IASD
App

OPTIMIZATION FOR MACHINE LEARNING

October 5, 2023

Today: Last lecture of the first block

Second block: November 8 - November 16

NONSMOOTH OPTIMIZATION AND REGULARIZATION

what can we do when we don't have gradients?

How can we guarantee that a solution has a specific structure?

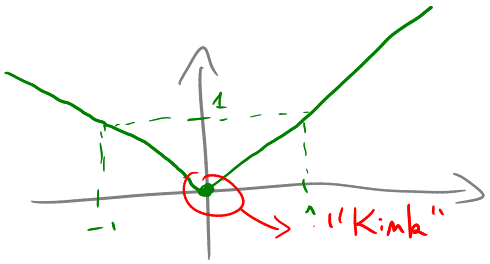
(1) Optimizing without gradients

minimize $f(w)$ $f: \mathbb{R}^d \to \mathbb{R}$
$w \in \mathbb{R}^d$

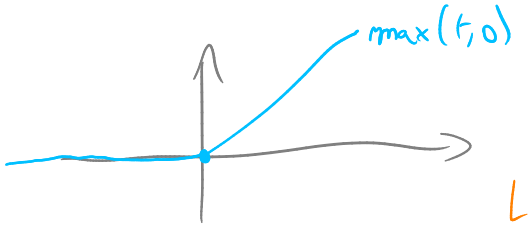
→ So far we have assumed that f was C^1

→ But many problems involve functions that are not C^1 and that do not even have a gradient at certain points

Ex) $t \mapsto |t|$ in $\mathbb{R}$
does not have a derivative (i.e. a gradient) at $t=0$



$t \mapsto \mathrm{ReLU}(t) = \max(t,0)$

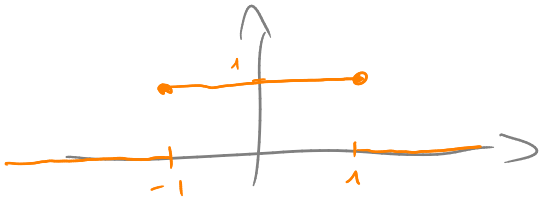


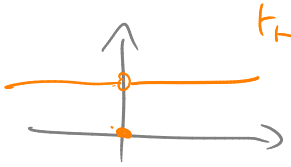
L-smooth $= C^{1,1}_L$
smooth $\equiv C^1$

Def (for this course) A function $f: \mathbb{R}^d \to \mathbb{R}$ is nonsmooth if there exists a vector $w \in \mathbb{R}^d$ at which f does not have a gradient

⚠ A nonsmooth function need not be discontinuous
(Ex: $t \mapsto |t|$)

$t \mapsto \begin{cases} 1 \text{ if } |t| \le 1 \\ 0 \text{ otherwise} \end{cases}$





$t \mapsto \begin{cases} 0 \text{ if } t=0 \\ 1 \text{ otherwise} \end{cases}$

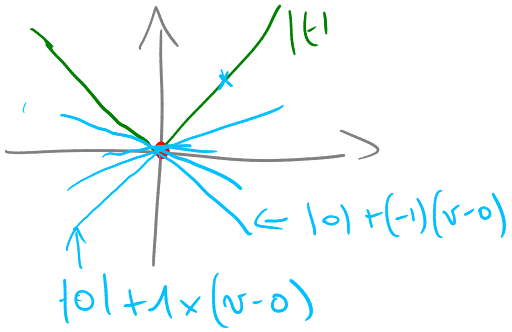
↳ We will focus on convex nonsmooth functions, that are necessarily continuous

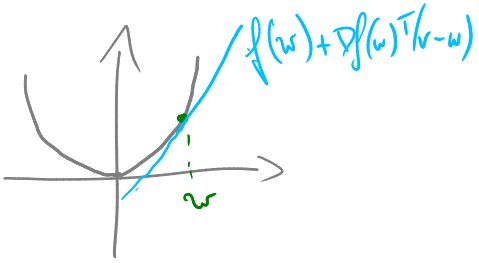
Def: Let $f: \mathbb{R}^d \to \mathbb{R}$ be convex (hence continuous) function. and let $w \in \mathbb{R}^d$.

A vector $g \in \mathbb{R}^d$ is called a subgradient of f at w if $\forall v \in \mathbb{R}^d$, $f(v) \geq f(w) + g^T(v-w)$.

The set of all subgradients of f at w is called the subdifferential of f at w, and is denoted by $\partial f(w) \subseteq \mathbb{R}^d$

↳ For C^1 convex f, we have $f(v) \geq f(w) + \nabla f(w)^T(v-w)$ $\forall v \in \mathbb{R}^d$
In that case, $\partial f(w) = \{\nabla f(w)\}$.





$\partial |\cdot|(0) = [-1, 1]$
$\partial |\cdot|(1) = \{1\}$ ⇒ If f has a gradient at w, then $\partial f(w) = \{\nabla f(w)\}$

Th: If $f: \mathbb{R}^d \to \mathbb{R}$ is a nonsmooth convex function, then

$[\bar{w} \in \mathbb{R}^d$ is a global minimum of $f] \iff [0_{\mathbb{R}^d} \in \partial f(\bar{w})]$

↳ Generalization of the C^1 setting, equivalent when f is C^1 ($\partial f(\bar{w}) = \{\nabla f(\bar{w})\}$ when f is C^1)

↳ Using the notion of subgradient, we can build a class of algorithms for minimizing convex nonsmooth functions. ⇒ Subgradient methods

Basic subgradient method iteration:

$$\forall k \geq 0, \quad w_{k+1} = w_k - \alpha_k g_k \quad \text{where } g_k \in \partial f(w_k)$$

$$\alpha_k > 0$$

⇒ This is the GD iteration when $\partial f(w_k) = \{\nabla f(w_k)\}$!

Some results about subgradient methods

↳ Does not always work : depends on the choice of subgradient g_k!

Ex) $f(w) = |w|$ in dimension 1 $\quad w_k = 0 \quad \alpha_k > 0$

$\partial f(w_k) = [-1, 1]$

If $g_k = 1 \in \partial f(w_k)$, then $w_{k+1} < 0$

$|w_{k+1}| > |w_k| = 0$

↳ A good strategy for choosing g_k:

$$g_k \in \underset{g \in \mathbb{R}^d}{\operatorname{argmin}} \{ \|g\| \mid g \in \partial f(w_k) \}$$

⊕ If $0 \in \partial f(w_k)$, then $g_k = 0$

⊖ This is an optimization problem to be solved and it may require to compute the entire subdifferential ⇒ possibly expensive

↳ Possible to derive convergence rates for subgradient methods

Recall: For f C^1 convex $\quad f(w_K) - \min_{w \in \mathbb{R}^d} f(w) \leq O\left(\frac{1}{K}\right)$

after K iterations of GD

Ex) For f convex nonsmooth, after $K \geq 1$ iterations of the subgradient method with well-chosen subgradients, then

$$f\left(\frac{1}{\sum_{k=0}^{K}\alpha_k}\sum_{k=0}^{K}\alpha_k w_k\right) - \min_{w\in\mathbb{R}^d} f(w) \leq O\left(\frac{1}{\sqrt{K}}\right)$$

slower than the GD rate

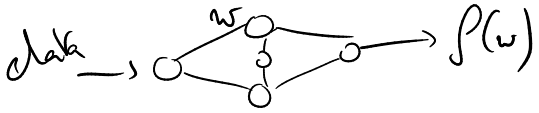
↓ weighted average of the iterates (if $\alpha_k = \alpha > 0$, that is $\frac{1}{K+1}\sum_{k=0}^{K} w_k$)

Computing gradients (or subgradients)

- Most deep learning libraries rely on automatic differentiation to compute gradients (or subgradients)
- Typical architecture:

Automatic differentiation (AD)
If you have a code that computes $f(w)$, you can apply AD to compute $\nabla f(w)$ at the cost of at most 5 times that of $f(w)$

→ Forward pass to evaluate $f(w)$



→ Reverse/Backward pass to compute $\nabla f(w)$ without re-accessing the data

$w \mapsto b \mapsto a \mapsto f$

Key concept: Chain rule

$$\frac{\partial f}{\partial w} = \frac{\partial f}{\partial a}\frac{\partial a}{\partial b}\frac{\partial b}{\partial w}$$

Example: $d=1$, $f(w) = (1 + w^2 e^{-w})^{1/3}$ Goal: Compute $\nabla f(0) = f'(0)$

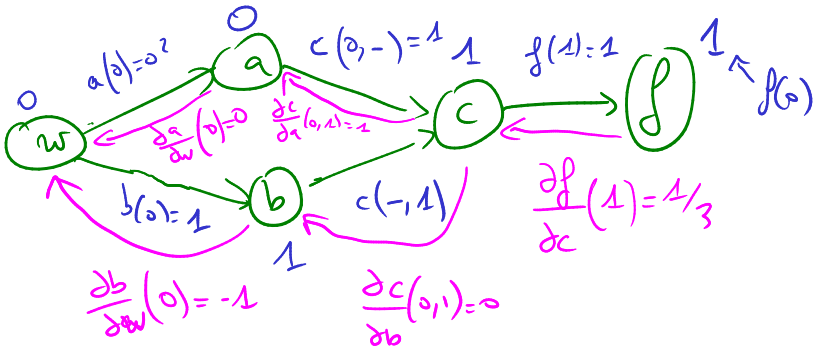
1) Build a computational graph/network to compute $f(w)$

$a = w^2$

$b = e^w$

$c = 1 + ab$

$f = c^{1/3}$



- Forward pass $f(0)$
- Backward pass $f'(0)$

→ what you code to evaluate f

← generated automatically by automatic differentiation

$$\frac{\partial f}{\partial w}(0) = \frac{\partial f}{\partial c}(\ )\left(\frac{\partial c}{\partial b}(\ )\frac{\partial b}{\partial w}(\ ) + \frac{\partial c}{\partial a}(\ )\frac{\partial a}{\partial w}(\ )\right)$$

Takeaway:

→ there exists software that, given a code to compute $f(w)$, constructs automatically a code to compute $\nabla f(w)$

→ But recently (2021!), researchers found that the way this is done in neural networks does not lead to useful subgradient values for optimization

(2) Regularization

↳ minimize $f(w)$ where $f: \mathbb{R}^d \to \mathbb{R}$ depends on some data
$w \in \mathbb{R}^d$

Ex) $f(w) = \frac{1}{2n} \|Xw - y\|^2$

X: data, $X \in \mathbb{R}^{n \times d}$, $y \in \mathbb{R}^n$

↳ So far we have seen examples of data-fitting problems

↳ In ML, the goal is generally not limited to "fit the data"

Other objectives:
- generalization: model should apply to unseen data
- simplicity: model should have as few parameters as possible
- robustness: model should be robust to outliers

⇒ In optimization, one way to incorporate these objectives is through regularization

Regularized optimization problem

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) + \lambda \Omega(w)$$

λ: regularization parameter > 0, controls the balance between data-fitting and regularization

$f(w)$: data-fitting term, typically a finite sum over data points (e.g. $\frac{1}{n} \sum_{i=1}^{n} (x_i^T w - y_i)^2$)

$\Omega: \mathbb{R}^d \to \mathbb{R}$ regularization term, penalizes vectors that do not satisfy desirable properties

↳ Ω represents some "soft" constraints on w

minimize $f(w)$ s.t. $w = 0$ ⟹ minimize $f(w) + \frac{1}{2} \|w\|^2$

hard constraints — soft constraint/penalty

Important example $\Omega(w) = \frac{1}{2}\|w\|^2 = \frac{1}{2}\sum_{j=1}^{d} w_j^2$

ℓ_2 regularization / ridge regularization / Tychonov regularization/etc

Aka batch normalization in the context of stochastic gradient

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) + \frac{\lambda}{2}\|w\|^2 \quad \text{for } \lambda > 0$$

- If f convex, then the regularized objective is λ-strongly convex
  $\Rightarrow$ 1 unique global minimum
  (often helpful for generalization purposes)
- As $\lambda \to \infty$, the problem becomes closer to $\underset{w}{\text{minimize}} \frac{\lambda}{2}\|w\|^2$ and $\operatorname{argmin}(\ ) \to \{0\}$
- More broadly, this regularization penalizes solutions of the UN-regularized problem that are large in norm
- It reduces the variance of the solution(s) with respect to the data

Sparsity and regularization

→ Sparse models ($w \in \mathbb{R}^d$ with many zero components) are often desirable because they are simple and even more interpretable

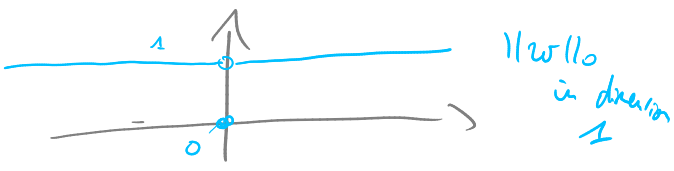
→ One model for sparsity

$$\underset{w \in \mathbb{R}^d}{\text{minimize}} \quad f(w) + \lambda \|w\|_0$$

ℓ_0 regularizer

$\|w\|_0$ = cardinality (number of elements) $= \left| \{ j \in \{1, \dots, d\} \mid w_j \neq 0 \} \right|$

$\|w\|_0$ is nonconvex
discontinuous (nonsmooth)
combinatorial



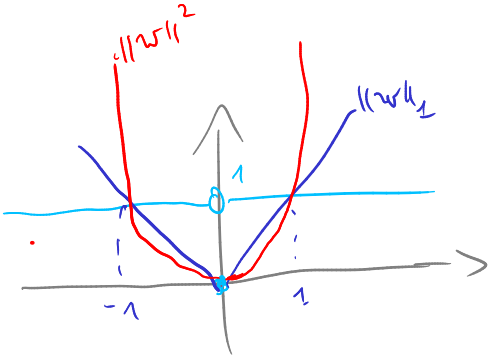
$\Rightarrow$ Hard to optimize

↳ A more tractable formulation uses a convex approximation of $\|w\|_0$ called the ℓ_1 norm

minimize $_{w \in \mathbb{R}^d}$ $f(w) + \lambda \|w\|_1$ → ℓ_1 regularization, LASSO regularization

$$\|w\|_1 = \sum_{j=1}^{d} |w_j|$$

continuous, convex, nonsmooth



- As $\lambda \nearrow$, solutions to the problem become sparser
- Unlike ℓ_2 regularization, in which all components become small but nonzero as $\lambda \nearrow$, for ℓ_1 regularization the number of zero components increases

(3) Proximal gradient methods for regularized optimization

Setup: minimize $_{w \in \mathbb{R}^d}$ $f(w) + \lambda \Omega(w)$ $\quad \lambda > 0$

where $f : \mathbb{R}^d \to \mathbb{R}$ is C^1 (smooth)

and $\Omega : \mathbb{R}^d \to \mathbb{R}$ is convex, possibly nonsmooth

PG

Proximal gradient iteration

$$w_{k+1} \in \underset{w \in \mathbb{R}^d}{\operatorname{argmin}} \left\{ f(w_k) + \nabla f(w_k)^T (w - w_k) + \frac{1}{2\alpha_k} \|w - w_k\|^2 + \lambda \Omega(w) \right\}$$

$f(w_k) + \nabla f(w_k)^T (w - w_k) \approx f(w)$ around w_k

$\frac{1}{2\alpha_k} \|w - w_k\|^2$: Proximal term: penalizes vectors that are far from w_k

$\lambda \Omega(w)$: left untouched

Every iteration of PG solves an auxiliary optimization problem (a subproblem)

→ If $\Omega(w) = 0 \quad \forall w \in \mathbb{R}^d$, then you can show that w_{k+1} is uniquely defined and that $w_{k+1} = w_k - \alpha_k \nabla f(w_k)$ GD

→ the PG subproblem can be solved explicitly for classical choices of Ω regardless of the value of f

Ex) $\Omega(w) = \|w\|_1$

PG subproblem $w_{k+1} \in \operatorname{argmin}_w \left\{ f(w_k) + \nabla f(w_k)^T (w - w_k) + \frac{1}{2\alpha_k} \|w - w_k\|^2 + \lambda \|w\|_1 \right\}$

has a unique solution defined coordinatewise by

$$\forall j = 1..d, \quad [w_{k+1}]_j = \begin{cases} [w_k - \alpha_k \nabla f(w_k)]_j - \lambda \alpha_k & \text{if } [w_k - \alpha_k \nabla f(w_k)]_j > \lambda \alpha_k \\ [w_k - \alpha_k \nabla f(w_k)]_j + \lambda \alpha_k & \text{if } [w_k - \alpha_k \nabla f(w_k)]_j < -\lambda \alpha_k \\ 0 & \text{otherwise} \end{cases}$$

⇒ In that particular case, proximal gradient iterations have a closed form and the method is called ISTA in image and signal processing (Iterative Soft-thresholding Algorithm)

NB: ISTA + acceleration on convex problems ⇒ Fast ISTA (FISTA)