

# Numerical Optimization

<https://pcombet.math.ncsu.edu/confab.html>

IN [A Few Useful Things to Know about Machine Learning](#), Pedro Domingos put up a relation:

*LEARNING* = *REPRESENTATION* + *EVALUATION* + *OPTIMIZATION*.

- Representation as the core of the note is the general (mathematical) **model** that computer can handle.
- Evaluation is **criteria**. An evaluation function (also called objective function, cost function or scoring function) is needed to distinguish good classifiers from bad ones.
- Optimization is aimed to find the parameters that optimizes the evaluation function, i.e.

$$\arg \min_{\theta} f(\theta) = \{\theta^* | f(\theta^*) = \min_{\theta} f(\theta)\} \text{ or } \arg \max_{\theta} f(\theta) = \{\theta^* | f(\theta^*) = \max_{\theta} f(\theta)\}.$$

---

The objective function to be minimized is also called cost function.

Evaluation is always attached with optimization; the evaluation which cannot be optimized is not a good evaluation in machine learning.

- [https://www.wikiwand.com/en/Mathematical\\_optimization](https://www.wikiwand.com/en/Mathematical_optimization)
- [https://web.stanford.edu/~boyd/cvxbook/bv\\_cvxbook.pdf](https://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf)
- <http://www.cs.cmu.edu/~pradeepr/convexopt/>
- [An interactive tutorial to numerical optimization](#)
- <http://awibisono.github.io/2016/06/06/world-of-optimization.html>
- <http://awibisono.github.io/2016/06/13/gradient-flow-gradient-descent.html>
- <http://awibisono.github.io/2016/06/20/accelerated-gradient-descent.html>

## Gradient Descent and More

Each iteration of a line search method computes a search direction  $p^k$  and then decides how far to move along that direction. The iteration is given by

$$x^{k+1} = x^k + \alpha_k p^k \quad (\text{Line search})$$

where the positive scalar  $\alpha_k$  is called the step length. The success of a line search method depends on effective choices of both the direction  $p^k$  and the step length  $\alpha_k$ .

*Note:* we use the notation  $x^k$  and  $\alpha_k$  to represent the  $k$ th iteration of the vector variables  $x$  and  $k$ th step length, respectively. Most line search algorithms require  $p^k$  to be a descent direction — one for which  $\langle p^k, \nabla f_k \rangle < 0$  — because this property guarantees that the function  $f$  can be reduced along this direction, where  $\nabla f_k$  is the gradient of objective function  $f$  at the  $k$ th iteration point  $x_k$  i.e.  $\nabla f_k = \nabla f(x^k)$ .

---

Gradient descent and its variants are to find the local solution of the unconstrained optimization problem:

$$\min f(x)$$

where  $x \in \mathbb{R}^n$ .

It is not difficult to observe that

$$f(x) \approx f(x^k) + (x - x^k)^T \nabla f(x^k) + \frac{s_k}{2} \|x - x^k\|_2^2$$

by Taylor expansion of  $f$  near the point  $x^k$  for some constant  $s_k$ .

Let  $x^{k+1} = \arg \min_x f(x^k) + (x - x^k)^T \nabla f(x^k) + \frac{s_k}{2} \|x - x^k\|_2^2$ , we will obtain  $x^{k+1} = x^k - \frac{1}{s_k} \nabla_x f(x^k)$ . However, the constant  $s_k$  is difficult to estimate.

And the general gradient descent methods are given by

$$x^{k+1} = x^k - \alpha_k \nabla_x f(x^k)$$

where  $x^k$  is the  $k$ th iterative result,  $\alpha_k \in \{\alpha | f(x^{k+1}) < f(x^k)\}$  and particularly  $\alpha_k = \arg \min_{\alpha} f(x^k - \alpha \nabla_x f(x^k))$  so that  $f(x^{k+1}) = \min_{\alpha} f(x^k - \alpha \nabla_x f(x^k))$ .

- <http://59.80.44.100/www.seas.ucla.edu/~vandenbe/236C/lectures/gradient.pdf>
- [http://wiki.fast.ai/index.php/Gradient\\_Descent](http://wiki.fast.ai/index.php/Gradient_Descent)



There are many ways to choose some proper step or learning rate sequence  $\{\alpha_k\}$ .

The proof of convergence or complexity is often based on the convex case where the objective function is convex, i.e.,

$$f(tx + (1-t)y) \leq tf(x) + (1-t)f(y), \quad t \in [0, 1].$$

And this optimization is called convex optimization.

---

Some variants of gradient descent methods are not line search method.

For example, the **heavy ball methods or momentum methods**:

$$x^{k+1} = x^k - \alpha_k \nabla_x f(x^k) + \rho_k (x^k - x^{k-1})$$

where the momentum coefficient  $\rho_k \in [0, 1]$  generally and the step length  $\alpha_k$  cannot be determined by line search.

**Nesterov accelerated gradient method** at the  $k$ th step is given by:

$$\begin{array}{ll} x^k = y^k - \alpha_{k+1} \nabla_x f(y^k) & \text{Descent} \\ y^{k+1} = x^k + \rho_k (x^k - x^{k-1}) & \text{Momentum} \end{array}$$

where the momentum coefficient  $\rho_k \in [0, 1]$  generally.

They are called as **inertial gradient methods** or **accelerated gradient methods**. [And there are some different forms.](#)

#### Inventor of Nesterov accelerated Gradient

<img src=https://upload.wikimedia.org/wikipedia/commons/thumb/9/9e/Nesterov\_yurii.jpg/440px-Nesterov\_yurii.jpg  
width = 60% />

- <https://blogs.princeton.edu/imabandit/2013/04/01/acceleratedgradientdescent/>
- <https://blogs.princeton.edu/imabandit/2014/03/06/nesterovs-accelerated-gradient-descent-for-smooth-and-strongly-convex-optimization/>
- <https://blogs.princeton.edu/imabandit/2015/06/30/revisiting-nesterovs-acceleration/>
- <https://blogs.princeton.edu/imabandit/2018/11/21/a-short-proof-for-nesterovs-momentum/>
- <https://blogs.princeton.edu/imabandit/2019/01/09/nemirovskis-acceleration/>

- 
- <https://zhuanlan.zhihu.com/p/41263068>
  - <https://zhuanlan.zhihu.com/p/35692553>
  - <https://zhuanlan.zhihu.com/p/35323828>
  - On Gradient-Based Optimization: Accelerated, Asynchronous, Distributed and Stochastic
  - <https://jlmelville.github.io/mize/nesterov.html>
  - <https://distill.pub/2017/momentum/>
  - [http://www.optimization-online.org/DB\\_FILE/2018/11/6938.pdf](http://www.optimization-online.org/DB_FILE/2018/11/6938.pdf)
  - <https://www.mat.univie.ac.at/~neum/glopt/mss/MasAi02.pdf>
  - <https://www.cs.cmu.edu/~ggordon/10725-F12/slides/09-acceleration.pdf>
  - <https://saugatbhattarai.com.np/what-is-gradient-descent-in-machine-learning/>
  - <https://www.fromthegenesis.com/gradient-descent-part-2/>
  - <http://www.deepideas.net/deep-learning-from-scratch-iv-gradient-descent-and-backpropagation/>
  - <https://ee227c.github.io/notes/ee227c-lecture08.pdf>

## Variable Metric Methods

### Newton's Method

NEWTON'S METHOD and QUASI-NEWTON METHODS are classified to variable metric methods.

It is also to find the solution of unconstrained optimization problems, i.e.

$$\min f(x)$$

where  $x \in \mathbb{R}^n$ . Specially, the objective function  $f(x)$  is convex so that the local minima is global minima.

---

If  $x^*$  is the extrema of the cost function  $f(x)$ , it is necessary that  $\nabla f(x^*) = 0$ .

So if we can find all the solution of the equation system  $\nabla f(x) = 0$ , it helps us to find the solution to the optimization problem  $\arg \min_{x \in \mathbb{R}^n} f(x)$ .

**Newton's method** is one of the fixed-point methods to solve nonlinear equation system.

It is given by

$$\begin{aligned} x^{k+1} &= \arg \min_x f(x^k) + (x - x^k)^T \nabla_x f(x^k) + \frac{1}{2\alpha_{k+1}} (x - x^k)^T H(x^k) (x - x^k) \\ &= x^k - \alpha_{k+1} H^{-1}(x^k) \nabla_x f(x^k) \end{aligned}$$

where  $H^{-1}(x^k)$  is inverse of the Hessian matrix of the function  $f(x)$  at the point  $x^k$ .

It is called **Newton–Raphson algorithm** in statistics.

Especially when the log-likelihood function  $\ell(\theta)$  is well-behaved,

a natural candidate for finding the MLE is the **Newton–Raphson algorithm** with quadratic convergence rate.

## The Fisher Scoring Algorithm

In maximum likelihood estimation, the objective function is the log-likelihood function, i.e.

$$\ell(\theta) = \sum_{i=1}^n \log P(x_i|\theta)$$

where  $P(x_i|\theta)$  is the probability of realization  $X_i = x_i$  with the unknown parameter  $\theta$ .

However, when the sample random variable  $\{X_i\}_{i=1}^n$  are not observed or realized, it is best to

replace negative Hessian matrix (i.e.  $-\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^T}$ )

of the likelihood function with the **observed information matrix**:

$$J(\theta) = \mathbb{E}\left(-\frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^T}\right) = - \int \frac{\partial^2 \ell(\theta)}{\partial \theta \partial \theta^T} f(x_1, \dots, x_n|\theta) dx_1 \cdots dx_n$$

where  $f(x_1, \dots, x_n|\theta)$  is the joint probability density function of  $X_1, \dots, X_n$  with unknown parameter  $\theta$ .

And the **Fisher scoring algorithm** is given by

$$\theta^{k+1} = \theta^k + \alpha_k J^{-1}(\theta^k) \nabla_{\theta} \ell(\theta^k)$$

where  $J^{-1}(\theta^k)$  is the inverse of observed information matrix at the point  $\theta^k$ .

See <http://www.stats.ox.ac.uk/~steffen/teaching/bs2HT9/scoring.pdf> or <https://wiseodd.github.io/techblog/2018/03/11/fisher-information/>.

**Fisher scoring algorithm** is regarded as an example of **Natural Gradient Descent** in information geometry as shown in <https://wiseodd.github.io/techblog/2018/03/14/natural-gradient/> and <https://www.zhihu.com/question/266846405>.

## Quasi-Newton Methods

Quasi-Newton methods, like steepest descent, require only the gradient of the objective function to be supplied at each iterate.

By measuring the changes in gradients, they construct a model of the objective function that is good enough to produce super-linear convergence.

The improvement over steepest descent is dramatic, especially on difficult problems. Moreover, since second derivatives are not required, quasi-Newton methods are sometimes more efficient than Newton's method.<sup>[^11]</sup>

In optimization, quasi-Newton methods (a special case of **variable-metric methods**) are algorithms for finding local maxima and minima of functions. Quasi-Newton methods are based on Newton's method to find the stationary point of a function, where the gradient is 0.

In quasi-Newton methods the Hessian matrix does not need to be computed. The Hessian is updated by analyzing successive gradient vectors instead. Quasi-Newton methods are a generalization of the secant method to find the root of the first derivative for multidimensional problems.

In multiple dimensions the secant equation is under-determined, and quasi-Newton methods differ in how they constrain the solution, typically by adding a simple low-rank update to the current estimate of the Hessian.

One of the chief advantages of quasi-Newton methods over Newton's method is that the Hessian matrix (or, in the case of quasi-Newton methods, its approximation)  $\mathbf{B}$  does not need to be inverted. The Hessian approximation  $\mathbf{B}$  is chosen to satisfy

$$\nabla f(x^{k+1}) = \nabla f(x^k) + B(x^{k+1} - x^k),$$

which is called the **secant equation** (the Taylor series of the gradient itself).

In more than one dimension  $B$  is underdetermined. In one dimension, solving for  $B$  and applying the Newton's step with the updated value is equivalent to the **secant method**.

The various quasi-Newton methods differ in their choice of the solution to the **secant equation** (in one dimension, all the variants are equivalent).

The unknown  $x_k$  is updated applying the Newton's step calculated using the current approximate Hessian matrix  $B_k$ :

1.  $\Delta x_k = -\alpha_k B_k^{-1} \nabla f(x_k)$ , with  $\alpha$  chosen to satisfy the Wolfe conditions;
2.  $x_{k+1} = x_k + \Delta x_k$ ;
3. The gradient computed at the new point  $\nabla f(x_{k+1})$ , and  $y_k = \nabla f(x_{k+1}) - \nabla f(x_k)$  is used to update the approximate Hessian  $B_{k+1}$ , or directly its inverse  $H_{k+1} = B_{k+1}^{-1}$  using the Sherman–Morrison formula.

A key property of the BFGS and DFP updates is that if  $B_k$  is positive-definite, and  $\alpha_k$  is chosen to satisfy the Wolfe conditions, then  $B_{k+1}$  is also positive-definite.

For example,

|               |             |                            |
|---------------|-------------|----------------------------|
| <b>Method</b> | $B_{k+1} =$ | $H_{k+1} = B_{k+1}^{-1} =$ |
|---------------|-------------|----------------------------|

---

| Method | $B_{k+1} =$   | $H_{k+1} = B_{k+1}^{-1} =$   |
|--------|---|--|
| DFP    | $(I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k}) B_k (I - \frac{\Delta x_k y_k^T}{y_k^T \Delta x_k}) + \frac{y_k y_k^T}{y_k^T \Delta x_k} \Delta x_k$ | $H_k + \frac{\Delta x_k \Delta x_k^T}{\Delta x_k^T y_k} - \frac{H_k y_k y_k^T H_k}{y_k^T H_k y_k}$   |
| BFGS   | $B_k + \frac{y_k y_k^T}{y_k^T \Delta x_k} - \frac{B_k \Delta x_k (B_k \Delta x_k)^T}{\Delta x_k^T B_k \Delta x_k}$                                    | $(I - \frac{\Delta x_k^T y_k}{y_k^T \Delta x_k}) H_k (I - \frac{y_k \Delta x_k^T}{y_k^T \Delta x_k}) + \frac{\Delta x_k \Delta x_k^T}{y_k^T \Delta x_k}$ |
| SR1    | $B_k + \frac{(y_k - B_k \Delta x_k)(y_k - B_k \Delta x_k)^T}{(y_k - B_k \Delta x_k)^T \Delta x_k}$  | $H_k + \frac{(\Delta x_k - H_k y_k)(\Delta x_k - H_k y_k)^T}{(\Delta x_k - H_k y_k)^T y_k}$  |



- [Quasi-Newton methods in Wikipedia page](http://aria42.com/images/bfgs.png)
- <http://59.80.44.98/www.seas.ucla.edu/~vandenbe/236C/lectures/qnewton.pdf>
- [http://fa.bianp.net/teaching/2018/eecs227at/quasi\\_newton.html](http://fa.bianp.net/teaching/2018/eecs227at/quasi_newton.html)

### The Barzilai-Borwein method

Consider the gradient iteration form

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k)$$

which can be written as

$$x^{k+1} = x^k - D_k \nabla f(x^k)$$

where  $D_k = \alpha_k I$ .

In order to make the matrix  $D_k$  have quasi-Newton property, we compute  $\alpha_k$  such that

$$\min \|s_{k-1} - D_k y_{k-1}\|$$

which yields

$$\alpha_k = \frac{\langle s_{k-1}, y_{k-1} \rangle}{\langle y_{k-1}, y_{k-1} \rangle} \quad (1)$$

where  $s_{k-1} = x_k - x_{k-1}$ ,  $y_{k-1} = \nabla f(x^k) - \nabla f(x^{k-1})$ .

By symmetry, we may minimize  $\|D_k^{-1} s_{k-1} - y_{k-1}\|$  with respect to  $\alpha_k$  and get

$$\alpha_k = \frac{\langle s_{k-1}, s_{k-1} \rangle}{\langle s_{k-1}, y_{k-1} \rangle}. \quad (2)$$

In short, the iteration formula of Barzilai-Borwein method is given by

$$x^{k+1} = x^k - \alpha_k \nabla f(x^k)$$

where  $\alpha_k$  is determined by (1) or (2).

It is easy to see that in this method no matrix computations and no line searches (except  $k = 0$ ) are required.

- [https://mp.weixin.qq.com/s/G9HH29b2-VBnk\\_Sqze8pDg](https://mp.weixin.qq.com/s/G9HH29b2-VBnk_Sqze8pDg)
- <http://www.math.ucla.edu/~wotaoyin/math273a/slides/>
- [http://bicmr.pku.edu.cn/~wenzw/courses/WenyuSun\\_YaxiangYuan\\_BB.pdf](http://bicmr.pku.edu.cn/~wenzw/courses/WenyuSun_YaxiangYuan_BB.pdf)
- <https://www.math.lsu.edu/~hozhang/papers/cbb.pdf>

- 
- [Wikipedia page on Newton Method](#)
  - [Newton-Raphson Visualization \(1D\)](#)
  - [Newton-Raphson Visualization \(2D\)](#)
  - [Using Gradient Descent for Optimization and Learning](#)

## Natural Gradient Descent

Natural gradient descent is to solve the optimization problem  $\min_{\theta} L(\theta)$  by

$$\theta^{(t+1)} = \theta^{(t)} - \alpha_{(t)} F^{-1}(\theta^{(t)}) \nabla_{\theta} L(\theta^{(t)})$$

where  $F^{-1}(\theta^{(t)})$  is the inverse of **Riemann metric** at the point  $\theta^{(t)}$ .

And **Fisher scoring** algorithm is a typical application of **Natural Gradient Descent** to statistics.

**Natural gradient descent** for manifolds corresponding to exponential families can be implemented as a first-order method through **mirror descent** (<https://www.stat.wisc.edu/~raskutti/publication/MirrorDescent.pdf>).

### Originator of Information Geometry: Shunichi Amari



- [Natural gradient descent and mirror descent](#)
- [Online Natural Gradient as a Kalman Filter](#)
- <https://www.zhihu.com/question/266846405>
- [2016 PKU Mini-Course: Information Geometry](#)
- [Information Geometry and Natural Gradients](#)
- [Tutorial on Information Geometry and Algebraic Statistics](#)
- [True Asymptotic Natural Gradient Optimization](#)
- [Shun-ichi Amari's CV in RIKEN](#)
- [Energetic Natural Gradient Descent](#)
- [Natural gradients and K-FAC](#)

- <http://www.divergence-methods.org/>
- [http://www.deeplearningpatterns.com/doku.php?id=natural\\_gradient\\_descent](http://www.deeplearningpatterns.com/doku.php?id=natural_gradient_descent)
- [NATURAL GRADIENTS AND STOCHASTIC VARIATIONAL INFERENCE](#)
- [谈谈优化算法 - 郑思座的文章 - 知乎](#)
- [Accelerating Natural Gradient with Higher-Order Invariance](#)

## Trust Region Methods

Trust-region methods define a region around the current iterate within which they trust the model to be an adequate representation of the objective function, and then choose the step to be the approximate minimizer of the model in this region.

If a step is not acceptable, they reduce the size of the region and find a new minimizer.

In general, the direction of the step changes whenever the size of the trust region is altered.

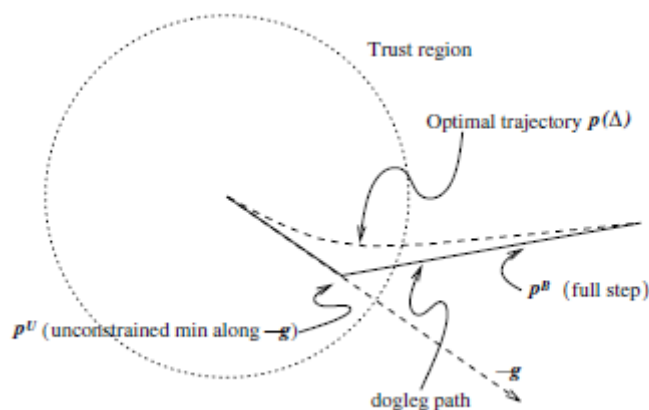
$$\min_{p \in \mathbb{R}^n} m_k(p) = f_k + g_k^T p + \frac{1}{2} p^T B_k p, \quad s.t. \quad \|p\| \leq \Delta_k,$$

where  $f_k = f(x^k)$ ,  $g_k = \nabla f(x^k)$ ;  $B_k$  is some symmetric matrix as an approximation to the Hessian  $\nabla^2 f(x^k + tp)$ ;  $\Delta_k > 0$  is the trust-region radius.

In most of our discussions, we define  $\|\cdot\|$  to be the Euclidean norm.

Thus, the trust-region approach requires us to solve a sequence of subproblems in which the objective function and constraint are both quadratic.

Generally, we set  $B_k = \nabla^2 f(x^k)$ .



- [https://optimization.mccormick.northwestern.edu/index.php/Trust-region\\_methods](https://optimization.mccormick.northwestern.edu/index.php/Trust-region_methods)
- [Concise Complexity Analyses for Trust-Region Methods](#)
- [https://www.nmr-relax.com/manual/Trust\\_region\\_methods.html](https://www.nmr-relax.com/manual/Trust_region_methods.html)

## Expectation Maximization Algorithm



**Expectation-Maximization algorithm**, popularly known as the **EM algorithm** has become a standard piece in the statistician's repertoire.

It is used in incomplete-data problems or latent-variable problems such as Gaussian mixture model in maximum likelihood estimation.

The basic principle behind the **EM** is that instead of performing a complicated optimization, one augments the observed data with latent data to perform a series of simple optimizations.

It is really popular for Bayesian statistician.

Let  $\ell(\theta|Y_{obs}) \triangleq \log L(\theta|Y_{obs})$  denote the log-likelihood function of observed datum  $Y_{obs}$ .

We augment the observed data  $Y_{obs}$  with latent variables  $Z$  so that both the complete-data log-likelihood  $\ell(\theta|Y_{obs}, Z)$  and the conditional predictive distribution  $f(z|Y_{obs}, \theta)$  are available.

Each iteration of the **EM** algorithm consists of an expectation step (E-step) and a maximization step (M-step)

Specifically, let  $\theta^{(t)}$  be the current best guess at the MLE  $\hat{\theta}$ . The E-step is to compute the **Q** function defined by

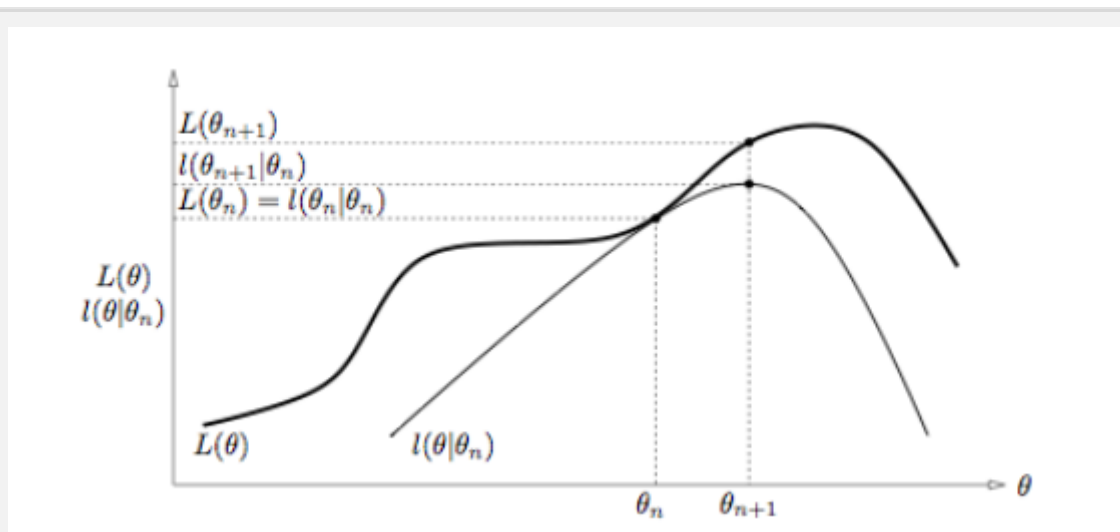
$$\begin{aligned} Q(\theta|\theta^{(t)}) &= \mathbb{E}(\ell(\theta|Y_{obs}, Z)|Y_{obs}, \theta^{(t)}) \\ &= \int_Z \ell(\theta|Y_{obs}, Z) \times f(z|Y_{obs}, \theta^{(t)}) dz, \end{aligned}$$

and the M-step is to maximize **Q** with respect to  $\theta$  to obtain

$$\theta^{(t+1)} = \arg \max_{\theta} Q(\theta|\theta^{(t)}).$$

- [https://www.wikiwand.com/en/Expectation-maximization\\_algorithm](https://www.wikiwand.com/en/Expectation-maximization_algorithm)
- <http://cs229.stanford.edu/notes/cs229-notes8.pdf>
- EM算法存在的意义是什么？ - 史博的回答 - 知乎

Diagram of EM algorithm



## Generalized EM Algorithm

Each iteration of the **generalized EM** algorithm consists of an expectation step (E-step) and a maximization step (M-step)

Specifically, let  $\theta^{(t)}$  be the current best guess at the MLE  $\hat{\theta}$ . The E-step

is to compute the  $Q$  function defined by

$$\begin{aligned} Q(\theta|\theta^{(t)}) &= \mathbb{E}[\ell(\theta|Y_{obs}, Z)|Y_{obs}, \theta^{(t)}] \\ &= \int_Z \ell(\theta|Y_{obs}, Z) \times f(z|Y_{obs}, \theta^{(t)}) dz, \end{aligned}$$

and the another step is to find  $\theta$  that satisfies  $Q(\theta^{t+1}|\theta^t) > Q(\theta^t|\theta^t)$ , i.e.

$$\theta^{(t+1)} \in \{\hat{\theta} | Q(\hat{\theta}|\theta^t) \geq Q(\theta|\theta^t)\}.$$

It is not to maximize the conditional expectation.

See more on the book [The EM Algorithm and Extensions, 2nd Edition](#) by Geoffrey McLachlan , Thriyambakam Krishna.

! [projection EM] ([https://pic4.zhimg.com/80/v2-468b515b4d26ebc4765f82bf3ed1c3bf\\_hd.jpg](https://pic4.zhimg.com/80/v2-468b515b4d26ebc4765f82bf3ed1c3bf_hd.jpg) =360x328)

- <https://www.stat.berkeley.edu/~aldous/Colloq/lange-talk.pdf>

### Quadratic Lower Bound

- <http://www.cs.cmu.edu/afs/cs/user/dwoodruf/www/w10b.pdf>

## Projected Gradient Method and More

We will focus on **projected gradient descent** and its some non-Euclidean generalization in order to solve some simply constrained optimization problems.

If not specified, all these methods are aimed to solve convex optimization problem with explicit constraints, i.e.

$$\arg \min_{x \in \mathbb{S}} f(x)$$

where  $f$  is convex and differentiable,  $\mathbb{S} \subset \mathbb{R}^n$  is convex.

The optimal condition for this constrained optimization problem is that the feasible direction is not the descent or profitable direction: if  $x^* \in \mathbb{S}$  is the solution to the problem, we can assert that **variational inequality** holds:

$$\forall x \in \mathbb{S}, \langle \nabla f(x^*), x - x^* \rangle \geq 0.$$

And it is the optimal condition of constrained optimization problem.

## Projected Gradient Descent

**Projected gradient descent** has two steps:

$$\begin{aligned} z^{k+1} &= x^k - \alpha_k \nabla_x f(x^k) && \text{Descent} \\ x^{k+1} &= Proj_{\mathbb{S}}(z^{k+1}) = \arg \min_{x \in \mathbb{S}} \|x - z^{k+1}\|^2 && \text{Projection} \end{aligned}$$

or in the compact form

$$\begin{aligned}x^{k+1} &= \arg \min_{x \in \mathbb{S}} \|x - (x^k - \alpha_k \nabla_x f(x^k))\|^2 \\&= \arg \min_x \{\|x - (x^k - \alpha_k \nabla_x f(x^k))\|^2 + \delta_{\mathbb{S}}(x)\}\end{aligned}$$

where  $\delta_{\mathbb{S}}$  is the indicator function of the set  $\mathbb{S}$

$$h(x) = \delta_{\mathbb{S}}(x) = \begin{cases} 0, & \text{if } x \in \mathbb{S}; \\ \infty, & \text{otherwise.} \end{cases}$$

Each projection is an optimization so that the iterative points satisfy the optimal conditions, which also restricts the projection method into the case where the projection is available or simple to compute.

And it is natural to search better Descent step or Projection step.

The following links are recommended if you want more theoretical proof:

- <http://maths.nju.edu.cn/~hebma/slides/03C.pdf>
- <http://maths.nju.edu.cn/~hebma/slides/00.pdf>

For the non-differentiable but convex function such as the absolute value function  $f(x) = |x|$ , we therefore consider sub-gradients in place of the gradient, where sub-gradient  $\phi$  at the point  $\hat{x}$  is defined as the elements in the domain of convex function  $f$  (denoted as  $D_f$ ), satisfying

$$\langle \phi, x - \hat{x} \rangle \leq f(x) - f(\hat{x}), \forall x \in D_f.$$

## Mirror descent

**Mirror descent** can be regarded as the non-Euclidean generalization via replacing the  $\ell_2$  norm or Euclidean distance in projected gradient descent by [Bregman divergence](#).

Bregman divergence is induced by convex smooth function  $h$ :

$$B_h(x, y) = h(x) - h(y) - \langle \nabla h(y), x - y \rangle$$

where  $\langle \cdot, \cdot \rangle$  is inner product and it also denoted as  $D_h$ .

The function  $h$  is usually required to be strongly convex. And if the convex function  $h$  is not differentiable, one element of the sub-gradient  $\partial h(y)$  may replace the gradient  $\nabla h(y)$ .

It is convex in  $x$  and  $\frac{\partial B_h(x, y)}{\partial x} = \nabla h(x) - \nabla h(y)$ .

Especially, when  $h$  is quadratic function, the Bregman divergence induced by  $h$  is

$$B_h(x, y) = x^2 - y^2 - \langle 2y, x - y \rangle = x^2 + y^2 - 2xy = (x - y)^2$$

i.e. the square of Euclidean distance.

A wonderful introduction to **Bregman divergence** is **Meet the Bregman Divergences** by [Mark Reid](http://mark.reid.name/blog/meet-the-bregman-divergences.html) at <http://mark.reid.name/blog/meet-the-bregman-divergences.html>.

The Bregman projection onto a convex set  $C \subset \mathbb{R}^n$  given by

$$y' = \arg \min_{x \in C} B(x, y)$$

is unique.

A **generalized Pythagorean theorem** holds: for convex  $C \subset \mathbb{R}^n$  and for all  $x \in C$  and  $y \in \mathbb{R}^n$  we have

$$B(x, y) \geq B(x, y') + B(y', y)$$

where  $y'$  is the Bregman projection of  $y$ , and equality holds when the convex set  $C$  defining the projection  $y'$  is affine.

<img src = "[https://upload.wikimedia.org/wikipedia/commons/2/2e/Bregman\\_divergence\\_Pythagorean.png](https://upload.wikimedia.org/wikipedia/commons/2/2e/Bregman_divergence_Pythagorean.png)" width=80%>

It is given in the projection form:

$$\begin{aligned} z^{k+1} &= x^k - \alpha_k \nabla_x f(x^k) && \text{Gradient descent ;} \\ x^{k+1} &= \arg \min_{x \in \mathbb{S}} B(x, z^{k+1}) && \text{Bregman projection.} \end{aligned}$$

In another compact form, mirror gradient can be described in the proximal form:

$$x^{k+1} = \arg \min_{x \in \mathbb{S}} \{f(x^k) + \langle g^k, x - x^k \rangle + \frac{1}{\alpha_k} B(x, x^k)\} \quad (1)$$

with  $g^k = \nabla f(x^k)$ .

Note that the next iteration point  $x^{k+1}$  only depends the current iteration point  $x^k$  no more previous iteration points.

By the optimal conditions of equation (1), the original "mirror" form of mirror gradient method is described as

$$\nabla h(x^{k+1}) = \nabla h(x^k) - \alpha_k \nabla f(x^k), x \in \mathbb{S}$$

where the convex function  $h$  induces the Bregman divergence.

One special method is called **entropic mirror descent**(**Multiplicative Weights Update, Exponential Gradient Descent**) when the Bregman divergence induced by  $e^x$  and the constraint set  $\mathbb{S} \subset \mathbb{R}^n$  is simplex, i.e.

$$\sum_{i=1}^n x_i = 1, \forall x_i \geq 0.$$

**Entropic descent method** at step  $k$  is given as follows:

$$x_i^{k+1} = \frac{x_i^k \exp(-\alpha \nabla f(x^k))}{\sum_{j=1}^n x_j^k \exp(-\alpha \nabla f(x^k))}, i = 1, 2, \dots, n.$$

See more on the following link list.

- [Bregman Divergence and Mirror Descent, Xinhua Zhang\(张歆华\)](#)
- [CS 294 / Stat 260, Fall 2014: Learning in Sequential Decision Problems](#)
- [ELE522: Large-Scale Optimization for Data Science , Yuxin Chen, Princeton University, Fall 2019](#)
- [Mirror Descent and the Multiplicative Weight Update Method, CS 435, 201, Nisheeth Vishnoi](#)
- <http://www.divergence-methods.org/>
- <https://zhuanlan.zhihu.com/p/34299990>
- <https://blogs.princeton.edu/imabandit/2013/04/16/orf523-mirror-descent-part-iii/>
- <https://blogs.princeton.edu/imabandit/2013/04/18/orf523-mirror-descent-part-iiii/>
- <https://tlienart.github.io/pub/csml/cvxopt/mda.html>
- <https://web.stanford.edu/class/cs229t/2017/Lectures/mirror-descent.pdf>
- <https://www.cs.ubc.ca/labs/lci/mlrg/slides/mirrorMultiLevel.pdf>

## Proximal Gradient Method

The **proximal mapping (or prox-operator)** of a convex function  $h$  is defined as

$$\text{prox}_h(x) = \arg \min_u \{h(u) + \frac{1}{2} \|x - u\|_2^2\}$$

Unconstrained problem with cost function split in two components:

$$\text{minimize} \quad f(x) = g(x) + h(x)$$

- $g$  is convex, differentiable;
- $h$  is closed, convex, possibly non-differentiable while  $\text{prox}_h(x)$  is inexpensive.

### Proximal gradient algorithm

$$x^k = \text{prox}_{t_k h} \{x^{k-1} - t_k \nabla g(x^{k-1})\}$$

$t_k > 0$  is step size, constant or determined by line search.

$$x^+ = \text{prox}_{t_h}(x - t \nabla g(x))$$

from the definition of proximal mapping:

$$\begin{aligned}
x^+ &= \arg \min_u (h(u) + \frac{1}{2} \|u - (x - t \nabla g(x))\|_2^2) \\
&= \arg \min_u (h(u) + g(x) + \nabla g(x)^T (u - x) + \frac{1}{2t} \|u - x\|_2^2) \\
&= \arg \min_u (h(u) + \nabla g(x)^T (u - x) + \frac{1}{2t} \|u - x\|_2^2)
\end{aligned}$$

$x^+$  minimizes  $h(u)$  plus a simple quadratic local model of  $g(u)$  around  $x$ .

And projected gradient method is a special case of proximal gradient methods with

$$h(x) = \delta_C(x) = \begin{cases} 0, & \text{if } x \in C; \\ \infty, & \text{otherwise.} \end{cases}$$

And it is natural to consider a more general algorithm by replacing the squared Euclidean distance in definition of **proximal mapping** with a Bregman distance:

$$\arg \min_u \{h(u) + \frac{1}{2} \|x - u\|_2^2\} \rightarrow \arg \min_u \{h(u) + B(x, u)\}.$$

so that the primary proximal gradient methods are modified to the Bregman version, which is called as **Bregman proximal gradient** method.

- <http://www.seas.ucla.edu/~vandenbe/236C/lectures/proxgrad.pdf>
- <https://people.eecs.berkeley.edu/~elghaoui/Teaching/EE227A/lecture18.pdf>
- <https://arxiv.org/abs/1808.03045>
- [A unified framework for Bregman proximal methods: subgradient, gradient, and accelerated gradient schemes](#)
- [A collection of proximity operators implemented in Matlab and Python.](#)
- [Proximal Algorithms, N. Parikh and S. Boyd](#)
- [For more \(non exhaustive\) informations regarding the proximity operator and the associated proximal algorithms](#)

## Proximal and Projected Newton Methods

A fundamental difference between gradient descent and Newton's method was that the latter also iteratively minimized quadratic approximations, but these used the local Hessian of the function in question.

Proximal Newton method can be also applied to such optimization problem:

$$\begin{aligned}
y^k &= \text{prox}_{H_{k-1}}(x^{k-1} - H_{k-1}^{-1} \nabla g(x^{k-1})) \\
x^k &= x^{k-1} + t_k(y^k - x^{k-1}).
\end{aligned}$$

- <http://www.stat.cmu.edu/~ryantibs/convexopt-S15/lectures/24-prox-newton.pdf>

---

*Note:* the projection from a point  $x^0$  into a subset  $C \subset \mathbb{R}^n$  is defined in proximal operator as

$$x^+ = \arg \min_x \{ \delta_C(x) + \frac{1}{2} \|x - x^0\|_2^2 \}$$

while it can also written in the following form:

$$x^+ = \arg \min_x \{ \frac{1}{1_C(x)} \cdot \|x - x^0\|_2^2 \}$$

where

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

How we can generalize this form into the proximal form? And what is the difference with the original addition proximal operator?

If  $x^0$  is in the set  $C$ , the projection can be rewritten in proximal operator:

$$x^+ = \arg \min_x \{ \exp[\delta_C(x)] \cdot \frac{1}{2} \|x - x^0\|_2^2 \}.$$

How we can generalize the function  $\delta_C$  to more general convex function? What are the advantages of this generalization? As likelihood and log-likelihood, we can transfer the product of some functions to sum by taking logarithm.

$$\begin{aligned} x^+ &= \arg \min_x \{ \exp[\delta_C(x)] \cdot \frac{1}{2} \|x - x^0\|_2^2 \} \\ &= \arg \min_x \log \{ \exp[\delta_C(x)] \cdot \frac{1}{2} \|x - x^0\|_2^2 \} \\ &= \arg \min_x \{ \delta_C(x) + \log(\|x - x^0\|_2^2) \}. \end{aligned}$$

## Lagrange Multipliers and Duality

It is to solve the constrained optimization problem

$$\arg \min_x f(x), \quad s. t. \quad g(x) = b.$$

The barrier or penalty function methods are to add some terms to  $f(x) + \Omega(g(x) - b)$  [converting constrained problems into unconstrained problems by introducing an artificial penalty for violating the constraint](#).

For example,

$$P(x, \lambda) = f(x) + \lambda \|g(x) - b\|_2^2$$

where the penalty function  $\Omega(x) = \|x\|_2^2$ ,  $\lambda \in \mathbb{R}^+$ .

We can regard it as a surrogate loss technique.

Although the penalty function is convex and differentiable, it is more difficult than directly optimizing  $f(x)$  when the constraint is complicated.

What is the simplest additional term when the constraint is linear? In following context we will talk the optimization problem with linear constraints:

$$\begin{aligned} \arg \min_x f(x) \\ s. t. Ax = b \end{aligned}$$

## Lagrange Multipliers and Generalized Lagrange Function

The penalty function methods do not take the optimal conditions into consideration although it works.

If  $x^*$  is in the solution set of the optimization problem above, it is obvious that  $Ax^* = b$  and  $L(x^*, \lambda) = f(x^*)$  where

$$L(x, \lambda) = f(x) + \lambda^T(Ax - b).$$

In another direction, we want to prove that  $x^*$  is the optima of the optimization problem if

$$L(x^*, \lambda^*) = \min_x [\max_{\lambda} L(x, \lambda)].$$

By the definition,

$L(x^*, \lambda^*) \geq L(x^*, \lambda) = f(x^*) + \lambda^T(Ax^* - b)$ , which implies that  $\lambda^T(Ax^* - b) = 0$  i.e.,  $Ax^* = b$ . And

$L(x^*, \lambda^*) = f(x^*) \leq L(x, \lambda^*) \forall x$  if  $Ax = b$

thus  $x^*$  is the solution to the primary problem.

It is dual problem is in the following form:

$$\max_{\lambda} [\min_x L(x, \lambda)].$$

Note that

$$\begin{aligned} \min_x L(x, \lambda) \leq L(x, \lambda) \leq \max_{\lambda} L(x, \lambda) \implies \\ \max_{\lambda} [\min_x L(x, \lambda)] \leq \max_{\lambda} L(x, \lambda) \leq \min_x [\max_{\lambda} L(x, \lambda)]. \end{aligned}$$

And note that necessary condition of extrema is that the gradients are equal to 0s:

$$\begin{aligned} \frac{\partial L(x, \lambda)}{\partial x} &= \frac{\partial f(x)}{\partial x} + A\lambda = 0 \\ \frac{\partial L(x, \lambda)}{\partial \lambda} &= Ax - b = 0 \end{aligned}$$



**Dual Ascent** takes advantages of this properties:

$$1. x^{k+1} = \arg \min_x L(x, \lambda);$$



$$2. \lambda^{k+1} = \lambda^k + \alpha_k (Ax^{k+1} - b).$$

If the constraints are more complex, **KKT theorem** may be necessary.

- [Duality](#)
- [https://cs.stanford.edu/people/davidknowles/lagrangian\\_duality.pdf](https://cs.stanford.edu/people/davidknowles/lagrangian_duality.pdf)
- <https://people.eecs.berkeley.edu/~elghaoui/Teaching/EE227A/lecture7.pdf>
- <https://www.svm-tutorial.com/2016/09/duality-lagrange-multipliers/>
- [https://www.cs.jhu.edu/~svitlana/papers/non\\_refereed/optimization\\_1.pdf](https://www.cs.jhu.edu/~svitlana/papers/non_refereed/optimization_1.pdf)
- [http://web.mit.edu/dimitrib/www/lagr\\_mult.html](http://web.mit.edu/dimitrib/www/lagr_mult.html)
- <https://zhuanlan.zhihu.com/p/50823110>

*exponential augmented Lagrangian method*

- [An exponential augmented Lagrangian method with second order convergence](#)
- [On the convergence of the exponential multiplier method for convex programming](#)

## Splitting Methods

Alternating direction method of multipliers is called **ADMM** shortly.

It is aimed to solve the following convex optimization problem:

$$\begin{aligned} \min F(x, y) \{ &= f(x) + g(y) \} && \text{(cost function)} \\ Ax + By &= b && \text{(constraint)} \end{aligned}$$

where  $f(x)$  and  $g(y)$  is convex;  $A$  and  $B$  are matrices.

Define the augmented Lagrangian:

$$L_\beta(x, y) = f(x) + g(y) - \lambda^T (Ax + By - b) + \frac{\beta}{2} \|Ax + By - b\|_2^2.$$

---

Augmented Lagrange Method at step  $k$  is described as following

$$\begin{aligned} 1. (x^{k+1}, y^{k+1}) &= \arg \min_{x \in \mathbf{X}} L_\beta(x, y, \lambda^k); \\ 2. \lambda^{k+1} &= \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b). \end{aligned}$$


---

ADMM at step  $t$  is described as following:

1.  $x^{k+1} = \arg \min_{x \in \mathbf{X}} L_{\beta}(x, y^k, \lambda^k);$
2.  $y^{k+1} = \arg \min_{y \in \mathbf{Y}} L_{\beta}(x^{k+1}, y, \lambda^k);$
3.  $\lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b).$

The convergence proof of ADMM in convex optimization can be reduced to [verifying the stability of a dynamical system](#) or based on the optimal condition [variational inequality](#) like [On the O\(1/t\) convergence rate of alternating direction method](#).

### Linearized ADMM

Note that the  $x$  subproblem in ADMM

$$\begin{aligned}
 & \arg \min_x L_{\beta}(x, y^k, \lambda^k) \\
 &= \arg \min_x \{f(x) + g(y^k) + \lambda^{kT}(Ax + By^k - b) + \frac{\beta}{2} \|Ax + By^k - b\|_2^2\} \\
 &= \arg \min_x f(x) + \frac{\beta}{2} \|Ax + By^k - b - \frac{1}{\beta} \lambda^k\|_2^2
 \end{aligned} \tag{1}$$

However, the

solution of the subproblem (1) does not have the closed form solution because of the general structure of the matrix  $A$ . In this case, we linearize the quadratic term of

$$\frac{\beta}{2} \|Ax + By^k - b - \frac{1}{\beta} \lambda^k\|_2^2$$

at  $x^k$  and add a proximal term  $\frac{r}{2} \|x - x^k\|_2^2$  to the objective function.

In another word, we solve the following  $x$  subproblem if ignoring the constant term of the objective function:

$$\min_x f(x) + \beta(Ax)^T(Ax^k + By^k - b - \frac{1}{\lambda^k}) + \frac{r}{2} \|x - x^k\|_2^2.$$

1.  $x^{k+1} = \arg \min_{x \in \mathbf{X}} f(x) + \beta(Ax)^T(Ax^k + By^k - b - \frac{1}{\lambda^k}) + \frac{r}{2} \|x - x^k\|_2^2,$
2.  $y^{k+1} = \arg \min_{y \in \mathbf{Y}} L_{\beta}(x^{k+1}, y, \lambda^k),$
3.  $\lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b).$

For given  $\beta > 0$ , choose  $r$  such that

the matrix  $rI_1 - \beta A^T A$  is definitely positive, i.e.,

$$rI_1 - \beta A^T A \geq 0.$$

We can also linearize the  $y$  subproblem:

$$1. x^{k+1} = \arg \min_{x \in \mathbf{X}} L_{\beta}(x, y^k, \lambda^k),$$

2.  $y^{k+1} = \arg \min_{y \in \mathbf{Y}} g(y) + \beta (By)^T (Ax^{k+1} + By^k - b - \frac{1}{\beta} \lambda^k) + \frac{r}{2} \|y - y^k\|^2,$
3.  $\lambda^{k+1} = \lambda^k - \beta (Ax^{k+1} + By^{k+1} - b).$

For given  $\beta > 0$ , choose  $r$  such that  
the matrix  $rI_2 - \beta B^T B$  is definitely positive, i.e.,

$$rI_2 - \beta B^T B \geq 0.$$

---

Taking  $\mu \in (0, 1)$  (usually  $\mu = 0.9$ ), the **Symmetric ADMM** is described as

1.  $x^{k+1} = \arg \min_{x \in \mathbf{X}} L_\beta(x, y^k, \lambda^k),$
2.  $\lambda^{k+\frac{1}{2}} = \lambda^k - \mu \beta (Ax^{k+1} + By^k - b),$
3.  $y^{k+1} = \arg \min_{y \in \mathbf{Y}} L_\beta(x^{k+1}, y, \lambda^{k+\frac{1}{2}}),$
4.  $\lambda^{k+1} = \lambda^{k+\frac{1}{2}} - \mu \beta (Ax^{k+1} + By^{k+1} - b).$

- [http://www.optimization-online.org/DB\\_FILE/2015/05/4925.pdf](http://www.optimization-online.org/DB_FILE/2015/05/4925.pdf)

Thanks to Professor He Bingsheng who taught me those.[^9]



He Bingsheng

---

One of the particular ADMM is also called **Split Bregman** methods. And **Bregman ADMM** replace the quadratic penalty function with Bregman divergence:

$$L_\beta^\phi(x, y) = f(x) + g(y) - \lambda^T (Ax + By - b) + \frac{\beta}{2} B_\phi(b - Ax, By).$$

where  $B_\phi$  is the Bregman divergence induced by the convex function  $\phi$ .

## BADMM

1.  $x^{k+1} = \arg \min_{x \in \mathbf{X}} L_{\beta}^{\phi}(x, y^k, \lambda^k);$
2.  $y^{k+1} = \arg \min_{y \in \mathbf{Y}} L_{\beta}^{\phi}(x^{k+1}, y, \lambda^k);$
3.  $\lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b).$

- <https://arxiv.org/abs/1306.3203>
- <https://www.swmath.org/software/20288>

## Relaxed, Inertial and Fast ADMM

- <http://bipop.inrialpes.fr/people/malick/Docs/15-titan-iutzeler.pdf>
- <http://www.iutzeler.org/pres/osl2017.pdf>
- <https://www.mia.uni-saarland.de/Publications/goldstein-cam12-35.pdf>
- [The Classical Augmented Lagrangian Method and Nonexpansiveness](#)

## Multi-Block ADMM

Firstly we consider the following optimization problem

$$\begin{aligned} \min & f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n) \\ \text{s. t.} & A_1x_1 + A_2x_2 + \cdots + A_nx_n = b, \\ & x_i \in X_i \in \mathbb{R}^{d_i}, i = 1, 2, \cdots, n. \end{aligned}$$

We defined its augmented Lagrangian multipliers as

$$L_{\beta}^n(x_1, x_2, \cdots, x_n | \lambda) = \sum_{i=1}^n f_i(x_i) - \lambda^T \left( \sum_{i=1}^n A_i x_i - b \right) + \frac{\beta}{2} \left\| \sum_{i=1}^n A_i x_i - b \right\|_2^2.$$

Particularly, we firstly consider the case when  $n = 3$ :

$$\begin{aligned} L_{\beta}^3(x, y, z | \lambda) &= f_1(x) + f_2(y) + f_3(z) - \lambda^T (A_1x + A_2y + A_3z - b) \\ &\quad + \frac{\beta}{2} \|A_1x + A_2y + A_3z - b\|_2^2. \end{aligned}$$

It is natural and computationally beneficial to extend the original ADMM directly to solve the general n-block problem. A counter-example shows that this method diverges.

---

And [Professor Bingsheng He](#), who taught me this section in his class, and his coauthors proposed some schemes for this problem based on his unified frame work for convex optimization and monotonic variational inequality.

[Parallel splitting augmented Lagrangian method](#) (abbreviated to **PSALM** ) is described as follows:

1.  $x^{k+1} = \arg \min_x \{L_\beta^3(x, y^k, z^k, \lambda^k) \mid x \in \mathbb{X}\};$
2.  $y^{k+1} = \arg \min_x \{L_\beta^3(x^{\textcolor{red}{k+1}}, y, z^k, \lambda^k) \mid y \in \mathbb{Y}\};$
3.  $z^{k+1} = \arg \min_x \{L_\beta^3(x^{\textcolor{red}{k+1}}, y^{\textcolor{yellow}{k}}, z, \lambda^k) \mid z \in \mathbb{Z}\};$
4.  $\lambda^{k+1} = \lambda^k - \beta(A_1 x^{k+1} + A_2 y^{k+1} + A_3 z^{k+1} - b).$

We can add one more correction step

$$v^{k+1} := v^k - \alpha(v^k - v^{k+1}), \alpha \in (0, 2 - \sqrt{2})$$

where  $v^{k+1} = (y^{k+1}, z^{k+1}, \lambda^{k+1})$ .

Another approach is to add an regularized terms:

1.  $x^{k+1} = \arg \min_x \{L_\beta^3(x, y^k, z^k, \lambda^k) \mid x \in \mathbb{X}\},$
2.  $y^{k+1} = \arg \min_x \{L_\beta^3(x^{\textcolor{red}{k+1}}, y, z^k, \lambda^k) + \frac{\tau}{2}\beta\|A_2(y - y^k)\|^2 \mid y \in \mathbb{Y}\},$
3.  $z^{k+1} = \arg \min_x \{L_\beta^3(x^{\textcolor{red}{k+1}}, y^{\textcolor{yellow}{k}}, z, \lambda^k) + \frac{\tau}{2}\beta\|A_3(z - z^k)\|^2 \mid z \in \mathbb{Z}\},$
4.  $\lambda^{k+1} = \lambda^k - \beta(A_1 x^{k+1} + A_2 y^{k+1} + A_3 z^{k+1} - b),$

where  $\tau > 1$ .

- <http://scis.scichina.com/en/2018/122101.pdf>
- <http://maths.nju.edu.cn/~hebma/slides/17C.pdf>
- <http://maths.nju.edu.cn/~hebma/slides/18C.pdf>
- <https://link.springer.com/article/10.1007/s10107-014-0826-5>

### Davis-Yin three operator splitting

If  $f_1$  is strongly convex, then apply Davis-Yin (to dual problem) gives:

1.  $x^{k+1} = \arg \min_x \{L^3(\textcolor{green}{x}, y^k, z^k, \lambda^k) \mid x \in \mathbb{X}\};$
2.  $y^{k+1} = \arg \min_x \{L_\beta^3(x^{k+1}, \textcolor{green}{y}, z^k, \lambda^k) \mid y \in \mathbb{Y}\};$
3.  $z^{k+1} = \arg \min_x \{L_\beta^3(x^{k+1}, y^{k+1}, \textcolor{green}{z}, \lambda^k) \mid z \in \mathbb{Z}\};$
4.  $\lambda^{k+1} = \lambda^k - \beta(A_1 x^{k+1} + A_2 y^{k+1} + A_3 z^{k+1} - b).$

where the notation  $L^3(x, y, z, \lambda)$  is deefined by

$$L^3(x, y, z, \lambda) = f_1(x) + f_2(y) + f_3(z) - \lambda^T(A_1 x + A_2 y + A_3 z - b)$$

is the Lagrangian rather than augmented Lagrangian.

- <http://fa.bianp.net/blog/2018/tos/>
  - <https://link.springer.com/article/10.1007%2Fs11228-017-0421-z>
  - [Three-Operator Splitting and its Optimization Applications](#)
  - <ftp://ftp.math.ucla.edu/pub/camreport/cam15-13.pdf>
  - [A Three-Operator Splitting Scheme and its Applications](#)
- 

**Randomly Permuted ADMM** given initial values at round  $k$  is described as follows:

1. Primal update:

- Pick a permutation  $\sigma$  of  $1, \dots, n$  uniformly at random;
- For  $i = 1, 2, \dots, n$ , compute  $x_{\sigma(i)}^{k+1}$  by

$$x_{\sigma(i)}^{k+1} = \arg \min_{x_{\sigma(i)}} L(x_{\sigma(1)}^{k+1}, \dots, x_{\sigma(i-1)}^{k+1}, x_{\sigma(i)}, x_{\sigma(i+1)}^{k+1}, \dots \mid \lambda^k).$$

2. Dual update. Update the dual variable by

$$\lambda^{k+1} = \lambda^k - \mu \left( \sum_{i=1}^n A_i x_i - b \right)$$

- [Randomly Permuted ADMM](#)
  - [http://opt-ml.org/oldopt/papers/OPT2015\\_paper\\_47.pdf](http://opt-ml.org/oldopt/papers/OPT2015_paper_47.pdf)
  - <https://arxiv.org/abs/1503.06387>
  - [Multi-Block ADMM and its Convergence](#)  
[Random Permutation Helps-A talk by Ye](#)
- 

- <http://maths.nju.edu.cn/~hebma/>
- <http://stanford.edu/~boyd/admm.html>
- [A General Analysis of the Convergence of ADMM](#)
- [用ADMM实现统计学习问题的分布式计算](#)
- [https://www.wikiwand.com/en/Augmented\\_Lagrangian\\_method](https://www.wikiwand.com/en/Augmented_Lagrangian_method)
- [凸优化：ADMM\(Alternating Direction Method of Multipliers\)交替方向乘子算法](#)
- [Splitting methods and ADMM, Thibaut Lienart](#)
- [Split Bregman](#)
- [Accelerated Bregman operator splitting with backtracking](#)
- [Splitting Algorithms, Modern Operator Theory, and Applications \(17w5030\)](#)
- [17w5030 Workshop on](#)  
[Splitting Algorithms, Modern Operator Theory,](#)

**Primal-dual fixed point algorithm and Primary Dual Hybrid Gradient**

- [A primal-dual fixed point algorithm for multi-block convex minimization](#)
  - [A primal–dual fixed point algorithm for convex separable minimization](#)
  - [A Unified Primal-Dual Algorithm Framework Based on Bregman Iteration](#)
  - [Proximal ADMM](#)
  - [PDHG](#)
- 

The methods such as ADMM, proximal gradient methods do not optimize the cost function directly. For example, we want to minimize the following cost function

$$f(x, y) = g(x) + h(y)$$

with or without constraints.

Specially, if the cost function is additionally separable,  $f(x) = f_1(x_1) + f_2(x_2) + \cdots + f_n(x_n)$ , we would like to minimize the sub-function or component function  $f_i(x_i)$ ,  $i = 1, 2, \dots, n$  rather than the cost function itself

$$\min \sum_i f_i(x_i) \leq \sum_i \min f_i(x_i).$$

And ADMM or proximal gradient methods are to split the cost function to 2 blocks, of which one is differentiable and smooth while the other may not be differentiable. In another word, we can use them to solve some non-smooth optimization problem.

However, what if there is no constraints application to the optimization problem?

**Coordinate descent** is aimed to minimize the following cost function

$$f(x) = g(x) + \sum_i^n h_i(x_i)$$

where  $g(x)$  is convex, differentiable and each  $h_i(x)$  is convex.

We can use coordinate descent to find a minimizer: start with some initial guess  $x^0$ , and repeat for  $k = 1, 2, 3, \dots$ :

---

1.  $x_1^k \in \arg \min_{x_1} f(x_1, x_2^{k-1}, x_3^{k-1}, \dots, x_n^{k-1});$
2.  $x_2^k \in \arg \min_{x_2} f(x_1^k, x_2, x_3^{k-1}, \dots, x_n^{k-1});$
3.  $x_3^k \in \arg \min_{x_3} f(x_1^k, x_2^k, x_3, \dots, x_n^{k-1});$
4.  $\vdots$
5.  $x_n^k \in \arg \min_{x_n} f(x_1^k, x_2^k, x_3^k, \dots, x_n).$

It can be extended to block coordinate descent (BCD) if the variables  $x_1, x_2, \dots, x_n$  are separable in some blocks.

---

- <http://bicmr.pku.edu.cn/conference/opt-2014/index.html>
- [https://calculus.subwiki.org/wiki/Additively\\_separable\\_function](https://calculus.subwiki.org/wiki/Additively_separable_function)
- <https://www.cs.cmu.edu/~ggordon/10725-F12/slides/25-coord-desc.pdf>
- [http://bicmr.pku.edu.cn/~wenzw/opt2015/multiconvex\\_BCD.pdf](http://bicmr.pku.edu.cn/~wenzw/opt2015/multiconvex_BCD.pdf)

## Surrogate Optimization

It is a unified principle that we optimize an objective function via sequentially optimizing surrogate functions such as **EM**, **ADMM**.

It is obvious that the choice of optimization methods relies on the objective function. Surrogate loss transform the original problem  $\min_x f(x)$  into successive trackable subproblems:

$$x^{k+1} = \arg \min_x Q_k(x).$$

We will call  $Q_k(x)$  surrogate function. Surrogate function is also known as **merit function**.

A good surrogate function should:

- Approximate the objective function.
- Easy to optimize.

This always leads to the **convexification technique** where the surrogate function is convex. For example, we can rewrite gradient descent in the following form

$$x^{k+1} = \arg \min_x \{f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2\alpha_k} \|x - x^k\|_2^2\}.$$

In Newton's method, we approximate the objective with a quadratic surrogate of the form:

$$Q_k(x) = f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2\alpha_k} (x - x^k)^T H_k (x - x^k).$$

Note that the Hessian matrix  $H_k$  is supposed to be positive definite. The quasi-Newton methods will approximate the Hessian matrix with some inverse symmetric matrix. And they can rewrite in the principle of surrogate function, where the surrogate function is convex in the form of linear function + squared functions in some sense.

Note that momentum methods can be rewrite in the surrogate function form:

$$\begin{aligned} x^{k+1} &= x^k - \alpha_k \nabla_x f(x^k) + \rho_k (x^k - x^{k-1}) \\ &= \arg \min_x \{f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{2\alpha_k} \|x - x^k - \rho_k (x^k - x^{k-1})\|_2^2\}. \end{aligned}$$



*PS:* How we can extend it to Nesterov gradient methods or stochastic gradient methods?

Discover acceleration

It is natural to replace the squared function with some non-negative function such as mirror gradient methods

$$x^{k+1} = \arg \min_x \{f(x^k) + \langle \nabla f(x^k), x - x^k \rangle + \frac{1}{\alpha_k} B(x, x^k)\}.$$

More generally, auxiliary function  $g_k(x)$  may replace the Bregman divergence  $B(x, x^k)$ , where the auxiliary functions  $g_k(x)$  have the properties  $g_k(x) \geq 0$  and  $g_k(x^{k-1}) = 0$ .

- <http://fa.bianp.net/teaching/2018/eecs227at/>
- <http://fa.bianp.net/teaching/2018/eecs227at/newton.html>
- <http://faculty.uml.edu/cbyrne/NBHSeminar2015.pdf>

The parameters  $\alpha_k$  is chosen so that the surrogate function is convex by observing that the condition number of Hessian matrix is determined by the factor  $\frac{1}{\alpha_k}$ . And gradient are used in these methods to construct the surrogate function. There are still two problems in unconstrained optimization:

1. If the cost function is not smooth or differential such as absolute value function, the gradient is not available so that it is a problem to construct a convex surrogate function without gradient;
2. In another hand, there is no unified principle to construct a convex surrogate function if we know more information of higher order derivatives.

---

**Moreau envelope (or Moreau-Yosida regularization)** of a convex function  $f$  with parameter  $\mu > 0$  is defined as

$$M_{\mu f}(x) = \inf_z \{f(z) + \frac{1}{2\mu} \|z - x\|_2^2\}.$$

Minimizing  $f$  and  $M_f$  are equivalent.

$prox_{\mu f}(x)$  is unique point that achieves the infimum that defines

$M_{\mu f}$ , i.e.

$$M_{\mu f}(x) = f(prox_{\mu f}(x)) + \frac{1}{2} \|prox_{\mu f}(x) - x\|_2^2.$$

**Moreau envelope**  $M_f$  is continuously differentiable with gradients

$$\nabla M_{\mu f} = \frac{1}{\mu} (prox_{\mu f}(x) - x).$$

This means

$$prox_{\mu f}(x) = x - \mu \nabla M_{\mu f}$$

i.e.,  $\text{prox}_{\mu f}(x)$  is gradient step for minimizing  $M_{\mu f}$ .

**Fenchel Conjugate** of a function  $h$  is the function  $h^*$  defined by

$$h^*(x) = \sup_z \{ \langle z, x \rangle - h(z) \}.$$

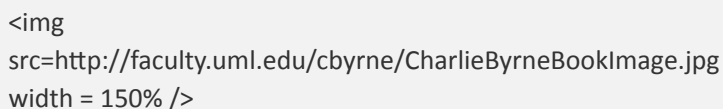
- [Smoothing for nonsmooth optimization, ELE522: Large-Scale Optimization for Data Science](#)
- [Smoothing, EE236C \(Spring 2013-14\)](#)
- [Smoothing and First Order Methods: A Unified Framework](#)

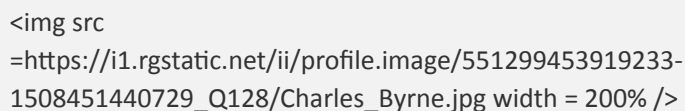
---

[Charles Byrne](#) gives a unified treatment of some iterative optimization algorithms such as auxiliary function methods.

**Young**

**Recent**

 `<img src=http://faculty.uml.edu/cbyrne/CharlieByrneBookImage.jpg width = 150% />`

 `<img src=https://i1.rgstatic.net/ii/profile.image/551299453919233-1508451440729_Q128/Charles_Byrne.jpg width = 200% />`

He is a [featured author of CRC press](#) and [professor in UML](#)

**Auxiliary-Function Methods** minimize the function

$$G_k(x) = f(x) + g_k(x)$$

over  $x \in \mathbb{S}$  for  $k = 1, 2, \dots$  if  $f(x) : \mathbb{S} \mapsto \mathbb{R}$  and  $g_k(x)$  is auxiliary function.

We do not linearize the cost function as the previous methods.

Auxiliary-Function Methods (AF methods) include

- Barrier- and Penalty-Function Algorithms such as sequential unconstrained minimization (SUM) methods, interior-point methods exterior-point methods;
- Proximal Minimization Algorithms such as majorization minimization.

And an AF method is said to be in the SUMMA class if the SUMMA Inequality holds:

$$G_k(x) - G_k(x^k) \geq g_{k+1}(x), \forall x \in \mathbb{S}.$$

Proximal minimization algorithms using Bregman distances, called here **PMAB**, minimize the function

$$G_k(x) = f(x) + B(x, x^{k-1}), \forall x \in \mathbb{S}.$$

The **forward-backward splitting (FBS)** methods is to minimize the function

$$f_1(x) + f_2(x)$$

where both functions

are convex and  $f_2(x)$  is differentiable with its gradient  $L$ -Lipschitz continuous in the Euclidean norm. The iterative step of the FBS algorithm is

$$x^k = \text{prox}_{\gamma f_1}(x^{k-1} - \gamma \nabla f_2(x^{k-1})).$$

It is equivalent to minimize

$$G_k(x) = f(x) + \frac{1}{2\gamma} \|x - x^{k-1}\|_2^2 - B(x, x^{k-1}),$$

where  $B(x, x^{k-1}) = f_1(x) - f_1(x^{k-1}) - \langle \nabla f_1(x^{k-1}), x - x^{k-1} \rangle$ ,  $0 < \gamma \leq \frac{1}{L}$ .

**Alternating Minimization (AM)** can be regarded as coordinate optimization methods with 2 blocks:

- $p^{n+1} = \arg \min_{p \in P} f(p, q^n)$ ,
- $q^{n+1} = \arg \min_{q \in Q} f(p^{n+1}, q)$ ,

where  $f(p, q), p \in P, q \in Q$  is the objective function. It is proved that the sequence  $f(p^n, q^n)$  converge to the minimizer if the

**Five-Point Property** hold:

$$f(p, q) + f(p, q^{n-1}) \geq f(p, q^n) + f(p^n, q^{n-1}).$$

For each  $p$  in the set  $P$ , define  $q(p)$  in  $Q$  as a member of  $Q$  for which  $f(p; q(p)) \leq f(p; q)$ , for all  $q \in P$ . Let  $\hat{f}(p) = f(p; q(p))$ .

At the  $n$ th step of AM we minimize

$$G_n(p) = f(p; q(p)) + [f(p; q^{n-1}) - f(p; q(p))]$$

to get  $p^n$ , where  $g_n(p) = f(p; q^{n-1}) - f(p; q(p))$  is the auxiliary function. So we can write that  $G_n(p) = f(p; q(p)) + g_n(p)$ .

See [Iterative Convex Optimization Algorithms; Part Two: Without the Baillon–Haddad Theorem](#)

or [Alternating Minimization, Proximal Minimization and Optimization Transfer Are Equivalent](#) for more information on AF methods.

Expectation-Maximization can be classified to AF method.

- $Q(\theta|\theta^{(t)}) = \mathbb{E}(\ell(\theta|Y_{obs}, Z)|Y_{obs}, \theta^{(t)})$ ;
- $\theta^{(t+1)} = \arg \min_{\theta} Q(\theta|\theta^{(t)})$ .

The  $Q(\theta|\theta^{(t)})$  function is log-likelihood function of complete data  $(Y_{os}, Z)$  given  $(Y_{obs}, \theta^{(t)})$ .

- [Iterative Convex Optimization Algorithms; Part Two: Without the Baillon–Haddad Theorem](#)
- [Alternating Minimization, Proximal Minimization and Optimization Transfer Are Equivalent](#)
- [A unified treatment of some iterative algorithms in signal processing and image reconstruction](#)

- [Convergence Rate of Expectation-Maximization](#)
- 

[Optimization using surrogate models](#) applied to Gaussian Process models (Kriging).

- [Convexification Procedure and Decomposition Methods for Nonconvex Optimization Problem](#)
  - [A method to convexify functions via curve evolution](#)
  - [Convexification and Global Optimization of Nonlinear Programs](#)
  - [Optimization using surrogate models](#)
- 

- [Surrogate loss function](#)
- [Divergences, surrogate loss functions and experimental design](#)
- [Surrogate Regret Bounds for Proper Losses](#)
- [Bregman Divergences and Surrogates for Learning](#)
- [Meet the Bregman Divergences](#)
- [Some Theoretical Properties of an Augmented Lagrangian Merit Function](#)
- <https://people.eecs.berkeley.edu/~wainwrig/stat241b/lec11.pdf>
- <http://fa.bianp.net/blog/2014/surrogate-loss-functions-in-machine-learning/>

## Fixed Point Iteration Methods

The fixed point algorithm is initially to find approximate solutions of the equation

$$f(x) = 0 \tag{1}$$

where  $f : \mathbb{R} \rightarrow \mathbb{R}, x \in \mathbb{R}^n$ .

In this method, we first rewrite the question(1) in the following form

$$x = g(x) \tag{2}$$

in such a way that any solution of the equation (2), which is a fixed point of the function  $g$ , is a solution of equation (1). For example, we can set  $g(x) = f(x) + x, g(x) = x - f(x)$ .

Then consider the following algorithm.

1. Give the initial point  $x^0$ ;
2. Compute the recursive procedure  $x^{n+1} = g(x^n), n = 1, 2, \dots$

So that finally we obtain an sequence  $\{x^0, x^1, \dots, x^n, \dots\}$ . There are many methods to test whether this sequence is convergent or not as learn in calculus.

In high dimensional space, it is a little different. Fixed point iteration as well as the fixed point itself arises in many cases such as [<https://arxiv.org/pdf/1511.06393.pdf>].

The contracting mapping  $F : \mathbb{R}^d \rightarrow \mathbb{R}^d$  is defined as

$$\|F(x) - F(y)\| \leq \alpha \|x - y\|, \forall x, y \in \mathbb{R}, \alpha \in [0, 1).$$

$$\text{Thus } \lim_{\|x-y\| \rightarrow 0} \frac{\|F(x)-F(y)\|}{\|x-y\|} \leq \alpha \in [0, 1).$$

Now we rewrite the necessary condition of unconstrained optimization problems  $\nabla f(x) = 0$  to the fixed point equation:

$$\begin{aligned} \nabla f(x) = 0 &\Rightarrow x - \alpha \nabla f(x) = x \\ \nabla f(x) = 0 &\Rightarrow g(x) - \alpha \nabla f(x) = g(x) \Rightarrow x - g^{-1}(\alpha \nabla f(x)) = x \\ \nabla f(x) = 0 &\Rightarrow H(x)x - \alpha \nabla f(x) = H(x)x \Rightarrow x - \alpha H(x)^{-1} \nabla f(x) = x \\ \nabla f(x) = 0 &\Rightarrow M(x) \nabla f(x) = 0 \Rightarrow x - \alpha M(x) \nabla f(x) = x \end{aligned}$$

where  $H(x)$  is the lambda-matrix.

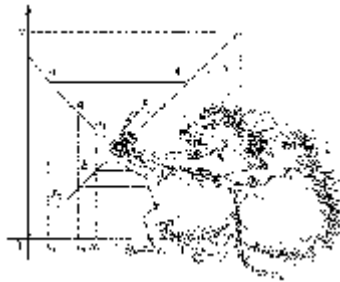
These correspond to gradient descent, mirror gradient methods, Newton's methods and quasi-Newton's methods, respectively.

And the projected (sub)gradient methods are in the fixed-point iterative form:

$$x = Proj_{\mathbb{S}}(x - \alpha \nabla f(x))$$

as well as the mirror gradient and proximal gradient methods different from the projection operator.

Expectation maximization is also an accelerated [fixed point iteration](#) as well as Markov chain.



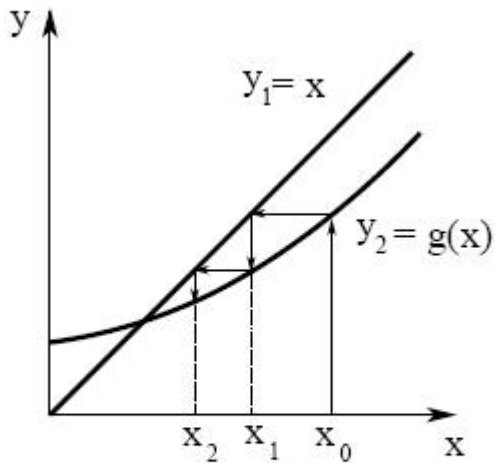
<http://www.drkhamsi.com/fpt/books.html>

The following figures in the table is form [Formulations to overcome the divergence of iterative method of fixed-point in nonlinear equations solution](#)

## Fixed Point Iterations

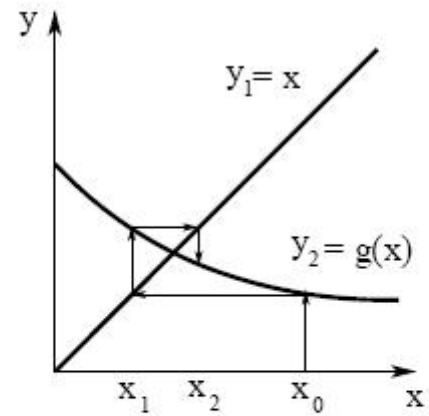
---

## Fixed Point Iterations



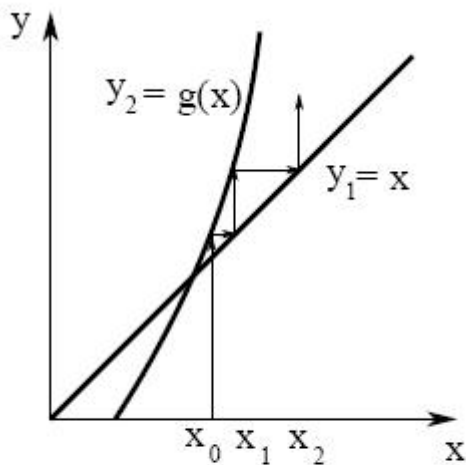
**Figure 1.** Convergence for  $0 < g'(x) < 1$  (monotone behavior).

Source: own work.



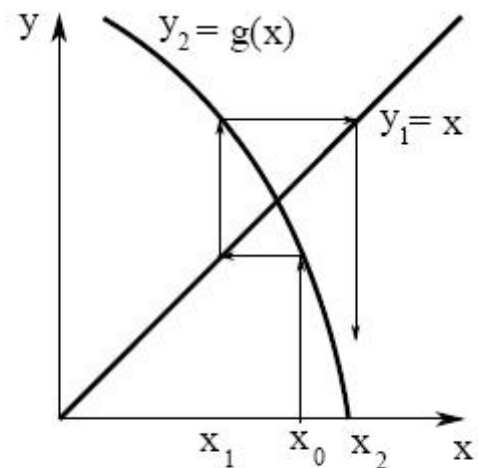
**Figure 2.** Convergence for  $-1 < g'(x) < 0$  (oscillatory behavior).

Source: own work.



**Figure 3.** Divergence for  $g'(x) > 1$  (monotone behavior).

Source: own work.



**Figure 4.** Divergence for  $g'(x) < -1$  (oscillatory behavior).

Source: own work.

- [https://www.wikiwand.com/en/Fixed-point\\_theorem](https://www.wikiwand.com/en/Fixed-point_theorem)
- [Fixed-Point Iteration](#)
- [Lecture 8 : Fixed Point Iteration Method, Newton's Method](#)
- [FixedPoint: A suite of acceleration algorithms with Application](#)
- [Books on Fixed Point Theory](#)

## ISTA and FASTA

The  $\ell_1$  regularization is to solve the ill-conditioned equations such as

$$\min_x \|Ax - b\|_2^2 + \lambda \|x\|_1.$$

It is less sensitive to outliers and obtain much more sparse solutions (as opposed to  $\ell_2$  regularization).  
 Its application includes and is not restricted in *LASSO*, *compressed sensing* and *sparse approximation of signals*.

It is clear that the absolute value function is not smooth or differentiable everywhere even the objective function  $\|Ax - b\|_2^2 + \lambda\|x\|_1$  is convex.

It is not best to solve this problem by gradient-based methods.

**Iterative Shrinkage-Threshold Algorithms (ISTA)** for  $\ell_1$  regularization is

$$x^{k+1} = \mathbf{T}_{\lambda t}(x^k - tA^T(Ax - b))$$

where  $t > 0$  is a step size and  $\mathbf{T}_\alpha$  is the shrinkage operator defined by

$$\mathbf{T}_\alpha(x)_i = (x_i - \alpha)_+ \text{sgn}(x_i)$$

where  $x_i$  is the  $i$  th component of  $x \in \mathbb{R}^n$ .

**FISTA with constant stepsize**

- $x^k = p_L(y^k)$  computed as ISTA;
- $t_{k+1} = \frac{1 + \sqrt{1 + 4t_k^2}}{2}$ ;
- $y^{k+1} = x^k + \frac{t_k - 1}{t_{k+1}}(x^k - x^{k-1})$ .

- [A Fast Iterative Shrinkage Algorithm for Convex Regularized Linear Inverse Problems](#)
- [https://pylops.readthedocs.io/en/latest/gallery/plot\\_ista.html](https://pylops.readthedocs.io/en/latest/gallery/plot_ista.html)
- [ORF523: ISTA and FISTA](#)

This will lead to the operator splitting methods analysed by [Wotao Yin](#) and others.

Wotao Yin wrote a summary on [First-order methods and operator splitting for optimization](#):

*First-order methods are described and analyzed with gradients or subgradients, while second-order methods use second-order derivatives or their approximations.*

*During the 70s–90s the majority of the optimization community focused on second-order methods since they are more efficient for those problems that have the sizes of interest at that time. Beginning around fifteen years ago, however, the demand to solve ever larger problems started growing very quickly. Many large problems are further complicated by non-differentiable functions and constraints. Because simple first-order and classic second-order methods are ineffective or infeasible for these problems, operator splitting methods regained their popularity.*

Operators are used to develop algorithms and analyze them for a wide spectrum of problems including optimization problems, variational inequalities, and differential equations. Operator splitting is a set of ideas that generate algorithms through decomposing a problem that is too difficult as a whole into two or more smaller and simpler subproblems. During the decomposition, complicated structures like non-differentiable functions, constraint sets, and distributed problem structures end up in different subproblems and thus can be handled elegantly. We believe ideas from operator splitting provide the most effective means to handle such complicated structures for computational problem sizes of modern interest.

- [ORIE 6326: Convex Optimization Operator Splitting](#)
- [Monotone Operator Splitting Methods](#)
- [A Course on First-Order, Operator Splitting, and Coordinate Update Methods for Optimization](#)
- [Operator Splitting Methods for Convex Optimization Analysis and Implementation](#)
- [Some Operator Splitting Methods for Convex Optimization](#)

Krasnosel'skii-Mann(KM, or averaged) iterations update  $x^k$  in iteration  $k$  to

$$x^{k+1} = (1 - \alpha_k)x^k + \alpha_k T(x^k), \alpha_k \in (0, 1)$$

for the fixed point problem (2). Specially,  $T(x^k) = x^k - \alpha_k \nabla f(x^k)$  for the convex optimization problems.

If the operator  $T$  is non-expansive, then the sequence  $\{x^k \mid k = 0, 1, 2, \dots\}$  is convergent by **Krasnosel'skii-Mann Theorem**.

In 1953, [Mann](#) defined an iterative method:

$$x^{k+1} = M(x^k, \alpha_k, T) = (1 - \alpha_k)x^k + \alpha_k T(x^k),$$

$$\alpha_k \in [0, 1) \text{ satisfying } \sum_{k=1}^{\infty} \alpha_k = \infty.$$

The sequence  $\{x^k\}$  defined by

$$x^{k+1} = (1 - \alpha_k)x^k + \alpha_k T^k(x^k), 0 < a \leq \alpha_k \leq b < 1$$

(additionally  $\sum_{k=1}^{\infty} \alpha_k = \infty$ ) is called a **modified Mann iteration**.

Take  $\{\alpha_k, \beta_k\}$  two sequences in  $[0, 1]$  satisfying

$$\sum_{k=1}^{\infty} \alpha_k \beta_k = \infty, \lim_{k \rightarrow \infty} \beta_k = 0, 0 \leq \alpha_k \leq \beta_k \leq 1.$$

Then the sequence  $\{x^k\}$  defined by



$$\begin{aligned}x^{k+1} &= (1 - \alpha_k)x^k + \alpha_k T^k(y^k), \\y^k &= (1 - \beta_k)x^k + \beta_k T(x^k),\end{aligned}$$

is called the **Ishikawa iteration**.

[Hojjat Afshari and Hassen Aydi](#) proposes another Mann type iteration:

$$x^{k+1} = \alpha_k x^k + \beta_k T(x^k) + \gamma_k T^2(x^k)$$

where  $\alpha_k + \beta_k + \gamma_k = 1$ ,  $\alpha_k, \beta_k, \gamma_k \in [0, 1)$  for all  $k \geq 1$ ,  $\sum_k (1 - \alpha_k) = \infty$ ,  $\sum_{k=1}^{\infty} \gamma_k < \infty$ .

If the Mann type iteration  $\{x^k\}$  converges strongly to a point  $p$ , then  $p$  is a fixed point of  $T$ .

- [Krasnoselskii-Mann method for non-self mappings](#)
- <http://www.krasnoselskii.iitp.ru/papereng.pdf>
- [Strong convergence of the modified Mann iterative method for strict pseudo-contractions](#)
- [Some results on a modified Mann iterative scheme in a reflexive Banach space](#)
- [Some results about Krasnosel'skiĭ-Mann iteration process](#)
- [Convergence theorems for inertial KM-type algorithms](#)

There is an acceleration framework of fixed point iterations for the problem (2) called **Anderson Acceleration** or **regularized nonlinear acceleration** :

1.  $F_k = (h_{k-m_k}, \dots, h_k)$  where  $h_i = g(x_i) - x_i$ ;
2. Determine  $\alpha^k = (\alpha_0^k, \dots, \alpha_{m_k}^k)^T$  that solves  $\min_{\alpha^k} \|F_k \alpha^k\|_2^2$  s.t.  $\sum_{i=0}^{m_k} \alpha_i^k = 1$ ;
3. Set  $x_{k+1} = \sum_{i=0}^{m_k} \alpha_i^k g(x_{k-m_k+i})$ .

It is maybe interesting to introduce some Bregman divergence  $D_f(\cdot, \cdot)$  instead of the squared  $\ell_2$  norm when choosing  $\alpha^k$  so that

$$\alpha^k = \arg \min_{\alpha^k} \{D_f(F_k \alpha^k) \mid \sum_{i=0}^{m_k} \alpha_i^k = 1\}.$$

It is proved that the Anderson acceleration converges if the fixed point mapping is contractive.

- [Anderson acceleration for fixed point iterations](#)
- [Anderson Acceleration](#)
- [MS142 Anderson Acceleration and Applications](#)
- [Anderson Acceleration of the Alternating Projections Method for Computing the Nearest Correlation Matrix](#)
- [Convergence Analysis For Anderson Acceleration](#)
- [Using Anderson Acceleration to Accelerate the Convergence of Neutron Transport Calculations with Anisotropic Scattering](#)
- [Comments on "Anderson Acceleration, Mixing and Extrapolation"](#)

The Baillon-Haddad Theorem provides an important link between convex optimization and fixed-point iteration, which proves that if the gradient of a convex and continuously differentiable function is nonexpansive, then it is actually **firmly nonexpansive**.

- [The Baillon-Haddad Theorem Revisited](#)
- [Recent Advances in Convex Optimization and Fixed Point Algorithms by Jean-Christophe Pesquet](#)
- [A Generic online acceleration scheme for Optimization algorithms via Relaxation and Inertia](#)
- [RELAXATION AND INERTIA IN FIXED-POINT ITERATIONS WITH APPLICATION](#)
- [Monotonicity, Acceleration, Inertia, and the Proximal Gradient algorithm](#)
- [Globally Convergent Type-I Anderson Acceleration for Non-Smooth Fixed-Point Iterations](#)
- [On the Convergence Rate of the Halpern-Iteration](#)
- [Iterative Convex Optimization Algorithms; Part One: Using the Baillon–Haddad Theorem](#)

However, it is best to think from the necessary condition of optima in non-convex optimization in my opinion. Another question is to generalize the fixed point iteration to stochastic gradient methods.

## Dynamical Systems

We will focus on the optimization methods in the form of fixed point iteration and dynamical systems. It is to minimize the following function

$$f(x), x \in \mathbb{R}^p, \quad \nabla f(x) = g(x), \quad \nabla^2 f(x) = H(x).$$

| Iteration                              | ODE                              |
|--|----------------------------------|
| $x^k = x^k - \alpha_k g(x^k)$          | $\dot{x}(t) = -g(x(t))$          |
| $x^k = x^k - \alpha_k H_k^{-1} g(x^k)$ | $\dot{x}(t) = -H^{-1}(x)g(x(t))$ |

Like Newton interpolation, more points can compute higher order derivatives. The dynamics of accelerated gradient methods are expected to correspond to higher order differential equations.

Some acceleration methods are iterations of the corresponding algorithms of **Asymptotic Vanishing Damping** called by [Hedy Attouch](#):

$$(AVD)_\alpha \quad \ddot{x}(t) + \frac{\alpha}{t} \dot{x}(t) + \nabla \Phi(x(t)) = 0. \tag{AVD}$$

where  $\Phi(x(t))$  is dependent on the objective function;  $\alpha > 0$  is constant in  $\mathbb{R}$ .

The fast minimization properties of the trajectories of the second-order evolution equation is also studied by Hedy's group in 2016:

$$\ddot{x}(t) + \frac{\alpha}{t} \dot{x}(t) + \nabla^2 \Phi(x(t)) \dot{x}(t) + \nabla \Phi(x(t)) = 0 \tag{HDD}$$

When it comes to numerical solution to differential equations, it is to find the solution of the equations  $x(t)$  so that the equations hold; in optimization, the optima is our goal so that the focus is limit order

$$\lim_{t \rightarrow t_0} x(t) = x^*$$

if possible where  $x^*$  optimizes the cost/objective function  $f(x)$  specially  $t_0 = \infty$ .



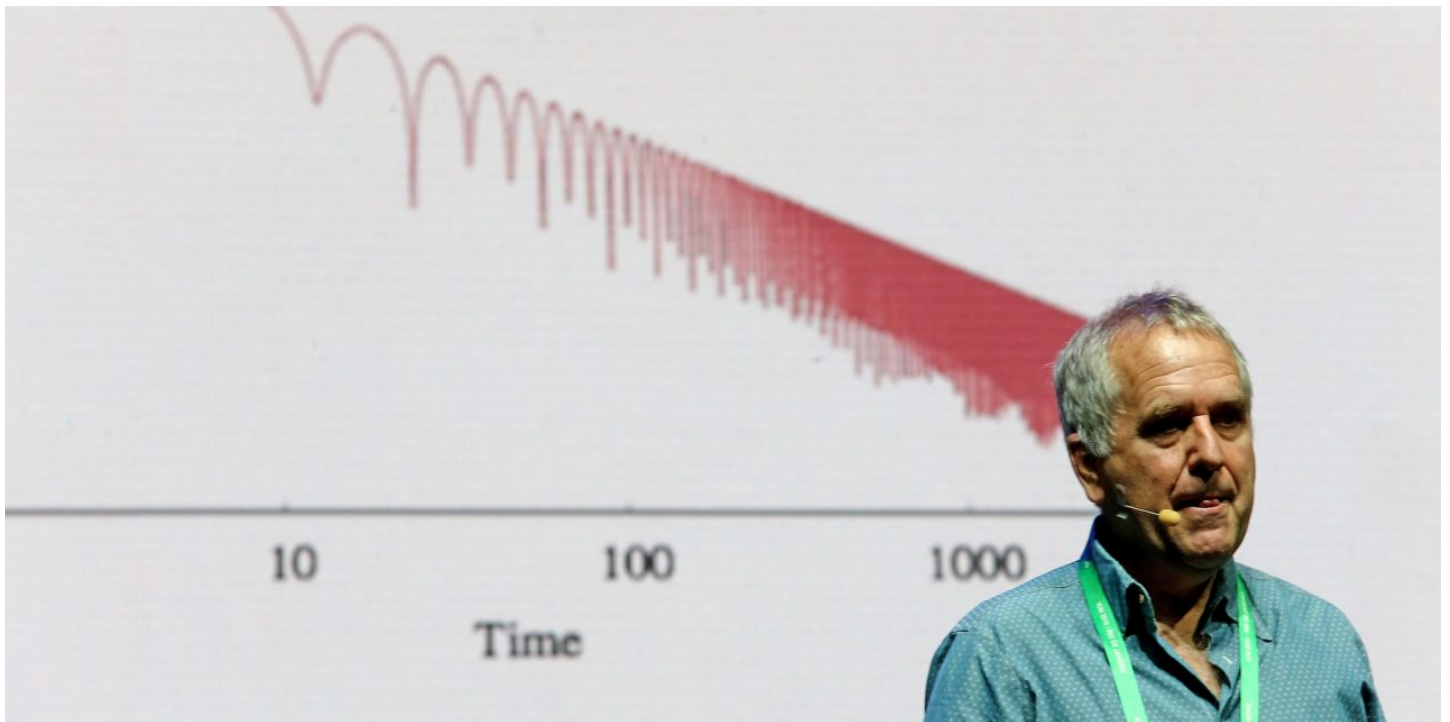
- <https://perso.math.univ-toulouse.fr/spot/resumes/>
- [Fast convex optimization via inertial dynamics with Hessian driven damping](#)
- [A proximal-Newton method for monotone inclusions in Hilbert spaces with complexity  \$O\(1/k^2\)\$](#)
- [https://www.researchgate.net/profile/Hedy\\_Attouch](https://www.researchgate.net/profile/Hedy_Attouch)
- <https://www.ljll.math.upmc.fr/~plc/sestri/>
- [Hedy Attouch at https://biography.omicsonline.org](#)
- [Rate of convergence of the Nesterov accelerated gradient method in the subcritical case  \$\alpha \leq 3\$](#)
- [A Dynamical Approach to an Inertial Forward-Backward Algorithm for Convex Minimization](#)
- [An Inertial Proximal Method for Maximal Monotone Operators via Discretization of a Nonlinear Oscillator with Damping](#)
- [THE HEAVY BALL WITH FRICTION METHOD, I. THE CONTINUOUS DYNAMICAL SYSTEM: GLOBAL EXPLORATION OF THE LOCAL MINIMA OF A REAL-VALUED FUNCTION BY ASYMPTOTIC ANALYSIS OF A DISSIPATIVE DYNAMICAL SYSTEM](#)
- [Viscosity Solutions of Minimization Problems](#)
- [Rate of convergence of the Nesterov accelerated gradient method in the subcritical case  \$\alpha \leq 3\$](#)
- [A dynamic approach to a proximal-Newton method for monotone inclusions in Hilbert spaces, with complexity  \$O\(\frac{1}{n^2}\)\$](#)
- [FAST CONVERGENCE OF INERTIAL DYNAMICS AND ALGORITHMS WITH ASYMPTOTIC VANISHING DAMPING](#)

---

It is difficult to generalize these methods to stochastic cases.

There is a wonderful summary [DYNAMICAL, SYMPLECTIC AND STOCHASTIC](#)

[PERSPECTIVES ON GRADIENT-BASED OPTIMIZATION](#) given by Micheal I Jordan at ICM 2018.



<http://www.icm2018.org/wp/2018/08/09/jordan-on-data-is-there-an-optimal-way-to-optimize/>

Some new connections between dynamical systems and optimization is found.

- Variational and Dynamical Perspectives On Learning and Optimization
- Continuous and Discrete Dynamics For Online Learning and Convex Optimization
- DYNAMICAL, SYMPLECTIC AND STOCHASTIC PERSPECTIVES ON GRADIENT-BASED OPTIMIZATIONs
- On Symplectic Optimization
- A variational perspective on accelerated methods in optimization
- A Dynamical Systems Perspective on Nesterov Acceleration
- The Physical systems Behind Optimization Algorithms
- <https://people.eecs.berkeley.edu/~jordan/optimization.html>

**Weijie J. Su** (joint with Bin Shi, Simon Du, and Michael Jordan) introduced a set of high-resolution differential equations to model, analyze, interpret, and design accelerated optimization methods.

 <http://stat.wharton.upenn.edu/~suw/WeijieSu.jpg> width = "30%" />

$$\ddot{x}(t) + 2\sqrt{\mu}\dot{x}(t) + \sqrt{s}\nabla^2 f(x)\dot{X}(t) + (1 + \sqrt{\mu s})\nabla f(X) = 0 \quad (\text{high-res ODE})$$

- A Differential Equation for Modeling Nesterov's Accelerated Gradient Method: Theory and Insights
- Acceleration via Symplectic Discretization of High-Resolution Differential Equations
- Understanding the Acceleration Phenomenon via High-Resolution Differential Equations
- Global Convergence of Langevin Dynamics Based Algorithms for Nonconvex Optimization

- Sampling as optimization in the space of measures: The Langevin dynamics as a composite optimization problem
  - Optimization and Dynamical Systems
  - Direct Runge-Kutta Discretization Achieves Acceleration
- 

- ESAIM: Control, Optimization and Calculus of Variations (ESAIM: COCV)
- MCT'03 Louisiana Conference on Mathematical Control Theory
- International Workshop “Variational Analysis and Applications”, ERICE, August 28 – September 5, 2018
- Games, Dynamics and Optimization, 2019
- System dynamics & optimization
- Introduction to Dynamical Systems by John K. Hunter, Department of Mathematics, University of California at Davis
- Special Focus on Bridging Continuous and Discrete Optimization

<https://eta.impa.br/dl/028.pdf>

## Stochastic Gradient Descent

### Clearing the Jungle of Stochastic Optimization

Stochastic gradient descent takes advantages of stochastic or estimated gradient to replace the true gradient in gradient descent.

It is **stochastic gradient** but may not be **descent**.

The name **stochastic gradient methods** may be more appropriate to call the methods with stochastic gradient.

It can date back to **stochastic approximation** in statistics.

It is aimed to solve the problem with finite sum optimization problem, i.e.

$$\arg \min_{\theta} \frac{1}{n} \sum_{i=1}^n f(\theta|x_i)$$

where  $n < \infty$  and  $\{f(\theta|x_i)\}_{i=1}^n$  are in the same function family and  $\{x_i\}_{i=1}^n \subset \mathbb{R}^d$  are constants while  $\theta \in \mathbb{R}^p$  is the variable vector.

The difficulty is  $p$ , that the dimension of  $\theta$ , is tremendous. In other words, the model is **overparameterized**. And the number  $n$  is far larger than  $p$  generally, i.e.  $n \gg p \gg d$ .

What is worse, the functions  $\{f(\theta|x_i)\}_{i=1}^n$  are not convex in most case.

---

The stochastic gradient method is defined as

$$\theta^{k+1} = \theta^k - \alpha_k \frac{1}{m} \sum_{j=1}^m \nabla f(\theta^k|x'_j)$$

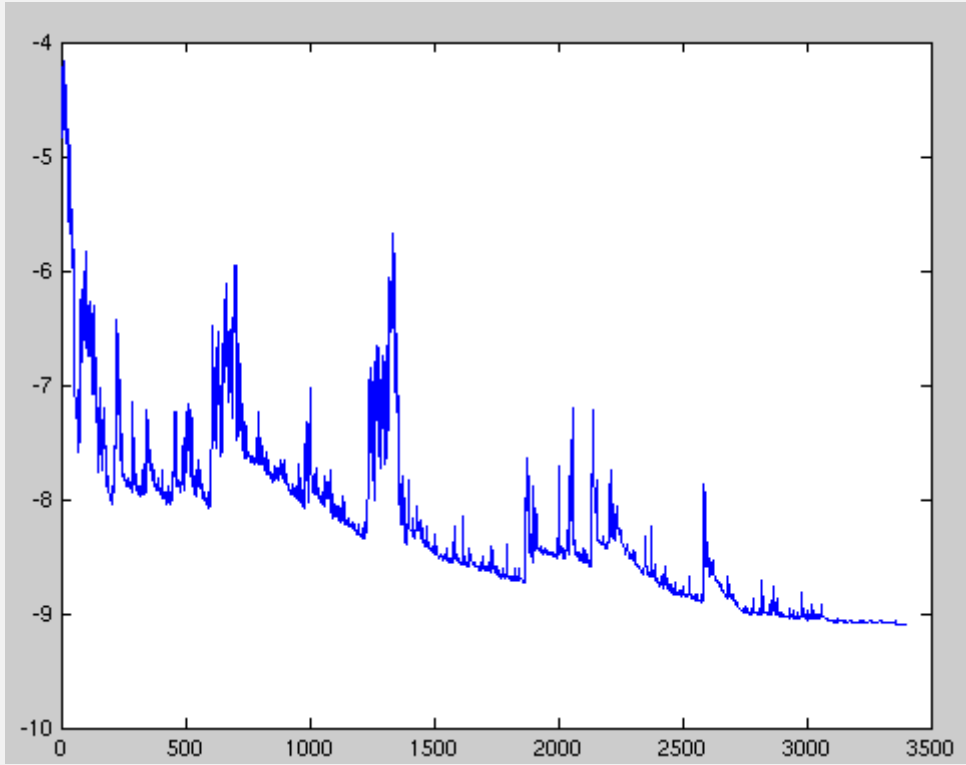
where  $x'_j$  is draw from  $\{x_i\}_{i=1}^n$  and  $m \ll n$  on random.

It is the fact  $m \ll n$  that makes it possible to compute the gradient of finite sum objective function and its side effect is that the objective function is not always descent.

There is fluctuations in the total objective function as gradient steps with respect to mini-batches are taken.

---

The fluctuations in the objective function as gradient steps with respect to mini-batches are taken



An heuristic proposal for avoiding the choice and for modifying the learning rate while the learning task runs is the **bold driver (BD) method**[<sup>14</sup>].

The learning rate increases *exponentially* if successive steps reduce the objective function  $f$ , and decreases rapidly if an “accident” is encountered (if objective function  $f$  increases), until a suitable value is found.

After starting with a small learning rate, its modifications are described by the following equation:

$$\alpha_{k+1} = \begin{cases} \rho \alpha_k, & f(\theta^{k+1}) < f(\theta^k); \\ \eta^n \alpha_k, & f(\theta^{k+1}) > f(\theta^k) \text{ using } \alpha_k, \end{cases}$$

where  $\rho$  is close to 1 such as  $\rho = 1.1$  in order to avoid frequent “accidents” because the objective function computation is wasted in these cases,  $\eta$  is chosen to provide a rapid reduction ( $\eta = 0.5$ ), and  $n$  is the minimum integer such that the reduced rate  $\eta^n$  succeeds in diminishing the objective function.[<sup>13</sup>]

---

The fact that the sample size is far larger than the dimension of parameter,  $n \gg p$ , that makes it expensive to compute total objective function  $f(\theta) = \sum_{i=1}^n f(\theta|x_i)$ .

Thus it is not clever to determine the learning rate  $\alpha_k$  by line search.

And most stochastic gradient methods are to find proper step length  $\alpha_k$  to make it converge at least in convex optimization. The variants of gradient descent such as momentum methods or mirror gradient methods have their stochastic counterparts.

- It is simplest to set the step length a constant, such as  $\alpha_k = 3 \times 10^{-3} \forall k$ .
- There are decay schemes, i.e. the step length  $\alpha_k$  diminishes such as  $\alpha_k = \frac{\alpha}{k}$ , where  $\alpha$  is constant.
- And another strategy is to tune the step length adaptively such as *AdaGrad*, *ADAM*.

**PS:** the step length  $\alpha_k$  is called **learning rate** in machine learning. Additionally, stochastic gradient descent is also named as **increment gradient methods** in some case.

We can see some examples to see the advantages of incremental method such as the estimation of mean.

Given  $x_1, x_2, \dots, x_n$  the mean is estimated as  $\bar{x} = \frac{\sum_{i=1}^n x_i}{n}$ . If now we observed more data  $y_1, y_2, \dots, y_m$  from the population, the mean could be estimated by  $\frac{n\bar{x}}{m+n} + \frac{\sum_{j=1}^m y_j}{m+n} = \frac{n\bar{x} + \sum_{j=1}^m y_j}{m+n}$ . It is not necessary to summarize  $x_1, \dots, x_n$ .

Another example, it is **Newton interpolation formula** in numerical analysis. The task is to fit the function via polynomials given some point in the function  $(x_i, f(x_i)), i = 1, 2, \dots, n$ .

The Newton form of the interpolating polynomial is given by

\$\$

$$P_n(x) = a_0 + a_1(x-x_1) + a_2(x-x_1)(x-x_2) + \dots$$

- $a_n(x-x_1)(x-x_2)(x-x_3)\dots(x-x_n)$ .

\$\$

This form is incremental and if another points  $(x_{n+1}, f(x_{n+1}))$  is observed we will fit the function  $f$  more precisely just by adding another term  $a_{n+1}(x-x_1)(x-x_2)\dots(x-x_n)(x-x_{n+1})$

where the coefficients  $a_0, a_1, \dots, a_n$  are determined by  $f(x_1), f(x_2), \dots, f(x_n)$ .

See the following links for more information on *stochastic gradient descent*.

- [https://www.wikiwand.com/en/Stochastic\\_gradient\\_descent](https://www.wikiwand.com/en/Stochastic_gradient_descent)
- <https://www.bonaccorso.eu/2017/10/03/a-brief-and-comprehensive-guide-to-stochastic-gradient-descent-algorithms/>
- <https://leon.bottou.org/projects/sgd>
- <https://leon.bottou.org/research/stochastic>
- <https://leon.bottou.org/papers/bottou-bousquet-2008>
- <http://runder.io/optimizing-gradient-descent/>
- <https://zhuanlan.zhihu.com/p/22252270>
- <https://henripal.github.io/blog/stochasticdynamics>
- <https://henripal.github.io/blog/langevin>
- [http://fa.bianp.net/teaching/2018/eecs227at/stochastic\\_gradient.html](http://fa.bianp.net/teaching/2018/eecs227at/stochastic_gradient.html)
- [VR-SGD](#)

## The Differences of Gradient Descent and Stochastic Gradient Descent



# ADAM

- <https://arxiv.org/abs/1412.6980>
- <http://ruder.io/deep-learning-optimization-2017/>

## The Differences of Stochastic Gradient Descent and its Variants



- <http://blavatnikawards.org/honorees/profile/leon-bottou/>



*Leon Bottou*

*PS:* [Zeyuan Allen-Zhu](#) and others published much work on acceleration of stochastic gradient descent.

- <https://arxiv.org/pdf/1811.03962.pdf>
- <http://www.arxiv-sanity.com/search?q=Zeyuan+Allen-Zhu>

## Distributed Optimization Methods

Beginning around ten years ago, the single-threaded CPU performance stopped improving significantly, due to physical limitations; it is the numbers of cores in each machine that continue to arise. Today we can buy 8-core phones, 64-core workstations, and 2.5k-core GPUs at affordable prices. On the other hand, most of our algorithms are still single-threaded, and because so, their running time is about the same as it was to ten years ago and will stay so in the future, say, ten or twenty years. To develop faster algorithms, especially for those large-scale problems that arise in signal processing, medical imaging, and machine learning, it is inevitable to consider parallel computing.

Machine learning especially deep learning requires more powerful **distributed and decentralized** optimization methods.

Large scale supervised machine learning methods, which are based on gradient to improve their performance, need online near-real-time feedback from massive users.

- ☐ [Math 285J: A First Course on Large-Scale Optimization Methods](#)
- ☐ <http://bicmr.pku.edu.cn/~wenzw/bigdata2019.html>
- ☐ [Data Science meet optimization](#)



- ☐ Distributed optimization for control and learning
- ☐ ME 555: Distributed Optimization
- ☐ <https://www.euro-online.org/websites/dso/>
- ☐ <https://labs.criteo.com/2014/09/poh-part-3-distributed-optimization/>
- ☐ Convex and Distributed Optimization
- ☐ (749g) Accelerated Parallel Alternating Method of Multipliers (ADMM) for Distributed Optimization
- ☐ 8th IEEE Workshop Parallel / Distributed Computing and Optimization (PDCO 2018)
- ☐ Distributed Optimization and Control
- ☐ ADOPT: Asynchronous Distributed Constraint Optimization with Quality Guarantees

On Distributed Optimization in  
Networked Systems

Distributed Optimization and Control using Operator Splitting Methods

Foundations of Distributed and Large Scale Computing Optimization

Distributed Optimization of Large-Scale Complex Networks

<http://principlesofoptimaldesign.org/>

Walkman: A Communication-Efficient Random-Walk Algorithm for Decentralized Optimization

Proportional-Integral Distributed Optimization

<http://ecee.colorado.edu/marden/files/dist-opt-journal.pdf>

<http://is4.tecnico.ulisboa.pt/~is4.daemon/tasks/distributed-optimization/>

<http://shivaram.org/publications/hemingway-mlsys-2016.pdf>



Parallelizing Stochastic Gradient Descent

- Hogwild: A Lock-Free Approach to Parallelizing Stochastic Gradient Descent
- Asynchronous Decentralized Parallel Stochastic Gradient Descent
- <https://arxiv.org/abs/1905.05920v1>
- [https://ray.readthedocs.io/en/latest/distributed\\_sgd.html](https://ray.readthedocs.io/en/latest/distributed_sgd.html)
- <https://www.podc.org/data/podc2018/podc2018-tutorial-alistarh.pdf>

Stochastic ADMM

Linearly constrained stochastic convex optimization is given by

$$\begin{aligned} \min_{x,y} \mathbb{E}_{\vartheta}[F(x, \vartheta)] + h(y), \\ s. t. Ax + By = b, x \in \mathbb{X}, y \in \mathbb{Y}. \end{aligned}$$

where typically the expectation  $\mathbb{E}_{\vartheta}[F(x, \vartheta)]$  is some loss function and  $h$  is regularizer to prevent from over-fitting.

The first problem is that the distribution of  $\vartheta$  is unknown as well as the expectation  $\mathbb{E}_{\vartheta}[F(x, \vartheta)]$  in the objective function.

**Modified Augmented Lagrangian**

Linearize  $f(x)$  at  $x_k$  and add a proximal term :

$$L_{\beta}^k(x, y, \lambda) := f(x_k) + \langle x_k, g_k \rangle + h(y) - \langle \lambda, Ax + By - b \rangle + \frac{\beta}{2} \|Ax + By - b\|_2^2 + \frac{1}{2\eta_k} \|x - x_k\|^2$$

where  $g_k$  is a stochastic (sub)gradient of  $f$ .

1.  $x^{k+1} = \arg \min_{x \in \mathbf{X}} L_{\beta}^k(x, y^k, \lambda^k);$
2.  $y^{k+1} = \arg \min_{y \in \mathbf{Y}} L_{\beta}^k(x^{k+1}, y, \lambda^k);$
3.  $\lambda^{k+1} = \lambda^k - \beta(Ax^{k+1} + By^{k+1} - b).$

- Stochastic ADMM
- Accelerated Variance Reduced Stochastic ADMM
- Towards optimal stochastic ADMM or the talk in ICML
- V-Cycle or Double Sweep ADMM
- <https://arxiv.org/abs/1903.01786>

Distributed ADMM

Let us assume that the function  $F$  has the following decomposition where each  $x_i$  has its own dimension.

$$\min_x F(x) \{ = \sum_{i=1}^n f_i(x_i) \},$$

$$s. t. \quad x \in \mathbf{C}$$

We can reformulate the problem to get

$$\min_x F(x) + g(y),$$

$$s. t. \quad x = y$$

where  $g(z)$  is the indicator function of set  $\mathbf{C}$ .

We can define an augmented Lagrangian

$$L_{\beta}(x, y, \lambda) = F(x) + g(y) - \lambda^T(x - y) + \frac{\beta}{2} \|x - y\|_2^2$$

$$= \sum_{i=1}^n [f_i(x_i) - \lambda_i(x_i - y_i) + \frac{\beta}{2} (x_i - y_i)^2] + g(y)$$

$$= \sum_{i=1}^n L_{(\beta, i)}(x_i, y, \lambda_i).$$

We can split the optimization over  $x_i$ :

1.

$$x_i^{k+1} = \arg \min_{x_i} L_{(\beta,i)}(x_i, y^k, \lambda_i^k) \quad i = 1, 2, \dots, n;$$

2.

$$y^{k+1} = \arg \min_{x_i} L_{(\beta,i)}(x_i^{k+1}, y, \lambda_i^k);$$

3.

$$\lambda^{k+1} = \lambda^k + \lambda(x^{k+1} - y^{k+1}).$$

Then optimization over  $x_i$  is done in parallel. Subsequently, the results are communicated back to a master node which performs the  $y$  update (usually just a projection as in the example above) and returns back the result to other worker nodes. The update over  $\lambda_i$  is again done independently.

- <http://users.isr.ist.utl.pt/~jmota/DADMM/>
- <http://repository.ust.hk/ir/Record/1783.1-66353>
- <https://ieeexplore.ieee.org/document/7039496>
- [http://www.iutzeler.org/pres/sem\\_louvain.pdf](http://www.iutzeler.org/pres/sem_louvain.pdf)
- Distributed Optimization and Statistical Learning via the Alternating Direction Method of Multipliers by S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein
- Asynchronous Distributed ADMM for Consensus Optimization
- Notes on Distributed ADMM

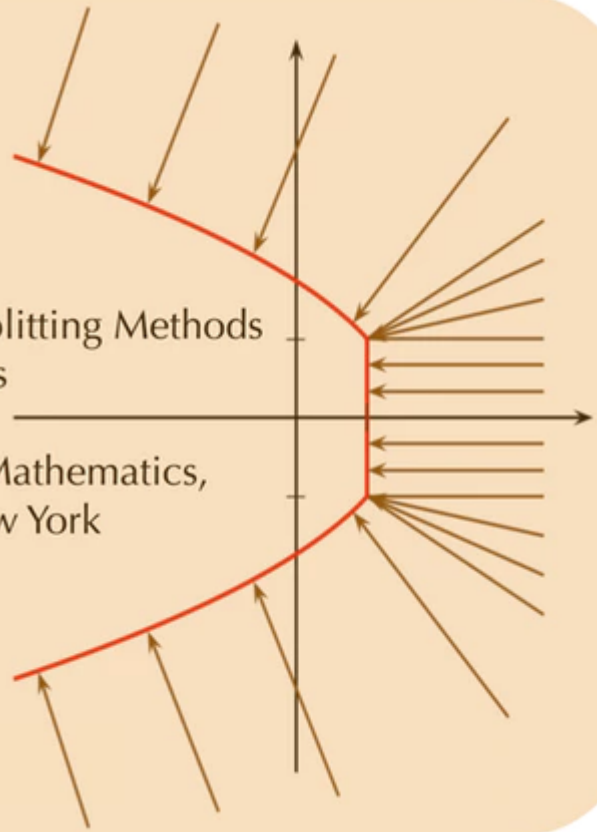
---

## Monotone Operator Splitting Methods for Optimization

Operator splitting is to decompose one complicated operator(procedure) into some simple operators (procedures).

## 2nd Workshop on Operator Splitting Methods in Data Analysis

Center for Computational Mathematics,  
Flatiron Institute, New York



- [https://web.stanford.edu/class/ee364b/lectures/monotone\\_split\\_slides.pdf](https://web.stanford.edu/class/ee364b/lectures/monotone_split_slides.pdf)
- Operator Splitting by Professor Udell @ORIE 6326: Convex Optimization
- A note on the equivalence of operator splitting methods
- Splitting methods for monotone operators with applications to parallel optimization
- <http://www.syscop.de/files/2015ss/numopt/splitting.pdf>
- <https://staffportal.curtin.edu.au/staff/profile/view/Jie.Sun/>
- Operator Splitting Methods in Data Analysis

## Gradient Free Optimization Methods

As shown in **Principle of Optimal Design**, **nongradient methods** are classified into 3 classes:

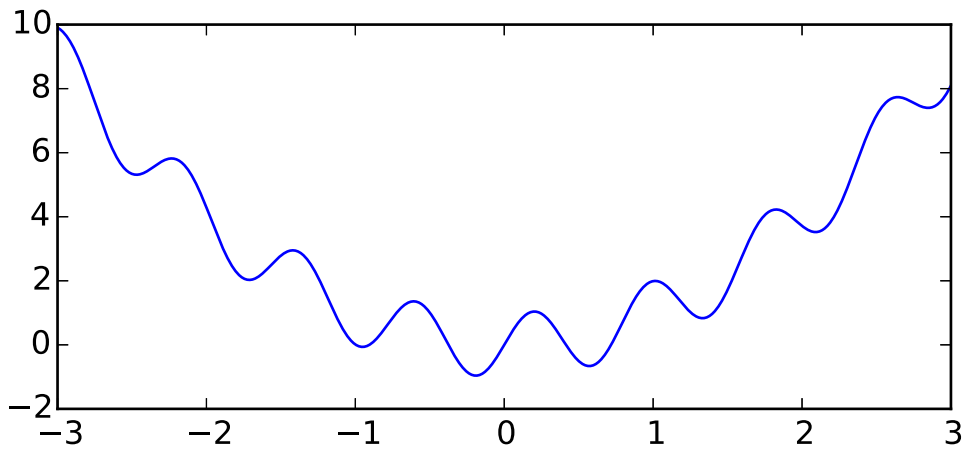
We organize the discussion of nongradient methods in three parts, direct search methods, heuristic methods, and black-box methods. Direct search methods rely on ranking the objective function values rather than using the objective values themselves. Heuristic methods use some random process to generate new candidate solutions, and so the iteration paths and even the solutions obtained can change each time we perform a search. Black-box methods deal with problems that have no known model function structure that we can exploit. For example, functions generated by simulations have no discernible mathematical properties (like convexity), and so we refer to them as black-box functions. In this sense, all nongradient methods can be used for black-box problems. The two black-box methods described in this chapter were created to address design problems with expensive simulations, and so their main goal is to find an optimum quickly with few function evaluations.

**Heuristic methods** will be introduced in computational intelligence.

Let us start with the example and suppose that we want to

$$\arg \min_x f(x) = x^2 + 4 \sin(2x). \quad (1)$$

The objective function  $f(x)$ , a non-convex function, has many local minimizer or extrema. The function (1) is upper bounded by  $x^2 + 4$  and lower bounded by  $x^2 - 4$ .



Another insightful example is to minimize the following cost function:

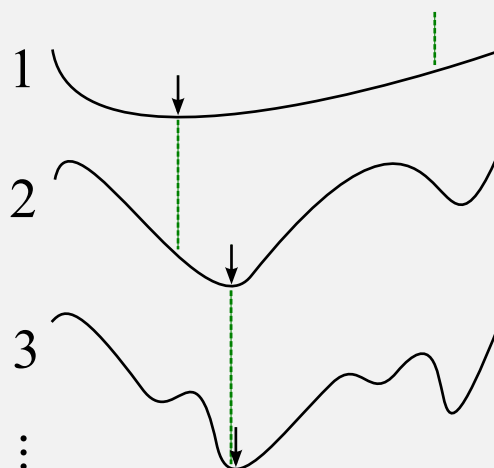
$$x^2 + 4 \sin(2x) - 1001 \mathbb{I}_{\sqrt{2}}(x) \quad (2)$$

where the last part  $\mathbb{I}_{\sqrt{2}}(x)$  is equal to 1 when  $x = \sqrt{2}$  and 0 otherwise, a Dirac function. It is almost equal to the first function (1) except at the point  $x = \sqrt{2}$ . The minimal value of the above function is  $2 + 4 \sin(2\sqrt{2}) - 1001$  when  $x = \sqrt{2}$ . And these two functions are two different kinds of non-convex functions.

[Optimization and Assumptions @ Freemind](#) | [Test functions for optimization](#)

Another related method is **graduated optimization**, which is a global optimization technique that attempts to solve a difficult optimization problem by initially solving a greatly simplified problem, and progressively transforming that problem (while optimizing) until it is equivalent to the difficult optimization problem.

### Graduated Optimization



**Multi-Level Optimization** is to optimize a related cheap function  $\hat{f}$  when the objective function  $f$  is very expensive to evaluate.

- [Multilevel Optimization: Convergence Theory, Algorithms and Application to Derivative-Free Optimization](#)
- [multilevel optimization iosotech](#)
- [OptCom: A Multi-Level Optimization Framework for the Metabolic Modeling and Analysis of Microbial Communities](#)
- [Laplacian Smoothing Gradient Descent](#)
- [Entropy SGD](#)
- [Deep Relaxation: partial differential equations for optimizing deep neural networks](#)
- [Deep Relaxation tutorials](#)
- [CS 369S: Hierarchies of Integer Programming Relaxations](#)
- [The Multiplicative Weights Update framework](#)
- <https://pratikac.github.io/>
- [https://www.wikiwand.com/en/Numerical\\_continuation](https://www.wikiwand.com/en/Numerical_continuation)
- [Convex Relaxations and Integrality Gaps](#)
- [LP/SDP Hierarchies Reading Group](#)
- [Proofs, beliefs, and algorithms through the lens of sum-of-squares](#)
- [Iterative Convex Optimization Algorithms; Part Two: Without the Baillon–Haddad Theorem](#)
- [Sequential quadratic programming](#)
  
- [Zeroth-Order Method for Distributed Optimization With Approximate Projections](#)
- [Derivative-Free Optimization \(DFO\)](#)
- [Derivative Free Optimization / Optimisation sans Gradient](#)
- [Delaunay-based derivative-free optimization via global surrogates, part I: linear constraints](#)
- [Delaunay-based derivative-free optimization via global surrogates, part II: convex constraints](#)
- <https://projects.coin-or.org/Dfo>
- [https://nlopt.readthedocs.io/en/latest/NLOpt\\_Algorithms/](https://nlopt.readthedocs.io/en/latest/NLOpt_Algorithms/)
- [http://adl.stanford.edu/aa222/Lecture\\_Notes\\_files/chapter6\\_gradfree.pdf](http://adl.stanford.edu/aa222/Lecture_Notes_files/chapter6_gradfree.pdf)
- <https://code.fb.com/ai-research/nevergrad/>
- <https://github.com/facebookresearch/nevergrad>
- <https://www.kthohr.com/optimlib.html>
- <https://www.infoq.com/presentations/black-box-optimization>

never grad

*never grad*

---

And there are more topics on optimization such as [this site](#).

And more courses on optimization:

- <http://bicmr.pku.edu.cn/~wenzw/opt-2018-fall.html>
- <http://math.sjtu.edu.cn/faculty/xqzhang/html/teaching.html>
- <http://mat.uab.cat/~alseda/MasterOpt/>

- <http://www.math.ucla.edu/~wotaoyin/summer2016/>
- <http://www.math.ucla.edu/~wotaoyin/math273c/>
- <http://www.math.ucla.edu/~lvese/273.1.10f/>
- <http://www.seas.ucla.edu/~vandenbe/ee236b/ee236b.html>
- <https://people.eecs.berkeley.edu/~elghaoui/Teaching/EECS127/index.html>
- [https://lavaxi.ieor.berkeley.edu/Course\\_IEOR262B\\_Spring\\_2019.html](https://lavaxi.ieor.berkeley.edu/Course_IEOR262B_Spring_2019.html)
- <https://web.stanford.edu/~boyd/teaching.html>
- <http://web.stanford.edu/class/ee364b/lectures.html>
- <http://www.stat.cmu.edu/~ryantibs/convexopt/>
- <https://nisheethvishnoi.wordpress.com/convex-optimization/>
- Optimization by Vector Space Methods
- Algorithms for Convex Optimization

- 
- ☐ <http://www.optimization-online.org/>
  - ☐ <http://convexoptimization.com/>
  - ☐ More [Optimization Online Links](#)
  - ☐ **TUTORIALS AND BOOKS** at <http://plato.asu.edu/sub/tutorials.html>.
  - ☐ Provable Nonconvex Methods/Algorithms
  - ☐ [https://optimization.mccormick.northwestern.edu/index.php/Main\\_Page](https://optimization.mccormick.northwestern.edu/index.php/Main_Page)
  - ☐ <https://arxiv.org/pdf/1712.07897.pdf>
  - ☐ <https://arxiv.org/pdf/1707.02444.pdf>
  - ☐ <http://www.vision.jhu.edu/assets/HaeffeleCVPR17.pdf>
  - ☐ <https://www.springer.com/us/book/9783319314822>
  - ☐ <https://core.ac.uk/display/83849901>
  - ☐ <https://core.ac.uk/display/73408878>
  - ☐ <https://zhuanlan.zhihu.com/p/51514687>
  - ☐ <http://math.cmu.edu/~hschaeff/research.html>
  - ☐ <https://people.eecs.berkeley.edu/~elghaoui/Teaching/EECS127/index.html>
  - ☐ <https://blogs.princeton.edu/imabandit/>
  - ☐ <http://www.probablistic-numerics.org/research/index.html>