La nouvelle règle de mise à jour devient : $v_{k+1} = \alpha v_k - \eta \nabla f(x_k)$ (nouvelle vitesse) avec $v_0 = 0$ ($v_1 = -\eta \nabla f(x_k)$)

$x_{k+1} = x_k + v_{k+1}$

$$\Leftrightarrow \boxed{x_{k+1} = \underbrace{x_k - \eta \nabla f(x_k)}_{\text{descente classique}} + \underbrace{\alpha v_k}_{\text{momentum}}}$$

α est le paramètre de momentum : $\alpha \in [0,1[$. En général, on prend $\alpha = 0{,}9$ (ordre de grandeur)

Si $\alpha = 0$, on retombe sur la descente de gradient classique

On peut écrire explicitement la nouvelle itération : $x_{k+1} = x_k + v_{k+1}$

$$\begin{aligned} &= x_k - \eta \nabla f(x_k) + \alpha v_k \\ &= x_k - \eta \nabla f(x_k) + \alpha(\alpha v_{k-1} - \eta \nabla f(x_{k-1})) \\ &= x_k - \eta(\nabla f(x_k) + \alpha \nabla f(x_{k-1})) + \alpha^2 \underbrace{v_{k-1}}_{\alpha v_{k-2} - \eta \nabla f(x_{k-2})} \\ &= \dots \\ &= x_k - \eta \underbrace{\sum_{i=0}^{k} \alpha^{k-i} \nabla f(x_i)} \end{aligned}$$

Toutes les itérations précédentes s'accumulent avec un poids de plus en plus faible (puisque $\alpha < 1$) au fur et à mesure qu'on remonte les itérations

Accélération de Nesterov

L'idée est la même que la descente avec momentum, mais la nouvelle direction de descente n'est pas calculée en x_k mais en $x_k + \alpha v_k$ (position avancée du fait de l'inertie)

La règle de mise à jour devient : $y_k = x_k + \alpha v_k$ — position avancée du fait de l'inertie

$v_{k+1} = \alpha v_k - \eta \nabla f(y_k)$ — nouvelle pente calculée en $y_k = x_k + \alpha v_k$

$x_{k+1} = x_k + v_{k+1}$ — idem descente avec momentum

$$\rightarrow \boxed{x_{k+1} = \underbrace{x_k + \alpha v_k}_{\text{idem momentum}} - \eta \nabla f(\underbrace{x_k + \alpha v_k}_{y_k})}$$

Dans la règle de mise à jour ci dessus, les hyperparamètres η et α sont fixés. Nesterov a montré que lorsque η et α étaient choisis judicieusement (variables en fonction de l'itération), alors la convergence de la descente devenait quadratique, et a prouvé qu'il était impossible de l'accélérer encore plus.

Ceci dit, même pour α = constante, la méthode de Nesterov est plus rapide que la descente avec momentum : dans l'analogie avec la balle, cette dernière devient "intelligente" et adapte sa trajectoire en avance ($\nabla f(x_k + \alpha v_k)$ vs $\nabla f(x_k)$), ce qui lui donne une meilleure robustesse au bruit (cadre d'apprentissage des réseaux de neurones par exemple).

Dans les accélérations avec momentum et de Nesterov, le learning rate η est fixe et le même pour toutes les variables
→ les méthodes plus avancées (Adagrad, RMSprop, Adam, etc) permettent d'adapter le learning rate en fonction de la variable

Adagrad (adaptive gradient algorithm)

Si on note $x_k = (x_k^1, \dots, x_k^n)^T \in \mathbb{R}^n$ le point de $\mathbb{R}^n$ après k itérations, adagrad permet d'adapter le pas en fonction de x_k^i

L'itération classique $x_{k+1} = x_k - \eta \nabla f(x_k) \iff \forall i = 1, \dots, n,\ x_{k+1}^i = x_k^i - \eta \frac{\partial f}{\partial x_i}(x_k)$ est remplacée par

$$x_{k+1} = x_k - \frac{\eta}{\sqrt{\operatorname{diag}(G_k) + \varepsilon}} \odot \nabla f(x_k)$$ avec $\odot$ le produit d'Hadamard → multiplication par éléments de deux vecteurs

$x \odot y = (x_1 y_1, \dots, x_n y_n)^T$

$$\iff x_{k+1}^i = x_k^i - \frac{\eta}{\sqrt{G_k^{ii} + \varepsilon}} \frac{\partial f}{\partial x_i}(x_k)$$

→ La règle de mise à jour d'Adagrad fait intervenir une matrice $G_k \in \mathbb{R}^{n \times n}$ qui correspond à l'accumulation de tous les gradients passés $G_k = \sum_{i=0}^{k} \nabla f(x_i) \nabla f(x_i)^T$ et $\varepsilon > 0$ un tout petit nombre (pour éviter une potentielle division par 0)

G_k^{ii} est le i^e élément sur la diagonale de G_k : $G_k^{ii} = \sum_{j=0}^{k} \left(\frac{\partial f}{\partial x_i}(x_j) \right)^2$

L'accumulation des gradients au dénominateur au fur et à mesure des itérations fait que ce dernier devient de plus en plus grand, donc le learning rate de plus en plus petit, en particulier pour des variables avec des forts gradients (variables qui évoluent rapidement)

6) Algorithme du gradient conjugué

La méthode du gradient conjugué a été développée pour résoudre des problèmes quadratiques du type $\min_{x \in \mathbb{R}^n} \frac{1}{2} x^T A x - b^T x$ avec A symétrique définie positive

→ descente de gradient lente, surtout si A est mal conditionnée

→ possible (et facile) de calculer le pas optimal dans le cas quadratique

Si $f(x) = \frac{1}{2} x^T A x - b^T x$, on a $\nabla f(x) = Ax - b$. Donc pour l'itéré $x_k \in \mathbb{R}^n$, $\nabla f(x_k) = Ax_k - b$

Le long de la direction de descente d_k, le pas optimal $\eta^* = \min_\eta f(x_k + \eta d_k)$ est donné par $\varphi'(\eta^*) = 0$ avec $\varphi(\eta) = f(x_k + \eta d_k)$

→ $\varphi'(\eta) = \frac{d}{d\eta} f(x_k + \eta d_k) = d_k^T \nabla f(x_k + \eta d_k)$ (dérivation en chaîne d'une fonction composée)

→ $\varphi'(\eta^*) = d_k^T \nabla f(x_k + \eta^* d_k) = 0$

$$\iff d_k^T (A(x_k + \eta^* d_k) - b) = 0$$

$$\iff d_k^T (\underbrace{Ax_k - b}_{\nabla f(x_k)} + \eta^* A d_k) = 0$$

$$\iff d_k^T \nabla f(x_k) + \eta^* d_k^T A d_k = 0$$, d'où le pas optimal $\boxed{\eta^* = -\frac{d_k^T \nabla f(x_k)}{d_k^T A d_k}}$

L'intérêt de la méthode du gradient conjugué est qu'elle converge en au plus n itérations si $A \in \mathbb{R}^{n \times n}$, moyennant des directions de descente adaptées (pas opposées au gradient en chaque itéré)

Définition: direction conjuguées

Soit $Q \in \mathbb{R}^{n \times n}$ une matrice symétrique. Deux vecteurs $x, y \in \mathbb{R}^n$ sont conjugués pour Q (ou Q-orthogonaux) si et seulement si $x^T Q y = 0$

→ Si $Q = I_d$, on retrouve l'orthogonalité classique