Introduction à l'optimisation pour le machine learning

Rodolphe Le Riche[1], Dédji Brian Whannou[2], Espéran Padonou[3]

[1] CNRS LIMOS at Mines Saint-Etienne, France
[2] KPMG France
[3] Fondation Vallet

Juillet 2021
Ecole d'Eté en Intelligence Artificielle
fondation Vallet
Cotonou, Bénin

Plan du cours

Introduction à l'optimisation pour le machine learning

Objectifs du cours

- Donner des bases pour l'optimisation numérique
- en faisant le lien avec le machine learning
- pour un public de bac+1
- avec quelques exemples de programmes en R/python codés à partir de 0.
- Limites: les algorithmes ne seront pas exactement ceux utilisés en pratique pour le deep learning, mais les principaux concepts y seront.

Bibliographie du cours

Ce cours doit beaucoup à

- [Minoux, 2008] : un classique sur l'optimisation, écrit avant l'avènement du machine learning mais une vraie base (niveau bac+3)
- [Ravikumar and Singh, 2017] : présentation détaillée des algorithmes d'optimisation utiles en machine learning (niveau bac+3)
- [Bishop, 2006] : un excellent livre d'introduction au machine learning avec quelques commentaires sur l'optimisation (niveau bac+3)
- [Schmidt et al., 2007] : techniques pour la régularisation L1 (article de recherche)
- [Sun, 2019] : panorama des méthodes d'optimisation pour les réseaux de neurones, rétro-propagation de gradient (article de recherche)

Nous en simplifierons le contenu et emprunterons des illustrations.

Optimisation = formalisation de la décision

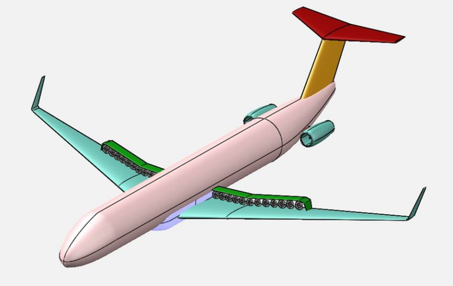
L'optimisation est une[1] formalisation mathématique de la décision

$$\min_{x \in \mathcal{S}} f(x)$$

- x vecteur des paramètres de la décision : dimensions, somme investie, réglage d'une machine/code, ...
- $f(x)$: coût de la décision x
- $\mathcal{S}$: ensemble des valeurs possibles de x, espace de recherche

[1] non unique, contestable par rapport à l'humain et la vie

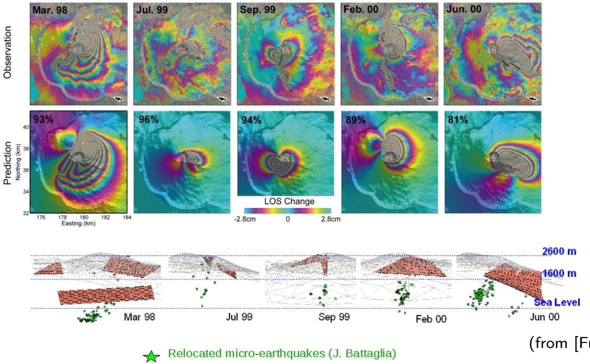
Optimization example: design



(from [Sgueglia et al., 2018])

x = aircraft parameters (here distributed electrical propulsion)
$f() = -1\times$ performance metric (agregation of $-1\times$ range, cost, take-off length, ...)
At the minimum, the design is "optimal".

Optimization example: model identification

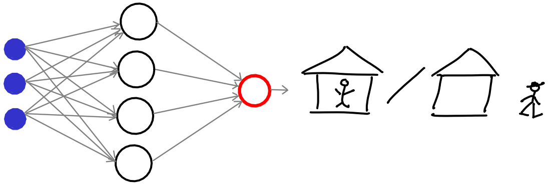


(from [Fukushima et al., 2010])

x = dike position, geometry, internal pressure
$f()$ = distance between measures (from RADARSAT-1 satellite) and model (boundary elements, non trivial computation)
At the minimum, the model best matches measurements and should correspond to the underground phenomenon.

Optimization example: neural net classification

Predict if a person stays at home or goes out based on longitude, latitude and temperature = a 2 classes classification problem.



x = neural network (NN) weights and biases
$f()$ = an error of the NN predictions (a cross-entropy error):

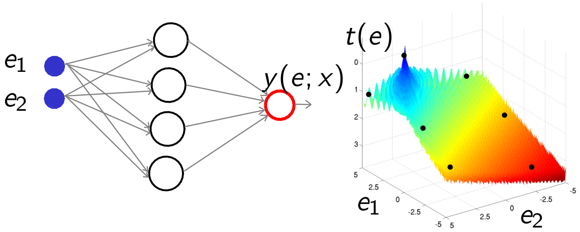
- e entries: e_1 longitude, e_2 latitude, e_3 temperature
- $t = 1$ if person stays, $t = 0$ otherwise
- Observed data set: (e^i, t^i), $i = 1, \dots, N$
- $y(e; x)$: output of the NN, the probability that $t(e) = 1$
- $f(x) = -\sum_{i=1}^{N} \{t^i \log(y(e^i; x)) + (1 - t^i) \log(1 - y(e^i; x))\}$

- View the relationship between the entry e and the class t as probabilistic (generalizes deterministic functions): $t(e)$ is a Bernoulli variable with a given probability that $t(e) = 1$
- The NN models this probability: $y(e; x)$ is the probability that $t(e) = 1$, $1 - y(e; x)$ is the proba that $t(e) = 0$, $0 \leq y(e; x) \leq 1$.
- The probability of t knowing e can be written $y(e; x)^t + (1 - y(e; x))^{1-t}$
- The likelihood of the N i.i.d observations is $\prod_{i=1}^{N} \left[y(e^i; x)^{t^i} + (1 - y(e^i; x))^{1-t^i} \right]$, to be maximized
- The likelihood is turned into an error, to be minimized, by taking $-\log(\text{likelihood})$,

$$f(x) = -\sum_{i=1}^{N} \{ t^i \log(y(e^i; x)) + (1 - t^i) \log(1 - y(e^i; x)) \}$$

Optimization example: neural net regression

learn a function from a discrete limited set of observations



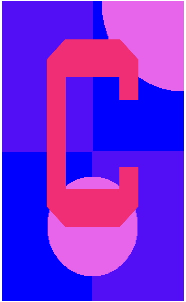
x = neural network (NN) weights and biases
$f()$ = an error of the NN predictions (sum-of-squares error):

- e entries, $t(e)$ target function to learn
- observed data set, "." : (e^i, t^i), $i = 1, \dots, N$
- $y(e;x)$: output of the NN, the expected value of $t(e)$
- $f(x) = 1/2 \sum_{i=1}^{N} (t^i - y(e^i;x))^2$

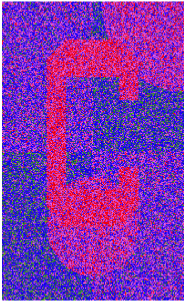
Optimization example: image denoising

$$\min_x f(x) \quad , \quad f(x) = \frac{1}{2} \sum_{i=1}^{N_{\text{pixels}}} (y_i - x_i)^2 + \lambda \sum_{i=1}^{N_{\text{pixels}}} \sum_{j \text{ near } i} |x_i - x_j|$$

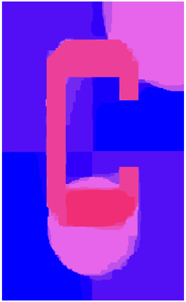
$\lambda \geq 0$ regularization constant



target image



noisy (observed) $= y_i$'s



denoised (optimized) $= x^\star$

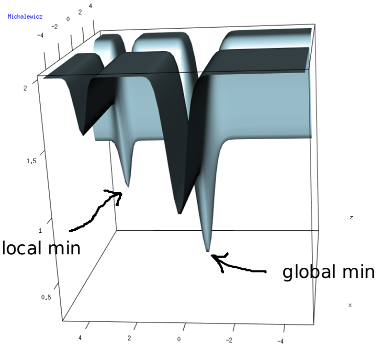
(from [Ravikumar and Singh, 2017])

Optimization basics

1. Introduction
   - Objectifs, remerciements
   - Formulation d'un problème d'optimisation
   - Examples of optimization usages
   - Optimization basics
2. Steepest descent
   - Algorithm
   - Application neural network
3. Improved gradient searches
4. Prise en compte des contraintes d'optimisation
5. Conclusions
6. Bibliography

Local versus global optimum

$$\min_{x \in \mathcal{S} \subset \mathbb{R}^d} f(x)$$

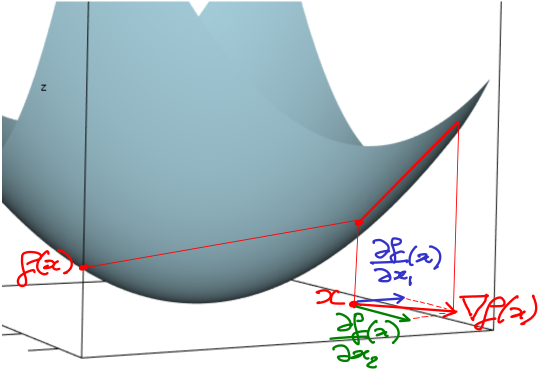


R code to generate the plot given in the project folder

Gradient of a function

Gradient of a function = direction of steepest ascent = vector of partial derivatives

$$\nabla f(x) = \begin{pmatrix} \frac{\partial f}{\partial x_1}(x) \\ \dots \\ \frac{\partial f}{\partial x_d}(x) \end{pmatrix}$$



Numerical approximation of the gradient

By forward finite differences

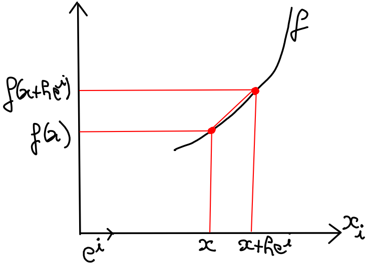
$$\frac{\partial f}{\partial x_i} f(x) \approx \frac{f(x+he^i) - f(x)}{h}$$

Proof: by Taylor,

$f(x+he^i) = f(x) + he^{i\top}\nabla f(x) + h^2/2 e^{i\top}\nabla^2 f(x+\rho h e^i) e^i$, $\rho \in]0,1[$

$\nabla f(x) = \frac{f(x+he^i)-f(x)}{h} - h/2 e^{i\top}\nabla^2 f(x+\rho h e^i) e^i$

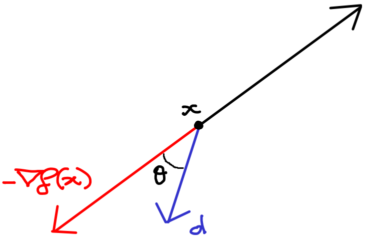
and make h very small $\square$



Other (better but more difficult to implement) schemes: central differences, automatic differentiation (e.g., in TensorFlow or PyTorch), (semi-)analytic differentiation (e.g., backpropagation in NN).

Descent direction

A search direction d which makes an acute angle with $-\nabla f(x)$ is a descent direction, i.e., for a small enough step f is guaranteed to decrease!

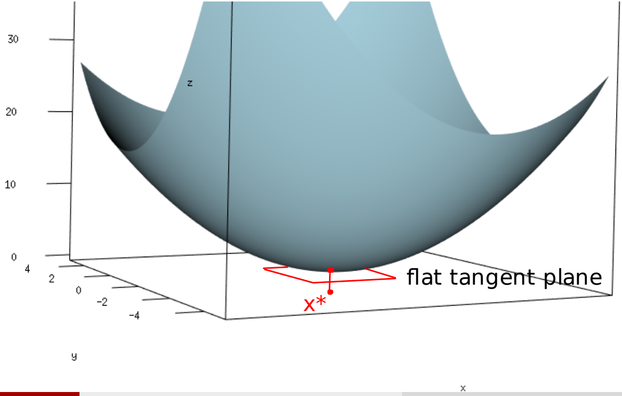


Proof: by Taylor, $\forall \alpha \leq 0,\ \exists \epsilon \in [0,1]$ such that
$f(x+\alpha d) = f(x) + \alpha d^\top \nabla f(x) + \frac{\alpha^2}{2} d^\top \nabla^2 f(x+\alpha\epsilon d) d$
$\lim_{\alpha \to 0^+} \frac{f(x+\alpha d)-f(x)}{\alpha} = d^\top \nabla f(x) = -1 \times \|\nabla f(x)\| \cos(d, -\nabla f(x))$
is negative if the cosine is positive $\square$

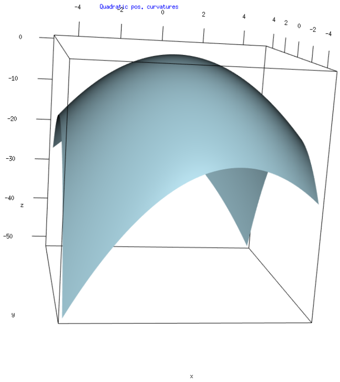
Necessary optimality condition (1)

A necessary condition for a differentiable function to have a minimum at $x^\star$ is that it is flat at this point, i.e., its gradient is null

$$x^\star \in \arg\min_{x \in \mathcal{S}} f(x) \Rightarrow \nabla f(x^\star) = 0$$

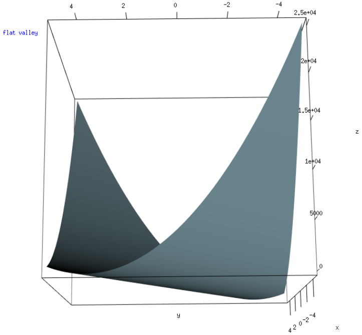


Necessary optimality condition (2)



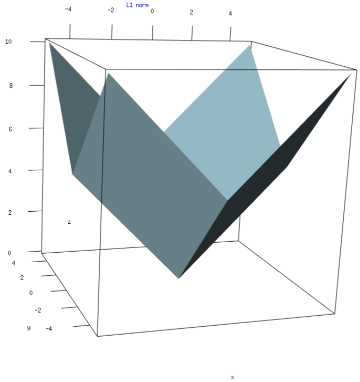
necessary is not sufficient (works with a max)

Necessary optimality condition (3)



$\nabla f(x^\star) = 0$ does not make $x^\star$ unique (flat valley)

Necessary optimality condition (4)



$\nabla f()$ not defined everywhere, example with L1 norm $= \sum_i^d |x_i|$

Plan du cours

An introduction to optimization for machine learning

Steepest descent algorithm

$$\min_{x \in \mathcal{S}} f(x)$$

(work in progress)present algo

Comments on steepest descent

(work in progress)comment on bound limits, handled with gradient projection

show that (with perfect line search) consecutive search directions are perpendicular: tendency to oscillate, sensitive to bad conditionning

other flaws: no convergence on nondifferentiable functions, gets trapped in local minima

Plan du cours

An introduction to optimization for machine learning

Plan du cours

Introduction à l'optimisation pour le machine learning

Conclusions

- L'optimisation numérique est une technique fondamentale associée à la décision optimale et à la modélisation statistique (machine learning).
- Avec l'enthousiasme autour du machine learning, de nombreux algorithmes ont été conçus que nous n'avons pas couverts ici: l'optimisation bayésienne (Bayesian optimization) pour le réglage des hyper-paramètres (paramètres de régularisation, nombre de couches du réseau de neurone, type de neurones, paramètres de l'algorithme d'optimisation des poids).

References I

Bishop, C. M. (2006).
Pattern recognition and machine learning.

Fukushima, Y., Cayol, V., Durand, P., and Massonnet, D. (2010).
Evolution of magma conduits during the 1998–2000 eruptions of piton de la fournaise volcano, réunion island.
Journal of Geophysical Research: Solid Earth, 115(B10).

Minoux, M. (2008).
Programmation mathématique. Théorie et algorithmes.
Lavoisier.

Ravikumar, P. and Singh, A. (2017).
Convex optimization.
http://www.cs.cmu.edu/~pradeepr/convexopt/.

Schmidt, M., Fung, G., and Rosales, R. (2007).
Fast optimization methods for l1 regularization: A comparative study and two new approaches.
In *European Conference on Machine Learning*, pages 286–297. Springer.

References II

Sgueglia, A., Schmollgruber, P., Bartoli, N., Atinault, O., Benard, E., and Morlier, J. (2018).
Exploration and sizing of a large passenger aircraft with distributed ducted electric fans.
In *2018 AIAA Aerospace Sciences Meeting*, page 1745.

Sun, R. (2019).
Optimization for deep learning: theory and algorithms.
arXiv preprint arXiv:1912.08957.