Optimization - Algorithm Summary

Thu Nguyen

July 11, 2020

Summary

This was intend as a personal reference guide on some of the fundamental optimization algorithms and their variants. As such, this write-up contains the high-level information: the motivation of the optimization problems, the approaches, the mathematical formulation, and the algorithms. I decided not to include the theoretical results (more often on the convergence) and instead provided reference where appropriate.

Contents

A note on references, one of the major reference sources is of course *Convex Optimization*, [Boyd and Vandenberghe, 2004]. The book provides both a broad and deep introduction to the problem of convex optimization.

That said, that book does not cover algorithms such as *conjugate gradient* or *dual ascent*. I used papers as the primary sources for those sections and provided references where appropriate.

1 Unconstrained Optimization

Consider the problem

$$\min_{x \in \mathbb{R}^d} \; F(x)$$

where F is some *objective function*, not necessarily *convex*. We note that the *convexity* condition often helps guaranteeing the convergence of the algorithms.

We will assume that F is sufficiently smooth, ie. F is differentiable, at least once. Given f, we denote by ∇F the gradient of F.

Let $x^* = \arg\min_{x \in \mathbb{R}^d} F(x)$, ie. x^* is the (global) optimal minimer. We remark that while it is desirable to precisely get to x^*, it is often infeasible (or inefficient) to do so with numerical optimization. We instead have a number of alternating choices for the convergence condition:

1. The change in F, $F\left(x^{(t+1)}\right) - F\left(x^t\right)$ is small enough, usually less than some predefined ε.
2. The norm of the gradient $\left\|\nabla F\left(x^{(t)}\right)\right\|$ is small enough (we recall that $\nabla F(x^*) = 0$, and hence the norm is 0 at the optimal x^*).
3. Predefining the number of iterations.
4. Etc.

Hence, in the specification of the algorithms, *convergence condition* usually refers to one or more of those conditions being satisfied.

1.1 Gradient Descent

Algorithm 1 Gradient Descent (GD)

1: Initialize: $x^{(0)} \in \mathbb{R}^d$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Update

$$x^{(t+1)} \;=\; x^{(t)} - \alpha_t \nabla F(x^{(t)})$$

4: until convergence

where α_t is the *step size*, which can be fixed (which in turn does not guaranteed convergence even if f is convex, but does not require any computing) or updated with *backtracking line search*.

1.1.1 Stochastic Gradient Descent

Stochastic gradient descent is a much more efficient (cheap to implement) variant of the base *gradient descent*. Suppose there are N data points, let f_i be the loss function associated with the i^{th} data point. This gives us

$$F(x) = \frac{1}{N}\sum_{i=1}^{N} f_i(x)$$

ie. F can be thought of as an aggregate of individual losses. Let us denote by $f_i(x^{(t)}$ be the loss from the i^{th} data point when evaluated at $x^{(t)}$.

We note that at each iteration t, f_i is sampled uniformly randomly from all the $\{f_i\}_{i=1}^{N}$.

Algorithm 2 Stochastic Gradient Descent (SGD)

1: Initialize: $x^{(0)} \in \mathbb{R}^d$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Sample f_i randomly from $\{f_i\}_{i=1}^{N}$, update

$$x^{(t+1)} \;=\; x^{(t)} - \alpha_t \nabla f_i(x^{(t)})$$

4: until convergence

We note that at each iteration, SGD updates based on only 1 data point, compared to all the N data points in the GD. An implication is that it is not guaranteed that the loss function F is always decreasing, although it should nevertheless is on a downward sloping trend.

1.1.2 Minibatch Gradient Descent

We observe that although *stochastic gradient descent* generally works as expected, the process can be sensitive to the random sampling of the data points. *Minibatch gradient descent* handles that problem by sampling some k data points at each iteration, versus just 1 in SGD. In practice, k is often chosen to be 16 or 32.

This gives stability to the process while not introducing significantly more computation cost.

Algorithm 3 Minibatch Gradient Descent

1: Initialize: $x^{(0)} \in \mathbb{R}^d$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Sample k f_{i_j} randomly from $\{f_i\}_{i=1}^{N}$, update

$$x^{(t+1)} \;=\; x^{(t)} - \alpha_t \sum_{j=1}^{k} \nabla f_{i_j}(x^{(t)})$$

4: until convergence

1.2 Variants of Gradient Descent

Let us now take a second look at the *gradient descent* and consider how we can speed up the convergence rate by observing the behavior of the t^{th} update.

Note on reference: the main source of reference on the algorithms in this section is from [Bottou et al., 2018].

1.2.1 Gradient Descent with Momentum

An idea is to take advantage of the previous search direction, which is now added to the current update, ie. a combination of the steepest descent direction and the most recent iterate displacement, effectively giving the *momentum* to the update.

Algorithm 4 Gradient Descent with Momentum

1: Initialize: $x^{(0)} \in \mathbb{R}^d$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Update

$$x^{(t+1)} = x^{(t)} - \alpha_t \nabla F(x^{(t)}) + \beta_t \left(x^{(t)} - x^{(t-1)} \right)$$

4: until convergence

where $\beta_t > 0$ is the parameter controlling the momentum rate. We note that when $\beta_t = 0$, this reduces to the *gradient descent*.

1.2.2 Gradient Descent with Nesterov Acceleration

Similar to the idea of *momentum*, in which we want to accelerate the update, the *Nesterov acceleration* reverses the order of computation: here we apply the momentum first and then apply a steepest descent step.

Algorithm 5 Gradient Descent with Nesterov Acceleration

1: Initialize: $x^{(0)} \in \mathbb{R}^d$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Update

$$\begin{aligned} x^{(t)} &= x^{(t)} + \beta_t \left(x^{(t)} - x^{(t+1)} - x^{(t-1)} \right) \\ x^{(t+1)} &= x^{(t)} - \alpha_t \nabla F(x^{(t)}) \end{aligned}$$

4: until convergence

1.3 Conjugate Gradient

2 Constrained Optimization

Consider the original problem of minimizing some objective function f, but this time subject to additional conditions

$$\begin{aligned} \min_{x \in \mathbb{R}^d} \quad & f(x) \\ \text{subject to} \quad & g_i(x) \leq 0, \quad \text{for } i = 1, \ldots, m \\ & h_i(x) = 0, \quad \text{for } i = 1, \ldots, p \end{aligned} \tag{P}$$

where f is the *objective function*, and g_i, h_i are the *constraint functions*. In principle, all of f, g_i, h_i need not be convex, although such assumptions often help guarantee the convergence of the algorithms, for example the Karush-Kuhn-Tucker (KKT) conditions.

We will approach this problem with *Duality theory*.

Let us consider the *Lagrangian function* $L : \mathbb{R}^d \times \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ such that

$$L(x, \lambda, \nu) \;=\; f(x) + \left(\sum_{i=1}^{m} \lambda_i g_i(x) \right) + \left(\sum_{i=1}^{p} \nu_i h_i(x) \right)$$

where $\lambda \in \mathbb{R}^m$ and $\nu \in \mathbb{R}^p$ are the Lagrangian multipliers. We define the *Lagrange dual function* $\hat{f} : \mathbb{R}^m \times \mathbb{R}^p \to \mathbb{R}$ as

$$\hat{f}(\lambda, \nu) \;=\; \inf_{x \in \mathbb{R}^d} L(x, \lambda, \nu).$$

We remark that $\hat{f}$ is *concave*, regardless of whether or not the original function f is convex.

Solving the dual function gives us the lower bounds on the the optimal x^*, ie. let $\tilde{x}$ be a feasible point (a point which satisfies all the g_i and h_i conditions), then

$$\forall \tilde{x} : \quad \hat{f}(\lambda, \nu) \;=\; \inf_{x \in \mathbb{R}^d} L(x, \lambda, \nu) \;\leq\; L(\tilde{x}, \lambda, \nu) \;\leq\; f(\tilde{x})$$

Under appropriate conditions (for example the KKT conditions), we observe the desired equality

$$\hat{f}(\lambda, \nu) \;=\; f(\tilde{x})$$

which implies that by solving solving the unconstrained dual problem

$$\max_{\lambda, \nu} \quad \hat{f}(\lambda, \nu),$$

we will have simultaneously solved the prior problem in (P).

Note on reference: the main source of reference on the algorithms in this section is from [Boyd et al., 2011].

2.1 Dual Ascent

Let us consider a special case of (P), in which the only constraints are the *equality constraints*:

$$\begin{aligned} \min_{x \in \mathbb{R}^d} \quad & f(x) \\ \text{subject to} \quad & Ax = b, \quad \text{for some matrix } A \in \mathbb{R}^{m \times d} \end{aligned}$$

Let $y \in \mathbb{R}^m$, it follows that the Lagrangian and its dual function are

$$\begin{aligned} L(x, y) &= f(x) + y^T (Ax - b) \\ \hat{f} &= \inf_{x \in \mathbb{R}^d} L(x, y) = \inf_{x \in \mathbb{R}^d} \Big(f(x) + y^T (Ax - b) \Big) \end{aligned}$$

The *dual ascent* algorithm is thus:

Algorithm 6 Dual Ascent

1: Initialize: $x^{(0)} \in \mathbb{R}^d, y \in \mathbb{R}^m$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Update

$$\begin{aligned} x^{(t+1)} &= \arg\min_{x \in \mathbb{R}^d} L\left(x, y^{(t)}\right) \\ y^{(t+1)} &= y^{(t)} + \alpha_t \left(A x^{(t+1)} - b \right) \end{aligned}$$

4: until convergence

2.2 Method of Multipliers

Method of multipliers is a variant of *dual ascent*, in which the *Lagrangian* has an additional term, namely the *penalty*, which should remind us of the L_2 regularization commonly seen in the literature.

Let us adapt the same problem as in section 2.1 Dual Ascent. Due to the addition of a penalty, what follows is the *augmented Lagrangian*:

$$L_\rho(x, y) = f(x) + y^T (Ax - b) + (\rho/2) \|Ax - b\|^2$$

where ρ is the *parameter parameter*, which can be modified as desired.

The *method of multipliers* algorithm is thus:

Algorithm 7 Method of Multipliers (MM)

1: Initialize: $x^{(0)} \in \mathbb{R}^d, y^{(0)} \in \mathbb{R}^m$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Update

$$\begin{aligned} x^{(t+1)} &= \arg\min_{x \in \mathbb{R}^d} \; L_\rho \left(x, y^{(t)} \right) \\ y^{(t+1)} &= y^{(t)} + \rho \left(A x^{(t+1)} - b \right) \end{aligned}$$

4: until convergence

2.3 Alternating Direction Method of Multipliers

We first have some remarks on *dual ascent* versus *method of multipliers*:

1. *Dual ascent:* time complexity can be reduced greatly if the objective function f is *separable* (with respect to a partition of the variables) whereas that is not possible with *method of multipliers*
2. *Method of multipliers:* convergence of the optimal x^* happens under far more general conditions that *dual ascent*.

We observe that *dual ascent* has advantages in run time complexity and *method of multipliers* in superior convergence. It is naturally desired that we could make use of both advantages. Such are the motivation for the *alternating direction method of multipliers*.

Let us consider a special case of section 2.1 Dual Ascent, one in which both the objective function and the constraints are decomposable. Let $A \in \mathbb{R}^{p \times n}, B \in \mathbb{R}^{p \times m}$, consider

$$\begin{aligned} \min_{x \in \mathbb{R}^n, z \in \mathbb{R}^m} \quad & f(x) + g(z) \\ \text{subject to} \quad & Ax + Bz = c \end{aligned}$$

Let $y \in \mathbb{R}^p$, the augmented Lagrangian is

$$L_\rho(x, z, y) \;=\; f(x) + g(z) + y^T \left(Ax + Bz - c \right) + (\rho/2) \| Ax + Bz - c \|^2$$

The *alternating direction method of multipliers* algorithm is thus:

Algorithm 8 Alternating Direction Method of Multipliers (ADMM)

1: Initialize: $x^{(0)} \in \mathbb{R}^n, z^{(0)} \in \mathbb{R}^m, y^{(0)} \in \mathbb{R}^p$
2: **for** $t = 0, 1, \ldots$ **do**:
3: Update

$$\begin{aligned} x^{(t+1)} &= \underset{x \in \mathbb{R}^n}{\arg\min}\ L_\rho\left(x, z^{(t)}, y^{(t)}\right) \\ z^{(t+1)} &= \underset{z \in \mathbb{R}^m}{\arg\min}\ L_\rho\left(x^{(t+1)}, z, y^{(t)}\right) \\ y^{(t+1)} &= y^{(t)} + \rho\left(Ax^{(t+1)} + By^{(t+1)} - c\right) \end{aligned}$$

4: until convergence

Remarks on ADMM versus *method of multipliers*: under MM, the update rules would have been

$$\begin{aligned} \left(x^{(t+1)}, z^{(t+1)}\right) &= \underset{x,z}{\arg\min}\ L_\rho\left(x^{(t+1)}, z, y^{(t)}\right) \\ y^{(t+1)} &= y^{(t)} + \rho\left(Ax^{(t+1)} + By^{(t+1)} - c\right) \end{aligned}$$

ie. the variables x and z are updated simultanously, compared to the sequential updates of x and then z in ADMM.

3 References

[Bottou et al., 2018] Bottou, L., Curtis, F. E., and Nocedal, J. (2018). Optimization methods for large-scale machine learning. *Siam Review*, 60(2):223–311.

[Boyd et al., 2011] Boyd, S., Parikh, N., Chu, E., Peleato, B., and Eckstein, J. (2011). *Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers.*

[Boyd and Vandenberghe, 2004] Boyd, S. and Vandenberghe, L. (2004). *Convex Optimization.* Cambridge University Press.