

Optimisation Stochastique

Chapitre I : Introduction

pierre.weiss@cnrs.fr

$$X: \text{entrées} \quad Y: \text{sorties} \quad Z = X \cdot Y$$

On suppose que les entrées X , les sorties Y sont des variables aléatoires de loi $P(X, Y)$.

En apprennentage, ce qui nous intéresse est d'approcher la loi conditionnelle $P(Y|X)$.

Idealement, on aimerait trouver $P(Y|X)$ mais c'est souvent trop lourd numériquement.

Dans ce cours, on suppose que X et Y sont liés approximativement par une application déterministe T

$$\begin{aligned} T: X &\rightarrow Y \\ x &\mapsto T(x) \end{aligned}$$

exemple $T = E[Y|X]$

"le meilleur estimateur de $P(Y|X)$ "

Notre objectif sera de déterminer une application $h \approx T$

h est appelée fonction de prédiction.

Étant donnée une fonction de perte $l: Y \times Y \rightarrow \mathbb{R} \cup \{+\infty\}$ on peut quantifier la qualité d'un prédicteur h par une énergie :

$$E(h) = \int l(h(x), y) dP(x, y) = E(l(h(x), y))$$

Le meilleur prédicteur possible h^* est défini par :

$$h^* = \arg \min_{h: X \rightarrow Y} E(h)$$

Il est incalculable en général car :

- la loi $P(x, y)$ est inconnue (exemple : loi d'une image)
- l'ensemble des fonctions $h: X \rightarrow Y$ est infini

Pour résoudre ces problèmes on va considérer deux outils :

- Risque empirique
- Régularisation

Risque Empirique : Comme la loi jointe est inconnue, on va considérer une base de données $(x_n, y_n)_{1 \leq n \leq N}$. Le risque empirique est défini par :

$$E_N(h) = \frac{1}{N} \sum_{n=1}^N l(h(x_n), y_n)$$

Régularisation : Plutôt que de minimiser $E_N(h)$ par rapport à toutes les applications, on minimise sur une famille de dimension finie H_D , $D \in \mathbb{N}$. On arrive au problème :

$$\min_{h \in H_D} E_N(h)$$

ex: Ψ_1, \dots, Ψ_D $\Psi_d: X \rightarrow Y$
 $H_D = \text{vect}(\Psi_1, \dots, \Psi_D)$
 $h \in H_D \Leftrightarrow h = \sum_{d=1}^D w_d \Psi_d$

Questions :

- Est-ce grave de minimiser E_N plutôt que E ?
- Quel est l'effet de la régularisation par H_D ?
- Comment résoudre (P_N) ?

I . 2) Quelques exemples

1.2.1 Reconnaissance de caractères

X : espace d'imagette $X = \mathbb{R}^{28 \times 28}$

$$Y = \{0, 1, 2, \dots, 9\}$$

$h: X \rightarrow Y$

$$\boxed{3} \rightarrow 3$$

$$l(h(x), y) = \begin{cases} 0 & \text{si } h(x) = y \\ 1 & \text{sinon} \end{cases}$$

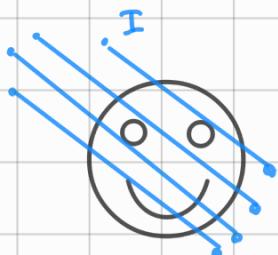
Cela ne fonctionne pas car E_N est constante par morceau : presque tout point est critique ! (gradient nul).

$$h: X \rightarrow \Delta_g = \left\{ y \in \mathbb{R}^{10} \text{ tq } \forall i, y_i \geq 0 \text{ et } \sum_{i=0}^g y_i = 1 \right\}$$

$\ell(h(x_n), y_n) = \text{dist}(h(x_n), \delta_{y_n})$ où dist est une distance entre lois de probabilité

Exemple: cross-entropy

1.2.2 Régression (exemple: problème inverse)



X : ensemble des coeff d'absorption

Y : espace des mesures en sortie

$$y = Lx + b \text{ où } L: X \rightarrow Y \text{ est un opérateur}$$

qui décrit comment sont absorbés les rayons X et b est du "bruit" de capteur.

Pour reconstruire x à partir de y , on peut chercher à résoudre le problème: $\inf_{x \in X} \frac{1}{2} \|Lx - y\|_2^2$

On souhaite que x soit consistant avec y . Le souci est qu'on manque d'information suivant certains angles et qu'on obtient des images de mauvaise qualité.

En apprentissage machine, une autre solution est de construire une base de données (x_n, y_n) avec $y_n = Lx_n + b_n$

$$\inf_{h \in H} \frac{1}{N} \sum_{i=1}^n \ell(h(y_n), x_n) \text{ avec par exemple}$$

$$\ell(h(y_n), x_n) = \|h(y_n) - x_n\|_2 \text{ et } h \text{ est un réseau de neurones.}$$

I.3. Quelques exemples de fonction perte

Classification binaire

- Fonction 0-1 $l(\hat{y}, y) = \begin{cases} 0 & \text{si } \hat{y} = y \\ 1 & \text{sinon} \end{cases}$ $\mathcal{Y} = \{-1, 1\}$

- Fonction Hinge $l(\hat{y}, y) = \max(0, 1 - \hat{y}y)$

- Entropie croisée: qui permet de traiter le cas
 $\mathcal{Y} = \{0, \dots, I\}$ où I est le nombre de labels

Régression

- Quadratique: $\frac{1}{2} \|\hat{y} - y\|_2^2$

- Robuste: $\frac{1}{2} \|\hat{y} - y\|_1$

- Réseau neurones $l(\hat{y}, y) = \text{NN}(\hat{y}, y)$

I.4) Quelques exemples de prédicteurs

Un prédicteur est une fonction $h: X \times W \rightarrow \mathcal{Y}$
 $(x, w) \mapsto h(x, w)$

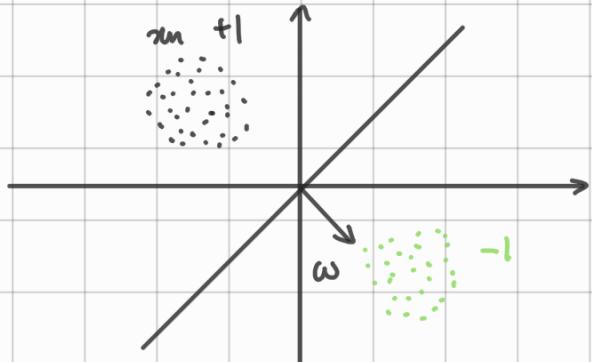
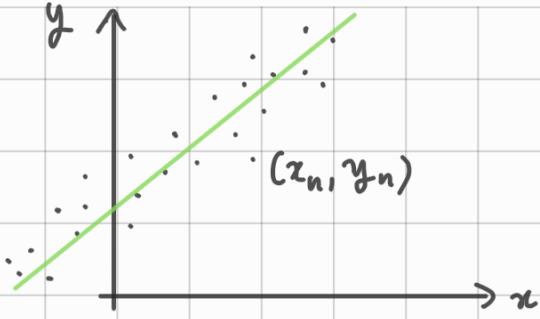
W espace des "poids" ou "paramètres" qui définissent le prédicteur
On suppose $W \subseteq \mathbb{R}^D$

1.4.1 Régression linéaire

On suppose $X = \mathbb{R}^N$ $\mathcal{Y} = \mathbb{R}^M$

On peut considérer $H_D = \{ \text{applications affines de } \mathbb{R}^N \text{ dans } \mathbb{R}^M \}$
 $D = NM + M$ car toute application affine s'écrit sous la forme
 $h(x) = Ax + b$ avec $A \in \mathbb{R}^{M \times N}$, $b \in \mathbb{R}^M$

Dans ce cadre on peut poser $w = (A, b) \in \mathbb{R}^D$

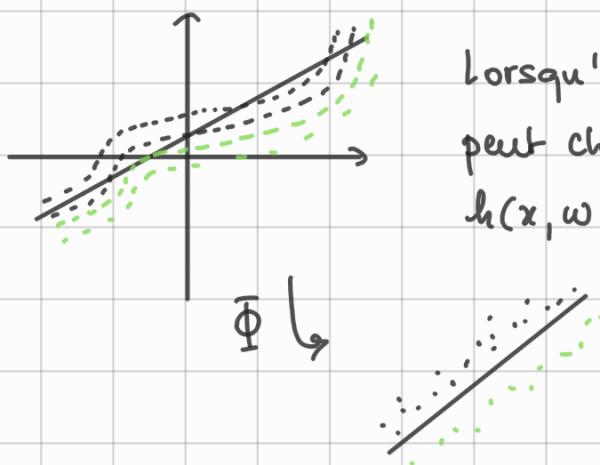


1.4. 2 Classification binaire

$$X = \mathbb{R}^N \quad Y = \{-1; 1\}$$

$h(x) = \text{signe } (\langle w, x \rangle) \in \{-1, 0, 1\}$ (SVM) $w \in \mathbb{R}^N$

Le prédicteur h permet de scinder l'espace en deux (gauche et droite de l'hyperplan) $\{x \in \mathbb{R}^N, \langle w, x \rangle = 0\}$



Lorsqu'on ne peut pas séparer les données on peut choisir un "noyau" $\Phi: \mathbb{R}^N \rightarrow \mathbb{R}^D$, $D \geq N$

$$h(x, w) = \text{signe } (\langle \Phi(x), w \rangle)$$

$\underbrace{\Phi}_{\in \mathbb{R}^D}: \mathbb{R}^N \rightarrow \mathbb{R}^D$

1.4. 3 Réseaux de neurones

Un réseau de neurones est défini comme une suite d'opérations du type :

$$x = x^{(0)} \xrightarrow{c\ell_1} x^{(1)} \xrightarrow{c\ell_2} x^{(2)} \xrightarrow{\dots} x^{(J)}$$

$$x \in \mathbb{R}^N \quad (X = \mathbb{R}^N)$$

$$x^{(J)} \in \mathbb{R}^M \quad (Y = \mathbb{R}^M)$$

$$\text{NN}(x) = [c\ell_1 \circ \dots \circ c\ell_J](x)$$

J est la profondeur du réseau

Chaque application $c\ell_j$ est une couche du réseau.

Réseaux linéaires :

Chaque application $c\ell_j$ est affine ou linéaire :

$$c_j(x^{(j-1)}) = A_j x^{j-1} + b_j \quad A_j \in \mathbb{R}_{\sum_j}^{P_j \times P_{j-1}} \quad b_j \in \mathbb{R}^{P_j} \rightarrow \text{biais}$$

$$D = \sum_{j=1}^J P_j P_{j-1} + P_j$$

Réseaux non-linéaires

$c_j(x^{(j-1)}) = \sigma(A_j x^{j-1} + b_j) \quad \sigma: \mathbb{R} \rightarrow \mathbb{R} \quad \text{fonction d'activation}$
 appliquée coordonnée par coordonnée
 Par exemple ReLU (Rectified Linear Unit)
 où $\sigma(t) = \max(t, 0)$



Réseau convolutionnels

Un soucis des réseaux linéaires est le nombre trop important de paramètres. Il ne sont pas implémentables si M et N sont grands.

$$c_j(x^{(j-1)}) = \sigma(\Psi_j * x^{j-1} + b_{j-1})$$

Où Ψ_j est un filtre de convolution.

On a réduit le # de paramètres de la couche j de

$P_j(P_{j-1}+1)$ à $P_j + q_j$ où q_j est le # de param du filtre Ψ_j .

Rappel convolution $a \in \mathbb{R}^N$ $b \in \mathbb{R}^Q$ $(a * b)[i] = \sum_{k=1}^Q a[i-k] b[k]$
 ↳ invariante par translation.

I.6 Quelques exemples d'Énergies = fonction objectif = $f \circ \text{côt}$...

Régression linéaire

$$\inf_{\substack{A \in \mathbb{R}^{M \times N} \\ b \in \mathbb{R}^M}} \frac{1}{2} \sum_{n=1}^N \|Ax_n + b - y_n\|_2^2 = F(A, b)$$

On peut vouloir régulariser les poids.

$$\inf_{\substack{A \in \mathbb{R}^{M \times N} \\ b \in \mathbb{R}^M}} \frac{1}{2} \sum_{n=1}^N \|Ax_n + b - y_n\|_2^2 + \frac{\lambda}{2} (\|A\|_F^2 + \|b\|_{\mathbb{R}^M}^2)$$

Classification binaire

$$\inf_{w \in \mathbb{R}^D} \frac{1}{2} \|w\|_2^2 \quad \text{tq} \quad \begin{cases} \langle w, \phi(x_n) \rangle > 0 & \text{si } y_n = 1 \\ \langle w, \phi(x_n) \rangle < 0 & \text{si } y_n = -1 \end{cases}$$

Réseau de neurones

$$\inf_{w \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^N \ell(h(x_n, w), y_n) + R(w)$$

Calcul

$$\inf_{A \in \mathbb{R}^{M \times N}} F(A) = \frac{1}{2} \sum_{n=1}^N \|Ax_n - y_n\|_2^2 = \sum_{n=1}^N F_n(A)$$

$$F_n(A) = \frac{1}{2} \|Ax_n - y_n\|_2^2$$

$$\begin{aligned} F_n(A+H) &= \frac{1}{2} \langle (A+H)x_n - y_n, (A+H)x_n - y_n \rangle_{\mathbb{R}^n} \\ &= F_n(A) + \langle Hx_n, Ax_n - y_n \rangle_{\mathbb{R}^n} + O(\|H\|_F^2) \end{aligned}$$

$$\begin{aligned} \partial F_n(A)(H) &= \langle Hx_n, Ax_n - y_n \rangle_{\mathbb{R}^n} \\ &= \langle H, (Ax_n - y_n) \otimes x_n \rangle_F \\ &= \langle H, (Ax_n - y_n) x_n^T \rangle_F \end{aligned}$$

Chapitre 2 : les erreurs d'apprentissage

$$h^* = \underset{h: X \rightarrow Y}{\operatorname{argmin}} E(h)$$

idéal

$$h_{H_D}^{*0} = \underset{h \in H_D}{\operatorname{argmin}} E(h)$$

famille

$$h_m^{*0} = \underset{h \in H_D}{\operatorname{argmin}} E_m(h)$$

Risque Empirique

On note E les erreurs d'apprentissage

$$\mathcal{E} = E(\tilde{h}_n) - E(h^*) \geq 0$$

$$= \underbrace{E(\tilde{h}_n) - E(h_m^*)}_{\mathcal{E}_{\text{opt}}} + \underbrace{E(h_m^*) - E(h_{H_D}^{*0})}_{\mathcal{E}_{\text{estimation}}} + \underbrace{E(h_{H_D}^{*0}) - E(h^*)}_{\mathcal{E}_{\text{approximation}}}$$

$\mathcal{E}_{\text{opt}} \downarrow$ avec N

$\mathcal{E}_{\text{app}} \downarrow$ avec D

II. 1 Erreurs d'approximation

Les erreurs d'approximation caractérisent la capacité de la famille H_D à approcher le prédicteur h^* .

⚠ choix de la famille H_D demande souvent un énorme travail d'analyse et de modélisation.

2.1.1 Largeur de Kolmogorov

Une méthode pour construire des prédicteurs consiste à considérer un sous espace vectoriel H_D .

$h \in H_D \Leftrightarrow h = \sum_{d=1}^D w_d \Psi_d$ où $\Psi_d: X \rightarrow Y$ sont des applications prédefinies.

Exemples :

- Polynômes
- Polynômes trigonométriques
- Ondelettes

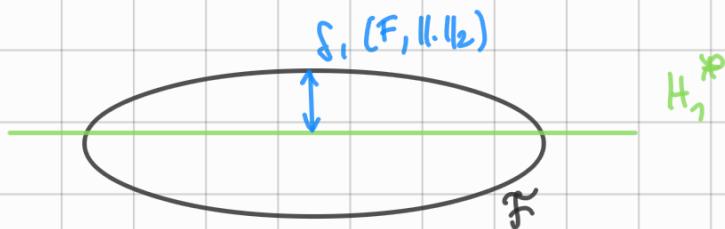
- Splines

Supposons qu'on sache par avance que $h^* \in F$ où F est la famille de fonctions connue.

La largeur de Kolmogorov de F permet de quantifier à quel point la famille F peut être approchée par un espace de dimension D .

Définition Largeur de Kolmogorov

$$\delta_d(F, \|\cdot\|) = \inf_{\dim(H_d)=d} \sup_{f \in F} \inf_{h \in H_d} \|h - f\|$$



Proposition

- $\forall F \quad \delta_d(F) \downarrow$
- $\delta_d(\alpha F) = |\alpha| \delta_d(F) \quad \forall \alpha$
- $\delta_d(F) = \delta_d(\text{conv}(F)) \quad \text{conv}(F) = \text{enveloppe convexe de } F$
- F compacte $\Leftrightarrow \delta_d(F) \rightarrow 0 \quad \text{et } F \text{ bornée}$
 $d \rightarrow +\infty$

Définition Espace de Sobolev

On note $W_p^n([0,1]) = \{ h \in C^{(n-1)}([0,1]), h^{(n-1)} \text{ absolument continue et } h^{(n)} \in L^p([0,1]) \}$

$$W_2^0 = L^2$$

$$W_p^0 = L^p$$

$$W_\infty^1 = \{ \text{fonct}^\circ \text{ lipschitz} \}$$

Théorème

$$S_d(B_p^n) \propto d^{-n} \text{ pour } 1 \leq p \leq +\infty$$

$$\text{avec } B_p^n = \{ f \in W_p^n([0,1]) \text{ tq } \|f^{(n)}\|_{L^p} \leq 1 \}$$

les sev optimaux sont les polynômes trigonométriques
les polynômes, les ondelettes et les splines

En dimension P

$$\text{Soit } H^n = W_2^n([0,1]^P) = \{ h \in C^{(n-1)}([0,1]^P), h^{(n)} \in L^2([0,1]^P) \}$$

$$\text{Soit } B_p^n = \{ f \in H^n, \|f\|_{L^p}^n \leq 1 \}$$

$$\text{Théorème: } S_D(B_p^n, \|\cdot\|_2) = O(D^{-n/p})$$

Fléau de dimension ! $P = 10000$ $n = 100$ (très régulier)

Erreur d'approximation $O(D^{-1/100})$

Pour faire une erreur de $1/100$ il faut

$$D = 10^\alpha \quad D^{-1/100} = 10^{-\alpha/100} = 10^{200}$$

Il faut un sev de dimension $D=100^{200}$ pour obtenir une erreur d'approximation au $1/100$.

2.1.2 Expressibilité des réseaux de neurones

Théorème d'approximation universelle:

Soit $f: \mathbb{R}^P \rightarrow \mathbb{R}^Q$ une fct continue. Soit $K \subset \mathbb{R}^P$ un ensemble compact et $\epsilon > 0$ une précision arbitraire.

Alors $\exists f_\epsilon$ de la forme : $f_\epsilon = W_2 \circ \sigma \circ W_1$ avec W_1 et W_2 affines

σ une fct arbitraire qui n'est pas un polynôme

telle que $\|f_\epsilon - f\|_{L^\infty(K)} \leq \epsilon$

$$\sup_{x \in K} |f_\epsilon(x) - f(x)|$$

Exemple: $\sigma = \text{ReLU}$ $\sigma(x) = x_+$



$$P=Q=1$$

$$W_1(x) = \begin{pmatrix} a_1 x + b_1 \\ \vdots \\ a_D x + b_D \end{pmatrix} \quad W_2(y) = \langle c, y \rangle + d$$

$$W_2 \circ \sigma \circ W_1(x) = \sum_{d=1}^D c_d \sigma(a_d x + b_d) + d$$

Donc la fonction $W_2 \circ \sigma \circ W_1$ est une fonction linéaire par morceau. Elle contient D morceaux.

II.2 Erreur d'estimation

Théorème: Supposons que $H = \{h = \sum_{d=1}^D w_d \psi_d\}$
où (ψ_d) est une famille arbitraire. Dans ce cas on a:

$$\sup_{h \in H} |E(h) - E_N(h)| \leq c \sqrt{\frac{D}{N}} \quad \text{où } c \text{ est une constante}$$

⚠ Il faut beaucoup d'échantillons N pour avoir une erreur d'estimation faible.

Chapitre 3 : gradient stochastique et variantes

$$\min_{w \in \mathbb{R}^D} \frac{1}{N} \sum_{n=1}^N f_n(w) \quad \text{typiquement: } f_n(w) = l(h(x_n, w), y_n)$$

Si chaque des fct f_n est différentiable, on peut effectuer la descente du gradient

$$w_{k+1} = w_k - \frac{\alpha_k}{N} \sum_{n=1}^N \nabla f_n(w_k)$$

Problèmes :

- si N est immense, la somme est trop coûteuse
- si D est grand ($w \in \mathbb{R}^D$), calculer juste 1 gradient $\nabla f_n(w)$ peut être très coûteux.

III. 1 Le gradient stochastique

3.1.1 le principe

Plutôt que d'effectuer la descente / tous les paramètres, on en choisit 1 au hasard :

$$\begin{aligned} w_0 &\in \mathbb{R}^D \text{ point de départ} \\ &\text{choisir un indice } i_k \text{ au hasard} \\ w_{k+1} &= w_k - \alpha_k \nabla f_{i_k}(w_k) \end{aligned}$$

⚠ Cet algorithme n'est pas un algorithme de descente.

$$f(x) = \frac{1}{2} \underbrace{(2x^2)}_{f_1(x)} + \frac{1}{2} \underbrace{(-x^2)}_{f_2(x)} = \frac{x^2}{2}$$

En pratique, on préfère souvent faire des descentes par batch :

- choisir un sous ensemble d'indices $I_k \subseteq \{1, \dots, N\}$
- $w_{k+1} = w_k - \alpha_k \nabla_{I_k} f(w_k)$

$$\text{avec } \nabla_{I_k} f(w) = \frac{1}{|I_k|} \sum_{i \in I_k} \nabla f_i(w)$$

En particulier si $I_k = \{1, \dots, N\}$ la descente du gradient stochastique coïncide avec la descente du gradient.

On parle d'une epoch lorsque le nombre d'itérations K est tel que $\sum_{k=1}^K |I_k| = N$

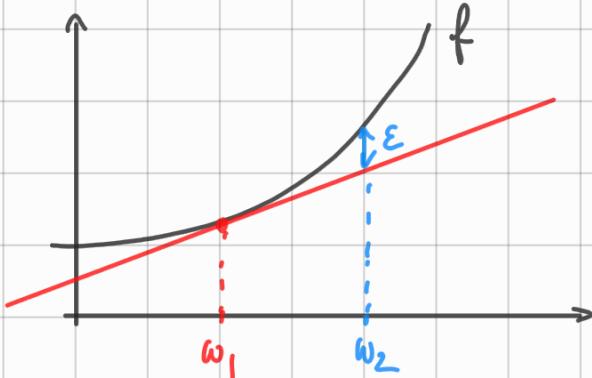
En apprentissage, on appelle α_k le learning rate (pas de la descente)

3.1.2 Garanties théoriques dans le cas fortement convexe

Hypothèses 3.1 les fonctions f_n sont dans $C^1(\mathbb{R}^D)$ et la fonction

$f = \sum f_n$ est à gradient Lipschitz :

$$\|\nabla f(w_1) - \nabla f(w_2)\|_2 \leq L \|w_1 - w_2\|_2$$



Rappel : Sous cette condition :

$$f(w_2) \leq f(w_1) + \langle \nabla f(w_1), w_2 - w_1 \rangle + \frac{L}{2} \|w_2 - w_1\|_2^2$$

proposition : sous l'hypothèse 3.1 on a :

$$\begin{aligned} E_{i_k} (f(w_{k+1})) &\leq f(w_k) - \alpha_k \langle \nabla f(w_k), E_{i_k} (\nabla f_{i_k}(w_k)) \rangle \\ &+ \frac{L}{2} \alpha_k^2 E_{i_k} (\|\nabla f_{i_k}(w_k)\|_2^2) \end{aligned}$$

H. 3.2 . Les indices i_k sont tirés indépendamment les uns les autres

- $E_{i_k} (\nabla_{i_k} f(w)) = \nabla f(w)$
- $\text{var} (\|\nabla f_{i_k}(w_k)\|_2) \leq \sigma^2$ pour $\sigma > 0$

Exemple : Si on prend i_k iid suivant la loi uniforme sur $\{1, \dots, N\}$ on a :

$$E_{i_k} (\nabla f_{i_k}(w)) = \frac{1}{N} \sum_{n=1}^N \nabla f_n(w) = \nabla f(w)$$

Dans ce cas on obtient :

$$E_{i_k} (f(w_{k+1})) \leq f(w_k) - \alpha_k \|\nabla f(w_k)\|_2^2 + \frac{L}{2} \alpha_k^2 E_{i_k} (\|\nabla f_{i_k}(w_k)\|_2^2)$$

$$\text{et } \text{Var} (\|\nabla f_{i_k}(w_k)\|_2) = E_{i_k} (\|\nabla f_{i_k}(w_k)\|_2^2) - \|\nabla f(w_k)\|_2^2 \leq \sigma^2$$

$$\text{Donc } E_{i_k} (f(w_{k+1})) \leq f(w_k) - \alpha_k \|\nabla f(w_k)\|_2^2 + \frac{L}{2} \alpha_k^2 (\sigma^2 + \|\nabla f(w_k)\|_2^2)$$

pour α_k suffisamment petit, le terme :

$\alpha \|\nabla f(w_k)\|_2^2 - \frac{L}{2} \alpha_k^2 (\sigma^2 + \|\nabla f(w_k)\|_2^2)$ est positif et on obtient bien une méthode de descente en espérance

H.3.3 Il existe un paramètre $\mu > 0$ tel que f est μ - fortement convexe : $f(w_2) \geq f(w_1) + \langle \nabla f(w_1), w_2 - w_1 \rangle + \frac{\mu}{2} \|w_2 - w_1\|_2^2$

proposition : sous ces hypothèses :

- $\|\nabla f(w)\|_2^2 \geq 2\mu(f(w) - f^*)$
- $\|\nabla f(w)\|_2 \geq \mu \|w - w^*\|_2$ où $w^* = \underset{w}{\operatorname{argmin}} f(w)$ et $f^* = f(w^*)$

Théorème : Sous H 3.1, 3.2, 3.3, on a :

$$E(f(w_k) - f(w^*)) \leq \underbrace{\frac{\alpha L \sigma^2}{2\mu}}_{\not\rightarrow 0} + (1-\alpha\mu)^k \left(f(w_0) - f^* - \frac{\alpha L \sigma^2}{2\mu} \right)$$

(ici on a pris $\alpha_k = \alpha$ où $\alpha \leq \frac{1}{L}$)

On a $1 \geq \alpha\mu > 0$. Donc le terme $(1-\alpha\mu)^k \xrightarrow[k \rightarrow \infty]{} 0$

Conclusion : si on choisit bien le pas, la descente de gradient stochastique "converge" dans une boule de rayon $\frac{\alpha L \sigma^2}{\mu}$ autour de w^* puis elle fait une marche aléatoire dans cette boule.

Dans une descente de gradient ordinaire, le rayon de la boule est μL .

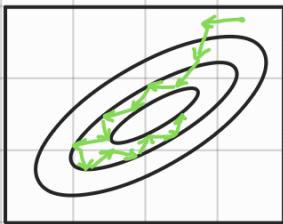
Rappel : descente du gradient stochastique $w_{k+1} = w_k - \frac{\alpha_k}{|\mathcal{I}_k|} \sum_{i \in \mathcal{I}_k} \nabla f_i(w_k)$ (1)

Soit $f = \sum f_i$

Si $f \in \mathcal{C}^1$, ∇f L -lipschitz ($\|\nabla f(w)\| \leq L \quad \forall w$)

f μ -fortement convexe ($Hf(w) \geq \mu I + w$)

Alors (1) "converge" : $E[f(x_k) - f^*] \leq \frac{\alpha\sigma^2}{2} \frac{L}{\mu} + (1-\alpha)\mu^{-k} C$



→ converge bien au début puis

Convergence avec pas décroissants :

Théorème : Sous les hypothèses précédentes, si :

$$\sum_{k=0}^{+\infty} \alpha_k = +\infty \text{ et } \sum_{k=0}^{+\infty} \alpha_k^2 < +\infty \text{ alors } w_k \xrightarrow[k \rightarrow +\infty]{} w^* \text{ (le minimiseur unique global)}$$

(exemple : $\alpha_k = \frac{1}{k}$ est optimal en opti stochastique)

Théorème : Si $\alpha_k = \frac{\beta}{k+r}$ $\beta \geq \frac{1}{\mu}$ $\alpha_0 = \frac{\beta}{r} \leq \frac{1}{L}$

$$E[f(x_k) - f^*] \leq \frac{\gamma}{r+k} \quad \gamma \text{ constante}$$

Sous les mêmes hypothèses, la descente de gradient converge linéairement (i.e beaucoup + rapidement)

$$f(x_k) - f^* \leq (1-\mu\alpha)^k [f(w_0) - f^*]$$

Cas non convexe

Rappel : une descente de gradient dans le cas non convexe assure la convergence critique avec des taux du type :

$$\min_{0 \leq k \leq K-1} \|\nabla f(w_k)\|_2 = O\left(\frac{1}{\sqrt{K}}\right)$$

Théorème Sous les hypothèses 3.1 et 3.2 et $0 \leq \alpha_k = \alpha \leq \frac{1}{L}$

$$\mathbb{E} \left[\frac{1}{K} \sum_{k=0}^{K-1} \|\nabla f(w_k)\|_2^2 \right] \leq \frac{\alpha L}{\mu} \sigma^2 + \frac{\mathbb{E} f(w_0) - f(w^*)}{\alpha K}$$

Note les règles de décaissement analytique (e.g. $\alpha_k = \frac{1}{k}$) sont peu utilisées en pratique. On leur préfère des méthodes adaptatives e.g. on mesure une moyenne glissante de $f(w_k)$ lorsque cette moyenne stagne, on divise le pas par 2.

Accélération et généralisation

Pour réduire le terme de variance $\alpha \sigma^2 \frac{L}{\mu}$, plusieurs stratégies sont possibles :

1) élargir la taille du batch m_b $\mathbb{E}[f(w_k) - f^*] \leq \frac{\alpha \sigma^2}{m_b} \frac{L}{\mu} + (1 - \alpha \mu)^k C$

2) Il existe des méthodes avancées (e.g. SAGA, SVRG) qui vont essayer d'évaluer le gradient complet à partir du gradient stochastique évalué en des points ≠ aux itérations précédentes.

3) Moyenner les itérés

Pour éviter les phénomènes d'oscillation quand on arrive proche du minimiseur on peut moyenner les itérés, i.e. calculer :

$$\hat{w}_k = \sum_{l=0}^k \lambda_l w_l, \quad \lambda_l \geq 0 \quad \sum_{l=0}^k \lambda_l = 1$$

On va favoriser des suites λ_k croissantes pour donner + de poids aux derniers itérés

ex: $\lambda_k = 1 - k + K \rightarrow$ gradient stochastique standard.

$\lambda_k = \frac{1}{k}$ moyenne de tous les gradients

$\lambda_k = c_k f^{k-k}$ favorise les derniers itérés

Recherche de pas

On peut effectuer des recherches de pas dans le cadre stochastique.

Par exemple, on peut effectuer une recherche linéaire sur le pas α , pour satisfaire la condition d'Armijo:

$$f_{ik}(w_k - \alpha \nabla f_{ik}(w_k)) \leq f_{ik}(w_k) - c\alpha \|\nabla f_{ik}(w_k)\|_2^2$$

avec c petit e.g. $c=0,1$

Préconditionnement diagonal

Comme en optimisation déterministe, on peut changer la métrique

$$w_{k+1} = w_k - M_k \nabla f_{ik}(w_k) \text{ où } M_k \text{ est une matrice SDP.}$$

Si $M_k = \text{Id}$, on obtient la descente de gradient standard.

Si $M_k = H f_{ik}(w_k)^{-1}$ on obtient la méthode de Newton stochastique.

(pas implémentable car Hf est en général trop coûteux)

Un exemple populaire en optimisation stochastique est la méthode

RMS Prop pour Root Mean Square propagation.

Dans cette méthode on va chercher à normaliser l'amplitude des composantes du gradient

$$\text{On pose } A_j^{(k)} = (1-\lambda) A_j^{(k-1)} + \lambda (g_j^{(k)})^2$$

avec $g_j^{(k)}$: gradient stochastique à l'itération

$$\text{eg } g_j^{(k)} = \begin{cases} \nabla f_{ik}(w_k) \\ \nabla f_{ik}(w_R) \end{cases} \quad \text{On choisir } \lambda \in [0, 1]$$

$$w_j^{(k+1)} = w_j^{(k)} - \frac{\alpha}{\sqrt{A_j^{(k)}} + \eta} g_j^{(k)} \quad \text{où } \eta \geq 0 \text{ param de régularisation.}$$

Par exemple : Si $\lambda=1$ et $\eta=0$

$$w_j^{(k+1)} = w_j^{(k)} - \alpha \frac{g_j^{(k)}}{|g_j^{(k)}|}$$

Méthodes inertielles

les méthodes inertielles peuvent être vues comme un système dynamique du 2nd ordre, alors que les descentes de gradient sont linéaires du 1^{er} ordre :

$$\ddot{w}(t) = F(w(t)) \text{ ou } F = \nabla f$$

contre $\ddot{w}(t) + \gamma \dot{w}(t) = F(w(t))$ (système inertiel)

Par ex: si $f = 0$ on retrouve la loi de Newton.

Après discréétisation, la méthode devient

$$w^{(k+1)} = w^{(k)} - \alpha \nabla f_k(w^{(k)}) + \beta (w^{(k)} - w^{(k-1)})$$

gradient choisi inertie

Quitte à réorganiser les termes:

$$w^{(k+1)} = w^{(k)} - \alpha \sum_{l=0}^{k-1} \beta^{k-l} \nabla f_k(w^{(l)})$$

On voit ici que la direction de descente est une combinaison linéaire des gradients aux itérations précédentes

Dans le cas déterministe convexe, ce genre de méthode permet de passer de taux de CV en $O(\frac{1}{K})$ à des taux optimaux en $O(\frac{1}{K^2})$

Exercice Soit (a_k) une suite

$$\text{On considère la suite: } m_k = \lambda m_{k-1} + (1-\lambda)a_k \quad m_0 = 0$$

Trouver l'expression de m_k en fonction de a_k

$$m_1 = (1-\lambda)a_1$$

$$\begin{aligned} m_2 &= (1-\lambda)\lambda a_1 + (1-\lambda)a_2 \\ &= (1-\lambda)(a_2 + \lambda a_1) \end{aligned}$$

$$m_k = (1-\lambda) \sum_{l=1}^k \lambda^{k-l} a_l$$

Pour que m_k représente une combinaison convexe (moyenne glissante)

des α_k il faut normaliser m_k par :

$$(1-\lambda) \sum_{k=1}^K \lambda^{k-k} = (1-\lambda) \frac{1-\lambda^K}{1-\lambda} = 1-\lambda^K$$

Donc $\frac{m_k}{1-\lambda^K}$ est une moyenne glissante des α_k .

"moralement" Si $\lambda = 0,9$ λ^k pour quel k est ce négligeable? $\frac{1}{1-\lambda}$
On effectue une moyenne glissante sur 10 elt
 $\lambda = 0,99$ " 100 elt.

La méthode ADAM

param α learning rate

$\beta_1 = 0,9$ inertie du premier ordre

$\beta_2 = 0,99$ " second "

ϵ petit paramètre

$m_0 = 0$ moment du 1^{er} ordre

$N_0 = 0$ " 2nd "

for $k=0$ à N

$$g_k = \nabla f_{I_k}(\omega_k)$$

$$m_k = \beta_1 m_{k-1} + (1-\beta_1) g_k$$

$$N_k = \beta_2 N_{k-1} + (1-\beta_2) g_k^2 \quad (\text{coordonnée par coordonnée})$$

$$\hat{m}_k = m_k / (1-\beta_1)^k$$

$$\hat{N}_k = N_k / (1-\beta_2)^k$$

$$\omega_k = \omega_{k-1} - \alpha \hat{m}_k / \sqrt{\hat{N}_k + \epsilon}$$