A Guide to Optimization for Machine Learning

XIU Shengjie

March 5, 2023

This is a note on optimization for machine learning. It is developed mainly based on the course COMP6211E Optimize for Machine Learning lectured by Tong ZHANG. The supplementary material include ELE522 Large-Scale Optimization for Data Science lectured by Yuxin CHEN.

Part I

Convex Theory

1 KKT Conditions and Duality

We consider the following constrained optimization problem

$$\begin{array}{ll} \underset{x}{\text{minimize}} & f(x) \\ \text{subject to} & g_j(x) \leq 0,\ j = 1, \ldots, k \\ & h_j(x) = 0,\ j = 1, \ldots, m. \end{array} \tag{1}$$

The Lagrangian function

$$L(x, \mu, \lambda) = f(x) + \mu^\top g(x) + \lambda^\top h(x).$$

1.1 General case

The KKT conditions in Theorem 1 are necessary conditions for a local optimal solution of (1).

Theorem 1. *(**General KKT**) Assume that $f(x)$, $g(x)$, $h(x)$ are continuously differentiable. Assume x_* is a local optimal solution of* (1)*. If the gradients of the active inequality constraints and the gradients of the equality constraints are linearly independent at x_*, then the following KKT conditions hold.*

- *Stationarity:*
$$\nabla_x L(x_*, \mu, \lambda) = 0.$$
- *Primal feasibility:*
$$g(x_*) \leq 0, \qquad h(x_*) = 0.$$
- *Dual feasibility:*
$$\mu_j \geq 0, \quad \forall j = 1, \ldots, k.$$
- *Complementary slackness:*
$$\mu_j g_j(x_*) = 0, \quad \forall j = 1, \ldots, k.$$

Proof. The proof is interesting so we attach here. First, primal feasibility is obvious. Second, complementary slackness can be considered in this way: If $g_j(x_*) < 0$, then this constraint isn't active, and thus we can set $\mu_j = 0$. Third, we need to prove stationary and dual feasibility. Simply assume $m = 0$, and consider $\Delta = x - x_*$. Imaging we move a small amount from the x_*, we have[1]

$$\nabla g(x_*)^\intercal \Delta \leq 0 \Rightarrow \nabla f(x_*)^\intercal \Delta \geq 0. \tag{2}$$

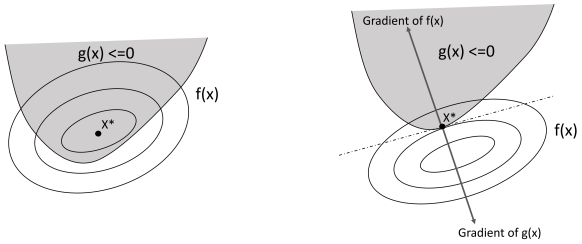
[1] This is quite intuitive. Refer to the note for formal description. Note that (2) violates one condition of the Farkas lemma.

Let $A = -\nabla g(x_*)^\intercal$ and $g = \nabla f(x_*)$, we know from the Farkas lemma that $\exists \mu \geq 0$ such that

$$-\nabla g(x_*)\,\mu = \nabla f(x_*).$$

This proves stationarity and dual feasibility. □

A geometric illustration is given as follows



From the right plot, we have

$$\begin{cases} \nabla f(x_*) + \mu \nabla g(x_*) = 0 \\ \nabla f(x_*)^\intercal \nabla g(x_*) \leq 0, \end{cases}$$

which means that we have $\mu \geq 0$.

1.2 Convex case

Theorem 2. ***(Convex KKT)*** *Assume that*

- *The objective function $f(x)$ is a convex but not necessarily differentiable (subgradient may not be unique).*
- *$g(x)$ is a continuously differentiable convex function.*
- *Each $h_j(x)$ is a linear constraint.*
- *The Slater's condition holds.*

Then x_* is an optimal solution of (1) *if and only if the KKT conditions of Theorem 1 are satisfied with a subgradient $\nabla f(x_*) \in \partial f(x_*)$.*

Part II

General Problem

2 *Gradient Descent*

2.1 Procedure

Target at unconstrained optimization problems:

$$\min_x \quad f(x).$$

Algorithm:

$$x_t = x_{t-1} - \eta_t \nabla f(x_{t-1}).$$

The gradient descent can be viewed as solving the following optimization problem at each step t:

$$\begin{aligned} x_t &= \arg\min_x f_t(x) \\ f_t(x) &= \left[f(x_{t-1}) + \nabla f(x_{t-1})^\intercal (x - x_{t-1}) + \frac{1}{2\eta_t} \parallel x - x_{t-1} \parallel_2^2 \right], \end{aligned} \tag{3}$$

where $f_t(x)$ is also a quadratic approximation of f.[1]

[1] replace $\nabla^2 f(x_{t-1})$ by $\frac{1}{\eta_t}$

- $f(x)$ is L-smooth:

$$f(x) \leq \left[f(x_{t-1}) + \nabla f(x_{t-1})^\intercal (x - x_{t-1}) + \frac{L}{2} \parallel x - x_{t-1} \parallel_2^2 \right],$$

then gradient descent reduces the objective value at each step when $\eta_t \leq 1/L$, because $f_t(x)$ is an upper bound of $f(x)$.[2] The following figure illustrates this.

[2] L-smooth means there exists a quadratic upper bound.

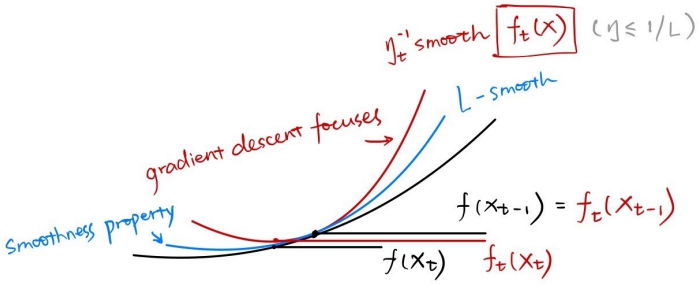


Figure 1: Illusion of the convergence of gradient descent

- $f(x)$ is λ-strongly convex:

$$f(x) \geq \left[f(x_0) + \nabla f(x_0)^\intercal (x - x_0) + \frac{\lambda}{2} \parallel x - x_0 \parallel_2^2 \right].$$

2.2 Convergence for Smooth or Strongly Convex Functions

Proposition 3. *If $f(x)$ is L-smooth, then*

$$f(x) - \min_x f(x) \geq \frac{1}{2L} \| \nabla f(x) \|_2^2.$$

Large gradient: has not optimized well.

Proposition 4. *If $f(x)$ is λ-strongly convex, then*

$$f(x) - \min_x f(x) \leq \frac{1}{2\lambda} \| \nabla f(x) \|_2^2.$$

Small gradient: problem is approximately solved.

2.3 Convergence for Smooth and Strongly Convex Functions

Theorem 5. *If $f(x)$ is L-smooth and λ-strongly convex, then it has an unique solution x_*. Given any fixed learning rate $\eta_t = \eta \leq 1/L$, we have*

$$[f(x_t) - f(x_*)] \leq (1 - \eta\lambda)^t [f(x_0) - f(x_*)].$$

If we set $\eta = 1/L$, then the number of iterations t to achieve an ϵ-suboptimal solution is

$$t = O\left(\frac{1}{\eta\lambda} \log(1/\epsilon)\right) = O\left(\frac{L}{\lambda} \log(1/\epsilon)\right) = \tilde{O}\left(\frac{L}{\lambda}\right), \tag{4}$$

where the number $\kappa = L/\lambda$ is called the condition number[1], and $\tilde{O}(\cdot)$ may hide a logarithmic factor in T. The rate of convergence for smooth and strongly convex functions in Theorem 5 is linear convergence.

[1] The condition number is usually large in practice.

Besides, we can also obtain the convergence of parameters.

Theorem 6. *If $f(x)$ is L-smooth and λ-strongly convex, then it has an unique solution x_*. Given any fixed learning rate $\eta_t = \eta \leq 1/L$, we have*

$$\|x_t - x_*\|_2^2 = (1 - \eta\lambda)^t \|x_0 - x_*\|_2^2.$$

2.4 Convergence for Smooth Convex Functions

For a smooth convex but not strongly-convex function, its solution may not be finite, and we cannot measure the convergence of x_t to x_*. Nevertheless, given any finite $\bar{x}$, we have the following result:

Theorem 7. *If $f(x)$ is L-smooth and convex, then given an arbitrary finite $\bar{x}$, and a fixed learning rate $\eta_t = \eta \leq 1/L$, we have*

$$\frac{1}{T} \sum_{t=1}^{T} f(x_t) \leq f(\bar{x}) + \frac{\| x_0 - \bar{x} \|_2^2}{2\eta T}.$$

The information is that the gradient descent method has a slower convergence rate[2] of $O(1/T)$ on average from $t = 1$ to $t = T$. Therefore to obtain an average

[2] The convergence rate is defined as the number of iterations to achieve sub-optimality of ϵ. The material by Raghu Meka shows that $f(x_T) \leq f(x_*) + \frac{\|x_0 - x_*\|_2^2}{2\eta T}$, and the material by Robert Gower shows that $\frac{1}{T}\sum_{t=1}^{T} f(x_t) \leq f(x_*) + \frac{\eta(f(x_0) - f(x_*))}{4LT} + \frac{\|x_0 - x_*\|_2^2}{\eta T}$.

sub-optimality of ϵ, we need

$$T = \frac{\| x_0 - \bar{x} \|_2^2}{2\eta} \cdot \frac{1}{\epsilon} \tag{5}$$

steps. The rate of convergence is sub-linear and slower than that of smooth and strongly convex functions.

Definition of rate of convergence

- First, the rate of convergence in wiki is the traditional asymptotic analysis. $\lim_{k\to\infty} \frac{|x_{k+1}-x_*|}{|x_k-x_*|} = \mu$, if $\mu \in (0,1)$ it means linear convergence, otherwise super-linear ($\mu = 1$) or sub-linear convergence ($\mu = 0$).
- As mentioned by Brian Borchers, it isn't directly comparable to the non-asymptotic analysis of first-order methods in terms of the number of iterations needed to obtain an epsilon-optimal solution (e.g. $O(1/\epsilon)$), or the reduction that can be obtained after k iterations (e.g. $O(1/k)$). This is because the first-order algorithms are typically stopped long before full convergence.
- Second, from Tibshirani's lecture, we know that if a bound of $f(x_t) - f(x_*) \leq \epsilon$ is achieved by
  - $O(\log(1/\epsilon))$ iterations, the rate is typically called "linear convergence;"
  - $O(1/\epsilon)$ iterations, the rate is typically called "sub-linear convergence."

3 *Quadratic Objective*

Target at quadratic optimization problems:

$$\min_x \quad Q(x) = \frac{1}{2} x^\intercal A x - b^\intercal x.$$

3.1 *Gradient Descent for Quadratic Optimization*

By applying the gradient descent, we have the iterative algorithm

$$x_t = x_{t-1} - \eta \left(A x_{t-1} - b \right).$$

Therefore, by plugging in $b = A x_*$, we obtain

$$x_t - x_* = (I - \eta A)(x_{t-1} - x_*),$$

and

$$x_t - x_* = (I - \eta A)^t (x_0 - x_*).$$

The convergence rate is established as

Corollary 8. *The gradient descent method converges for arbitrary x_0 if and only if $\eta \in (0, 2/L)$, which implies that the spectral norm of $I - \eta A$: $\rho = \max(1 - \eta\lambda, |1 - \eta L|) < 1$. We have*

$$\|x_t - x_*\|_2 = \rho^t \|x_0 - x_*\|_2.$$

It implies that we should set the learning rate $\eta < 2/L$ to ensure convergence:

- If $\eta = 1/L$, it matches the result of gradient descent in Theorem 6.

3.2 *Conjugate Gradient Method*

The conjugate gradient has the following iterative algorithm (from immediate gradient to weighted gradient for smoothing)

$$x_t = x_{t-1} + \alpha_t p_{t-1}$$
$$p_t = -\nabla Q(x_t) + \beta_t p_{t-1}.$$

- p_{t-1} is the search direction, and α_t is chosen to achieve steepest descent:

$$\alpha_t = \arg\min_{\alpha} Q\left(x_{t-1} + \alpha p_{t-1}\right).$$

- β_t is chosen so that the search directions p_t and p_{t-1} is orthogonal in A-norm:

$$p_t^\intercal A p_{t-1} = 0 \Longrightarrow \left((b - Ax_{t-1}) + \beta_t p_{t-1}\right)^\intercal A p_{t-1} = 0.$$

Understanding of conjugate direction [Shewchuk et al., 1994]:

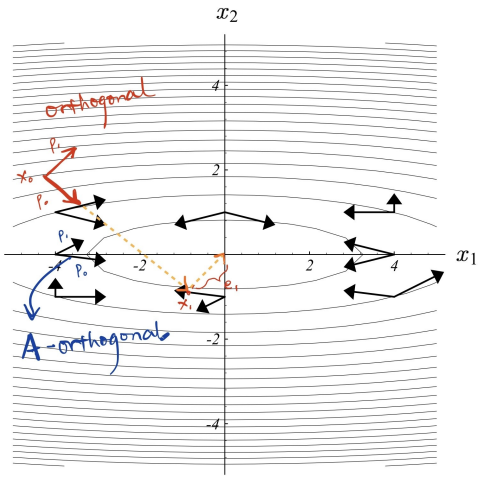


Figure 2: Illusion of the conjugate gradient.

- Why not the orthogonal search direction $p_t \perp p_{t-1}$? For example the 2-D case in Fig. 2, if we want find the optimal solution in two steps, we have to make sure that the first step should produce $p_0 \perp e_1$, where e_1 is the error after the first step $e_t = x_t - x$. The difficulty is that without knowing the optimal solution, we don't know e_1 in advance, which makes it almost impossible to determine α_1.
- Instead, using the conjugate direction, our new requirement is that e_1 be A-orthogonal to p_0. Finding p_0 is equivalent to finding the minimum point along the search direction. This is not coincidental with the proof

$$\begin{aligned}
&\frac{d}{d\alpha} Q\left(x_{t+1}\right) = 0 \\
\Rightarrow &Q'\left(x_{t+1}\right)^\intercal \frac{d}{d\alpha} x_{t+1} = 0 \\
\Rightarrow &- r_{t+1}^\intercal p_t \\
\Rightarrow &\left(A e_{t+1}\right)^\intercal p_t = 0,
\end{aligned}$$

where $r_t = b - Ax_t = -Ae_t$ is the residual.

Theorem 9. *Assume that A is positive definite. x_t achieves the minimum value of*

$$\min_x Q(x) \quad s.t.\ x \in span\left\{p_0, \ldots, p_{t-1}\right\}.$$

After at most $T = d$ iterations, we have $x_T = x_*$.

Pros:

- is an "optimal" (for linear systems, among all such methods) first order method that employs first order gradient information, and converges faster than regular gradient descent. It is applicable to sparse systems that are too large to be handled by a direct implementation or other direct methods such as the Cholesky decomposition.
- It also automatically sets the parameters α_t and β_t, which is an advantage for general linear systems.

Cons:

- Difficult to generalize to stochastic methods: difficult to generalize the computation of α_t and β_t in CG.
- Also CG does not handle non-smooth regularizer such as Lasso well.

3.3 Heavy Ball Method

The heavy ball method is very similar to CG,

$$x_t = x_{t-1} + \alpha_t p_{t-1}$$
$$p_t = -\nabla Q(x_t) + \beta_t p_{t-1}.$$

Set $\alpha_t = \alpha$ and $\beta_t = \beta$.

The convergence rate of the heavy-ball method with the optimal settings of α_t and β_t matches that of the CG algorithm, improving from $O(\kappa \log(1/\epsilon))$ to $O(\sqrt{\kappa}\log(1/\epsilon))$ compared with gradient descent in quadratic problems. Although the heavy ball method was stated for general nonlinear opThe conjugate gradient method and heavy ball method for this general problem employ the following form of recursion:

$$x_t = x_{t-1} + p_{t-1}$$
$$p_t = -\alpha_t \nabla f(x_t) + \beta_t p_{t-1},$$

or modified as:

$$y_t = x_{t-1} + \beta_t (x_{t-1} - x_{t-2})$$
$$x_t = y_t - \alpha_t \nabla f(x_{t-1}).$$

timization, only asymptotic convergence can be proved. Global convergence can be obtained only for quadratic functions.

In general, both momentum methods are first-order methods that can significantly accelerate the convergence of the vanilla gradient descent. These methods are extended by Nesterov for general convex functions, with global convergence proof.

4 Nesterov's Acceleration Method

Target at general unconstrained convex optimization problems:

$$\min_x \quad f(x).$$

The conjugate gradient method and heavy ball method for this general problem employ the following form of recursion:

$$\begin{aligned} x_t &= x_{t-1} + p_{t-1} \\ p_t &= -\eta_t \nabla f(x_t) + \beta_t p_{t-1}, \end{aligned}$$

or modified as:

$$\begin{aligned} y_t &= x_{t-1} + \beta_t (x_{t-1} - x_{t-2}) \\ x_t &= y_t - \nabla f(x_{t-1}). \end{aligned}$$

Nesterov's acceleration method modify the above equation as follows

$$\begin{aligned} y_t &= x_{t-1} + \beta_t (x_{t-1} - x_{t-2}) \\ x_t &= y_t - \eta_t \nabla f(y_t), \end{aligned}$$

with the difference in where to evaluate ∇f. With this modification, one can prove the global convergence of the resulting algorithm for convex functions.

4.1 Convergence for Smooth and Strongly Convex Functions

Theorem 10. *Assume $f(x)$ is L-smooth and λ-strongly convex. Let $\eta \leq 1/L$ and $\theta \leq \sqrt{\eta\lambda}$. Let $\alpha_t = \eta \leq 1/L$ and $\beta_t = \beta = (1-\theta)/(1+\theta)$. Then*[1]

$$f(x_t) \leq f(x_*) + (1-\theta)^t \left[f(x_0) - f(x_*) + \frac{\lambda}{2} \parallel x_0 - x_* \parallel_2^2 \right]. \tag{6}$$

[1] See the note of Adaptive and General Acceleration Methods. In material Rate of Convergence by Mark Schmidt, there is no $\frac{\lambda}{2} \parallel x_* - x_0 \parallel_2^2$ term.

It implies a convergence rate of

$$O(\sqrt{\kappa} \log(1/\epsilon)) = \tilde{O}(\sqrt{L/\lambda}). \tag{7}$$

For quadratic objective functions, this convergence rate matches that of CG and that of the heavy-ball method. However, the result also applies more generally to strongly convex functions.

4.2 Tuning of η and β

- η is learning rate.
- η is the decaying factor for the aggregated gradients (momentum term).

In practice, careful tuning of η and β are important.

- Right setting of β can significantly improve convergence, but inappropriate setting can lead to oscillation.
- The sensitivity to η is less severe with appropriate $\beta > 0$, because the gradients are aggregated.

5 Non-Smooth Convex Optimization

non-smooth but bounded gradient

We consider the general convex optimization problem:

$$\min_{x \in C} \quad f(x),$$

and assume that $f(x)$ is nonsmooth, but Lipschitz convex function in C.

$$\|\nabla f(x)\|_2 \leq G.$$

C is a convex set.[1]

[1] With the constraint, we can talk about the Lipschitz in the feasible set. Otherwise, for a quadratic function, $\nabla f(x)$ is unbounded.

Lipschitz and smooth

- Lipschitz:
$$\|\nabla f(x)\|_2 \leq G \Leftrightarrow |f(x) - f(x')| \leq G\|x - x'\|_2.$$
- Smooth:
$$\|\nabla^2 f(x)\|_2 \leq L \Leftrightarrow \|\nabla f(x) - \nabla f(x')\|_2 \leq L\|x - x'\|_2.$$

5.1 Subgradient Method

If the function is not smooth, the gradient can be replaced by a subgradient, but the step size must be reduced to achieve convergence. We can obtain a conver-

Algorithm 1: Subgradient Projection Method

1: **Input:** $f(x)$, x_0, and $\eta_1, \eta_2, \ldots$
2: **for** $t = 1, 2, \ldots, T$ **do**
3: Let $\tilde{x}_t = x_{t-1} - \eta_t g_t$, where $g_t \in \partial f(x_{t-1})$ is a subgradient
4: Let $x_t = \text{proj}_C(\tilde{x}_t)$
5: **end for**
6: **return** x_T

gence result:

Theorem 11. *(Non-smooth but bounded gradient) Assume that $f(x)$ is G-Lipschitz on C, then we have for all $\bar{x} \in C$:*

$$\frac{1}{\sum_{t=1}^T \eta_t} \sum_{t=1}^T \eta_t f(x_{t-1}) \leq f(\bar{x}) + \frac{\|x_0 - \bar{x}\|_2^2 + \sum_{t=1}^T \eta_t^2 G^2}{2\sum_{t=1}^T \eta_t}.$$

To understand the convergence result, we consider the case that we know T in advance, and choose a small constant learning rate $\eta_0/\sqrt{T}$. We obtain

$$\frac{1}{T} \sum_{t=1}^T f(x_{t-1}) \leq f(\bar{x}) + \frac{\|x_0 - \bar{x}\|_2^2 + \eta_0^2 G^2}{2\eta_0\sqrt{T}}.$$

We may optimize the bound in Theorem 11 over η_0 and takes $\eta_0 = \|x_0 - \bar{x}\|_2/G$, and achieve

$$\frac{1}{T} \sum_{t=1}^T f(x_{t-1}) \leq f(\bar{x}) + \frac{\|x_0 - \bar{x}\|_2 G}{\sqrt{T}}. \tag{8}$$

Theorem 12. *(Non-smooth but strongly convex) Assume that $f(x)$ is λ-strongly convex. Then with $\eta_t = 1/(\lambda t)$, we have*

$$\frac{1}{T}\sum_{t=1}^{T} f(x_{t-1}) \leq \inf_{\bar{x}\in C} f(\bar{x}) + \frac{(\ln T + 1)G^2}{2\lambda T}. \tag{9}$$

Proof. Proof of Theorem 11 and 12 are important paradigms. □

The number of iterations t to achieve an ϵ-suboptimal solution is:

- From (8), $t = O(G^2R^2/\epsilon^2)$, where R is the diameter of C ($\sup\{\|x - x_0\| : x \in C\}$)
- From (9), $t = \tilde{O}(G^2/\lambda\epsilon)$.

  There is no theory for acceleration method for nonsmooth problem.

5.2 Smoothing

It is possible to achieve a better convergence rate than the subgradient method by using the smoothing technique first, then Nesterov's acceleration method. This is also shown in *Lecture 9: General Unconstrained Convex Optimization*.

Definition 13. *$\tilde{f}(x)$ is an (L, ϵ)-smooth approximation of $f(x)$ if*

$$\tilde{f}(x) \leq f(x) \leq \tilde{f}(x) + \epsilon.$$

Approximate optimization: The following result shows that instead of optimizing with $f(x)$, we can obtain an approximation solution by optimizing with its smoothed version $\tilde{f}(x)$.

Theorem 14. *Let $\tilde{x}$ be an $\tilde{\epsilon}$-approximate solution of the minimization problem with respect to $\tilde{f}(x)$:*

$$\tilde{f}(\tilde{x}) \leq \min_x \tilde{f}(x) + \tilde{\epsilon},$$

then

$$\tilde{f}(\tilde{x}) \leq \min_x f(x) + \epsilon + \tilde{\epsilon}.$$

In particularly, for Lipschitz functions, there is an usual approximation method:

Conjecture 15. *If $f(x)$ is G-Lipschitz, then*

$$\tilde{f}(x) = \min_z \left[f(z) + \frac{L}{2} \parallel x - z \parallel_2^2 \right] \tag{10}$$

is an $(L, G^2/2L)$-smooth approximation of $f(x)$.

The smoothing is not unique, for example, there are different kinds of smoothing for ReLU. But the one in (10) is very handy.
general version with γ and θ

Proposition 16. *Assume $f(x)$ has an $(L = \tilde{G}^2/2\epsilon, \epsilon)$-smooth approximation $\tilde{f}(x)$ for some $\tilde{G} > 0$, and assume that $\nabla\tilde{f}(x)$ is easy to calculate. Let x_T be the solution obtained by Nesterov's acceleration algorithm with learning rate $\eta_t = 2\epsilon/G^2$. Then*

$$f(x_t) \leq f(\bar{x}) + \epsilon + \lambda_T \left[f(x_0) - f(\bar{x}) + \frac{\gamma_0}{2} \parallel x_0 - \bar{x} \parallel_2^2 \right],$$

where $\lambda_t = \prod_{s=1}^{t}(1 - \theta_s)$.

- For strongly convex function with $\gamma_0 = \lambda$, we have $\gamma_t = \lambda$ and $\theta_t = \theta = \sqrt{2\epsilon\lambda}/\tilde{G}$. This implies that with $T = O(\tilde{G}/\sqrt{\lambda\epsilon})$:

$$f(x_T) \leq f(x_*) + O(\epsilon).$$

  For strongly convex non-smooth optimization, we have

$$\text{smoothing} + \begin{cases} \text{GD: } \tilde{O}(L/\lambda) = \tilde{O}(\tilde{G}^2/\lambda\epsilon) & \text{match subgradient} \\ \text{AGD: } O(\tilde{G}/\sqrt{\lambda\epsilon}) & \text{better than subgradient} \end{cases}$$

- For convex function with $\lambda = 0$ and $\gamma_0 = 1/\eta = \tilde{G}^2/2\epsilon$, we have $\lambda_T = O(1/T^2)$. Therefore by choosing $T = O(\tilde{G}\|\bar{x} - x_0\|_2/\epsilon)$, we obtain

$$f(x_T) \leq f(\bar{x}) + O(\epsilon).$$

5.3 *Summary*

Smooth Problems		
	strongly-convex	non-strongly-convex
gradient descent	$\tilde{O}(L/\lambda)$	$\tilde{O}(LR^2/\epsilon)$
accelerated descent	$\tilde{O}(\sqrt{L/\lambda})$	$\tilde{O}(\sqrt{LR^2/\epsilon})$
Nonsmooth Problems		
	strongly-convex	non-strongly-convex
subgradient	$\tilde{O}(G^2/\lambda\epsilon)$	$\tilde{O}(G^2R^2/\epsilon^2)$
smoothing	$\tilde{O}(G/\sqrt{\lambda\epsilon})$	$\tilde{O}(GR/\epsilon)$

Table 1: Summary of complexity.

- From smooth problems to nonsmooth problems: let $L = G^2/\epsilon$.
- From strongly-convex to non-strongly-convex: let $\lambda = \epsilon$.

6 *Adaptive Learning Rate**

It was shown that for nonsmooth problems, the learning rate should decay when t increases, i.e., we should prevent large step sizes. Therefore it is useful to design methods that can tune learning rate automatically. If we do not know the smoothness parameter L of $f(x)$, we cannot set the learning rate easily.

The problem can be formulated as: given the current point y and a search direction p, we want to find a learning rate α.

A standard technique is **exact line search**:

$$\min_{\alpha} \quad f(y + \alpha p).$$

The second technique is **backtracking line search** (Armijo-Goldstein step size). If $f(x)$ is L-smooth (first inequality) and we take $p = -\nabla f(y)$, then when $\alpha \leq 1/L$, $c = 0.5$ (second inequality), we have

$$\begin{aligned} f(y + \alpha p) &\leq f(y) + \alpha \nabla f(y)^\intercal p + \frac{L}{2}\alpha^2 \parallel p \parallel_2^2 \\ &= f(y) + \alpha(1 - 0.5\alpha L)\nabla f(y)^\intercal p \\ &\leq f(y) + c\alpha \nabla f(y)^\intercal p. \end{aligned} \tag{11}$$

which is shown in Figure 3.

Algorithm 2: Backtracking Line Search Method

Input: $f(y)$, y, p, α_0, $\tau \in (0,1)$, $c \in (0,1)$ (default is $c = 0.5$)
Output: α

1: Let $\alpha = \alpha_0$
2: **while** $f(y + \alpha p) > f(y) + c\alpha \nabla f(y)^\intercal p$ **do**
3: Let $\alpha = \tau\alpha$
4: **end while**

The third technique is **Barzilai-Borwein step size**. The backtracking method is popular and simple to implement. However, it requires function value evaluation. In general, this can be avoided by only using gradient evaluations. It shows that the largest learning rate is equal to the right-hand side of the following equation, an estimate based on previous iterations:

$$\frac{1}{L} \leq \frac{\parallel \alpha p \parallel_2^2}{(\nabla f(y + \alpha p) - \nabla f(y))^\intercal (\alpha p)}.$$

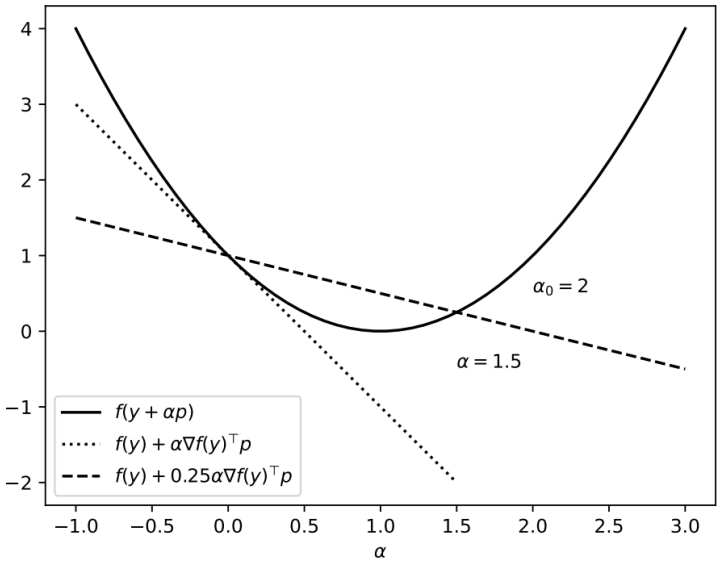


Figure 3: Illustration of Armijo-Goldstein condition

7 Quasi-Newton Methods*

[1]Target problem:

$$\min_{\mathbf{x}\in\mathbb{R}^n} \quad f(\mathbf{x}),$$

[1] This section mainly comes from Quasi-Newton Methods by Yuxin Chen

the Newton's method is

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t(\nabla^2 f(\mathbf{x}_t))^{-1}\nabla f(\mathbf{x}_t).$$

The problem is that it requires storing and inverting Hessian $\nabla^2 f(\mathbf{x}_t) \in \mathbb{R}^{n\times n}$.

The key idea is to approximate the Hessian matrix, $(\nabla^2 f(\mathbf{x}_t))^{-1}$ as $\mathbf{H}_t \succ 0$,

$$\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{H}_t \nabla f(\mathbf{x}_t),$$

using only gradient information.

7.1 Criterion of Ht

Consider the approximation of $f(\cdot)$:

$$\tilde{f}(\mathbf{x}) := f(\mathbf{x}_{t+1}) + \langle \nabla f(\mathbf{x}_{t+1}), \mathbf{x} - \mathbf{x}_{t+1}\rangle + \frac{1}{2}(\mathbf{x} - \mathbf{x}_{t+1})^\intercal \mathbf{H}_{t+1}^{-1}(\mathbf{x} - \mathbf{x}_{t+1}), \tag{12}$$

which satisfies

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

Criterion: gradient matching for the latest two iterates,

$$\text{criteria 1:} \quad \nabla\tilde{f}(\mathbf{x}_t) = \nabla f(\mathbf{x}_t)$$

$$\text{criteria 2:} \quad \nabla\tilde{f}(\mathbf{x}_{t+1}) = \nabla f(\mathbf{x}_{t+1}).$$

Criteria 2 holds automatically from (12). To satisfy criteria 1, we have the **secant equation**

$$\mathbf{H}_{t+1}^{-1}\underbrace{(\mathbf{x}_{t+1} - \mathbf{x}_t)}_{=:\mathbf{s}_t} = \underbrace{\nabla f(\mathbf{x}_{t+1}) - \nabla f(\mathbf{x}_t)}_{=:\mathbf{y}_t},$$

which is illustrated in Figure 4.

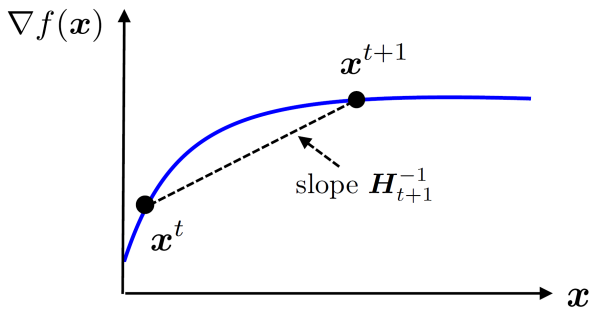


Figure 4: Secant equation.

$\mathbf{H}_{t+1}^{-1}$ shows the difference of the first derivative.

In addition to the two criterion, we want to choose $\mathbf{H}_{t+1}$ sufficiently close to $\mathbf{H}_t$, which can be seen as criteria 3:

$$\begin{aligned} \min_{\mathbf{H}} \quad & \|\mathbf{H} - \mathbf{H}_t\| \\ \text{s.t.} \quad & \mathbf{H} = \mathbf{H}^\intercal \\ & \mathbf{H}\mathbf{y}_t = \mathbf{s}_t, \end{aligned}$$

where different norms leads to different quasi-Newton methods.

7.2 BFGS

Broyden-Fletcher-Goldfarb-Shanno (BFGS) method chooses the norm with any weight matrix $\mathbf{W}$ obeying $\mathbf{W}\mathbf{s}_t = \mathbf{y}_t$,

$$\begin{aligned} \min_{\mathbf{H}} \quad & \|\mathbf{W}^{1/2}(\mathbf{H} - \mathbf{H}_t)\mathbf{W}^{1/2}\|_F \\ \text{s.t.} \quad & \mathbf{H} = \mathbf{H}^\intercal \\ & \mathbf{H}\mathbf{y}_t = \mathbf{s}_t, \end{aligned}$$

It has a closed-form expression (BFGS update rule)[1]

$$\mathbf{H}_{t+1} = \left(\mathbf{I} - \rho_t \mathbf{s}_t \mathbf{y}_t^\intercal\right) \mathbf{H}_t \left(\mathbf{I} - \rho_t \mathbf{y}_t \mathbf{s}_t^\intercal\right) + \rho_t \mathbf{s}_t \mathbf{s}_t^\intercal,$$

with $\rho_t = 1/(\mathbf{y}_t^\intercal \mathbf{s}_t)$.

[1] Derivation in BFGS Derivation Stanford (Unconstrained Optimization in CS 205A).

Algorithm 3: BFGS

1: **for** $t = 0, 1, \ldots$ **do**
2: line search: $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \mathbf{H}_t \nabla f(\mathbf{x}_t)$
3: update Hessian: $\mathbf{H}_{t+1} = \left(\mathbf{I} - \rho_t \mathbf{s}_t \mathbf{y}_t^\intercal\right) \mathbf{H}_t \left(\mathbf{I} - \rho_t \mathbf{y}_t \mathbf{s}_t^\intercal\right) + \rho_t \mathbf{s}_t \mathbf{s}_t^\intercal$
4: **end for**

Advantages of BFGS include:

- each iteration costs $\mathcal{O}(n^2)$, compared to $\mathcal{O}(n^3)$, in Newton's method;
- no need to solve linear systems or invert matrices;
- no magic formula for initialization; possible choices: approximate inverse Hessian at $\mathbf{x}_0$, or identity matrix

7.3 L-BFGS

Hessian matrices are usually dense (which means we cannot find an efficient storage of it), and thus for large-scale problems, even storing $\mathbf{H}$ (the approximated inverse Hessian in BFGS) is difficult, not to mention calculating the search direction $\mathbf{p}$

$$\mathbf{p} = -\mathbf{H}\mathbf{g}.$$

Limited memory BFGS (L-BFGS) maintains a compact approximation of $\mathbf{H}$, and calculate $\mathbf{p}$ instead of $\mathbf{H}$ directly. The advantage is that L-BFGS only needs to store a few $n \times 1$ vectors.

Let's start our derivation. I name the BFGS update as **one-look-back update**:

$$
\begin{aligned}
\underbrace{\mathbf{H}_t\mathbf{g}_t}_{:=\mathbf{p}_t} &= \left(\mathbf{I}-\rho_{t-1}\mathbf{s}_{t-1}\mathbf{y}_{t-1}^\intercal\right)\mathbf{H}_{t-1}\left(\mathbf{I}-\rho_{t-1}\mathbf{y}_{t-1}\mathbf{s}_{t-1}^\intercal\right)\mathbf{g}_t+\rho\mathbf{s}_{t-1}\mathbf{s}_{t-1}^\intercal\mathbf{g}_t \\
&= \left(\mathbf{I}-\rho_{t-1}\mathbf{s}_{t-1}\mathbf{y}_{t-1}^\intercal\right)\underbrace{\mathbf{H}_{t-1}\underbrace{\left(\mathbf{g}_t-\underbrace{\rho_{t-1}\mathbf{s}_{t-1}^\intercal\mathbf{g}_t}_{\alpha_{t-1}}\mathbf{y}_{t-1}\right)}_{\mathbf{q}_{t-1}}}_{\mathbf{p}_{t-1}}+\underbrace{\rho\mathbf{s}_{t-1}^\intercal\mathbf{g}_t}_{\alpha_{t-1}}\mathbf{s}_{t-1} \\
&= \left(\mathbf{I}-\rho_{t-1}\mathbf{s}_{t-1}\mathbf{y}_{t-1}^\intercal\right)\mathbf{p}_{t-1}+\alpha_{t-1}\mathbf{s}_{t-1} \\
&= \mathbf{p}_{t-1}+(\alpha_{t-1}-\beta_{t-1})\mathbf{s}_{t-1}
\end{aligned}
\tag{13}
$$

where

- $\alpha_{t-1}=\rho_{t-1}\mathbf{s}_{t-1}^\intercal\mathbf{q}_t$
- $\mathbf{q}_{t-1}=\mathbf{q}_t-\alpha_{t-1}\mathbf{y}_{t-1}$
- $\mathbf{p}_{t-1}=\mathbf{H}_{t-1}\mathbf{q}_{t-1}$
- $\beta_{t-1}=\rho_{t-1}\mathbf{y}_{t-1}^\intercal\mathbf{p}_{t-1}$, [1]

and we let $\mathbf{q}_t=\mathbf{g}_t$. $\{\alpha,\beta,\mathbf{q}\}$ are auxiliary variables. The key information of (13) is that $\mathbf{p}_t$ can be calculated without operations with $\mathbf{H}_t$ if we know $\mathbf{p}_{t-1}$. Also, $\mathbf{p}_{t-1}$ can be calculated from $\mathbf{p}_{t-2}$. The problem is that finally we have to calculate the initial $\mathbf{p}$ that may require a dense $\mathbf{H}$.

L-BFGS assumes that if we look back for a large history size m, the initial inverse Hessian $\mathbf{H}_k^0=\mathbf{H}_{k-m}$ (denotes the earliest guess of the $\mathbf{H}$ in the update of $\mathbf{H}_t$) can be approximated as an scaled identity matrix. The remaining question is how we can calculate $\mathbf{q}_{k-m}$ so that we can start our flow from $\mathbf{p}_{k-m}=\mathbf{H}_{k-m}\mathbf{q}_{k-m}$.

From (13) we notice that $\mathbf{q}_{t-1}$ can be calculated from $\mathbf{q}_t$ by

$$
\mathbf{q}_{t-1}=\mathbf{q}_t-\alpha_{t-1}\mathbf{y}_{t-1}, \tag{14}
$$

so the calculation of $\mathbf{q}_{k-m}$ is another flow.

The L-BFGS containing two recursions (13)(14) is summarized in Algorithm 4.[2]

With L-BFGS, we only need to store m most recent vector pairs $(\mathbf{s}_t,\mathbf{y}_t)$.

[1] You may be curious why $\mathbf{q}_{t-k-1}$ is not $\mathbf{g}_{t-k}-\alpha_{t-k}\mathbf{y}_{t-k-1}$, and α_{t-k} is not $\rho_{t-k-1}\mathbf{s}_{t-k-1}^\intercal\mathbf{g}_{t-k}$ but $\rho_{t-k-1}\mathbf{s}_{t-k-1}^\intercal\mathbf{q}_{t-k}$. Expand $\mathbf{H}_{t-1}$ as $\mathbf{H}_{t-2}$ and compare with Algorithm 4.

[2] Refer to Wikipedia: Limited-memory BFGS.

Algorithm 4: L-BFGS

$\mathbf{q} = \mathbf{g}_t$
for $i = t-1, t-2, \ldots, t-m$ **do**
    $\alpha_i = \rho_i \mathbf{s}_i^\intercal \mathbf{q}$
    $\mathbf{q} \leftarrow \mathbf{q} - \alpha_i \mathbf{y}_i$
end for
$\mathbf{H}_k^0 = \frac{\mathbf{s}_{t-1}^\intercal \mathbf{y}_{t-1}}{\mathbf{y}_{t-1}^\intercal \mathbf{y}_{t-1}} \mathbf{I}$
$\mathbf{p} = \mathbf{H}_k^0 \mathbf{q}$
for $i = t-m, t-m+1, \ldots, t-1$ **do**
    $\beta_i = \rho_i \mathbf{y}_i^\intercal \mathbf{p}$
    $\mathbf{p} \leftarrow \mathbf{p} + (\alpha_i - \beta_i)\mathbf{s}_i$
end for

Part III

Composite Problem

8 Proximal Gradient Descent Method

Target at composite convex optimization problems:

$$\min_x \quad \phi(x) = f(x) + g(x), \tag{15}$$

where $f(x)$ is a smooth convex function, and $g(x)$ may be a nonsmooth convex function, usually a regularizer.

Lipschitz and smooth

non-smooth problem $\begin{cases} \text{1. if proximal operator OK} & \rightarrow \text{better algorithm} \\ \text{1. smoothing} & \rightarrow \text{GD or AGD} \end{cases}$

8.1 Proximal Mapping

Here we define proximal operator as:

$$\text{prox}_{\eta g}(x) = \arg\min_z \left[\frac{1}{2} \parallel z - x \parallel_2^2 + \eta g(z) \right], \tag{16}$$

which means finding the proximal point z of x by minimizing there Euclidean distance and the regularizer on z. It is a mapping.

For a differentiable function f, recall that the gradient update $x_t = x_{t-1} - \eta_t \nabla f(x_{t-1})$ is derived using the quadratic approximation of f (3). In this case, $\phi(x)$ is not differentiable. Why don't we make a quadratic approximation (upper bound) to $f(x)$, leaving $g(x)$ alone?

$$\begin{aligned} x_t &= \arg\min_x \underbrace{f(x_{t-1}) + \nabla f(x_{t-1})^\intercal (x - x_{t-1}) + \frac{1}{2\eta} \parallel x - x_{t-1} \parallel_2^2}_{\tilde{f}_\eta(x)} + g(x) \\ &= \arg\min_x \frac{1}{2\eta} \parallel x - (x_{t-1} - \eta \nabla f(x_{t-1})) \parallel_2^2 + g(x) \\ &= \text{prox}_{\eta g}(x_{t-1} - \eta \nabla f(x_{t-1})), \end{aligned} \tag{17}$$

when $\eta \leq 1/L$.

Proximal gradient descent is almost the same as classical gradient descent (3). It also tries to minimize the upper bound, as shown in Figure 1. The only difference is that now the optimization problem has an additional term $g(x)$. The new optimization problem can be written in a proximal problem useful if $\text{prox}_\eta(\cdot)$ is inexpensive.

The benefit is that $\text{prox}_{\eta g}(x - \eta \nabla f(x))$ has a closed-form for many important functions g.

Algorithm 5: Proximal Gradient Descent

Input: $\phi(\cdot)$, $x_0 \in \text{dom}\phi$, and $\eta_1, \eta_2, \ldots$
for $t = 1, 2, \ldots, T$ **do**
    Let $\tilde{x}_t = x_{t-1} - \eta_t \nabla f(x_{t-1})$
    Let $x_t = \text{prox}_{\eta g}(\tilde{x}_t)$
end for
return x_T

- Mapping $\text{prox}_{\eta g}(x - \eta \nabla f(x))$ doesn't depend on g at all, only on f.
- Smooth part f can be complicated; we only need to compute its gradients.

Comparison between proximal gradient method and subgradient method:

$$\text{proximal: } x_t = \text{prox}_{\eta g}(x_{t-1} - \eta_t \nabla f(x_{t-1})) = \arg\min_z \left[\frac{1}{2} \|z - (x_{t-1} - \eta_t \nabla f(x_{t-1}))\|_2^2 + \eta g(z) \right]$$

$$\text{subgradient/gradient: } x_t = x_{t-1} - \eta_t \nabla f(x_{t-1}) = \arg\min_z \left[\frac{1}{2} \|z - (x_{t-1} - \eta_t \nabla f(x_{t-1}))\|_2^2 \right],$$

so we can see that proximal evaluates the difference between a proximal point and a gradient point with one more penalty.

The proximal gradient descent is viewed as a generalization of the projected gradient descent. It extends the hard threshold

$$\mathbb{1}_C(x) = \begin{cases} 0, & \text{if } x \in C \\ \infty, & \text{else} \end{cases}$$

to a more general $g(x)$. Thus, the analysis of projected gradient descent has two directions:

1. Section 5: non-smmoth
2. Section 8: proximal gradient descent

8.2 *Composite gradient mapping*

Definition 17. Consider (15), where both $f(x)$ and $g(x)$ are convex. For any $\eta > 0$, the operator $\text{dom}\phi \to \mathbb{R}^d$ defined as

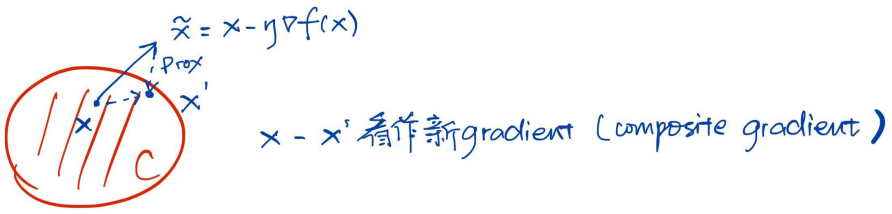
$$D_{\eta g} f(x) = \frac{1}{\eta} \left(x - \text{prox}_{\eta g}(x - \eta \nabla f(x)) \right)$$

is referred to as composite gradient mapping.

Using this notation, the proximal gradient method can be written as

$$x_t = x_{t-1} - \eta_t D_{\eta_t g} f(x_{t-1}),$$

which has a form similar to gradient descent. Therefore, $\|D_{\eta g} f(x)\|_2$ instead of $\|\nabla \phi(x)\|_2$ or $\|\nabla f(x)\|_2$ can be used for convergence checking.



8.3 Convergence

Theorem 18. *If $f(x)$ is L-smooth and λ-strongly convex function, and $g(x)$ is a λ'-strongly convex function. Let $\eta_t = \eta \leq 1/L$, then for all $\bar{x} \in dom\phi$:*

$$\phi(x_t) \leq \phi(\bar{x}) + (1-\theta)^t[\phi(x_0) - \phi(\bar{x})],$$

where $\theta = (\eta\lambda + \eta\lambda')/(1 + \eta\lambda')$.

Theorem 19. *If $f(x)$ is L-smooth. Let $\eta_t = \eta \leq 1/L$, then for all $\bar{x} \in dom\phi$:*

$$\frac{1}{T}\sum_{t=1}^{T}\phi(x_t) \leq \phi(\bar{x}) + \frac{\| x_0 - \bar{x} \|_2^2}{2\eta T}.$$

8.4 Backtracking line search

Observe that in the proof of Theorem 18, the convergence result holds as long as the learning rate satisfies the condition

$$\phi(x_t) \leq \tilde{f}_\eta(x_t) + g(x_t).$$

Instead, for convergence, we can use line-search method.[1]

[1] Line search is impractical for ML because it is only for deterministic methods but not stochastic methods.

9 Mirror Descent and Dual Averaging

Target at composite convex optimization problems:

$$\min_x \quad \phi(x) = f(x) + g(x),$$

where $f(x)$ is a smooth convex function, and $g(x)$ may be a nonsmooth convex function, usually a regularizer.

9.1 Mirror Descent

There is a clear path of development from gradient descent (GD) to proximal gradient descent and finally to mirror descent. Their objective functions are all in quadratic forms[2], and are summarized below:

[2] This conclusion is inspired by ELE522.

$$\begin{aligned}
\text{GD: } & \arg\min_x f(x_{t-1}) + \nabla f(x_{t-1})^\top (x - x_{t-1}) + \frac{1}{2\eta_t} \| x - x_{t-1} \|_2^2 \\
\text{PGD: } & \arg\min_x f(x_{t-1}) + \nabla f(x_{t-1})^\top (x - x_{t-1}) + \frac{1}{2\eta_t} \| x - x_{t-1} \|_2^2 + g(x) \qquad (18) \\
\text{MD: } & \arg\min_x f(x_{t-1}) + \nabla f(x_{t-1})^\top (x - x_{t-1}) + \frac{1}{\eta_t} D_h(x, x_{t-1}) + g(x).
\end{aligned}$$

Proximal gradient descent adds a regularizer to gradient descent, and mirror descent replaces the quadratic proximal term (discrepancy between $f(\cdot)$ and its first-order approximation) with a Bregman divergence metric defined as

$$D_h(x, y) = h(x) - h(y) - \nabla h(y)^\intercal (x - y),$$

which includes a huge family of variations with different types of $h(\cdot)$.

Key benefit: Adjust gradient updates to fit problem geometry. For example, Euclidean distance is in general not recommended for measuring the distance between probability vectors, and may prefer probability divergence metrics, e.g., Kullback-Leibler divergence, total-variation distance, and χ^2 divergence.

The convergence is achieved when the Bregman divergence is an "upper bound", which is the same for other methods that requires the quadratic proximal term is an "upper bound". For GD and PGD, we requires $f(\cdot)$ is a L-smooth function and $\eta_t \leq 1/L$; for MD, we requires $f(\cdot)$ is smooth with respect to a convex function $h(\cdot)$, defined as

$$f(x) \leq f(y) + \nabla f(y)^\intercal (x - y) + D_h(x, y).$$

If we define the generalization of proximal mapping as:

$$\text{prox}_g^h(x) = \arg\min_z \left[-x^\intercal z + h(z) + g(z) \right].$$

The proximal operator in Section 8 can be regarded as the special case of $h(z) = \frac{1}{2}\|z\|_2^2$. Then, the mirror descent can be further simplified as

$$\text{MD: } \text{prox}_{\eta g}^h(\nabla h(x_{t-1}) - \eta_t \nabla f(x_{t-1})).$$

Algorithm 6: Mirror Proximal Descent

Input: $f(\cdot)$, $g(\cdot)$, x_0, $h(\cdot)$ and $\{\eta_t \leq 1/L\}$
for $t = 1, 2, \ldots, T$ **do**
    Let $\tilde{x}_t = \nabla h(x_{t-1}) - \eta_t \nabla f(x_{t-1})$
    Let $x_t = \text{prox}_{\eta g}^h(\tilde{x}_t)$
end for
return x_T

9.2 *Dual Averaging*

The first order condition of x_t for being the solution of the general proximal mapping minimization problem is: [from (18)]

$$\nabla h(x_t) + \eta_t \nabla g(x_t) = \nabla h(x_{t-1}) - \eta_t \nabla f(x_{t-1}).$$

By summing over $s \leq t$, we obtain the following equation:

$$\nabla h(x_t) + \sum_{s=1}^{t} \eta_s \nabla g(x_s) = \nabla h(x_0) - \sum_{s=1}^{t} \eta_s \nabla f(x_{s-1}).$$

An alternative method is to replace each $\nabla g(x_s)$ by $\nabla g(x_t)$ as follows

$$\nabla h(x_t) + \sum_{s=1}^{t} \eta_s \nabla g(x_t) = \nabla h(x_0) - \underbrace{\sum_{s=1}^{t} \eta_s \nabla f(x_{s-1})}_{\text{momentum}}.$$

That is

$$x_t = \arg\min_x \left[-(\nabla h(x_0) - \sum_{s=1}^{t} \eta_s \nabla f(x_{s-1}))^\intercal x + h(x) + \tilde{\eta}_t g(x) \right],$$

where $\tilde{\eta}_t = \sum_{s=1}^{t} \eta_s$. The solution is

$$x_t = \text{prox}^h_{\tilde{\eta}_t g}(\nabla h(x_0) - \sum_{s=1}^{t} \eta_s \nabla f(x_{s-1})).$$

Here the variable

$$\tilde{\alpha}_t = \nabla h(x_0) - \sum_{s=1}^{t} \eta_s \nabla f(x_{s-1})$$

is referred to as dual variable, and thus the resulting regularized dual averaging (RDA) method is

Algorithm 7: Regularized Dual Averaging (RDA)

Input: $f(\cdot)$, $g(\cdot)$, x_0, h and $\{\eta_t\}$
Let $p_0 \in \partial h(x_0)$
Let $\tilde{\eta}_0 = \eta_0$
for $t = 1, 2, \ldots, T$ **do**
    Let $\tilde{\eta}_t = \tilde{\eta}_{t-1} + \eta_t$
    Let $p_t = p_{t-1} - \eta_t \nabla f(x_{t-1})$
    Let $x_t = \text{prox}^h_{\tilde{\eta}_t g}(p_t)$
end for
return x_T

10 Convex Duality Properties

10.1 Conjugate Function

The **conjugate function** (or dual) is defined as:

$$f^*(x) = \sup_y \left[y^\intercal x - f(y) \right],$$

which is convex even if $f(\cdot)$ is not convex. It is illustrated in Figure 5 ($f^*(y)$).

By definition, we always have the following Fenchel's inequality:

$$f^*(x) + f(y) \geq y^\intercal x.$$

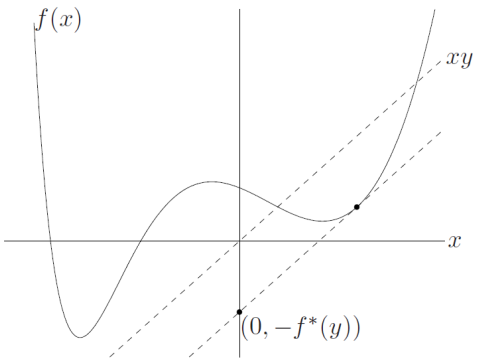


Figure 3.8 A function $f : \mathbf{R} \to \mathbf{R}$, and a value $y \in \mathbf{R}$. The conjugate function $f^*(y)$ is the maximum gap between the linear function yx and $f(x)$, as shown by the dashed line in the figure. If f is differentiable, this occurs at a point x where $f'(x) = y$.

Figure 5: Illustration of conjugate functions (from Boyd's book). The maximum distance along the vertical axis is equivalent to the maximum gap between two dashed lines.

Theorem 20. *If $f(x)$ is a closed convex function, then*

$$f^{**}(x) = f(x).$$

Moreover, for any pair (x, y), the following conditions are equivalent:[1]

- $x \in \partial f(y)$
- $y \in \partial f^*(x)$
- (x, y) satisfies the equality

$$f^*(x) + f(y) = y^\intercal x.$$

[1] When the "=" can be achieved, i.e., the supremum is equal to the maximum.

Given a norm $\|\cdot\|$, its **dual norm** is defined as:

$$\|y\|_* = \sup_{\|x\| \leq 1} y^\intercal x,$$

leading to the following inequality:

$$y^\intercal x = \|x\| \left(y^\intercal \frac{x}{\|x\|} \right) \leq \|x\| \|y\|_*$$

for all x and y.

10.2 Smoothness and Strong Convexity

We may generalize the smoothness and strong convex with respect to the $\|\cdot\|_2$ norm to an arbitrary norm as follows.

Definition 21. A function is L-smooth with respect to a norm $\|\cdot\|$ if

$$f(x') \leq f(x) + \nabla f(x)^\intercal (x' - x) + \frac{L}{2} \| x' - x \|^2$$

for all x and $x^{'}$.

A function is λ-strongly convex with respect to a norm $\|\cdot\|$ if

$$f(x^{'}) \geq f(x) + \nabla f(x)^{\intercal}(x^{'} - x) + \frac{\lambda}{2} \| x^{'} - x \|^2$$

for all x and $x^{'}$.

Theorem 22. *Consider a norm $\|\cdot\|$ and its dual norm $\|\cdot\|_*$. If $f(x)$ is L-smooth with respect to $\|\cdot\|$, then $f^*(y)$ is L^{-1} strongly convex with respect to $\|\cdot\|_*$. Similarly, if $f(x)$ is λ-strongly convex with respect to $\|\cdot\|$, then $f^*(y)$ is λ^{-1} smooth with respect to $\|\cdot\|_*$.*

10.3 Bregman Divergence

Given convex function f, and let

$$D_f(x^{'}, x) = f(x^{'}) - f(x) - y^{\intercal}(x^{'} - x)$$

be its Bregman divergence, where $y \in \partial f(x)$.

Theorem 23. *Let $y \in \partial f(x)$ and $y^{'} \in \partial f(x^{'})$. Then*

$$\begin{aligned} D_f(x^{'}, x) &= f(x^{'}) - f(x) - y^{\intercal}(x^{'} - x) \\ &= f^*(y) - f^*(y^{'}) - (x)^{'\intercal}(y - y^{'}) \\ &= D_{f^*}(y, y^{'}). \end{aligned}$$

10.4 Moreau's Identity

Given $g(x)$, we denote the proximal mapping by

$$\text{prox}_g(x) = \arg\min_z \left[\frac{1}{2} \| z - x \|_2^2 + g(z) \right].$$

$$\text{prox}_g(x) + \text{prox}_{g^*}(x) = x$$

It is a key relationship between proximal mapping and duality, and a generalization of orthogonal decomposition.[1]

[1] Refer to PROXIMAL GRADIENT METHODS by Yuxin Chen.

10.5 Fenchel's Duality

Consider the composite optimization problem

$$\phi(x) = f(x) + g(x),$$

and let x_* be its solution. We have the following important property:

$$\begin{aligned} \phi(x_*) = f(x_*) + g(x_*) &\geq \min_{x, x^{'}} \left[f(x) + g(x^{'}) + \alpha^{\intercal}(x - x^{'}) \right] \\ &= -f^*(-\alpha) - g^*(\alpha) \\ &= \phi_D(\alpha) \end{aligned}$$

where the inequality follows the Lagrangian, and the equality follows the definition of conjugate functions.

Theorem 24. *Given* $\alpha_* \in \arg\max_\alpha \phi_D(\alpha)$*, there exists* $x_* \in \arg\max_x \phi(x)$ *such that*

$$x_* \in \partial g^*(\alpha_*), \quad x_* \in \partial f^*(-\alpha_*),$$

and

$$\phi(x_*) = \phi_D(\alpha_*).$$

11 Dual and Primal-Dual methods

This section refers to CONVEX DUALITY by Tong Zhang, DUAL AND PRIMAL-DUAL METHODS by Yuxin Chen, and A TUTORIAL ON PRIMAL-DUAL ALGORITHM by Shenlong Wang. I put it into a separate section because the development of the theory provided by Tong Zhang is not clear enough.

11.1 Derivation

The target problem is still

$$\min_x \quad \phi(x) = f(x) + g(x),$$

where f and g are convex. We add an auxiliary variable x' and get an equivalent problem

$$\begin{aligned} \min_{x,z} \quad & f(x) + g(z) \\ \text{subject to} \quad & x = z. \end{aligned}$$

Next, we have a dual formulation by Lagrangian

$$\max_\alpha \min_{x,z} \quad f(x) + g(z) + \langle \alpha, x - z \rangle ,$$

and it can be further simplified with the introduction of conjugate functions:

$$\max_\alpha \min_x \quad f(x) + \langle \alpha, x \rangle - g^*(\alpha) \tag{19}$$

$$\max_\alpha \quad -f^*(-\alpha) - g^*(\alpha) \Leftrightarrow \min_\alpha \quad f^*(-\alpha) + g^*(\alpha) \tag{20}$$

Here (19) is the target of the primal dual method[1], while (20) is the target of the dual method[2].

[1] It contains both the primal variable and the dual variable.
[2] It contains only the dual variable.

11.2 Dual Method

To solve the dual problem (20), one can use the dual proximal gradient algorithm[3]. This dual method is useful if

- the proximal operator w.r.t. g is cheap (then we can use the Moreau's identity for $\text{prox}_{g^*}(x) = x - \text{prox}_g(x)$)
- f is smooth (or if f is strongly convex)

[3] There is a proximal dual ascent method in LAGRANGIAN DUALITY AND DUAL DECOMPOSITION METHODS, which has the same idea but for the general optimization problem.

11.3 Primal-Dual Method

We'll first examine the optimality condition for (19),[1]

$$\begin{cases} \alpha \in & -\partial f(x) \\ x \in & \partial g^*(\alpha), \end{cases}$$

and the key idea of the primal-dual method is to iteratively update (α, x) to reach an optimal point. There are different designs of the algorithm, and the book provides the following one:

[1] The same for dual problem as:

$$\begin{cases} x \in & \partial f^*(-\alpha) \\ x \in & \partial g^*(\alpha). \end{cases}$$

Algorithm 8: Primal-Dual Ascent Method

Input: $f(\cdot)$, $g(\cdot)$, x_0, $\eta_1, \eta_2, \dots$ and α_0
Output: x_T

1: **for** $t = 1, \dots, T$ **do**
2: Let $\alpha_t = (1 - \eta_{t-1})\alpha_{t-1} - \eta_{t-1}\nabla f(x_{t-1})$
3: Let $x_t = \arg\max_x [\alpha_t^\intercal x - g(x)] = \nabla g^*(\alpha_t)$
4: **end for**
5: **return** x_T

The update of α_t includes momentum.

Yuxin Chen provides the primal-dual proximal gradient method that takes advantage of both prox_f and prox_g.

12 Dual Decomposition and Dual Ascent Algorithm

Target at the following formulation,

$$\begin{aligned} \min_{x,z} \quad & \phi(x,z) = f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c, \end{aligned}$$

where two primal variables are coupled in the constraint.

12.1 Dual Decomposition

Its Lagrangian function is:

$$L(x, z, \alpha) = \phi(x, z) + \alpha^\intercal(Ax + Bz - c). \tag{21}$$

Its dual objective is ($\alpha \geq 0$):

$$\begin{aligned} \max_\alpha \phi_D(\alpha) &= \max_\alpha \left[\min_{x,z} f(x) + g(z) + \alpha^\intercal(Ax + Bz - c) \right] \\ &= \max_\alpha -f^*(-A^\intercal\alpha) - g^*(-B^\intercal\alpha) - c^\intercal\alpha \end{aligned} \qquad \alpha \in C_D, \tag{22}$$

where C_D is the domain of $\phi_D(\cdot)$. By taking the dual, we decompose a complex problem into multiple simpler problems.

12.2 *Dual Ascent Algorithms*

The optimality condition of (22) is

$$\begin{aligned} 0 &\in A\partial f^*(-A^\intercal\alpha) + B\partial g^*(-B^\intercal\alpha) - c \\ &\in Ax_t + Bz_t - c, \end{aligned}$$

where

$$\begin{aligned} x_t &= \arg\min_x \left[-\alpha_{t-1}^\intercal Ax + f(x)\right] \\ z_t &= \arg\min_z \left[-\alpha_{t-1}^\intercal Bz + g(z)\right]. \end{aligned}$$

In dual ascent algorithm, we have the update

$$\alpha_t = \alpha_{t-1} + \eta_t(Ax_t + Bz_t - c). \tag{23}$$

The dual ascent method can be written as

Algorithm 9: Dual Ascent Method

Input: $\phi(\cdot)$, A, B, α_0, c $\eta_1, \eta_2, \ldots$
Output: x_T

1: **for** $t = 1, \ldots, T$ **do**
2: Let $x_t = \arg\min_x \left[-\alpha_{t-1}^\intercal Ax + f(x)\right]$
3: Let $z_t = \arg\min_z \left[-\alpha_{t-1}^\intercal Bz + g(z)\right]$
4: Let $\alpha_t = \text{proj}_{C_D}\left(\alpha_{t-1} + \eta_t[Ax_t + Bz_t - c]\right)$
5: **end for**
6: **return** x_T

Here we will introduce the proximal dual ascent method.[1] In this case, we can assume that $g(\cdot)$ is strongly convex. It means that $g^*(\cdot)$ is smooth, and thus, we can use an upper bound of $g^*(\cdot)$ for proximal iteration and write (22) as follow:

[1] I am not sure the relationship between the upper bound and the update of x_t in the note.

$$\alpha_t = \arg\max_\alpha [-f^*(-A^\intercal\alpha) \underbrace{-g^*(-B^\intercal\alpha_{t-1}) - (-Bz_t)^\intercal(\alpha - \alpha_{t-1}) - \frac{1}{2\eta_t}\|\alpha - \alpha_{t-1}\|_2^2}_{-\text{upper bound of } g^*(\cdot)} - c^\intercal\alpha]. \tag{24}$$

There exists $x_t \in \partial f^*(-A^\intercal\alpha_t)$ so that

$$\alpha_t = \alpha_{t-1} + \eta_t(Ax_t + Bz_t - c),$$

the first order condition of (24). Since $-A^\intercal\alpha_t \in \partial f(x_t)$, we obtain

$$0 \in (\partial f(x_t) + A^T\alpha_{t-1}) + \eta_t A^\intercal(Ax_t + Bz_t - c),$$

which is the first order condition of the following optimization problem

$$x_t = \arg\min_x \left[\alpha_{t-1}^\intercal Ax + \frac{\eta_t}{2}\|Ax + Bz_t - c\|_2^2 + f(x)\right],$$

which is a proximal method. This leads to proximal dual ascent method

Compared with the first algorithm, the second one updates (x, z) iteratively but not simultaneously. In practice, one often employs an improved version of the second one called ADMM.

Algorithm 10: Proximal Dual Ascent Method (I rewrite the update of α_t since what the book writes is unreasonable.)

Input: $\phi(\cdot)$, A, B, α_0, c $\eta_1, \eta_2, \ldots$
Output: x_T

1: **for** $t = 1, \ldots, T$ **do**
2: Let $z_t = \arg\min_z \left[-\alpha_{t-1}^\intercal Bz + g(z)\right]$
3: Let $x_t = \arg\min_x \left[\alpha_{t-1}^\intercal Ax + 0.5\eta\|Ax + Bz_t - c\|_2^2 + f(x)\right]$
4: Let $\alpha_t = \text{proj}_{C_D}\left(\alpha_{t-1} + \eta_t[Ax_t + Bz_t - c]\right)$
5: **end for**
6: **return** x_T

13 *Alternating Direction Method of Multipliers*

Target at the same formulation,

$$\begin{aligned} \min_{x,z} \quad & \phi(x,z) = f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c. \end{aligned}$$

13.1 *Method of Multipliers*

To improve convergence, one often employs the augmented Lagrangian function instead of the regular Lagrangian function, where the primal problem is reformulated as an **equivalent** form:

$$\begin{aligned} \min_{x,z} \quad & \phi_\rho(x,z) \coloneqq \phi(x,z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 \\ \text{subject to} \quad & Ax + Bz = c. \end{aligned}$$

Its Lagrangian function is the augmented Lagrangian function:

$$L_\rho(x,z,\alpha) = \phi(x,z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2 + \alpha^\intercal(Ax + Bz - c). \tag{25}$$

Compared with the traditional Lagrangian function (21), the augmented Lagrangian function adds dual smoothness: it is ρ-strongly convex in $Ax + Bz$, which implies ρ^{-1} smoothness for the dual.

Algorithm 11: Method of Multipliers

Input: $\phi(\cdot)$, A, B, c, ρ, α_0
Output: x_T

1: **for** $t = 1, \ldots, T$ **do**
2: Let $[x_t, z_t] = \arg\min_{x,z} \left[-\alpha_{t-1}^\intercal[Ax + Bz] + \phi(x,z) + \frac{\rho}{2}\|Ax + Bz - c\|_2^2\right]$
3: Let $\alpha_t = \text{proj}_{C_D}\left(\alpha_{t-1} + \rho[Ax_t + Bz_t - c]\right)$
4: **end for**
5: **return** x_T

Benefit: It converges better than the dual ascent algorithm because of the added dual smoothness.[1]

Shortage: It is difficult to apply for many practical problems because x and z are coupled.

[1] I am confused about the relationship between 'converge better', 'convergence rate', and 'step size'.

13.2 ADMM

ADMM (Alternating Direction Method of Multipliers) remedies the difficulty of dealing with a coupled problem in the Method of Multipliers. Instead of solving a joint problem, we decouple x and z by solving the problem sequentially (alternating direction).

Algorithm 12: Alternating Direction Method of Multipliers (ADMM)

Input: $\phi(\cdot)$, A, B, c, ρ, α_0, x_0, z_0
Output: x_T

1: **for** $t = 1, \ldots, T$ **do**
2: Let $z_t = \arg\min_z \left[\alpha_{t-1}^\intercal Bz + g(z) + \frac{\rho}{2}\|Ax_{t-1} + Bz - c\|_2^2\right]$
3: Let $x_t = \arg\min_x \left[\alpha_{t-1}^\intercal Ax + f(x) + \frac{\rho}{2}\|Ax + Bz_t - c\|_2^2\right]$
4: Let $\alpha_t = \text{proj}_{C_D}\left(\alpha_{t-1} + \rho[Ax_t + Bz_t - c]\right)$
5: **end for**
6: **return** x_T

The solutions of x_t and z_t involve the following optimization problems:

$$\min_x \left[\frac{\rho}{2}\|Ax - \tilde{x}\|_2^2 + f(x)\right]$$
$$\min_z \left[\frac{\rho}{2}\|Bz - \tilde{z}\|_2^2 + g(z)\right].$$

Convergence and Stopping Criteria[1]**:** Assume

[1] Refer to Vini's talk and Boyd's book Boyd et al. [2011].

- f, g are convex, closed, proper
- L_0(unaugmented Lagrangian) has a saddle point

then ADMM converges:

- iterates approach feasibility: $Ax_t + Bz_t - c \to 0$
- objective approaches optimal value: $\phi(x_t, z_t) \to p^*$
- false (in general) statements: x converges, z converges.
- true statement: α converges.

In practice, ADMM is often slow to converge to high accuracy, but often converges to moderate accuracy within a few dozens of iterations, which is sufficient for most practical purposes.

Part IV

Stochastic Methods

14 Stochastic Dual Coordinate Ascent

Still target at the composite optimization problem, but with a finite sum structure:

$$\min_{w} \quad \phi(w) = \frac{1}{n}\sum_{i=1}^{n} f_i(X_i^\intercal w) + \lambda g(w), \tag{26}$$

where

- $w \in \mathbb{R}^d$ is the model parameter;
- $X_1, \ldots, X_n$ are training data in $\mathbb{R}^{d\times k}$ (k samples in a batch, and each sample has d features);
- $f_1, \ldots, f_n$ are a sequence of convex loss functions $\mathbb{R}^d \to \mathbb{R}$;
- g is a convex regularization function, and $\lambda \geq 0$.

14.1 Dual Formulation

The problem (26) can be reformulated as

$$\min_{\omega} \quad \phi(w, \{u_i\}) = \frac{1}{n}\sum_{i=1}^{n} f_i(u_i) + \lambda g(w), \qquad \text{s.t.} \quad X_i^\intercal w = u_i,$$

and its Lagrangian function is

$$L(w, \{u_i\}, \alpha) = \frac{1}{n}\sum_{i=1}^{n} f_i(u_i) + \lambda g(w) + \frac{1}{n}\sum_{i=1}^{n} \alpha_i^\intercal (u_i - X_i^\intercal w).$$

The dual objective function is

$$\phi_D(\alpha) = \min_{w,\{u_i\}} L(w, \{u_i\}, \alpha) = \frac{1}{n}\sum_{i=1}^{n} -f_i^*(-\alpha_i) - \lambda g^*\left(\frac{1}{\lambda n}\sum_{i=1}^{n} X_i\alpha_i\right), \tag{27}$$

with the optimal solution:

$$w^* \in \partial g^*\left(\frac{1}{\lambda n}\sum_{i=1}^{n} X_i\alpha_i^*\right).$$

14.2 Proximal SDCA

The dual coordinate ascent (DCA) maximizes the dual objective $\phi_D(\alpha)$ by optimizing one α_i at a time for a chosen i, like an instance of ADMM.

The stochastic dual coordinate ascent (SDCA) chooses α_i uniformly at random.

Here we introduce the proximal stochastic dual coordinate ascent (Prox-SDCA)[1],

[1] See the slide - Proximal Stochastic Dual Coordinate Ascent by Shai Shalev-Shwartz and Tong Zhang.

a stochastic version of SDCA. "Stochastic" means we perform randomly in choosing training sets by choosing corresponding α_i's. Assume $g(\cdot)$ is 1-strongly convex in norm $\|\cdot\|_P$, so $g^*(\cdot)$ is 1-smooth in dual norm $\|\cdot\|_D$. Update each α_i according to the prox-dual:

$$\begin{aligned}\Delta\alpha_i &= \arg\max_{\Delta\alpha_i} -\frac{1}{n} f_i^*(-(\alpha_i+\Delta\alpha_i)) \\ &\quad - \lambda\left(\underbrace{g^*(v^{(t-1)}) + \nabla g^*(v^{(t-1)})^\intercal\left(\frac{1}{\lambda n}X_i\Delta\alpha_i\right) + \frac{1}{2}\|\frac{1}{\lambda n}X_i\Delta\alpha_i\|_D^2}_{\text{upper bound of } g^*(\cdot)}\right) \\ &= \arg\max_{\Delta\alpha_i} -f_i^*(-(\alpha_i+\Delta\alpha_i)) - \left(w^{(t-1)}\right)^\intercal X_i\Delta\alpha_i - \frac{1}{2\lambda n}\|X_i\Delta\alpha_i\|_D^2 \\ v &= \frac{1}{\lambda n}\sum_{i=1}^n X_i\alpha_i.\end{aligned} \tag{28}$$

Prox-SDCA randomly picks i and update $\Delta\alpha_i$.

What is proximal: For complex $g^*(\cdot)$, directly maximizing the dual objective $\phi_D(\alpha)$ may not be easy. Instead, we try to construct a lower bound by finding an upper bound of $g^*(\cdot)$. The proximal method achieves this by the quadratic approximation (the quadratic term $\|\Delta\alpha_i\|_D^2$ is called the proximal term).

In previous lectures, the proximal has more specific meanings.The gradient descent applies quadratic approximation indirectly, though it originates from the gradient view. The proximal gradient descent considers the regularization term and starts from a view of quadratic approximation. It further encodes the reformulated bi-criterion optimization (f, g) into a proximal operator.

15 *Randomized Proximal Coordinate Descent*

Different to SDCA and Prox-SDCA which adds randomness in selecting training sets by considering the dual problem, the randomized coordinate descent adds randomness in selecting features by considering the primal problem. "Stochastic" $\to$ training sets; "randomized" $\to$ feature sets. The target problem has a model parameter $w \in \mathbb{R}^d$ which is split into p pieces $w = [w_1, \ldots, w_p]$. Data $A \in \mathbb{R}^{k\times d}$ is also split into p pieces accordingly alone the feature axis[1]

$$\min_w \quad \phi(w) = f(w) + g(w),$$

where

$$f(w) = \psi\left(\sum_{j=1}^p A_j w_j\right), \quad g(w) = \sum_{j=1}^p g_j(w_j).$$

We assume that $f(\cdot)$ is L_i-smooth with respect to w_i,[2] and $g(\cdot)$ is convex but may not be smooth. Note that if $\psi(\cdot)$ is L-smooth, and $\|A_i\|_2$ is the spectral norm[3] of A_i, then $L_i \le \|A_i\|_2^2 \cdot L$.

[1] In previous lecture, $X_i \in \mathbb{R}^{d\times k}$. The notations are different, which requires attention when dealing with regression problems.

[2] $i \in [1, p]$

[3] The spectral norm of a matrix $\mathbf{A}$ is the largest singular value. $\|\mathbf{A}\|_2 = \sqrt{\lambda_{\max}(\mathbf{A}^*\mathbf{A})} = \sigma_{\max}(\mathbf{A})$.

To solve this problem, the randomized proximal coordinate descent randomly select a variable i from 1 to p, and minimize the objective with respect to w_i using proximal gradient. That is, we select i, and optimize with respect to $w_i + \Delta w_i$:

$$\psi\left(\sum_{j=1}^{p} A_j w_j + A_i \Delta w_i\right) + \sum_{j=1}^{p} g_j(w_j + \Delta w_i \delta_i^j).$$

Given $\eta_i \leq 1/L_i$, we use an upper bound of $f(\cdot)$,

$$\psi\left(\sum_{j=1}^{p} A_j w_j\right) + \nabla\psi\left(\sum_{j=1}^{p} A_j w_j\right)^\intercal (A_i \Delta w_i) + \frac{1}{2\eta_i}\|\Delta w_i\|_2^2 + g_i(w_i + \Delta w_i).$$

Let

$$u = \sum_{j=1}^{p} A_j w_j,$$

then we can optimize

$$\begin{aligned}
\Delta w_i &= \arg\min_{\Delta w}\left[\left(A_i^\intercal \nabla f(u)\right)^\intercal \Delta w + \frac{1}{2\eta_i}\|\Delta w\|_2^2 + g_i(w_i + \Delta w)\right] \\
&= \arg\min_{\Delta w}\left[\frac{1}{2\eta_i}\|\Delta w + \eta_i A_i^\intercal \nabla f(u)\|_2^2 + g_i(w_i + \Delta w)\right] \\
&= \text{prox}_{\eta_i, g_i}\left(w_i - \eta_i A_i^\intercal \nabla f(u)\right) - w_i,
\end{aligned}$$

and the proximal operator is defined as

$$\text{prox}_{\eta_i, g_i}(w) = \arg\min_{z \in \mathbb{R}^{d_i}}\left[\frac{1}{2}\|z - w\|_2^2 + \eta_i g_i(z)\right].$$

The proximal operator = a new optimization problem to find a point that is closest to the target point with respect to the L_2 distance and a penalty.

16 *Stochastic Gradient Descent*

Target at the problem:

$$\min_w \quad \phi(w) = F(w) + g(w), \quad \underbrace{F(w) = \mathbb{E}_{\xi \sim D} f(w; \xi)}_{\text{expected error}},$$

where ξ is a random variable drawn from a distribution D of the training data, the randomness in problem. Suppose $f(w; \xi)$ is convex for every ξ, and hence $F(w)$ is convex. The expectation removes the randomness in training data.[1]

One common example of $F(w)$ is the empirical risk:

$$F(w) = \frac{1}{n}\sum_{i=1}^{n} f(w; \{x_i, y_i\}) = \frac{1}{n}\sum_{i=1}^{n} f_i(w)$$

[1] Similar to my ML guide: expected test error Err removes the randomness in test error $\text{Err}_{\mathcal{T}}$.

16.1 Stochastic Proximal Gradient Descent

The problem is that the D may be unknown; even if it is known, evaluating high-dimensional expectation is often expensive. Instead, we picks $\xi_t = (x, y)$ at a time, and works with this data point, where its gradient $g(w_t; \xi_t)$ is an unbiased estimate of the gradient of the expected error:

$$\mathbb{E}_{\xi_t}[g(w_t; \xi_t)] = \nabla F(w_t). \tag{29}$$

The algorithm is almost the same as Proximal Gradient Descent in Algorithm 5. The only difference is replacing $\nabla f(w_t)$ with $g(w_t; \xi_t)$.

Algorithm 13: Stochastic Proximal Gradient Descent (Proximal SGD)

Input: $f(\cdot)$, $g(\cdot)$, w_0, and $\eta_1, \eta_2, \ldots$
Output: w_T

1: **for** $t = 1, \ldots, T$ **do**
2: Randomly pick $\xi \sim D$
3: $w_t = \text{prox}_{\eta_t, g}(w_{t-1} - \eta_t \nabla g(w_{t-1}; \xi_{t-1}))$
4: **end for**
5: **return** w_T

In the finite sample case, each full gradient computation is

$$\nabla F(w) = \frac{1}{n} \sum_{i=1}^{n} \nabla f_i(w),$$

which requires n gradient evaluations per iteration. In contrast, Proximal SGD (or simply SGD) requires to compute only one

$$g(w; \xi) = \nabla f_i(w) \tag{30}$$

per iteration, and have the same speed of convergence.

16.2 Minibatch SGD

In practice, it is often more efficient to work with a minibatch $\mathcal{B}$ of m training samples instead of one data point per iteration, i.e., $\xi_t = \mathcal{B}_t$. The minibatch gradient is

$$g_{\mathcal{B}}(w; \xi) = \frac{1}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \nabla f_i(w),$$

which is also unbiased

$$\mathbb{E}_{\xi_t}[g_{\mathcal{B}}(w_t; \xi_t)] = \nabla F(w_t).$$

Then the minibatch SGD algorithm replaces $g(w_t; \xi_t)$ with $g_{\mathcal{B}}(w_t; \xi_t)$.

16.3 Convergence Analysis (Tong Zhang)

Theorem 25. *Consider minibatch SGD of size m. If $F(w)$ is convex and L smooth, $g(w)$ is convex. Let*

$$V = \sup_{w \in C} \mathbb{E}_{\xi \sim D} \|\nabla F(\xi, w) - \nabla F(w)\|_2^2.$$

If we choose $\eta_t < 1/L$ for all t, then for all $w \in C$

$$\sum_{t=1}^{T} \eta_t \mathbb{E}\left[\phi(w_t) - \phi(w)\right] \leq \sum_{t=1}^{T} \frac{\eta_t^2 V}{2\left(1 - \eta_t L\right) m} + \frac{1}{2}\|w - w_0\|_2^2.$$

16.4 Convergence Analysis (Yuxin Chen)

This convergence analysis is provided for problem:[1]

[1] See STOCHASTIC GRADIENT by Yuxin Chen.

$$\min_w \quad F(w), \quad \underbrace{F(w) = \mathbb{E}_{\xi \sim D} f(w; \xi)}_{\text{expected error}},$$

where we have the assumptions:

- $f(\cdot)$: λ-strongly convex, L-smooth
- $g(w_t; \xi_t)$ is an unbiased estimate of $\nabla F(w_t)$ given $\left\{\xi^0, \cdots, \xi^{t-1}\right\}$
- for all w,
$$\mathbb{E}_{\xi}\left[\|g(w;\xi)\|_2^2\right] \leq \sigma_g^2 + c_g\|\nabla F(w)\|_2^2,$$
which means there is a upper bound for the variance of the gradient estimator.

Theorem 26. *(Convergence of SGD for strongly convex problems; fixed stepsizes) Under the assumptions above, if $\eta_t \equiv \eta \leq \frac{1}{Lc_g}$, then SGD achieves*

$$\mathbb{E}\left[F(w_t) - F(w_*)\right] \leq \frac{\eta L \sigma_g^2}{2\lambda} + (1 - \eta\lambda)^t \left(F(w_0) - F(w_*)\right).$$

We have the following information:

- fast (linear) convergence at the very beginning[2]
- converges to some neighborhood of w_* - variation in gradient computation prevents further progress
- when gradient computation is noiseless (i.e. $\sigma_g = 0$), it converges linearly to optimal points
- smaller stepsizes η yield better converging points[3], which inspires us to reduce stepsizes whenever progress stalls.

[2] The first term is omitted, and it is same as GD shown in Theorem **??**.

[3] a smaller first term

Figure 6 illustrates the diminishing stepsizes strategy.

17 Variance Reduction

[4]Consider the finite-sum problem (no regularization):

[4] This chapter comes from VARIANCE REDUCTION by Yuxin Chen.

$$\min_w \quad F(w) := \frac{1}{n}\sum_{i=1}^{n} f_i(w),$$

where $f_i(\cdot)$ is convex and L-smooth, and $F(\cdot)$ is λ-strongly convex.

Problems with SGD:

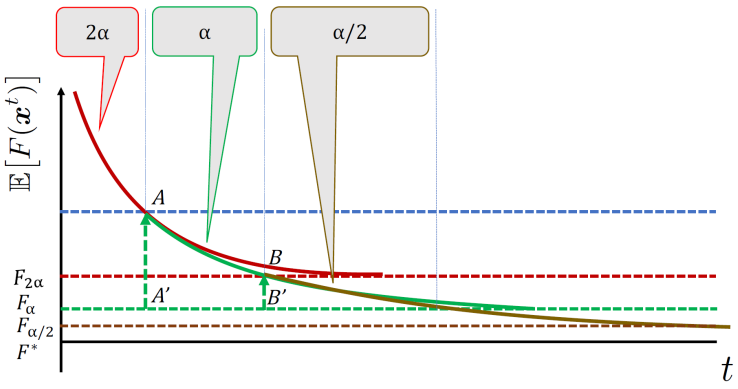


Figure 6: Diminishing stepsizes strategy.

- SGD with long stepsizes ($\eta_t = 1$) poorly suppresses noise, which tends to oscillate around the global minimizers due to the noisy nature of gradient computation.
- Choosing $\eta_t = 1/t$ mitigates oscillation, but is too conservative.

Recall that in SGD, the gradient of the expected error is approximated by the gradient of a training data (30), denoted as

$$g(w_t; \xi_t) = \nabla f_i(w_t).$$

From Theorem 26, we know that the problems with SGD above can be alleviated by reducing variability, so that

$$\mathbb{E}_\xi \left[\|g(w;\xi)\|_2^2 \right] \leq \mathbb{E}_\xi \left[\|\nabla f_i(w_t)\|_2^2 \right].$$

17.1 *Stochastic Variance Reduced Gradient*

A simple idea to reduce variability is that

$$g(w_t; \xi_t) = \nabla f_{i_t}(w_t) - v_t,$$

where v_t is zero-mean and positively correlated with $\nabla f_i(w_t)$.

Stochastic variance reduced gradient (SVRG) specifics it as

$$g(w_t; \xi_t) = \underbrace{\nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{\text{old}})}_{\to 0 \text{ if } w_t \approx w_{\text{old}}} + \underbrace{\nabla F(w_{\text{old}})}_{\to 0 \text{ if } w_{\text{old}} \approx w_*}, \tag{31}$$

which

- is an unbiased estimate of $\nabla F(w_t)$
- converges to 0 if $w_t \approx w_{\text{old}} \approx w_*$; the variability is reduced.[1]

[1] $\nabla f_{i_t}(w_t)$ may not converge to 0.

Intuition: If the current iterate is not too far away from previous iterates, then historical gradient info might be useful in producing such a v_t to reduce variance.

Algorithm 14: SVRG for finite-sum optimization

for $s = 1, 2, \ldots,$ epoch **do**
  batch gradient: $w_{\text{old}}^s \leftarrow w_m^{s-1}$, and compute $\nabla F(w_{\text{old}}^s)$
  initialize $w_0^s \leftarrow w_{\text{old}}^s$
  for $t = 0, \ldots, m-1$ **do**
    choose i_t uniformly from $\{1, \ldots, n\}$, and
    $w_{t+1}^s = w_t^s - \eta \left\{ \nabla f_{i_t}(w_t^s) - \nabla f_{i_t}(w_{\text{old}}^s) + \nabla F(w_{\text{old}}^s) \right\}$
  end for
end forreturn w_T

In other words, we calculate the batch gradient at the beginning to increase confidence, although we only calculate the stochastic gradient from one sample in the inner iteration within each epoch.

Each epoch contains $n + 2m$ gradient computations:

- the batch gradient is computed only once every m iterations
- the average per-iteration cost of SVRG is comparable to that of SGD if $m \gtrsim n$

17.2 *Proximal SVRG*

Consider the finite-sum optimization with regulation:

$$\min_w \quad F(w) + g(w), \quad F(w) = \frac{1}{n} \sum_{i=1}^n f_i(w).$$

Like the gradient descent method shown in (17), the only difference made by proximal SVRG compared with SVRG is to replace

$$w_{t+1}^s = w_t^s - \eta \left\{ \nabla f_{i_t}(w_t^s) - \nabla f_{i_t}(w_{\text{old}}^s) + \nabla F(w_{\text{old}}^s) \right\}$$

in Algorithm 14 with

$$w_{t+1}^s = \text{prox}_g \left(w_t^s - \eta \left\{ \nabla f_{i_t}(w_t^s) - \nabla f_{i_t}(w_{\text{old}}^s) + \nabla F(w_{\text{old}}^s) \right\} \right).$$

17.3 *Stochastic Recursive Gradient Algorithm*

Stochastic Recursive Gradient Algorithm (SARAH) is a variation of SVRG.

Key idea: recursive / adaptive updates of gradient estimates:

$$\begin{aligned} g_t &= \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{t-1}) + g_{t-1} \\ w_{t+1} &= w_t - \eta g_t, \end{aligned}$$

compared to SVRG (use a fixed snapshot point w_{old} for the entire epoch)

$$g_t = \nabla f_{i_t}(w_t) - \nabla f_{i_t}(w_{\text{old}}) + \nabla F(w_{\text{old}}).$$

Part V

Common Optimization Methods

18 Water-Filling

The water-filling algorithm is useful in power allocation. The constrained optimization problem for capacity-achieving is

$$\begin{aligned} \min_{\{p_i\}} \quad & \sum_{i=1}^{L} \log(1 + p_i c_i) \\ \text{s.t.} \quad & \sum_{i=1}^{L} p_i = P_T \\ & p_i \geq 0,\ 1 \leq i \leq L \end{aligned} \tag{32}$$

and its water filling solution is

$$p_i^* = \left(\frac{1}{\mu} - \frac{1}{c_i}\right)^+,\ 1 \leq i \leq L \tag{33}$$

where L is the number of subchannels, c_i is the gain of the i-th subchannel, and $\frac{1}{\mu}$ is the water level chosen to satisfy the power constraint with equality $\sum_{i=1}^{L} p_i = P_T$.

18.1 Derivation

Here we derive the water filling solution of (32). Its Lagrangian function is

$$L(\{p_i\}, \lambda_i, \mu) = \sum_{i=1}^{L} \log(1 + p_i c_i) - \sum_{i=1}^{L} \lambda_i p_i - \mu \left(\sum_{i=1}^{L} p_i - P_T\right).$$

The dual function is then given by

$$g(\lambda_i, \mu) = \sup_{\{p_i\}} L(\{p_i\}, \lambda_i, \mu),$$

and the zero gradient property is

$$\begin{aligned} \frac{\partial L}{\partial p_i} =& \frac{c_i}{1 + p_i c_i} - \lambda_i - \mu = 0 \\ \Rightarrow & 1 + p_i c_i = \frac{c_i}{\lambda_i + \mu} \\ \Rightarrow & p_i = \frac{1}{\lambda_i + \mu} - \frac{1}{c_i} \\ \Rightarrow & p_i^* = \left(\frac{1}{\mu} - \frac{1}{c_i}\right)^+. \end{aligned}$$

Since we manually requires $p_i^* \geq 0$, the constraint $p_i \geq 0$ is redundant, and so as the λ_i. A deep fading subchannel with a large $\frac{1}{c_i}$ will have a small p_i.

19 Active-set Method for QP

A general quadratic program (QP) has the form

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^\intercal Gx + x^\intercal c \\ \text{s.t.} \quad & a_i^\intercal x = b_i,\ i \in \mathcal{E} \\ & a_i^\intercal x \geq b_i,\ i \in \mathcal{I}. \end{aligned}$$

19.1 Primal Method

The beauty of the primal active set method is that it turns a inequality-constraint problem into a sequence of equality-constrained problem,

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^\intercal Gx + x^\intercal c \\ \text{s.t.} \quad & a_i^\intercal x = b_i,\ i \in \mathcal{W}_k, \end{aligned}$$

assuming that

$$\begin{aligned} \min_x \quad & q(x) = \frac{1}{2}x^\intercal Gx + x^\intercal c \\ \text{s.t.} \quad & a_i^\intercal x = b_i,\ i \in \mathcal{W}_k \\ & a_i^\intercal x \geq b_i,\ i \notin \mathcal{W}_k. \end{aligned} \tag{34}$$

- Given an iterate x_k and the working set $\mathcal{W}_k$, we compute a step (direction) p_k by solving an equality-constrained QP subproblem in which the constraints corresponding to the working set $\mathcal{W}_k$ are regarded as equalities and all other constraints are temporarily disregarded.
- If $p \neq 0$: We get an update direction p_k for x_k. However $x_k + p_k$ may not be feasible because it may violate other constraints $i \notin \mathcal{W}_k$, which requires to determine the step-length α_k.
  - If $x_k + p_k$ is feasible, then we set $x_{k+1} = x_k + p_k$.
  - If $x_k + p_k$ is infeasible, we have to carefully choose a step length so that no other constraints are violated.
    - If $a_i^\intercal p_k \geq 0$ for some $i \notin \mathcal{W}_k$, moving along this direction will not violate those constraints $a_i^\intercal x > b_i$;
    - If $a_i^\intercal p_k < 0$ for some $i \notin \mathcal{W}_k$, we have to choose a $\alpha_k < 1$, so that all other constraints will not be violated, i.e., remain primal feasibility. Add one of these blocking constraints to $\mathcal{W}_k$.
- If $p = 0$: Current point is already optimal in this subproblem. We only need to examine the signs of the multipliers corresponding to the inequality constraints in the working set.[1]
  - If these multipliers are all nonnegative, the dual feasibility is also satisfied; problem solved.

[1] (34): If we define the multipliers corresponding to the inequality constraints that are not in the working set ($i \notin \mathcal{W}_k$) to be zero. The stationary condition is satisfied when we solve the equality constrained problem; the primal feasibility is satisfied too during our careful choice of step length.

– If one or more of these multipliers is negative: we remove one of the corresponding constraints from the working set. [1]

[1] Nocedal and Wright [2006, Theorem 16.5] shows that this strategy produces a direction p at the next iteration that is feasible with respect to the dropped constraint.

19.2 *Dual Method*

Dual active-set method can be seen the primal active-set method on the dual problem. Here, we briefly introduce the implementation in Goldfarb and Idnani [1982] from a dual $\Longleftrightarrow$ primal perspective. We copy the words above and explain its relationship with primal subproblems.

Initially, it lets all the dual constraints to be active, i.e., dual working set $\mathcal{W}_k = U \Longleftrightarrow$ solve an unconstrained primal problem, with primal working set $\mathcal{A} = \emptyset$. $\mathcal{W}_k$ and $\mathcal{A}$ are complementary.

- If $p = 0$: Current point is already optimal in this subproblem. We only need to examine the signs of the multipliers corresponding to the inequality constraints in the working set $\Longleftrightarrow$ the signs of the primal constraints.
  - If these primal constraints are all satisfied, the primal feasibility is also satisfied; problem solved.
  - If one or more of these primal constraints q is negative: we remove one of the corresponding constraints from the working set, i.e., $\mathcal{W}_k \backslash q \Longleftrightarrow$ primal working set $\mathcal{A} \cup q$.
- If $p \neq 0$: We get an update direction p_k for ψ_k. However $\psi_k + p_k$ may not be feasible because it may violate other constraints $i \notin \mathcal{W}_k$, which requires to determine the step-length α_k.
  - If $\psi_k + p_k$ is feasible, then we set $\psi_{k+1} = \psi_k + p_k \Longleftrightarrow$ If the optimal point of $\mathcal{A} \cup q$ treating as equality constraints is the same as the one of $\mathcal{A} \cup q$ treating as inequality constraints (the corresponding dual Lagrangian multipliers are positive, i.e., dual feasible).
  - If $\psi_k + p_k$ is infeasible:
    - **Dual Method**: Remain dual feasibility along this step direction.
      - We need to make sure $\psi_i \geq 0$, $i \notin \mathcal{W}_k \Longleftrightarrow$ Find the point $\hat{\psi}_{k+1}$ on the line segment $[\psi_k, \psi_k + p_k]$ closest to $\psi_k + p_k$ (just choose a step length but not another step direction) which is an optimal solution to the subproblem when treating $\mathcal{A}$ as inequality constraints and q as equality constraints (i.e., requires $\psi_i \geq 0$, $i \in \mathcal{A}$). Because in dual method, we starts with removing a constraint q. Same as foot 1, the dropped q is feasible.
      - Then, add the blocking $q \notin \mathcal{W}_k$ into the working set, i.e.,$\mathcal{W}_k \cup q \Longleftrightarrow$ primal working set $\mathcal{A} \backslash q$.
    - **Primal Dual Method**: Use primal algorithm to solve $\mathcal{A} \cup q$ treating as inequality constraints. This gives a primal feasible point, and thus it is called primal-dual algorithm.

Part VI

Dual Cookbook

20 Dual Norm

Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The associated dual norm, denoted as $\|\cdot\|_*$, is defined as

$$\|\mathbf{z}\|_* = \sup\{\mathbf{z}^\intercal \mathbf{x} | \|\mathbf{x}\| \leq 1\}. \tag{35}$$

The dual norm can be interpreted as the operator norm of $\mathbf{z}^\intercal$.

From the definition we have the inequality

$$\mathbf{z}^\intercal \mathbf{x} = \|\mathbf{x}\| \left(\mathbf{z}^\intercal \frac{\mathbf{x}}{\|\mathbf{x}\|}\right) \leq \|\mathbf{x}\| \|\mathbf{z}\|_* \tag{36}$$

which holds for all $\mathbf{x}$ and $\mathbf{z}$. This inequality is tight, which means for any $\mathbf{x}$ there is a $\mathbf{z}$ that achieves the equality, and vice versa.

The dual of the Euclidean norm is the Euclidean norm, since

$$\sup\{\mathbf{z}^\intercal \mathbf{x} | \|\mathbf{x}\|_2 \leq 1\} = \|\mathbf{z}\|_2.$$

(This follows from the Cauchy–Schwarz inequality; for nonzero $\mathbf{z}$, the value of $\mathbf{x}$ that maximizes $\mathbf{z}^\intercal \mathbf{x}$ over $\|\mathbf{x}\|_2 \leq 1$ is $\frac{\mathbf{z}}{\|\mathbf{z}\|_2}$.)

Application: Equality constrained norm minimization

$$\begin{aligned} \min_{\mathbf{x}} \quad & \|\mathbf{x}\| \\ \text{s.t.} \quad & \mathbf{A}\mathbf{x} = \mathbf{b} \end{aligned}$$

Its dual function is

$$g(\mathbf{v}) = \inf_{\mathbf{x}} (\|\mathbf{x}\| - \mathbf{v}^\intercal \mathbf{A}\mathbf{x} + \mathbf{b}^\intercal \mathbf{v}) = \begin{cases} \mathbf{b}^\intercal \mathbf{v} & \|\mathbf{A}^\intercal \mathbf{v}\|_* \leq 1 \\ -\infty & \text{otherwise.} \end{cases}$$

It follows from

$$\inf_{\mathbf{x}} (\|\mathbf{x}\| - \mathbf{y}^\intercal \mathbf{x}) = \begin{cases} 0 & \|\mathbf{y}\|_* \leq 1 \\ -\infty & \text{otherwise,} \end{cases}$$

with the proof

- if $\|\mathbf{y}\|_* \leq 1$, then $\mathbf{x}^\intercal \mathbf{y} \leq \|\mathbf{x}\| \|\mathbf{y}\|_* \leq \|\mathbf{x}\|$ for all $\mathbf{x}$ (36);
- if $\|\mathbf{y}\|_* > 1$, choose $\mathbf{x} = t\mathbf{u}$, where $\|\mathbf{u}\| \leq 1$, $\mathbf{u}^\intercal \mathbf{y} = \|\mathbf{y}\|_* > 1$ (35):

$$\|\mathbf{x}\| - \mathbf{y}^\intercal \mathbf{x} = t(\|\mathbf{u}\| - \|\mathbf{y}\|_*) \to -\infty \quad \text{as } t \to -\infty$$

21 *Dual cone*[1]

[1] Boyd 2.6.1

Let K be a cone, then the set

$$K^* = \{\mathbf{y} | \mathbf{x}^\intercal \mathbf{y} \geq 0 \text{ for all } \mathbf{x} \in K\}$$

is called the dual cone of K. As the name suggests, K^* is a cone, and is always convex, even when the original cone K is not.

An example of self-dual cone is nonnegative orthant. The cone $\mathbb{R}^n_+$ is its own dual:

$$\mathbf{x}^\intercal \mathbf{y} \geq 0 \text{ for all } \mathbf{x} \succeq \mathbf{0} \iff \mathbf{y} \succeq \mathbf{0}.$$

Dual of a norm cone: Let $\|\cdot\|$ be a norm on $\mathbb{R}^n$. The dual of the associated cone $K = \{(\mathbf{x}, t) \in \mathbb{R}^{n+1} | \|\mathbf{x}\| \leq t\}$ is the cone defined by the dual norm, i.e.

$$K^* = \{(\mathbf{u}, v) \in \mathbb{R}^{n+1} | \|\mathbf{u}\|_* \leq v\}.$$

22 *Duality of SOCP*

Part VII

Appendix

A Nesterov's Appendix

A.1 Adaptive Tuning of Beta with Unknown λ

Problem type 1: Smooth and strongly convex functions

If we know λ, then for fast convergence, we should choose the largest θ possible such that the result (6) holds. This means we should choose a relatively small β so that $\theta = \sqrt{\eta\lambda}$.

However, if we do not know λ, then we have to guess it. To check whether θ is good, we cannot check (6), the convergence of function values, because we don't know x_* and $f(x_*)$. Instead, we can check the convergence of gradients:

$$\| \nabla f(x_t) \|_2^2 \leq \frac{2}{\theta^2}(1-\theta)^t \| \nabla f(x_0) \|_2^2 . \tag{37}$$

Then comes Adaptive Acceleration Method (Theoretically Motivated): and a

Algorithm 15: Adaptive Acceleration Method (Theoretically Motivated)

```
Let θ = 0.5
for t = 1, ..., T do
    Perform Nesterov's Acceleration Methods
    if the convergence of gradients is violated after △ steps then
        θ = θ/2
        Compute new △
    end if
end for
```

simpler and more aggressive method where we tune β using observed convergence rate, measured by

$$\gamma = \log \frac{\| \nabla f(y_t) \|_2^2}{\| \nabla f(y_{t-1}) \|_2^2},$$

leading to Adaptive Acceleration Method (Practically Simplified) :

Algorithm 16: Adaptive Acceleration Method (Practically Simplified)

```
Let γ = 0
for t = 1, ..., T do
    Let β = min(1, exp(γ))
    Perform Nesterov's Acceleration Methods
    Let γ = 0.8γ + 0.2 log(‖ ∇f(y_t) ‖_2^2 / ‖ ∇f(y_{t-1}) ‖_2^2)
end for
```

Problem type 2: Smooth but not strongly convex problems

The key idea of this algorithm is to start with a large θ, and gradually decrease it until $\theta \to \lambda$. This is similar to what is done in Algorithm 15. However, the

decrease of θ is manually controlled by a deterministic sequence and not adaptive. With this method, the following general theorem for convergence can be given.

Algorithm 17: Nesterov's General Acceleration Method

Theorem 27. *Assume $f(x)$ is L-smooth and λ-strongly convex. Then for all $x_* \in \mathbb{R}^d$, we have*

$$f(x_t) \leq f(x_*) + \left(\prod_{s=1}^{t}(1-\theta_s)\right)\left[f(x_0) - f(x_*) + \frac{\gamma_0}{2} \parallel x_0 - x_* \parallel_2^2\right].$$

[1] It can be shown that for nonstrongly convex functions, the rate of convergence is $O(1/t^2)$, faster than that of gradient descent $O(1/t)$ (Theorem 7).

[1] This contains much hidden information. See the note for details.

B Subgradient method's Appendix

Assume $f(x)$ is G-Lipschitz on C, then we have[2]

$$f(x_t^{best}) - f(x_*) \leq \frac{\parallel x_0 - x_* \parallel_2^2 + 2\sum_{t=1}^{T}\eta_t^2 G^2}{2\sum_{t=1}^{T}\eta_t}.$$

[2] This is taken from Subgradient Methods by Stephen Boyd etc. $f(x_t^{best})$ means the optimal value in the first t steps, since subgradient method may not reduce the objective value at each step.

As Boyd shows, generally there are 4 choices of step size:

- *Constant step size.* $\eta_t = h$ is a constant independent of t.
- *Constant step length.* $\eta_t = h/\|g_t\|_2$ so that $\|x_t - x_{t-1}\|_2 = h$.
- *Square summable but not summable.* $\sum_{t=1}^{\infty}\eta_t^2 < \infty$, $\sum_{t=1}^{\infty}\eta_t = \infty$.
- *Nonsummable diminishing.* $\sum_{t=1}^{\infty}\eta_t = \infty$, $\lim_{t\to\infty}\eta_t = 0$.

Corollary 28. *If we take $\eta_t = \eta = \eta_0/\sqrt{T}$, then*[3]

$$f(x_t^{best}) - f(x_*) \leq \frac{\parallel x_0 - x_* \parallel_2^2 + \eta_0^2 G^2}{2\eta_0\sqrt{T}}.$$

[3] nonsummable diminishing case

The rate of convergence is $O(1/\sqrt{T})$, compared to $O(1/T)$ shown in the smooth case (Theorem 7).

C Proximal Gradient Descent Method's Appendix

C.1 Backtracking line search

Similar to section 5.1, there are two line search methods. The first one is a generalization of the Armijo-Goldstein condition at $c = 0.5$. The condition can be written as

$$f(x_t) \leq f(x_{t-1}) + \nabla f(x_{t-1})^\intercal (x_t - x_{t-1}) + \frac{1}{2\eta_t} \parallel x_t - x_{t-1} \parallel_2^2,$$

$$x_t = \text{prox}_\eta(x_{t-1} - \eta_t \nabla f(x_{t-1})),$$

which an optimal η_t (decays from a larger value) should satisfy.

The Barzilai-Borwein method is similar.

References

Stephen Boyd, Neal Parikh, and Eric Chu. *Distributed optimization and statistical learning via the alternating direction method of multipliers*. Now Publishers Inc, 2011.

Donald Goldfarb and Ashok Idnani. Dual and primal-dual methods for solving strictly convex quadratic programs. In *Numerical analysis*, pages 226–239. Springer, 1982.

Jorge Nocedal and Stephen Wright. *Numerical optimization*. Springer Science & Business Media, 2006.

Jonathan Richard Shewchuk et al. An introduction to the conjugate gradient method without the agonizing pain, 1994.