Optimization methods

Unconstrained convex optimization

Unconstrained convex optimization problem

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \quad f(\boldsymbol{w})$$

(f is the convex objective function; feasible region is $\mathbb{R}^d$).

Gradient descent for differentiable objectives

- Start with some initial $\boldsymbol{w}^{(1)} \in \mathbb{R}^d$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.
  - Compute gradient of f at $\boldsymbol{w}^{(t)}$:
    $$\boldsymbol{\lambda}^{(t)} := \nabla f(\boldsymbol{w}^{(t)}) .$$
  - Update:
    $$\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} - \eta_t \boldsymbol{\lambda}^{(t)} .$$

($\eta_t > 0$ are step sizes.)

(Projected) (sub)gradient descent

Two issues

1. Objective function not necessarily differentiable everywhere.
2. Feasible region not necessarily $\mathbb{R}^d$.

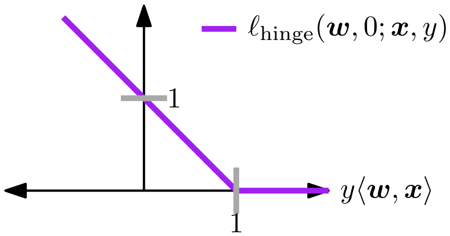
Non-differentiability

Non-differentiable convex objectives

Some convex functions f are not differentiable everywhere;
gradient descent not even well-specified for these problems.

Example: hinge loss

$$f(\boldsymbol{w}) \;=\; \ell_{\text{hinge}}(\boldsymbol{w}, 0; \boldsymbol{x}, y) \;=\; \Big[1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle\Big]_+ .$$



Not differentiable at $\boldsymbol{w} \in \mathbb{R}^d$ where $y\langle \boldsymbol{w}, \boldsymbol{x}\rangle = 1$.

Dealing with non-differentiability

Local optimization

- Locally change $\boldsymbol{x} \to \boldsymbol{x} + \boldsymbol{\delta}$.
- Hopefully improve objective value $f(\boldsymbol{x}) \to f(\boldsymbol{x} + \boldsymbol{\delta})$.

Subgradients

We say $\boldsymbol{\lambda} \in \mathbb{R}^d$ is a **subgradient** of a function $f \colon \mathbb{R}^d \to \mathbb{R}$ at $\boldsymbol{w}_0 \in \mathbb{R}^d$ if

$$f(\boldsymbol{w}) \;\geq\; f(\boldsymbol{w}_0) + \langle \boldsymbol{\lambda}, \boldsymbol{w} - \boldsymbol{w}_0\rangle \quad \text{for all } \boldsymbol{w} \in \mathbb{R}^d ,$$

i.e., $\boldsymbol{\lambda}$ specifies an **affine lower bound** on the function.

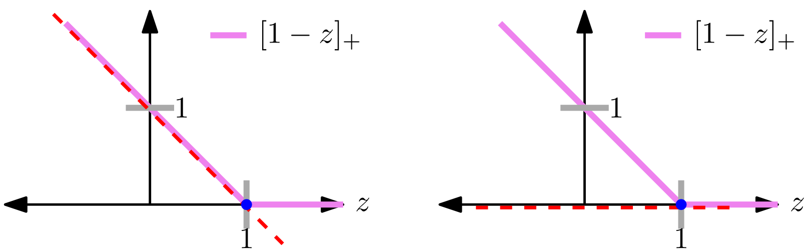
Although not every function f is differentiable everywhere,
every **convex function** f has **subgradients** *everywhere*.
(Some technical conditions apply.)

Call the entire set the **subdifferential of f at w_0**; denote it by

$$\partial f(\boldsymbol{w}_0) \;:=\; \{\boldsymbol{\lambda} \in \mathbb{R}^d : \boldsymbol{\lambda} \text{ is a subgradient of } f \text{ at } \boldsymbol{w}_0\} .$$

Example: max of two affine functions

Consider one-dimensional function $f(z) := [1 - z]_+ = \max\{0, 1 - z\}$.



Two subgradients of f at $z = 1$: -1 and 0.

$$\begin{aligned} f(z) \;&\geq\; f(1) + (-1) \cdot (z - 1) \;=\; 1 - z \;; \\ f(z) \;&\geq\; f(1) + (0) \cdot (z - 1) \;=\; 0 \,. \end{aligned}$$

In fact, every $\lambda \in [-1, 0]$ satisfies

$$f(z) \;\geq\; f(1) + \lambda \cdot (z - 1) \,.$$

We have $\partial f(1) = [-1, 0]$.

Subgradient descent

Subgradient descent for general convex objectives

- Start with some initial $\boldsymbol{w}^{(1)} \in \mathbb{R}^d$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.
  - Compute *any* subgradient $\boldsymbol{\lambda}^{(t)} \in \partial f(\boldsymbol{w}^{(t)})$.
  - Update:
$$\boldsymbol{w}^{(t+1)} \;:=\; \boldsymbol{w}^{(t)} - \eta_t \boldsymbol{\lambda}^{(t)} \,.$$

($\eta_t > 0$ are step sizes.)

Example: soft-margin (homogeneous) SVM

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} f(\boldsymbol{w}), \qquad f(\boldsymbol{w}) := \frac{\lambda}{2}\|\boldsymbol{w}\|_2^2 + \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S} \left[1 - y\langle \boldsymbol{w}, \boldsymbol{x}\rangle\right]_+$$

Question: How do we compute *a* subgradient $\boldsymbol{g}$ of f at a given point $\boldsymbol{w}_0 \in \mathbb{R}^d$?

$$\boldsymbol{g} = \lambda \boldsymbol{w}_0 + \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S} \begin{cases} \boldsymbol{0} & \text{if } 1 - y\langle \boldsymbol{w}_0, \boldsymbol{x}\rangle < 0\,; \\ -y\boldsymbol{x} & \text{if } 1 - y\langle \boldsymbol{w}_0, \boldsymbol{x}\rangle > 0\,; \\ \text{any conv. comb. of above} & \text{if } 1 - y\langle \boldsymbol{w}_0, \boldsymbol{x}\rangle = 0\,. \end{cases}$$

With "continuous" data, exact equalities like $1 - y\langle \boldsymbol{w}_0, \boldsymbol{x}\rangle = 0$ never come up.

Example: soft-margin (homogeneous) SVM

Subgradient descent algorithm for soft-margin SVM:

- Start with some initial $\boldsymbol{w}^{(1)} \in \mathbb{R}^d$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.

$$\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} - \eta_t \left(\lambda \boldsymbol{w}^{(t)} + \frac{1}{|S|}\sum_{(\boldsymbol{x},y)\in S} \begin{cases} \boldsymbol{0} & \text{if } 1 - y\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}\rangle < 0\,; \\ -y\boldsymbol{x} & \text{if } 1 - y\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}\rangle \geq 0 \end{cases} \right)$$

$$= (1 - \lambda\eta_t)\boldsymbol{w}^{(t)} + \eta_t \frac{1}{|S|} \sum_{\substack{(\boldsymbol{x},y)\in S: \\ y\langle \boldsymbol{w}^{(t)}, \boldsymbol{x}\rangle \leq 1}} y\boldsymbol{x}\,.$$

Note effect of regularization term $\frac{\lambda}{2}\|\boldsymbol{w}\|_2^2$ (whenever $\eta_t < 1/\lambda$):
Shrink $\boldsymbol{w}^{(t)}$ by a factor $1 - \lambda\eta_t$ before updating with subgradient of loss term.
Helps prevent $\boldsymbol{w}^{(t)}$ from becoming too long.

Constrained convex optimization

$$\begin{aligned} \min_{\boldsymbol{w}\in\mathbb{R}^d} \quad & f(\boldsymbol{w}) \\ \text{s.t.} \quad & \boldsymbol{w} \in A \end{aligned}$$

for convex function f and convex feasible set A (possibly a strict subset of $\mathbb{R}^d$).

Projected subgradient descent

- Start with some initial $\boldsymbol{w}^{(1)} \in A$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.
  - Compute *any* subgradient $\boldsymbol{\lambda}^{(t)} \in \partial f(\boldsymbol{w}^{(t)})$.
  - Update:

    $$\boldsymbol{w}^{(t+0.5)} := \boldsymbol{w}^{(t)} - \eta_t \boldsymbol{\lambda}^{(t)}\,.$$

  - Project to feasible region A:

    $$\boldsymbol{w}^{(t+1)} := \mathrm{Proj}_A(\boldsymbol{w}^{(t+0.5)})\,.$$

Projection onto convex set A: $\mathrm{Proj}_A(\boldsymbol{w}) := \arg\min_{\boldsymbol{w}'\in A} \|\boldsymbol{w} - \boldsymbol{w}'\|_2^2$
(another convex optimization problem, but hopefully simpler objective!).

Example: box constraints

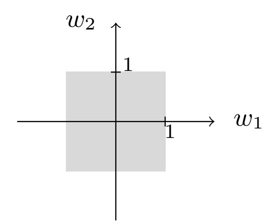
Suppose you want to prevent any feature from having a high weight:

- Constraints:

  $$|w_i| \leq 1 \quad \text{for all } i = 1, 2, \ldots, d\,.$$

- Feasible region:

  $$A := [-1, +1]^d$$



- $\mathrm{Proj}_A(\boldsymbol{w})$:
  - For each $i = 1, 2, \ldots, d$:
    - If $w_i \in [-1, +1]$, leave w_i alone.
    - If $w_i > +1$, set $w_i := 1$.
    - If $w_i < -1$, set $w_i := -1$.

Convergence and optimality conditions

Convergence of (projected) subgradient descent

In general, subgradient descent (with possibly non-differentiable convex objectives) can converge relatively slowly.

(Can help to return the *average* of the iterates $\frac{1}{T}\sum_{t=1}^{T} \boldsymbol{w}^{(t)}$.)

Optimality condition for general convex optimization problems also considerably more involved (e.g., more than just "$\nabla f_0(\boldsymbol{w}) = \mathbf{0}$").

But for machine learning purposes, we can use alternative criteria for stopping (e.g., hold-out error rate).

Stochastic gradient method

Minimizing sums of convex functions

In machine learning, we typically have objectives of the following form:

$$F(\boldsymbol{w}) \;:=\; \frac{1}{n}\sum_{i=1}^{n} \phi_i(\boldsymbol{w})$$

for convex functions $\phi_i \colon \mathbb{R}^d \to \mathbb{R}$ for $i = 1, 2, \dots, n$.

Example: ϕ_i is loss on i-th training example.

Computational cost of computing $\nabla F(\boldsymbol{w})$: linear in n. (Typically $O(nd)$).

Stochastic gradients

Key idea

Let I be a uniform random variable in $\{1, 2, \dots, n\}$.
Then for any $\boldsymbol{w}$,

$$F(\boldsymbol{w}) \;=\; \frac{1}{n}\sum_{i=1}^{n} \phi_i(\boldsymbol{w}) \;=\; \mathbb{E}\big[\phi_I(\boldsymbol{w})\big]$$

and by linearity,

$$\nabla F(\boldsymbol{w}) \;=\; \nabla\left\{\frac{1}{n}\sum_{i=1}^{n} \phi_i(\boldsymbol{w})\right\} \;=\; \frac{1}{n}\sum_{i=1}^{n} \nabla\phi_i(\boldsymbol{w}) \;=\; \mathbb{E}\big[\nabla\phi_I(\boldsymbol{w})\big]\,.$$

Upshot: $\nabla\phi_I(\boldsymbol{w})$ is an *unbiased* estimate of gradient of F at $\boldsymbol{w}$.

Computational cost of computing $\nabla\phi_I(\boldsymbol{w})$: **independent of** n. (E.g., $O(d)$).

Stochastic gradient method

Consider convex functions $\phi_i \colon \mathbb{R}^d \to \mathbb{R}$, $i = 1, 2, \ldots, n$;

$$\min_{\boldsymbol{w} \in A} F(\boldsymbol{w}), \qquad F(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} \phi_i(\boldsymbol{w}).$$

Stochastic (projected) (sub)gradient "descent" method

- Start with some initial $\boldsymbol{w}^{(1)} \in A$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.
  - Pick I_t uniformly at random from $\{1, 2, \ldots, n\}$.
  - Compute *any* subgradient $\boldsymbol{\lambda}^{(t)} \in \partial \phi_{I_t}(\boldsymbol{w}^{(t)})$.
  - Update:
    $$\boldsymbol{w}^{(t+0.5)} := \boldsymbol{w}^{(t)} - \eta_t \boldsymbol{\lambda}^{(t)}.$$
  - Project to feasible region A:
    $$\boldsymbol{w}^{(t+1)} := \operatorname{Proj}_A(\boldsymbol{w}^{(t+0.5)}).$$

Example: logistic regression

Maximum likelihood estimation for logistic regression:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \ln\Big(1 + \exp\big(-y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle\big)\Big).$$

Stochastic gradient method for logistic regression:

- Start with some initial $\boldsymbol{w}^{(1)} \in \mathbb{R}^d$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.
  - Pick I_t uniformly at random from $\{1, 2, \ldots, n\}$;
    $$\boldsymbol{w}^{(t+1)} := \boldsymbol{w}^{(t)} + \eta_t \big(1 - P_{\boldsymbol{w}^{(t)}}(Y = y_{I_t} \mid \boldsymbol{X} = \boldsymbol{x}_{I_t})\big) y_{I_t} \boldsymbol{x}_{I_t}.$$

Example: soft-margin (homogeneous) SVMs

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} \quad \frac{1}{n} \sum_{i=1}^{n} \left(\frac{\lambda}{2} \|\boldsymbol{w}\|_2^2 + \big[1 - y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle\big]_+ \right)$$

Stochastic gradient method for soft-margin SVMs:

- Start with some initial $\boldsymbol{w}^{(1)} \in \mathbb{R}^d$.
- For $t = 1, 2, \ldots$ until some stopping condition is satisfied.
  - Pick I_t uniformly at random from $\{1, 2, \ldots, n\}$;

$$\begin{aligned}
\boldsymbol{w}^{(t+1)} &:= \boldsymbol{w}^{(t)} - \eta_t \left(\lambda \boldsymbol{w}^{(t)} + \begin{cases} \boldsymbol{0} & \text{if } 1 - y_{I_t} \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_{I_t} \rangle < 0; \\ -y_{I_t} \boldsymbol{x}_{I_t} & \text{if } 1 - y_{I_t} \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_{I_t} \rangle \geq 0 \end{cases} \right) \\
&= (1 - \lambda \eta_t) \boldsymbol{w}^{(t)} + \begin{cases} \boldsymbol{0} & \text{if } y_{I_t} \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_{I_t} \rangle > 1; \\ y_{I_t} \boldsymbol{x}_{I_t} & \text{if } y_{I_t} \langle \boldsymbol{w}^{(t)}, \boldsymbol{x}_{I_t} \rangle \leq 1. \end{cases}
\end{aligned}$$

Why should this work?

Consider differentiable convex functions $\phi_i \colon \mathbb{R}^d \to \mathbb{R}$, $i = 1, 2, \ldots, n$:

$$\min_{\boldsymbol{w} \in \mathbb{R}^d} F(\boldsymbol{w}), \qquad F(\boldsymbol{w}) := \frac{1}{n} \sum_{i=1}^{n} \phi_i(\boldsymbol{w}).$$

Then

$$\boldsymbol{\lambda}^{(t)} = \nabla \phi_{I_t}(\boldsymbol{w}^{(t)}) = \nabla F(\boldsymbol{w}^{(t)}) + \underbrace{\Big(\nabla \phi_{I_t}(\boldsymbol{w}^{(t)}) - \mathbb{E}\big[\nabla \phi_{I_t}(\boldsymbol{w}^{(t)})\big] \Big)}_{\text{mean-zero "noise"}}.$$

"Noise" is "averaged away" when step sizes η_t are small (e.g., $\eta_t \sim 1/\sqrt{t}$).

Convergence theory

Consider convex functions $\phi_i \colon \mathbb{R}^d \to \mathbb{R}$, $i = 1, 2, \dots, n$;

$$\min_{\boldsymbol{w} \in A} F(\boldsymbol{w}) , \qquad F(\boldsymbol{w}) \ := \ \frac{1}{n} \sum_{i=1}^{n} \phi_i(\boldsymbol{w}) .$$

Run stochastic gradient method for T iterations, get iterates $\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_{T+1} \in A$.

Under some suitable conditions (e.g., on step sizes η_t),

$$\mathbb{E}\big[F(\boldsymbol{w}_{T+1})\big] \ \leq \ \min_{\boldsymbol{w} \in A} F(\boldsymbol{w}) + O\left(\frac{\log T}{\sqrt{T}}\right) .$$

Let $\bar{\boldsymbol{w}} := \frac{1}{T+1} \sum_{t=1}^{T+1} \boldsymbol{w}_t$ (average of the iterates). Under same conditions,

$$\mathbb{E}\big[F(\bar{\boldsymbol{w}})\big] \ \leq \ \min_{\boldsymbol{w} \in A} F(\boldsymbol{w}) + O\left(\frac{1}{\sqrt{T}}\right) .$$

Often really helps in practice!

Efficient implementation trick

Special case:

- $\phi_i(\boldsymbol{w}) = \lambda \|\boldsymbol{w}\|_2^2/2 + \ell(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle)$ for some convex function $\ell \colon \mathbb{R} \to \mathbb{R}$.
- $\boldsymbol{x}_i$ has few non-zero entries (i.e., $\boldsymbol{x}_i$ is "sparse").

Gradient of ϕ_i at $\boldsymbol{w}$:

$$\nabla \phi_i(\boldsymbol{w}) \ = \ \lambda \boldsymbol{w} + \ell'(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \, y_i \boldsymbol{x}_i .$$

Update ($I_t = i$):

$$\boldsymbol{w}_{t+1} \ := \ (1 - \lambda \eta_t) \boldsymbol{w}_t - \eta_t \ell'(y_i \langle \boldsymbol{w}, \boldsymbol{x}_i \rangle) \, y_i \boldsymbol{x}_i .$$

- Represent weight vector $\boldsymbol{w}_t$ as $c_t \boldsymbol{v}_t$ for scalar $c_t \in \mathbb{R}$ and vector $\boldsymbol{v}_t \in \mathbb{R}^d$.
- Sparsity-respecting update:

$$\begin{aligned} g_t &:= \ell'(y_t c_t \langle \boldsymbol{v}_t, \boldsymbol{x}_i \rangle) \\ c_{t+1} &:= (1 - \lambda \eta_t) c_t \\ \boldsymbol{v}_{t+1} &:= \boldsymbol{v}_t - \frac{\eta_t g_t}{c_{t+1}} y_i \boldsymbol{x}_i \end{aligned}$$

Other practical tips

- Pick $I_1, I_2, \dots, I_t$ u.a.r. **without replacement** from $\{1, 2, \dots, n\}$.
  I.e., randomly shuffle order of training data, then process in that order.
  If possible, re-shuffle after each pass through training data.
- For debugging purposes, periodically check overall objective value.
  (Should generally be improving, though not necessarily at every step.)
- Use hold-out error rate as a stopping criterion.
  (Stop when hold-out error rate does not improve after a while.)
- Use hold-out set to tune step sizes.

Other solvers

Many other algorithms for solving convex optimization problems

- *Newton-Raphson*: use Hessian to pick better descent directions.
- *Quasi-Newton methods* (e.g., conjugate gradient, "BFGS", "L-BFGS"): use efficient approximations of Hessians.
- Techniques for dealing with constraints:
  - *Barrier methods*: add penalties for constraint violations, slowly relax.
  - *Primal-dual methods*: start with dual-feasible point, iteratively improve until corresponding primal point is feasible.
  - ...
- ...

But remember: end goal in machine learning is *not* to minimize training error rate or training surrogate loss.

Key takeaways

1. Concept of subgradients, and generalizing gradient descent to non-differentiable objectives.
2. Subgradients for maximum of affine functions.
3. Extending gradient descent to constrained optimization problems.
4. Concept of stochastic gradients; stochastic gradient descent method; high-level idea of convergence theory.
5. Efficient implementation for sparse gradients.