Numerical Optimization

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.

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

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://web.stanford.edu/~boyd/cvxbook/bv_cvxbook.pdf
- http://www.cs.cmu.edu/~pradeepr/convexopt/

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

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

where the positive scalar α^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 α_k.

$Note$: we use the notation x^k and α_k to represent the kth iteration of the vector variables x and kth 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 ∇f_k is the gradient of objective function f at the kth 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.

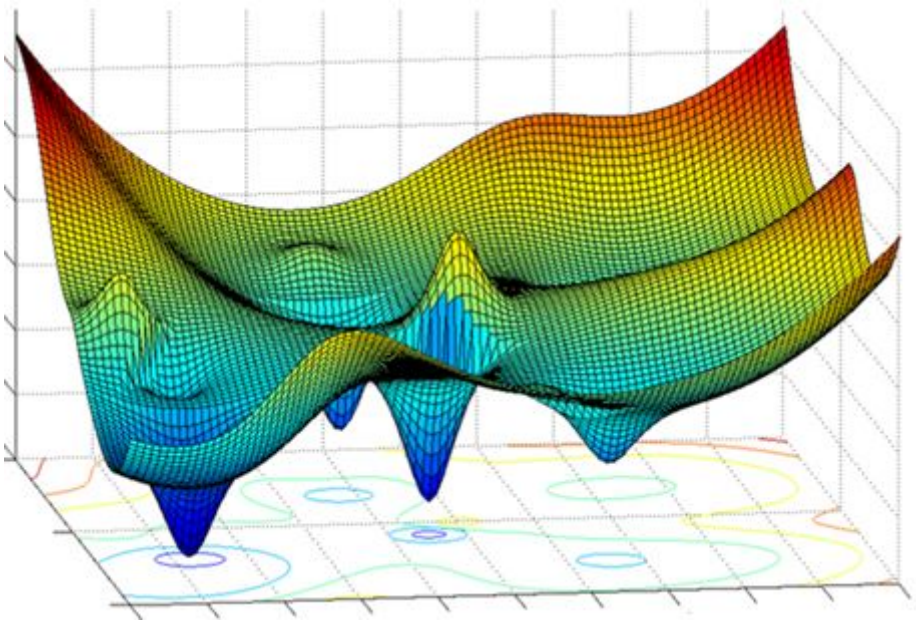
Let $x^{k+1} = \arg\min_x f(x^k) + (x - x^k)^T \nabla f(x^k) + \frac{s_k}{2} \|x - x^k\|_2^2$, we will obtain $x^{k+1} = x^k - \frac{1}{s_k} \nabla_x f(x^k)$. However, the constant s_k is difficult to estimate.

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 kth 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



There are many ways to choose some proper step pr 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 method**:

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

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

Nesterov accelerated gradient method at the kth step is given by:

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

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

}

- https://www.wikiwand.com/en/Gradient_descent
- http://wiki.fast.ai/index.php/Gradient_Descent
- https://blogs.princeton.edu/imabandit/2013/04/01/acceleratedgradientdescent/
- https://blogs.princeton.edu/imabandit/2015/06/30/revisiting-nesterovs-acceleration/
- http://awibisono.github.io/2016/06/20/accelerated-gradient-descent.html
- https://jlmelville.github.io/mize/nesterov.html
- http://awibisono.github.io/2016/06/13/gradient-flow-gradient-descent.html
- http://stat.wharton.upenn.edu/~suw/paper/Nesterov_ODE.pdf
- http://stat.wharton.upenn.edu/~suw/paper/symplectic_discretization.pdf
- http://stat.wharton.upenn.edu/~suw/paper/highODE.pdf
- https://zhuanlan.zhihu.com/p/41263068
- https://zhuanlan.zhihu.com/p/35692553
- https://zhuanlan.zhihu.com/p/35323828

- http://www.optimization-online.org/DB_FILE/2018/11/6938.pdf
- https://www.mat.univie.ac.at/~neum/glopt/mss/MasAi02.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/

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

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

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

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 θ.
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**:

$$J(\theta) = \mathbb{E}(-1\frac{\partial^2 \ell(\theta)}{\partial\theta\partial\theta^T}) = -\int \frac{\partial^2 \ell(\theta)}{\partial\theta\partial\theta^T} f(x_1, \cdots, x_n|\theta)\mathrm{d}x_1 \cdots \mathrm{d}x_n$$

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

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 θ^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 superlinear 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.[^11]

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 α 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 α_k is chosen to satisfy the Wolfe conditions, then B_{k+1} is also positive-definite.

For example,

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

BFGS

- Quasi-Newton methods in Wikipedia page
- http://59.80.44.98/www.seas.ucla.edu/~vandenbe/236C/lectures/qnewton.pdf
- 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 α_k such that

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

which yields

$$\alpha_k = \frac{\langle s_{k-1}, y_{k-1}\rangle}{\langle y_{k-1}, y_{k-1}\rangle} \tag{1}$$

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 α_k and get

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

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

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

where α_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
- http://www.math.ucla.edu/~wotaoyin/math273a/slides/
- 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

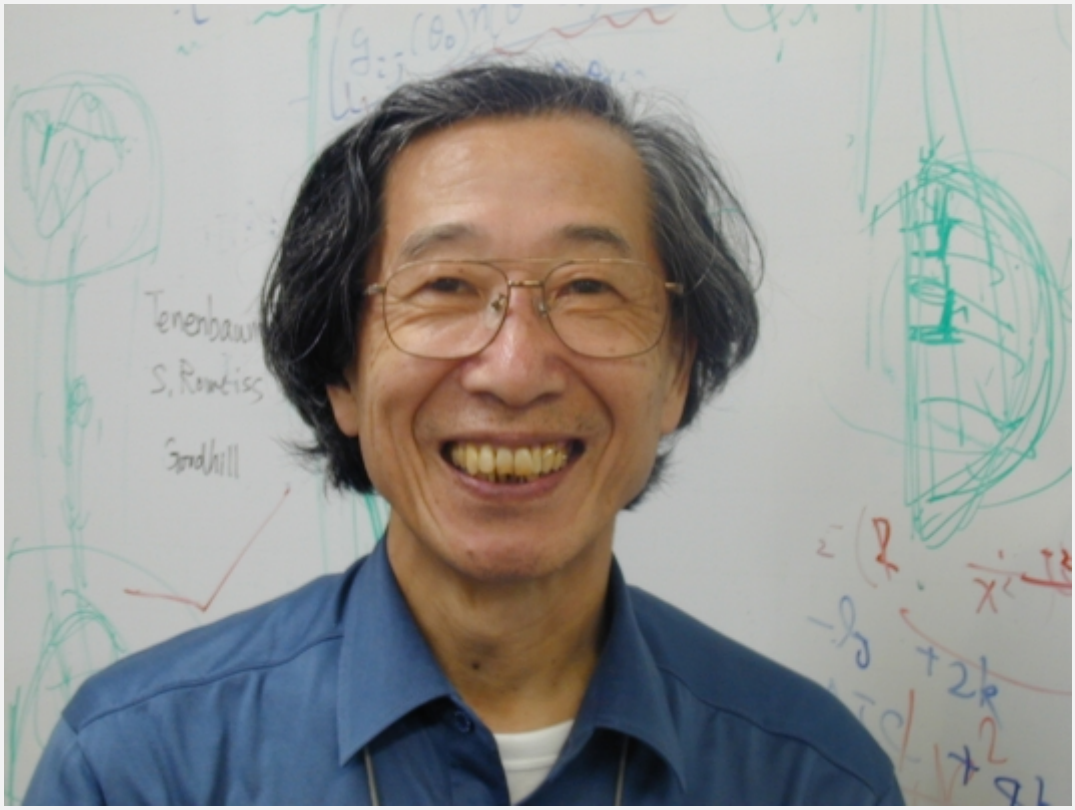
Natural gradient descent is to solve the optimization problem $\min_\theta L(\theta)$ by

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

where $F^{-1}(\theta^{(t)})$ is the inverse of `Remiann 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

Originator of Information Geometry



- http://www.yann-ollivier.org/rech/publs/natkal.pdf
- http://www.dianacai.com/blog/2018/02/16/natural-gradients-mirror-descent/
- https://www.zhihu.com/question/266846405
- http://bicmr.pku.edu.cn/~dongbin/Conferences/Mini-Course-IG/index.html
- http://ipvs.informatik.uni-stuttgart.de/mlr/wp-content/uploads/2015/01/mathematics_for_intelligent_systems_lecture12_notes_I.pdf
- http://www.luigimalago.it/tutorials/algebraicstatistics2015tutorial.pdf
- http://www.yann-ollivier.org/rech/publs/tango.pdf
- http://www.brain.riken.jp/asset/img/researchers/cv/s_amari.pdf
- https://people.cs.umass.edu/~pthomas/papers/Thomas2016b_ICML.pdf
- https://www.depthfirstlearning.com/assets/k-fac-tutorial.pdf
- http://www.deeplearningpatterns.com/doku.php?id=natural_gradient_descent

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

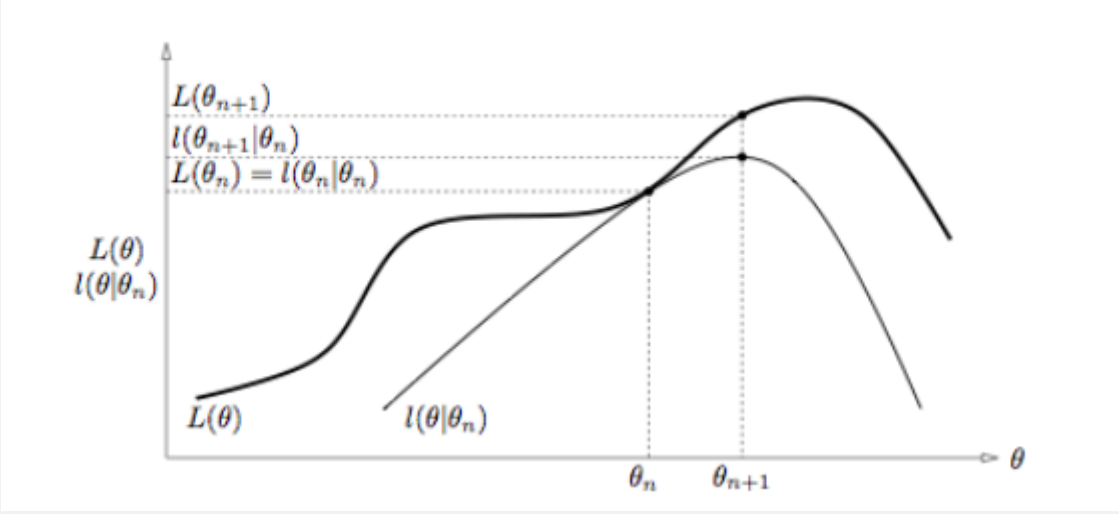
$$\begin{aligned} Q(\theta|\theta^{(t)}) &= \mathbb{E}(\ell(\theta|Y_{obs}, Z)|Y_{obs}, \theta^{(t)}) \\ &= \int_Z \ell(\theta|Y_{obs}, Z) \times f(z|Y_{obs}, \theta^{(t)}) \mathrm{d}z, \end{aligned}$$

and the M-step is to maximize **Q** with respect to θ to obtain

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

- https://www.wikiwand.com/en/Expectation–maximization_algorithm
- http://cs229.stanford.edu/notes/cs229-notes8.pdf
- https://www2.stat.duke.edu/courses/Spring06/sta376/Support/EM/EM.Mixtures.Figueiredo.2004.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

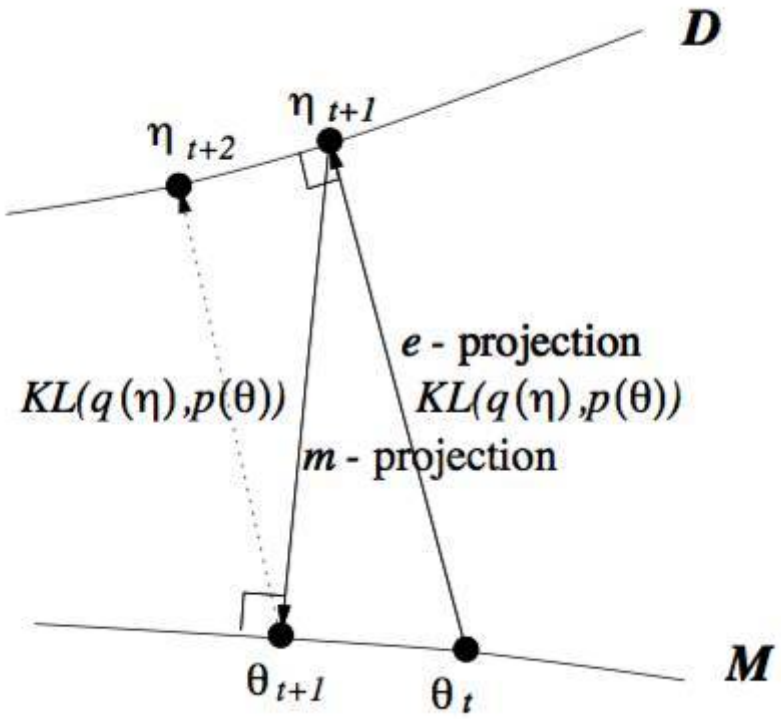
$$\begin{aligned} Q(\theta|\theta^{(t)}) &= \mathbb{E}[\ell(\theta|Y_{obs}, Z)|Y_{obs}, \theta^{(t)}] \\ &= \int_Z \ell(\theta|Y_{obs}, Z) \times f(z|Y_{obs}, \theta^{(t)}) \mathrm{d}z, \end{aligned}$$

and the another step is to find θ 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://www.stat.berkeley.edu/~aldous/Colloq/lange-talk.pdf

Qudratic Loweer 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 $f, \mathbb{S} \subset \mathbb{R}^n$ are convex.
The optimal condition for this constrained optimization problem is that the feasible direction is not the descent or profitable direction: if $x^\star \in \mathbb{S}$ is the solution to the problem, we can assert that `variational inequality` holds:

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

And it is the optimal condition of constrained optimization problem.

Projected Gradient Descent

Projected gradient descent has two steps:

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

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 indictor function of the set $\mathbb{S}$

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

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

Mirror descent

Mirror descent can be regarded as the non-Euclidean generalization via replacing the ℓ_2 norm or Euclidean distance in projected gradient descent by Bregman divergence.

Bregman divergence is induced by convex smooth function f:

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

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

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 Euclidean distance.

A wonderful introduction to **Bregman divergence** is **Meet the Bregman Divergences** by Mark Reid 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

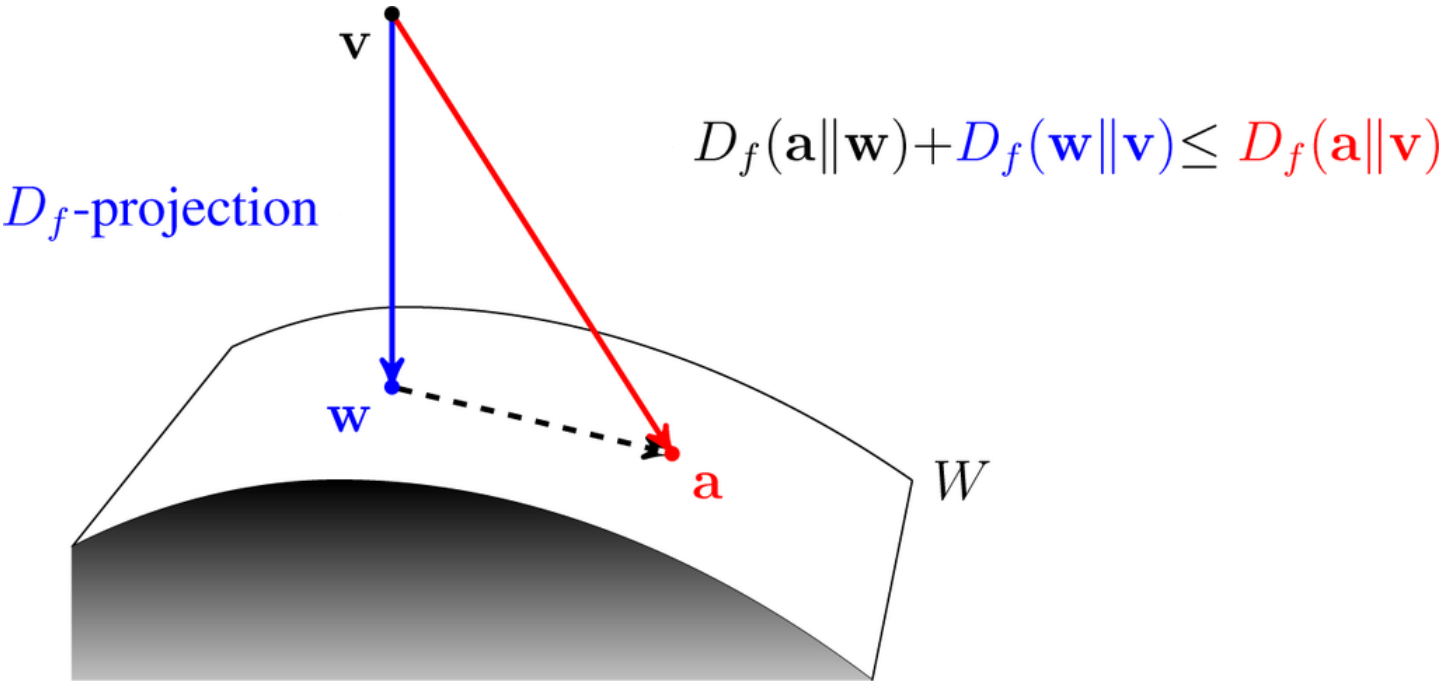
$$y' = \arg\min_{x \in C} B(x, y)$$

is unique.

A `generalised 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.



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:

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

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

The original "mirror" form of mirror gradient method is described as

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

where the convex function h induces the Bregman divergence.

One special method is called `entropic mirror descent(multiplicative weight)` 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 mirror descent(multiplicative weight) at step k is given as follows:

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

See more on the following link list.

- http://users.cecs.anu.edu.au/~xzhang/teaching/bregman.pdf
- 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://www.stat.berkeley.edu/~bartlett/courses/2014fall-cs294stat260/lectures/mirror-descent-notes.pdf
- http://www.princeton.edu/~yc5/ele538_optimization/lectures/mirror_descent.pdf
- http://users.cecs.anu.edu.au/~xzhang/teaching/bregman.pdf
- https://tlienart.github.io/pub/csml/cvxopt/mda.html
- https://web.stanford.edu/class/cs229t/2017/Lectures/mirror-descent.pdf

Proximal Gradient Method

The `proximal mapping (or prox-operator)` of a convex function $\mathbf{h}$ is defined as

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

Unconstrained problem with cost function split in two components:

$$minimize \qquad f(x) = g(x) + h(x)$$

- g is convex, differentiable;
- h is closed, convex, possibly non-differentiable while $prox_h(x)$ is inexpensive.

Proximal gradient algorithm

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

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

$$x^+ = prox_{th}(x - t\nabla g(x))$$

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

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

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:

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

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://web.stanford.edu/~boyd/papers/pdf/prox_algs.pdf
- https://arxiv.org/abs/1808.03045
- https://arxiv.org/abs/1812.10198

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

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

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 δ_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.

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

Proximal and Projected Newton Methods

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

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 constrain 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:

$$\arg\min_x f(x)$$
$$s.t.\ Ax = b$$

Lagrange Multipliers and Generalized Lagrange Function

The penalty function methods do not take the optimal conditions into consideration although it works.
If $x^\star$ is in the solution set of the optimization problem above, it is obvious that $Ax^\star = b$ and $L(x^\star, \lambda) = f(x^\star)$ where

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

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

$$L(x^\star, \lambda^\star) = \quad \min_x[\max_\lambda L(x, \lambda)].$$

By the definition,
$L(x^\star, \lambda^\star) \geq L(x^\star, \lambda) = f(x^\star) + \lambda^T(Ax^\star - b)$, which implies that $\lambda^T(Ax^\star - b) = 0$ i.e., $Ax^\star = b$. And
$L(x^\star, \lambda^\star) = f(x^\star) \leq L(x, \lambda^\star) \forall x$ if $Ax = b$
thus $x^\star$ 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

$$\min_x L(x, \lambda) \leq L(x, \lambda) \leq \max_\lambda L(x, \lambda) \implies$$
$$\max_\lambda[\min_x L(x, \lambda)] \leq \max_\lambda L(x, \lambda) \leq \min_x[\max_\lambda L(x, \lambda)].$$

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

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

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.

- http://www.ece.ust.hk/~palomar/ELEC5470_lectures/07/slides_Lagrange_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
- http://web.mit.edu/dimitrib/www/lagr_mult.html
- https://zhuanlan.zhihu.com/p/50823110

Alternating Direction Method of Multipliers

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

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

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

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

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

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

- http://www.optimization-online.org/DB_FILE/2015/05/4925.pdf

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

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_ϕ is the Bregman divergence induced by the convex function ϕ.

BADMM

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

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

Multi-Block ADMM

Firstly we consider the following optimization problem

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

We defined its augmented Lagrangian multipliers as

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

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

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

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:

$$\begin{aligned}
&1.\ x^{k+1} = \arg\min_x\{L_\beta^3(x, y^k, z^k, \lambda^k) \mid x \in \mathbb{X}\};\\
&2.\ y^{k+1} = \arg\min_x\{L_\beta^3(x^{k+1}, y, z^k, \lambda^k) \mid y \in \mathbb{Y}\};\\
&3.\ z^{k+1} = \arg\min_x\{L_\beta^3(x^{k+1}, y^k, z, \lambda^k) \mid z \in \mathbb{Z}\};\\
&4.\ \lambda^{k+1} = \lambda^k - \beta(A_1x^{k+1} + A_2y^{k+1} + A_3z^{k+1} - b).
\end{aligned}$$

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:

$$\begin{aligned}
&1.\ x^{k+1} = \arg\min_x\{L_\beta^3(x, y^k, z^k, \lambda^k) \mid x \in \mathbb{X}\},\\
&2.\ y^{k+1} = \arg\min_x\{L_\beta^3(x^{k+1}, y, z^k, \lambda^k) + \tfrac{\tau}{2}\beta\|A_2(y - y^k)\|^2 \mid y \in \mathbb{Y}\},\\
&3.\ z^{k+1} = \arg\min_x\{L_\beta^3(x^{k+1}, y^k, z, \lambda^k) + \tfrac{\tau}{2}\beta\|A_3(z - z^k)\|^2 \mid z \in \mathbb{Z}\},\\
&4.\ \lambda^{k+1} = \lambda^k - \beta(A_1x^{k+1} + A_2y^{k+1} + A_3z^{k+1} - b),
\end{aligned}$$

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:

$$
\begin{aligned}
&1.\ x^{k+1} = \arg\min_x \{L^3(\boldsymbol{x}, y^k, z^k, \lambda^k) \mid x \in \mathbb{X}\};\\
&2.\ y^{k+1} = \arg\min_x \{L^3_\beta(x^{k+1}, \boldsymbol{y}, z^k, \lambda^k) \mid y \in \mathbb{Y}\};\\
&3.\ z^{k+1} = \arg\min_x \{L^3_\beta(x^{k+1}, y^{k+1}, \boldsymbol{z}, \lambda^k) \mid z \in \mathbb{Z}\};\\
&4.\ \lambda^{k+1} = \lambda^k - \beta(A_1x^{k+1} + A_2y^{k+1} + A_3z^{k+1} - b).
\end{aligned}
$$

where $L^3(x, y^k, z^k, \lambda^k) = f_1(x) + f_2(y^k) + f_3(z^k) - \lambda^{k^T}(A_1x + A_2y^k + A_3z^k - b)$
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
- http://www.math.ucla.edu/~wotaoyin/papers/pdf/three_op_splitting_wotao_yin_40_min.pdf
- ftp://ftp.math.ucla.edu/pub/camreport/cam15-13.pdf
- http://www.math.ucla.edu/~wotaoyin/papers/pdf/three_operator_splitting_ICCM16.pdf

---

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

1. Primal update:
    - Pick a permutation σ of $1, \ldots, n$ uniformly at random;
    - For $i = 1, 2, \cdots, n$, compute $x^{k+1}_{\sigma(i)}$ by

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

2. Dual update. Update the dual variable by

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

- Randomly Permuted ADMM
- http://opt-ml.org/oldopt/papers/OPT2015_paper_47.pdf
- https://arxiv.org/abs/1503.06387
- https://community.apan.org/cfs-file/__key/docpreview-s/00-00-01-07-11/Ye.pdf

---

Stochastic ADMM

Linearly constrained stochastic convex optimization is given by

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

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 ϑ 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

- http://maths.nju.edu.cn/~hebma/
- https://www.ece.rice.edu/~tag7/Tom_Goldstein/Split_Bregman.html
- https://www.birs.ca/cmo-workshops/2017/17w5030/files/
- http://stanford.edu/~boyd/admm.html
- A General Analysis of the Convergence of ADMM
- http://shijun.wang/2016/01/19/admm-for-distributed-statistical-learning/
- https://www.wikiwand.com/en/Augmented_Lagrangian_method
- https://blog.csdn.net/shanglianlm/article/details/45919679
- https://tlienart.github.io/pub/csml/cvxopt/split.html

Primal-dual fixed point algorithm

- http://math.sjtu.edu.cn/faculty/xqzhang/Publications/PDFPM_JCM.pdf

- http://math.sjtu.edu.cn/faculty/xqzhang/publications/CHZ_IP.pdf
- https://link.springer.com/content/pdf/10.1007%2Fs10915-010-9408-8.pdf
- Proximal ADMM

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 additively 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 $f_i(x_i), i=1,2,\cdots,n$ 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 $k=1,2,3,\ldots$:

---

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

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

- http://bicmr.pku.edu.cn/conference/opt-2014/index.html
- https://calculus.subwiki.org/wiki/Additively_separable_function
- https://www.cs.cmu.edu/~ggordon/10725-F12/slides/25-coord-desc.pdf

- http://bicmr.pku.edu.cn/~wenzw/opt2015/multiconvex_BCD.pdf

Stochastic Gradient Descent

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 θ, 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

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

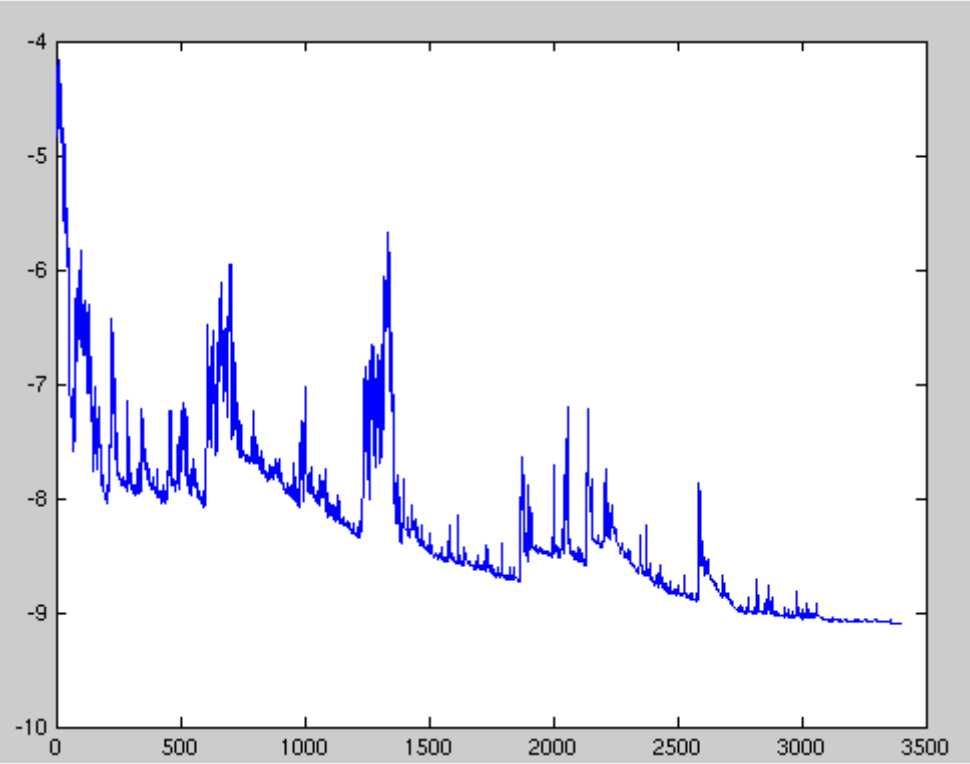
where x'_j is stochastically draw from $\{x_i\}_{i=1}^n$ and $m \ll n$.

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

---

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**[^14].
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 ρ 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, η is chosen to provide a rapid reduction ($\eta = 0.5$), and n is the minimum integer such that the reduced rate η^n succeeds in diminishing the objective function.[^13]

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 α_k by line search.
And most stochastic gradient methods are to find proper step length α_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 α_k diminishes such as $\alpha_k = \frac{\alpha}{k}$, where α is constant.
- And another strategy is to tune the step length adaptively such as *AdaGrad, ADAM*.

PS: the step length α_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 $x_1, x_2, \ldots, x_n$ the mean is estimated as $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$. If now we observed more data $y_1, y_2, \ldots, y_m$ 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 $x_1, \ldots, x_n$.

Another example, it is `Newton interpolation formula` in numerical analysis. The task is to fit the function via polynomials given some point in th function $(x_i, f(x_i)), i = 1, 2, \ldots, n$.
The Newton form of the interpolating polynomial is given by

$$P_n(x) = a_0 + a_1(x - x_1) + a_2(x - x_1)(x - x_2) + \cdots + a_n(x - x_1)(x - x_2)(x - x_3)\cdots(x - x_n).$$

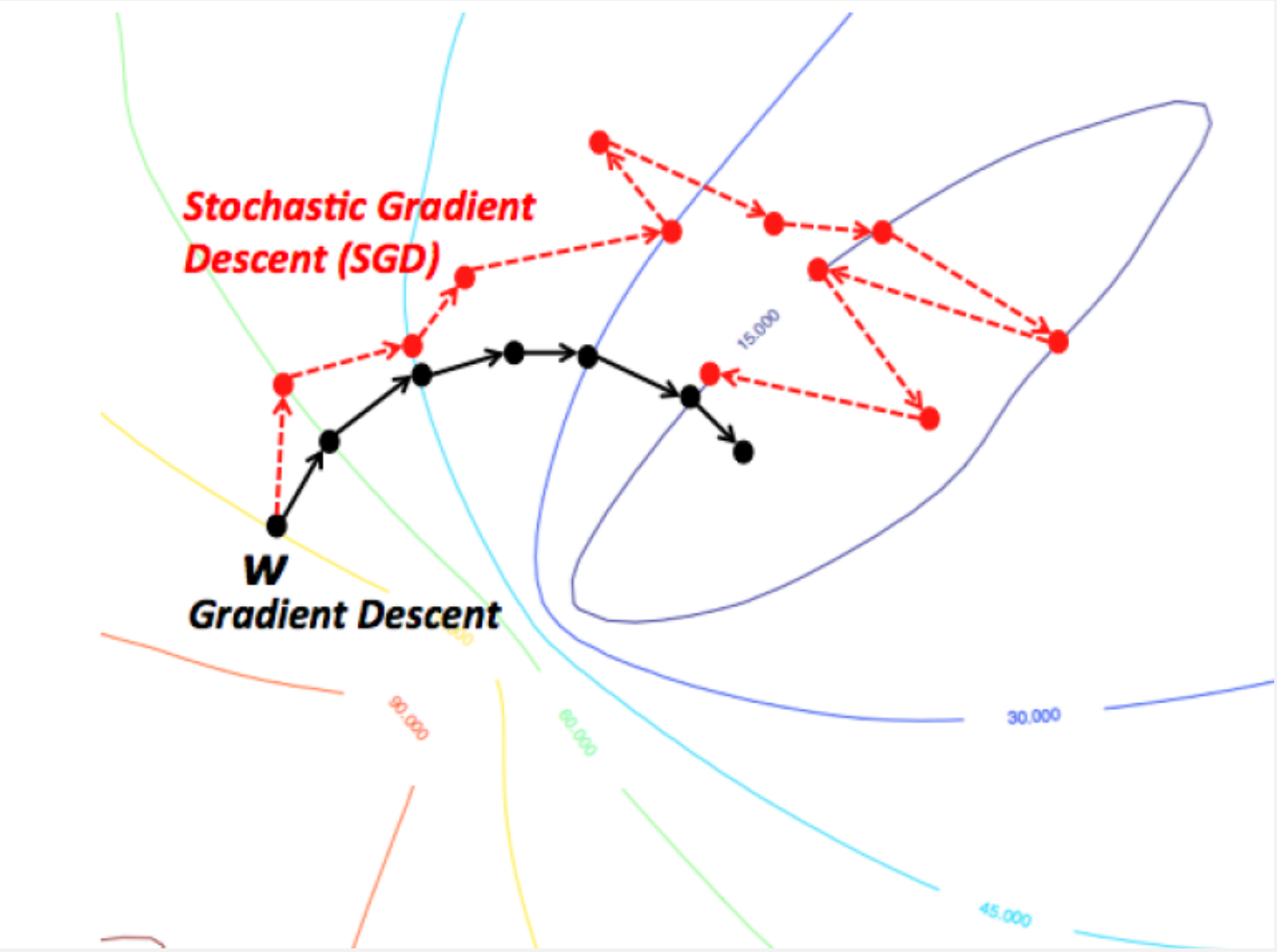
This form is incremental and if another points $(x_{n+1}, f(x_{n+1}))$ is observed we will fit the function f more precisely just by adding another term $a_{n+1}(x - x_1)(x - x_2)\cdots(x - x_n)(x - x_{n+1})$
where the coefficients $a_0, a_1, \cdots, a_n$ are determined by $f(x_1), f(x_2), \cdots, f(x_n)$.

See the following links for more information on *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://ruder.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
- VR-SGD

The Differences of Gradient Descent and Stochastic Gradient Descent

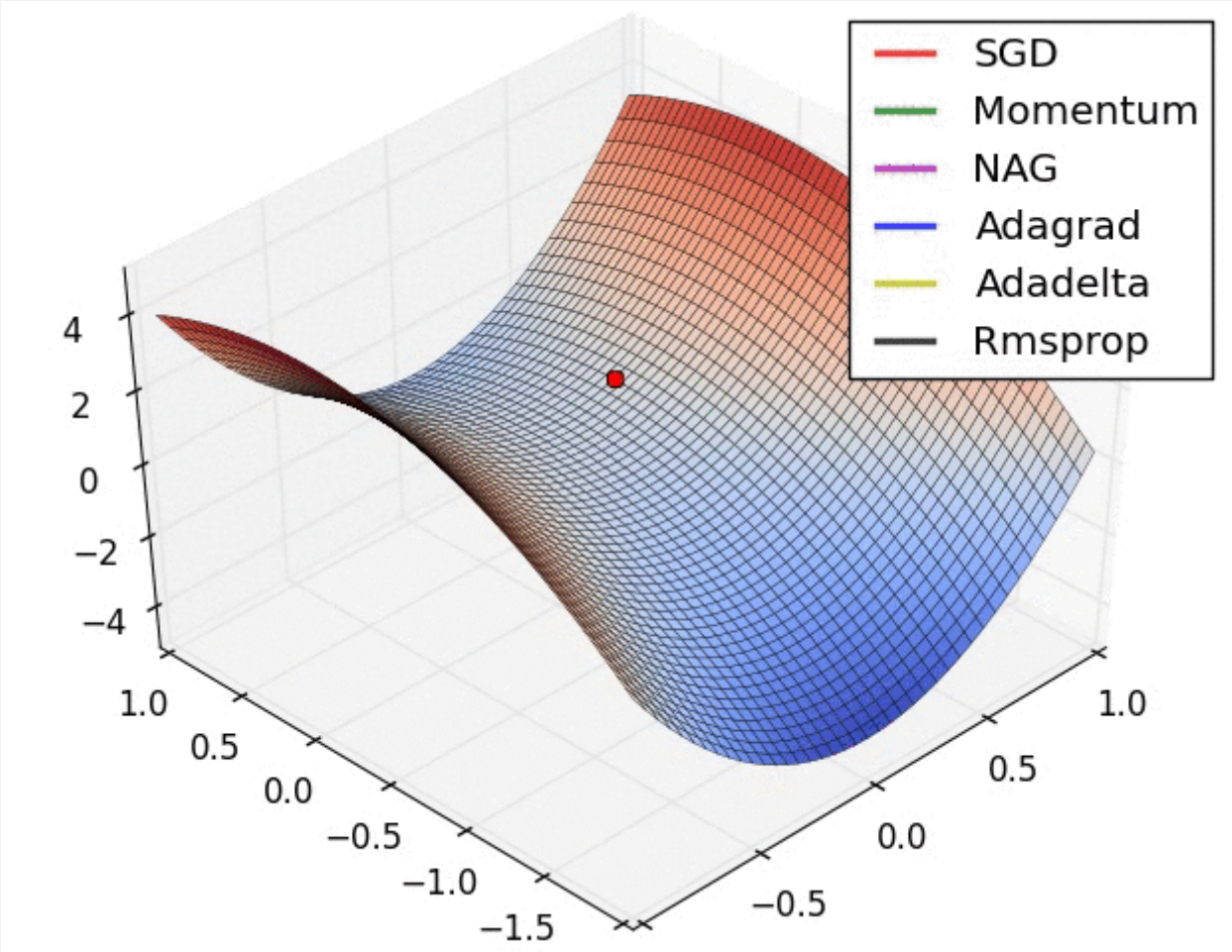
The Differences of Gradient Descent and Stochastic Gradient Descent



Adam

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

The Differences of Stochastic Gradient Descent and its Variants



PS: Zeyuan Allen-Zhu and others published much work on acceleration of stochastic gradient descent.

- https://arxiv.org/pdf/1811.03962.pdf

Surrogate Loss Functions

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

It is obvious that the choice of optimization methods relies on the objective function.

- Divergences, surrogate loss functions and experimental design
- http://fa.bianp.net/teaching/2018/eecs227at/
- 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

Fixed Point Iteration Methods

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

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

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

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). Then consider the following algorithm.

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

So that finally we obtain an sequence $\{x^0, x^1, \cdots, x^n, \cdots\}$. 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.

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

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

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

Now we consider the necessary condition of unconstrainted optimization problems:

$$\nabla f(x) = 0 \Rightarrow x - \alpha \nabla f(x) = x$$
$$\nabla f(x) = 0 \Rightarrow H(x)x - \alpha \nabla f(x) = H(x)x \Rightarrow x - \alpha H(x)^{-1} \nabla f(x) = x$$
$$\nabla f(x) = 0 \Rightarrow M(x) \nabla f(x) = 0 \Rightarrow x - \alpha M(x) \nabla f(x) = x$$

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

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

And the projected (sub)gradient methods are in the fixed-point ierative 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.

This will lead to the operator splitting methods analysed by Wotao Yin.
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.

- 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
- Anderson Acceleration

- [ ] 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://arxiv.org/pdf/1712.07897.pdf
- [ ] https://arxiv.org/pdf/1707.02444.pdf
- [ ] http://www.vision.jhu.edu/assets/HaeffeleCVPR17.pdf
- [ ] https://core.ac.uk/display/73408878
- [ ] https://www.springer.com/us/book/9783319314822

- [ ] https://core.ac.uk/display/83849901
- [ ] https://zhuanlan.zhihu.com/p/51514687
- [ ] http://math.cmu.edu/~hschaeff/research.html
- [ ] https://people.eecs.berkeley.edu/~elghaoui/Teaching/EECS127/index.html