ax \neq b \end{cases}$$

ADMM pour basis pursuit :

$$\begin{aligned} x^{k+1} &:= \Pi(z^k - u^k) \\ z^{k+1} &:= S_{1/\rho}(x^{k+1} + u^k) \\ u^{k+1} &:= u^k + x^{k+1} - z^{k+1}, \end{aligned}$$

où Π est la projection sur $\{x \in \mathbb{R}^d : Ax = b\}$

Converge vers une solution approximative en quelques dizaines d'itérations

Très efficace en pratique

Consensus

Consensus

$$\text{minimize} \quad f(x) = \sum_{i=1}^N f_i(x),$$

Nouveau problème: personne n'a accès à toutes les fonctions f_i !

Chaque agent a accès à une seule fonction f_i mais la variable x est commune

Comment peut-on decentraliser l'optimisation ?

On re-formule le problème pour utiliser ADMM et optimiser en parallèle

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^N f_i(x_i) \\ & \text{subject to} && x_i - z = 0, \quad i = 1, \dots, N. \end{aligned}$$

On introduit une variable artificielle z qui joint les x_i

Consensus

$$\begin{aligned} & \text{minimize} && \sum_{i=1}^N f_i(x_i) \\ & \text{subject to} && x_i - z = 0, \quad i = 1, \dots, N. \end{aligned}$$

Lagrangien augmenté :

$$L_\rho(x_1, \dots, x_N, z, y) = \sum_{i=1}^N \left(f_i(x_i) + y_i^T (x_i - z) + (\rho/2) \|x_i - z\|_2^2 \right),$$

ADMM:

$$x_i^{k+1} := \underset{x_i}{\operatorname{argmin}} \left(f_i(x_i) + y_i^{kT} (x_i - z^k) + (\rho/2) \|x_i - z^k\|_2^2 \right)$$

Les N agents peuvent effectuer ce calcul décentralisé

$$z^{k+1} := \frac{1}{N} \sum_{i=1}^N \left(x_i^{k+1} + (1/\rho) y_i^k \right)$$

Agent central (n'a pas besoin des fonctions f_i)

$$y_i^{k+1} := y_i^k + \rho(x_i^{k+1} - z^{k+1}).$$

décentralisé

Chaque agent a sa fonction privée f_i , sa variable x_i et son multiplicateur de Lagrange y_i

Merci !