Google Research

Beyond gradient descent

Mathieu Blondel

December 3, 2020

Outline

Measuring progress

- How to measure the progress made by an iterative algorithm for solving an optimization problem?

$$x^\star = \operatorname*{argmin}_{x \in \mathcal{X}} f(x)$$

- Non-negative error measure

$$E_t = ||x_t - x^\star|| \quad \text{or} \quad E_t = f(x_t) - f(x^\star)$$

- Progress ratio

$$\rho_t = \frac{E_t}{E_{t-1}}$$

- An algorithm makes strict progress on iteration t if $\rho_t \in [0, 1)$.

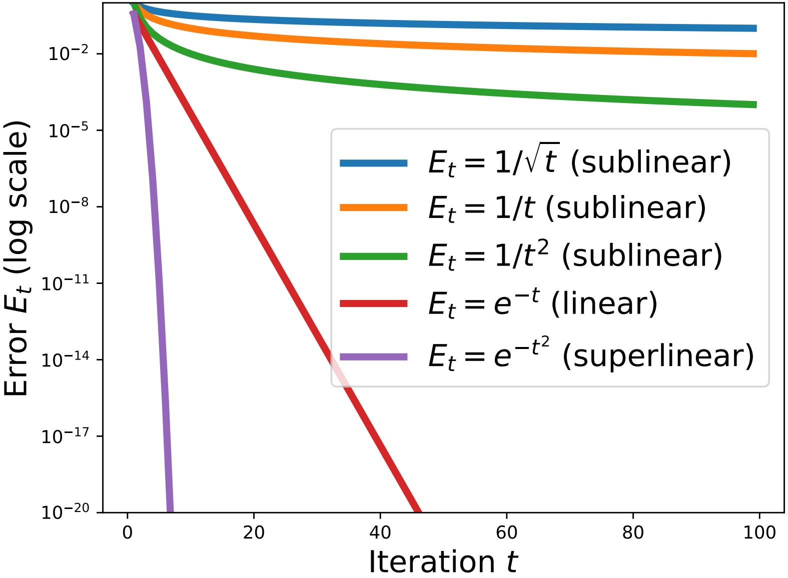
Types of convergence rates

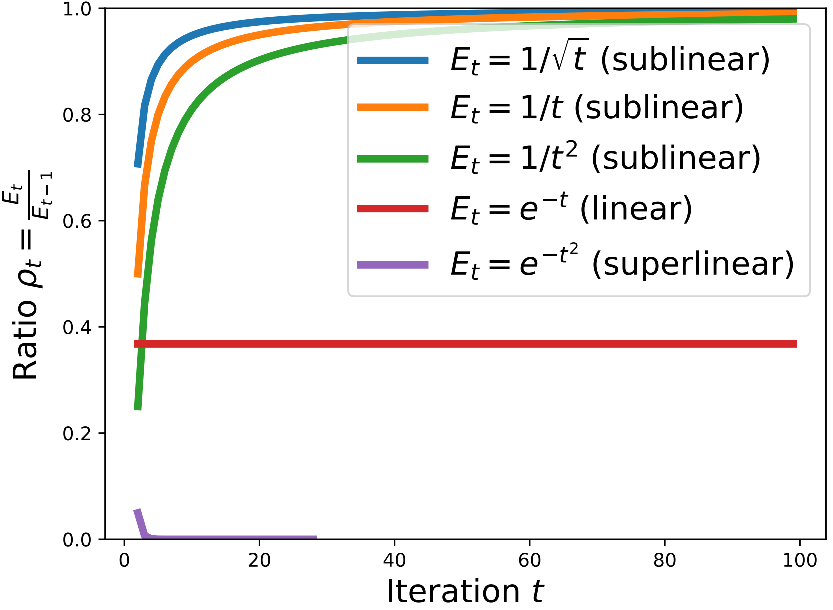
- Asymptotic convergence rate

$$\lim_{t \to \infty} \rho_t = \rho$$

i.e., the sequence $\rho_1, \rho_2, \rho_3, \ldots$ converges to ρ

- Sublinear rate: $\rho = 1$. The longer the algorithm runs, the slower it makes progress! (the algorithm decelerates over time)
- Linear rate: $\rho \in (0, 1)$. The algorithm eventually reaches a state of constant progress.
- Superlinear rate: $\rho = 0$. The longer the algorithm runs, the faster it makes progress! (the algorithm accelerates over time)


Error E_t (log scale)
10^{-2}
10^{-5}
10^{-8}
10^{-11}
10^{-14}
10^{-17}
10^{-20}
0
20
40
60
80
100
Iteration t
$E_t = 1/\sqrt{t}$ (sublinear)
$E_t = 1/t$ (sublinear)
$E_t = 1/t^2$ (sublinear)
$E_t = e^{-t}$ (linear)
$E_t = e^{-t^2}$ (superlinear)



$E_t = 1/\sqrt{t}$ (sublinear)
$E_t = 1/t$ (sublinear)
$E_t = 1/t^2$ (sublinear)
$E_t = e^{-t}$ (linear)
$E_t = e^{-t^2}$ (superlinear)
Ratio $\rho_t = \frac{E_t}{E_{t-1}}$
Iteration t
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100

Sublinear rates

- Error at iteration E_t, number of iterations to reach ε-error T_ε

$$E_t = O\left(\frac{1}{t^b}\right) \Leftrightarrow T_\varepsilon = O\left(\frac{1}{\varepsilon^{1/b}}\right) \quad b > 0$$

- $b = 1/2$

$$E_t = O\left(\frac{1}{\sqrt{t}}\right) \Leftrightarrow T_\varepsilon = O\left(\frac{1}{\varepsilon^2}\right)$$

- $b = 1$

$$E_t = O\left(\frac{1}{t}\right) \Leftrightarrow T_\varepsilon = O\left(\frac{1}{\varepsilon}\right)$$

  ex: gradient descent for smooth but not strongly-convex functions

- $b = 2$

$$E_t = O\left(\frac{1}{t^2}\right) \Leftrightarrow T_\varepsilon = O\left(\frac{1}{\sqrt{\varepsilon}}\right)$$

  ex: Nesterov's method for smooth but not strongly-convex functions

Linear rates

- The iteration number t now appears in the exponent

$$E_t = O(\rho^t) \qquad \rho \in (0, 1)$$

- Example:

$$E_t = O\left(e^{-t}\right) \Leftrightarrow T_\varepsilon = O\left(\log \frac{1}{\varepsilon}\right)$$

- "Linear rate" is kind of a misnomer: E_t is decreasing exponentially fast! On the other hand, $\log E_t$ is decreasing linearly.

- Ex: gradient descent on smooth and strongly convex functions

Superlinear rates

- We can further classify the order q of convergence rates

$$\lim_{t \to \infty} \frac{E_t}{(E_{t-1})^q} = M$$

- Superlinear ($q = 1$, $M = 0$)

$$E_t = O\left(e^{-t^k}\right) \Leftrightarrow T_\varepsilon = O\left(\log \frac{1}{\varepsilon}\right)^{1/k}$$

- Quadratic ($q = 2$)

$$E_t = O\left(e^{-2^t}\right) \Leftrightarrow T_\varepsilon = O\left(\log \log \frac{1}{\varepsilon}\right)$$

Outline

Coordinate descent

- Minimize only one variable per iteration, keeping all others fixed
- Well-suited if the one-variable sub-problem is easy to solve
- Cheap iteration cost
- Easy to implement
- No need for step size tuning
- State-of-the-art on the lasso, SVM dual, (non-negative) matrix factorization
- Block coordinate descent: minimize only a block of variables

Coordinate-wise minimizer

- A point is called coordinate-wise minimizer of f if f is minimized along all coordinates separately

$$f(x + \delta e_j) \geq f(x) \quad \forall \delta \in \mathbb{R}, j \in \{1, \ldots, d\}$$

- Does the coordinate-wise minimizer coincide with the global minimizer?

- f convex and differentiable: **yes**

$$\nabla f(x) = 0 \Leftrightarrow \nabla_j f(x) = 0 \quad j \in \{1, \ldots, d\}$$

- f convex but non-differentiable: **not always.** Coordinate descent can get stuck

- $f(x) = g(x) + \sum_{j=1}^{d} h_j(x_j)$ where g is differentiable but h_j is not: **yes**

$$0 \in \partial f(x) \Leftrightarrow -\nabla_j g(x) \in \partial h_j(x_j) \quad j \in \{1, \ldots, d\}$$

Coordinate descent

- On each iteration t, pick a coordinate $j_t \in \{1, \ldots, d\}$ and minimize (approximately) this coordinate while keeping others fixed

$$\min_{x_{j_t}} f(x_1^t, x_2^t, \ldots, x_{j_t}, \ldots, x_{d-1}^t, x_d^t)$$

- Coordinate selection strategies: random, cyclic, shuffled cyclic.
- Coordinate descent with exact updates: requires an “oracle” to solve the sub-problem
- Coordinate gradient descent: only requires first-order information (and sometimes a prox operator)

Coordinate descent with exact updates

- Suppose f is a quadratic function. Then

$$f(x+\delta e_j) = f(x) + \nabla_j f(x)\delta + \frac{\delta^2}{2}\nabla_{jj}^2 f(x)$$

- Minimizing w.r.t. δ, we get

$$\delta^\star = -\frac{\nabla_j f(x)}{\nabla_{jj}^2 f(x)} \Leftrightarrow x_j \leftarrow x_j - \frac{\nabla_j f(x)}{\nabla_{jj}^2 f(x)}$$

- Example: $f(x) = \frac{1}{2}\|Ax-b\|_2^2 + \frac{\lambda}{2}\|x\|_2^2$

$$\nabla f(x) = A^\top(Ax-b) + \lambda x \quad \Rightarrow \quad \nabla_j f(x) = A_{:,j}^\top(Ax-b) + \lambda x_j$$

$$\nabla^2 f(x) = A^\top A + \lambda I \quad \Rightarrow \quad \nabla_{jj}^2 f(x) = \|A_{:,j}\|_2^2 + \lambda$$

Coordinate descent with exact updates

- Computing $\nabla f(x)$ for gradient descent costs $O(nd)$ time
- Let us maintain the residual vector $r = Ax - b \in \mathbb{R}^n$
- When x_j is updated, synchronizing r takes $O(n)$ time
- When r in synchronized, we can compute $\nabla_j f(x)$ in $O(n)$ time
- The second derivatives $\nabla^2_{jj} f(x)$ can be pre-computed ahead of time, since it does not depend on x
- Doing a pass on all d coordinates therefore takes $O(nd)$ time, just like one iteration of gradient descent

Coordinate descent with exact updates

- If $f(x) = g(x) + \sum_{j=1}^d h_j(x_j)$ and g is quadratic, then

$$f(x + \delta e_j) = g(x) + \nabla_j g(x)\delta + \frac{\delta^2}{2}\nabla^2_{jj} g(x) + h_j(x_j + \delta)$$

- The closed form solution is

$$\delta^\star = \mathrm{prox}_{\frac{\lambda}{\nabla^2_{jj} f(x)} h_j}\left(x_j - \frac{\nabla_j g(x)}{\nabla^2_{jj} g(x)}\right) - x_j$$

  where we used the proximity operator

$$\mathrm{prox}_{\tau h_j}(u) = \operatorname*{argmin}_v \frac{1}{2}(u - v)^2 + \tau h_j(v)$$

- If $h_j = |\cdot|$, then prox is the soft-thresholding operator.
- State-of-the-art for solving the lasso!

Coordinate gradient descent

- If f is not quadratic, there typically does not exist a closed form
- If $\nabla_j f(x)$ is L_j-Lipschitz-continuous, recall that $\nabla_{jj}^2 f(x) \leq L_j$
- Key idea: replace $\nabla_{jj}^2 f(x)$ with L_j, i.e.,

$$x_j \leftarrow x_j - \frac{\nabla_j f(x)}{\nabla_{jj}^2 f(x)}$$

  becomes

$$x_j \leftarrow x_j - \frac{\nabla_j f(x)}{L_j}$$

- Each L_j is coordinate-specific (easier to derive and tighter than a global constant L)
- Convergence: $O(1/\varepsilon)$ under Lipschitz gradient and $O(\log(1/\varepsilon))$ under strong convexity (random or cyclic selection)

Coordinate gradient descent with prox

- Similarly, we can replace

$$x_j \leftarrow \text{prox}_{\frac{\lambda}{\nabla^2_{jj} g(x)} h_j}\left(x_j - \frac{\nabla_j g(x)}{\nabla^2_{jj} g(x)}\right)$$

  with

$$x_j \leftarrow \text{prox}_{\frac{\lambda}{L_j} h_j}\left(x_j - \frac{\nabla_j g(x)}{L_j}\right)$$

  where $\nabla^2_{jj} g(x) \leq L_j$ for all x

- Can be used for instance for L_1-regularized logistic regression
- If $h_j(x_j) = I_{\mathcal{C}_j}(x_j)$, where $\mathcal{C}_j$ is a convex set, then the prox becomes the projection onto $\mathcal{C}_j$.

Implementation techniques

- Synchronize “statistics” (e.g. residuals) upon each update
- Column-major format: Fortran-style array or sparse CSC matrix
- Regularization path and warm-start

$$\lambda_1 > \lambda_2 > \cdots > \lambda_m$$

  Since CD converges faster with big λ, start from λ_1, use solution to warm-start (initialize) λ_2, etc.

- Active set, safe screening: use optimality conditions to safely discard coordinates that are guaranteed to be 0

Outline

Newton's method for root finding

- Given a function g, find x such that $g(x) = 0$
- Such x is called a root of g
- Newton's method in one dimension:

$$x^{t+1} = x^t - \frac{g(x^t)}{g'(x^t)}$$

- Newton's method in d dimensions:

$$x^{t+1} = x^t - J_g(x^t)^{-1} g(x^t)$$

  where $J_g(x^t) \in \mathbb{R}^{d \times d}$ is the Jacobian matrix of $g \colon \mathbb{R}^d \to \mathbb{R}^d$

- The method may fail to converge to a root

Newton's method for optimization

- If we want to minimize f, we can set $g = f'$ or $g = \nabla f$
- Newton's method in one dimension:

$$x^{t+1} = x^t - \frac{f'(x^t)}{f''(x^t)}$$

- Newton's method in d dimensions:

$$x^{t+1} = x^t - \underbrace{\nabla^2 f(x^t)^{-1} \nabla f(x^t)}_{d^t}$$

- In practice, once solves the linear system of equations

$$\nabla^2 f(x^t) d^t = \nabla f(x^t)$$

- If f is non-convex, $\nabla^2 f(x^t)$ is indefinite (i.e., not psd)

Line search

- If f is a quadratic, Newton's method converges in one iteration
- Otherwise, Newton's method may not converge, even if f is convex
- Solution: use a step size

$$x^{t+1} = x^t - \eta^t d^t$$

- Backtracking line search: decrease η^t geometrically until $\eta^t d^t$ satisfies some conditions
- Examples: Armijo rule, strong Wolfe conditions
- Superlinear local convergence

Trust region methods

- Newton's method

$$x^{t+1} = x^t - \underbrace{\nabla^2 f(x^t)^{-1} \nabla f(x^t)}_{d^t}$$

is equivalent to solving a quadratic approximation of f around x^t

$$-d^t = \operatorname*{argmin}_d f(x^t) + \nabla f(x^t)^\top d + \frac{1}{2} d^\top \nabla^2 f(x^t) d$$

- Trust region method: add a ball constraint

$$-d^t = \operatorname*{argmin}_d f(x^t) + \nabla f(x^t)^\top d + \frac{1}{2} d^\top \nabla^2 f(x^t) d \quad \text{s.t.} \quad \|d\|_2 \leq \delta^t$$

- If $x^t - d^t$ increases f, reject the solution and decrease δ^t
- Similar convergence guarantees as line search methods

Hessian-free method

- If d (number of dimensions) is large, computing the Hessian matrix is expensive
- The conjugate gradient (CG) method can be used to solve $Ax = b$
- It only requires to know how to multiply with A, not A directly
- Since $A = \nabla^2 f(x^t)$, we need to multiply with the Hessian
- This can be done in a number of ways: manual derivation, finite difference, autodiff (cf. autodiff lecture)
- The resulting algorithm (with line search) is called Newton-CG

Sub-sampled Hessians

- In machine learning, f is often an average (finite expectation)

$$f(x) = \frac{1}{n}\sum_{i=1}^{n} f_i(x)$$

- On iteration t we can sub-sample a set $S^t \subseteq \{1, \ldots, n\}$ to compute unbiased estimates of the gradient and Hessian

$$\nabla f(x) \approx \frac{1}{|S^t|}\sum_{i \in S^t} \nabla f_i(x^t)$$

$$\nabla^2 f(x) \approx \frac{1}{|S^t|}\sum_{i \in S^t} \nabla^2 f_i(x^t)$$

- Can be combined with Hessian-free method

Quasi-Newton methods

- BFGS: replaces

$$x^{t+1} = x^t - \nabla^2 f(x^t)^{-1} \nabla f(x^t)$$

  with

$$x^{t+1} = x^t - H^t \nabla f(x^t)$$

  where $H^{t+1} \approx \nabla f(x^{t+1})^{-1}$ is built incrementally from H^t, $d^t = x^{t+1} - x^t$ and $v^t = \nabla f(x^{t+1}) - \nabla f(x^t)$ using the so-called secant equation

- L-BFGS: keep a history of only m pairs (d^t, v^t) and compute $H^t \nabla f(x^t)$ on the fly without materializing H^t in memory

- Local superlinear convergence rate

- One of the go-to algorithms in machine learning!

Outline

Projected gradient descent

- Consider the constrained optimization problem

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

where f is L-smooth convex and $\mathcal{C}$ is a closed convex set

- Projected gradient descent

$$x^{t+1} = P_{\mathcal{C}}\left(x^t - \frac{1}{L}\nabla f(x^t)\right)$$

where

$$P_{\mathcal{C}}(x) = \underset{y \in \mathcal{C}}{\operatorname{argmin}} \|x - y\|_2^2$$

is the projection of x onto $\mathcal{C}$

Projected gradient descent

- Recall that gradient descent's update can be seen as solving a (crude) local quadratic approximation of f around x^t

$$x^{t+1} = x^t - \frac{1}{L}\nabla f(x^t) = \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{L}{2} \underbrace{(x - x^t)^\top I (x - x^t)}_{||x - x^t||_2^2}$$

- Similarly

$$\begin{aligned} x^{t+1} = P_{\mathcal{C}}(x^t - \frac{1}{L}\nabla f(x^t)) &= P_{\mathcal{C}} \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{L}{2}||x - x^t||_2^2 \\ &= \operatorname*{argmin}_{x \in \mathcal{C}} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{L}{2}||x - x^t||_2^2 \end{aligned}$$

Frank-Wolfe

- A method for constrained optimization
- Based on a linear approximation instead of a quadratic one
- Projection free: a linear minimization oracle (LMO) is needed instead
- LMOs are typically cheaper to compute than projections
- Also known as conditional gradient method (not to be confused with conjugate gradient method)

Frank-Wolfe

- Initialize $x^0 \in \mathcal{C}$
- For $t \in \{0, 1, 2, \dots\}$

$$s = \operatorname*{argmin}_{s \in \mathcal{C}} f(x^t) + \nabla f(x^t)^\top (s - x^t) = \operatorname*{argmin}_{s \in \mathcal{C}} \nabla f(x^t)^\top s$$

$$x^{t+1} = (1 - \gamma^t) x^t + \gamma^t s \qquad \gamma^t = \frac{2}{2 + t}$$

- $\operatorname{argmin}_{s \in \mathcal{C}} g^\top s$ is called linear minimization oracle (LMO)
- $\mathcal{C}$ needs to be compact (closed and bounded), otherwise the LMO problem is not feasible (solution goes to infinity)
- How to compute the LMO?

Convex hulls

- Probability simplex

$$\triangle^m = \{p \in \mathbb{R}^m \colon \sum_{i=1}^m p_i = 1, p_i \geq 0 \; i \in \{1, \ldots, m\}\}$$

- v is a convex combination of $\{v_1, \ldots, v_m\}$ if

$$v = \sum_{i=1}^m p_i v_i \quad \text{for some } p \in \triangle^m$$

- The convex hull of $\mathcal{S}$ is the set of all convex combinations of $\mathcal{S}$

$$\text{conv}(\mathcal{S}) = \left\{ \sum_{i=1}^m p_i v_i \colon m \in \mathbb{N}; p \in \triangle^m; v_1, \ldots, v_m \in \mathcal{S} \right\}$$

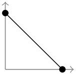
Convex polytopes

- A convex polytope is the convex hull of its vertices $V = \{v_1, \dots, v_m\}$

$$\mathcal{C} = \text{conv}(V)$$

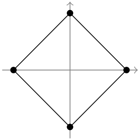
- Example 1: Probability simplex

$$\triangle^m = \text{conv}(\{e_1, \dots, e_m\})$$



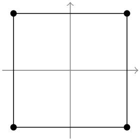
- Example 2: L_1-ball

$$\Diamond^m = \{x \colon \|x\|_1 \leq 1\} = \text{conv}(\{\pm e_1, \dots, \pm e_m\})$$



- Example 3: L_∞-ball

$$\square^m = \{x \colon \|x\|_\infty \leq 1\} = \text{conv}(\{-1, 1\}^m)$$



Linear minimization oracles

- If $\mathcal{C} = \text{conv}(V)$ where $V = \{v_1, \ldots, v_m\}$ then

$$\operatorname*{argmin}_{s \in \mathcal{C}} g^\top s \subseteq V$$

- Example 1: Probability simplex

$$e_i \in \operatorname*{argmin}_{s \in \triangle^m} g^\top s \qquad i \in \operatorname*{argmin}_j g_j$$

- Example 2: L_1-ball

$$\text{sign}(-g_i) e_i \in \operatorname*{argmin}_{s \in \Diamond^m} g^\top s \qquad i \in \operatorname*{argmax}_j |g_j|$$

- Example 3: L_∞-ball

$$\text{sign}(-g) \in \operatorname*{argmin}_{s \in \square^m} g^\top s$$

Example: sparse regression

- Consider the objective

$$\min_{\|w\|_1 \leq \tau} f(w) = \frac{1}{2}\|Xw - y\|_2^2 \qquad \nabla f(w) = X^\top (Xw - y)$$

- Initialize $w^0 = 0$
- For $t \in \{0, 1, 2, \dots\}$

$$g = \nabla f(w^t)$$

$$i \in \operatorname*{argmax}_j |g_j|$$

$$s = \tau \cdot \text{sign}(g_i) e_i$$

$$w^{t+1} = (1 - \gamma^t) w^t + \gamma^t s \qquad \gamma^t = \frac{2}{2+t}$$

- Pick a coordinate greedily and update it!
- Sparse solution: w^t contains at most t non-zero elements

Norm constraints

- Consider the case of norm constraints $\mathcal{C} = \{x \in \mathbb{R}^d \colon \|x\| \leq \tau\}$
- Note that

$$s \in \operatorname*{argmin}_{\|x\| \leq \tau} g^\top x = \tau \cdot \operatorname*{argmax}_{\|x\| \leq 1} -g^\top x$$

- Recall the definition of dual norm

$$\|y\|_* = \max_{\|x\| \leq 1} x^\top y$$

- Thus, up to the factor τ, s is the argument achieving the maximum in the dual norm
- We saw (last lecture) that this coincides with the subdifferential
- Therefore, $s \in \tau \cdot \partial \| - g\|_*$
- Example 3 (last slide): $\operatorname{sign}(-g) \in \partial \| - g\|_1$

Frank-Wolfe variants

- Vanilla FW has a slow sublinear rate

$$f(x^t) - f(x^\star) \leq O\left(\frac{1}{t}\right)$$

- Variants of FW that enjoy a linear rate of convergence exist under strong convexity assumptions on f

$$f(x^t) - f(x^\star) \leq O\left(\rho^t\right) \qquad \rho \in (0, 1)$$

- Full-corrective FW
- Away-steps FW
- Pairwise FW

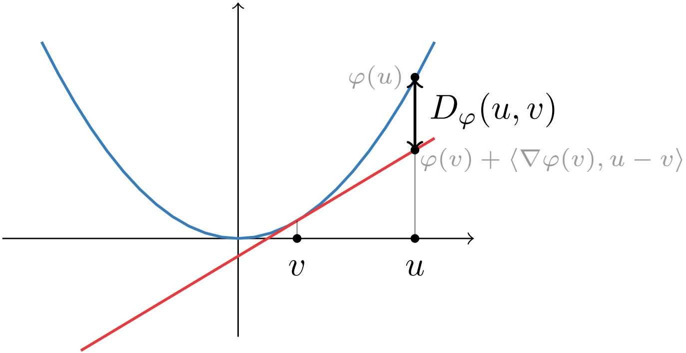
Outline

Bregman divergences

- The Bregman divergence generated by φ between u and v is

$$D_\varphi(u, v) = \varphi(u) - \varphi(v) - \langle \nabla\varphi(v), u - v \rangle$$

- It is the difference between $\varphi(u)$ and its linearization around v.



- Examples: $\varphi(x) = \frac{1}{2}\|x\|_2^2$ (squared Euclidean), $\varphi(x) = x^\top \log(x)$ (Kullback-Leibler)

Bregman projections

- Euclidean projection

$$P_{\mathcal{C}}(x) = \operatorname*{argmin}_{y \in \mathcal{C}} \|y - x\|_2^2$$

- Bregman projection onto $\mathcal{C} \subseteq \mathrm{dom}(\varphi)$

$$P_{\mathcal{C}}^{\varphi}(x) = \operatorname*{argmin}_{y \in \mathcal{C}} D_{\varphi}(y, x)$$

- Recovers Euclidean projections as a special case

- Example: $\varphi(x) = x^\top \log(x)$

$$P_{\mathcal{C}}^{\varphi}(x) = \operatorname*{argmin}_{y \in \mathcal{C}} \mathrm{KL}(y, x) \qquad x \in \mathbb{R}_+^d, \mathcal{C} \subseteq \mathbb{R}_+^d$$

Mirror descent

- Projected gradient descent

$$x^{t+1} = P_{\mathcal{C}}(x^t - \eta^t \nabla f(x^t)) = P_{\mathcal{C}} \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{1}{2\eta^t} ||x - x^t||_2^2$$

$$= \operatorname*{argmin}_{x \in \mathcal{C}} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{1}{2\eta^t} ||x - x^t||_2^2$$

- Mirror descent

$$\begin{aligned} x^{t+1} &= P_{\mathcal{C}}^{\varphi} \operatorname*{argmin}_{x \in \mathbb{R}^d} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{1}{\eta^t} D_\varphi(x, x^t) \\ &= \operatorname*{argmin}_{x \in \mathcal{C}} f(x^t) + \nabla f(x^t)^\top (x - x^t) + \frac{1}{\eta^t} D_\varphi(x, x^t) \\ &\neq P_{\mathcal{C}}^{\varphi}(x^t - \eta^t \nabla f(x^t)) \quad \text{(in general)} \end{aligned}$$

- Convergence rate is $O\left(\frac{L_{f,*}}{\sqrt{t}}\right)$, where $\|\nabla f(x)\|_* \leq L_{f,*}$ for all x when using $\eta_t = O\left(\frac{1}{L_{f,*}\sqrt{t}}\right)$

Example: optimization over the simplex

- If $\varphi(x) = x^\top \log(x)$ and $\mathcal{C} = \triangle^d$, then we have a closed form

$$x^{t+1} = \frac{x^t \exp(-\eta_t \nabla f(x^t))}{\sum_{j=1}^d x_j^t \exp(-\eta_t \nabla_j f(x^t))}$$

(the operations in the numerator are element-wise)

- Often called exponentiated gradient descent or entropic descent
- The KL case, for which $\|\cdot\| = \|\cdot\|_1$ and $\|\cdot\|_* = \|\cdot\|_\infty$, enjoys better convergence rate than the Euclidean case on the simplex
- Indeed, using $\|g\|_\infty \leq \|g\|_2 \leq \sqrt{d}\|g\|_\infty$, we get

$$\frac{1}{\sqrt{d}} \leq \frac{L_{f,\infty}}{L_{f,2}} \leq 1$$

Alternative view

- On iteration t

$$\begin{aligned}
\hat{x}^t &= \nabla\varphi(x^t) && \text{map primal point to dual} \\
\hat{y}^{t+1} &= \hat{x}^t - \eta^t \nabla f(x^t) && \text{take gradient step in the dual} \\
y^{t+1} &= \nabla\varphi^*(\hat{y}^{t+1}) && \text{map new dual point back to primal} \\
x^{t+1} &= P_{\mathcal{C}}^{\varphi}(y^{t+1}) && \text{project onto feasible set}
\end{aligned}$$

- $\nabla\varphi$ and $\nabla\varphi^*$ are called mirror maps

- Under technical assumptions on φ called "Legendre-type" (φ strictly convex, $\nabla\varphi = \infty$ on the boundary of $\mathrm{dom}(\varphi)$), we have

$$\nabla\varphi^* = (\nabla\varphi)^{-1}$$

Outline

Summary

- Convergence rates: important to familarize yourself with their classification
- Coordinate descent: ideal when regularizers or constraints are decomposable
- Newton's method: the Newton-CG algorithm (Hessian-free) is popularly used
- Frank-Wolfe: projection-free constrained optimization
- Mirror descent: generalization of projected gradient descent to non-Euclidean geometries

Lab work

- Recall that the dual of multiclass SVMs consists in maximizing

$$D(\beta) = -\sum_{i=1}^{n} [\Omega(\beta_i) - \Omega(y_i)] - \frac{1}{2\lambda} \|X^\top (Y - \beta)\|_2^2 \quad \text{s.t.} \quad \beta_i \in \triangle^k$$

  where $X \in \mathbb{R}^{n \times d}$ (features), $Y \in \mathbb{R}^{n \times k}$ (one-hot labels), $\Omega(\beta_i) = -\langle \beta_i, 1 - y_i \rangle$, $\lambda > 0$ (regularization parameter)

- The primal-dual link is $W^\star = \frac{1}{\lambda} X^\top (Y - \beta^\star) \in \mathbb{R}^{d \times k}$

- The gradient $\nabla D(\beta) \in \mathbb{R}^{n \times k}$ has rows as follows:

$$\nabla_i D(\beta) = -\nabla \Omega(\beta_i) + x_i^\top (\underbrace{\frac{1}{\lambda} X^\top (Y - \beta)}_{W}) \in \mathbb{R}^k$$

  where $\nabla \Omega(\beta_i) = y_i - 1$

Lab work

- We want to minimize $f(\beta) = -D(\beta)$, where $\nabla f(\beta) = -\nabla D(\beta)$
- Implement Frank-Wolfe for this problem.
  - Initialize $\beta_i^0 \in \triangle^k$, e.g., $\beta_i^0 = (1/k, \dots, 1/k)$, for $i \in \{1, \dots, n\}$
  - For $t \in \{0, 1, 2, \dots\}$

$$G = \nabla f(\beta^t) \in \mathbb{R}^{n \times k}$$

$$s_i = \operatorname*{argmin}_{s_i \in \triangle^k} g_i^\top s_i \quad i \in \{1, \dots, n\}$$

$$\beta^{t+1} = (1 - \gamma^t)\beta^t + \gamma^t S \qquad \gamma^t = \frac{2}{2+t}$$

- Implement mirror descent for this problem using the KL geometry.

$$\beta_i^{t+1} = \frac{\beta_i^t \exp(-\eta_t \nabla_i f(\beta^t))}{\sum_{j=1}^k \beta_{i,j}^t \exp(-\eta_t \nabla_{i,j} f(\beta^t))} \qquad i \in \{1, \dots, n\}$$

using $\eta_t = \eta/\sqrt{t}$ for some $\eta \in (0, 1]$