Matrix Derivatives and Descent Optimization Methods

Qiang Ning

Department of Electrical and Computer Engineering

Beckman Institute for Advanced Science and Techonology

University of Illinois at Urbana-Champaign

Fall 2013

Abstract

In the recent few weeks I keep working on the Variable Projection (VARPRO) method. Here I summarize the several main points I have met in these days: matrix/complex calculus, optimization methods and conjugate gradient descent method. As all of them are fundamental knowledge, hopefully this summary can serve as a reference in the future.

I. Multivariable Derivatives

A. Single Variable Scalar Function

We firstly review multivariable calculus for real functions before we move on to matrix and complex calculus. We know that the Taylor expansion of a function $f : \mathbb{R} \to \mathbb{R}$ is:

$$\begin{aligned} f(x) &= f(x_0) + \frac{df}{dx}|_{x=x_0}(x - x_0) + \frac{d^2f}{dx^2}|_{x=x_0}(x - x_0)^2 + ... \\ &= \sum_{n=0}^{\infty} \frac{d^n f}{dx^n}|_{x=x_0}(x - x_0)^n. \end{aligned} \tag{1}$$

This is an important result for analytic functions. Approximations using the partial sums of the series can simplify problems, which we can see in the following sections.

B. Multivariable Scalar Function

Then consider a multivariable function $f : \mathbb{R}^n \to \mathbb{R}$. Its second-order Taylor expansion can be written similarly as:

$$f(\mathbf{x}) = f(\mathbf{x}_0) + \bigtriangledown f|_{\mathbf{x}=\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^T H|_{\mathbf{x}=\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + ..., \tag{2}$$

where $\mathbf{x} \in \mathbb{R}^n$ and written as $\mathbf{x} = [x_1, x_2, ..., x_n]^T$, $\bigtriangledown f$ is the gradient of function f and defined as $\bigtriangledown f = [\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, ..., \frac{\partial f}{\partial x_n}]$, and H is a $n \times n$ matrix which is the so-called Hessian matrix:

$$H = \begin{pmatrix} \frac{\partial^2 f}{\partial^2 x_1} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial^2 x_2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial^2 x_n} \end{pmatrix}. \tag{3}$$

The general term of the Taylor expansion for multivariable functions is much more complex than the second-order expansion above:

$$f(\mathbf{x}) = \sum_{m_1=0}^{\infty} \sum_{m_2=0}^{\infty} \cdots \sum_{m_n=0}^{\infty} \frac{(x_1 - x_{1,0})^{m_1} \cdots (x_2 - x_{2,0})^{m_2}}{m_1! \cdots m_n!} (\frac{\partial^{m_1+m_2+\cdots+m_n} f}{\partial^{m_1} x_1 \cdots \partial^{m_n} x_n})|_{\mathbf{x}=\mathbf{x}_0}. \tag{4}$$

C. Multivariable Vector Function

A more general case is to expand a vector function $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$,

$$\mathbf{f}(\mathbf{x}) = f(\mathbf{x}_0) + J|_{\mathbf{x}=\mathbf{x}_0}(\mathbf{x} - \mathbf{x}_0) + o(\|\mathbf{x} - \mathbf{x}_0\|), \tag{5}$$

where the Jacobian matrix $J_{m \times n}$ is defined as:

$$J = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \cdots & \frac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \frac{\partial f_m}{\partial x_2} & \cdots & \frac{\partial f_m}{\partial x_n} \end{pmatrix}. \tag{6}$$

From the equations (1) (2) (5) above, we can find some consistence among them: both gradient $\bigtriangledown f$ and the Jacobian matrix J can be regarded as "first derivatives". The difference between them is that the gradient is the first derivative of a scalar multivariable function, and Jacobian matrix is of a vector multivariable function. Furthermore, the Hessian matrix, which is in the sense of second-derivatives, can be regarded as the Jacobian matrix of the gradient of a scalar function.

Later when we introduce the notation of matrix calculus, both the gradient and Jacobian can be written uniformly:

$$\bigtriangledown f = \frac{\partial f}{\partial \mathbf{x}}, \tag{7}$$

$$J(\mathbf{f}) = \frac{\partial \mathbf{f}}{\partial \mathbf{x}}. \tag{8}$$

TABLE I

MATRIX CALCULUS IN NUMERATOR LAYOUT [1]

	Scalar	Vector	Matrix
Scalar	$\frac{dy}{dx}$	$\frac{d\mathbf{y}}{dx} = [\frac{\partial y_i}{\partial x}]$	$\frac{d\mathbf{Y}}{dx} = [\frac{\partial y_{ij}}{\partial x}]$
Vector	$\frac{dy}{d\mathbf{x}} = [\frac{dy}{dx_j}]$	$\frac{d\mathbf{y}}{d\mathbf{x}} = [\frac{\partial y_i}{\partial x_j}]$	
Matrix	$\frac{dy}{d\mathbf{X}} = [\frac{\partial y}{\partial x_{ji}}]$		

And the Hessian matrix can be written as:

$$H = \frac{\partial(\bigtriangledown f^T)}{\partial \mathbf{x}} = J(\bigtriangledown f^T). \tag{9}$$

II. MATRIX DERIVATIVES

Matrix calculus is a useful notation to simplify the formulation of multivariable calculus.

A. Layout Conventions

Note that there are mainly two notation conventions of matrix calculus that are used in various of fields: numerator layout and denominator layout. For example, when handling the derivative of a vector with respect to a vector, i.e., $\frac{\partial \mathbf{y}}{\partial \mathbf{x}}$, where $y \in \mathbb{R}^m, x \in \mathbb{R}^n$. The result can either be arranged as an $m \times n$ matrix (numerator layout) or $n \times m$ matrix (denominator layout). The same issue arises whether to treat the gradient of a scalar function, $\frac{\partial f}{\partial \mathbf{x}}$, as a row vector or a column vector. The key concept is to lay the derivatives according to $\mathbf{y}$ and $\mathbf{x}^T$ (numerator layout) or $\mathbf{y}^T$ and $\mathbf{x}$ (denominator layout).

It does not matter which convention one chooses. However, consistence should be maintained throughout one complete work.

A lookup table for numerator layout (which we used in Eqn. (7)-(9)) is shown in TABLE I. Note that the indexes i, j represent the row index and column index, respectively.

B. Basic Examples

Some basic examples are shown in TABLE II, where we treat all vectors as a column vector. The proofs are trivial following the steps below:

1) Clarify the dimentions of both terms.

TABLE II

MATRIX CALCULUS EXAMPLES

Derivative	Numerator Layout	Denominator Layout
$\frac{\partial a^T x}{\partial x}$	a^T	a
$\frac{\partial Ax}{\partial x}$	A	A^T
$\frac{\partial a^T X b}{\partial X}$	ba^T	ab^T
$\frac{\partial x^T A x}{\partial x}$	$x^T(A + A^T)$	$(A + A^T)x$
$\frac{\partial x^T A x}{\partial x^T}$	$(A + A^T)x$	$x^T(A + A^T)$
$\frac{\partial^2 x^T A x}{\partial x \partial x^T}$	$A + A^T$	$A + A^T$
$\frac{\partial \|x\|_2^2}{\partial x}$	$2x^T$	$2x$
$\frac{\partial}{\partial X}\|X\|_F^2$	$2X^T$	$2X$

2) Element-wise calculate derivatives.

3) Put the results according to a specific layout convention.

More examples about derivatives of determinants, inverse matrices, eigenvalues, traces and norms can be found in the Matrix Cookbook by K. B. Petersen [2].

III. COMPLEX DERIVATIVES

A complex scalar function $f : \mathbb{C} \to \mathbb{C}$ is differentiable if the limit,

$$\lim_{z \to z_0} \frac{f(z) - f(z_0)}{z - z_0}, \tag{10}$$

exists over all possible approaches to z_0.

A function $f(x + iy) = u(x, y) + iv(x, y)$ is differentiable at point $z_0 = x_0 + iy_0$ if and only if the real functions u, v satisfy the Cauchy-Riemann equations at point z_0:

$$\frac{\partial u}{\partial x} = \frac{\partial v}{\partial y}, \ \frac{\partial v}{\partial x} = -\frac{\partial u}{\partial y}, \tag{11}$$

or equivalently,

$$\frac{\partial f}{\partial y} = i\frac{\partial f}{\partial x}. \tag{12}$$

In this case, we can calculate the derivative of f by:

$$\frac{df}{dz} = \frac{\partial u}{\partial x} + i\frac{\partial v}{\partial x}. \tag{13}$$

A function f is further said to be holomorphic or analytic[1] at z_0 if it is differentiable at every point in an open neighbor of z_0, which implies that the Eqn. (11) is valid throughout the region we are dealing with.

Another complex derivative, the conjugate complex derivative, is introduced as:

$$\frac{df}{d\bar{z}} = \frac{1}{2}(\frac{\partial f}{\partial x} + i\frac{\partial f}{\partial y}). \tag{14}$$

It is obvious that f is differentiable if $\frac{df}{d\bar{z}} = 0$, which implies roughly that f is functionally independent from the complex conjugate of z. When deriving a complex gradient, the conjugate complex derivative is useful. If f is a real function of a complex vector $\mathbf{z}$, then the complex gradient vector is[2]:

$$\begin{aligned} \bigtriangledown f(\mathbf{z}) &= 2\frac{\partial f}{\partial \bar{\mathbf{z}}} \\ &= \frac{\partial f}{\partial \mathbf{x}} + i\frac{\partial f}{\partial \mathbf{y}}. \end{aligned} \tag{15}$$

The same result holds if $\mathbf{z}$ is a matrix. These expressions can be used for gradient descent algorithms.

The chain rule for complex numbers can be expressed as following[2].

$$\begin{aligned} \frac{\partial g(u(x))}{\partial x} &= \frac{\partial g}{\partial u}\frac{\partial u}{\partial x} + \frac{\partial g}{\partial \bar{u}}\frac{\partial \bar{u}}{\partial x} \\ &= \frac{\partial g}{\partial u}\frac{\partial u}{\partial x} + (\frac{\partial \bar{g}}{\partial u})^*\frac{\partial u}{\partial x}, \end{aligned} \tag{16}$$

where $*$ also denotes complex conjugate. If the function $g(u)$ is analytic, the second term turns to be zero.

Some basic examples are shown in TABLE III.

[1]To be precise, the definition of holomorphic and analytic are different in analysis: A complex function is defined as holomorphic at a point a if it is differentiable at every point within some open neighbor of a, while is defined as analytic at a if it can be expanded as a converged power series in some open neighbor of a. However, one major theorem of complex analysis is that holomorphic functions equals analytic (which is not true for real functions). This is why holomorphic and analytic are interchangeable in the sense of complex analysis.

TABLE III
COMPLEX CALCULUS EXAMPLES

Function	Derivative
$z^k (k$ integer)	kz^{k-1}
e^z	e^z
$f(z) + g(z)$	$f'(z) + g'(z)$
$f(z)g(z)$	$f'(z)g(z) + f(z)g'(z)$
$\frac{1}{f}$	$-\frac{1}{f^2} f'$

IV. UNCONSTRAINED OPTIMIZATION METHODS

Unconstrained optimization problems usually focus on the minimization of a scalar function over multivariables $f : \mathbb{R}^n \to \mathbb{R}$, i.e.,

$$x^\star = arg\min f(x). \tag{17}$$

Descent methods are a main class of unconstrained optimization methods. Descent methods are to produce a minimizing sequence[3] $x^{(k)}, k = 1, 2, \ldots$ for function f, where

$$x^{(k+1)} = x^{(k)} + t^{(k)} \Delta x^{(k)} \tag{18}$$

and $t^{(k)} > 0$. t is called the step length and $\Delta x^{(k)}$ is called the search direction. The search direction in a descent method must satisfy

$$\bigtriangledown f(x^{(k)}) \Delta x^{(k)} < 0. \tag{19}$$

The general descent algorithm is as following.

1) Given a starting point $x^{(0)}$.
2) Repeat until stopping criterion is satisfied.
   a) Determine a descent search direction.
   b) Line search. Choose a step length $t > 0$.
   c) Update. $x := x + t\Delta x$.

A. Gradient Descent Method

A simple choice for the search direction is the negative gradient $\Delta x = - \bigtriangledown f(x)^T$, and the resulting algorithm is called the gradient descent method. According to the Chapter 9.3.1 [3], the converging rate of gradient descent method with exact line search would be linear:

$$f(x^{(k)}) - x^\star \leq c^k (f(x^{(0)}) - x^\star), \tag{20}$$

$$c = 1 - m/M < 1, \tag{21}$$

$$c^k \approx 1 - km/M, \tag{22}$$

where m and M come from the assumption of strong convexity,

$$mI \preceq H(f) \preceq MI.$$

B. Newton's Method

Another descent direction is called the Newton step

$$\Delta x_{nt} = -H(f(x))^{-1} \bigtriangledown f(x)^T. \tag{23}$$

Positive definiteness of $H(f(x))$ implies that Δx_{nt} satisfies the descent direction requirement (19),

$$\bigtriangledown f(x) \Delta x_{nt} = - \bigtriangledown f(x) H(f(x))^{-1} \bigtriangledown f(x)^T < 0, \tag{24}$$

unless x is optimal and $\bigtriangledown f(x) = 0$.

The motivation of Newton's Method comes from the second-order Taylor approximation as shown in Eqn. (2), which we rewrite here as

$$\hat{f}(x) = f(x_0) + \bigtriangledown f|_{x=x_0} \Delta x + \frac{1}{2} \Delta x^T H|_{x=x_0} \Delta x. \tag{25}$$

Referring to the third column of TABLE II of matrix calculus, the minimizer of $\hat{f}(x)$ satisfies

$$\frac{\partial \hat{f}}{\partial x} = \bigtriangledown f(x_0)^T + H(f(x_0)) \Delta x \equiv 0, \tag{26}$$

which directly leads to the Newton step (23).

It is also mentioned in [3] that the Newton step can be regarded as the steepest descent direction for the quadratic norm defined by the Hessian matrix. And the converging rate is said to be quadratic.

C. Quasi-Newton Method

The quasi-newton methods are based on Newton's method. While Newton's method uses the Hessian matrix, quasi-newton methods does not need to compute the Hessian matrix directly, but to update it by analyzing successive gradient vectors instead.

Rewriting $\frac{\partial \hat{f}}{\partial x}$ by $\bigtriangledown f(x)^T$ in Eqn. (26), we have,

$$\bigtriangledown f(x)^T = \bigtriangledown f(x_0)^T + H(f(x_0))\Delta x \equiv 0. \tag{27}$$

It is easy to see that an approximation of the Hessian matrix $H(\hat{f}(x))$ can be solved from the equation above. The various quasi-newton methods just differ in their choice of the solution of it. Some of the most common ones are named as DFP, BFGS, etc., which can be easily found through web search.

D. Non-linear Least Squares Problem

Linear least squares problem is to solve

$$x_0 = arg\min_{x \in \mathbb{F}^n} \|Ax - b\|, \tag{28}$$

where $A \in \mathbb{F}^{m \times n}$ ($\mathbb{F} = \mathbb{R}$ *or* $\mathbb{C}$, and $m \geq n$) and $b \in \mathbb{F}^n$. The fact is that the solution of x_0 to this least squares problem is

$$x_0 = A^{\dagger}b + y_0, \tag{29}$$

where $y_0 \in ker(A)$ [4]. Among all the selections of y_0, $x_0 = A^{\dagger}b$ is the unique solution of minimum length; specifically, if A has full rank of n, $x_0 = A^{\dagger}b$ is the unique solution to (28).

As we can see the linear least squares problem has been well handled, the general non-linear squares problem

$$\beta^{\star} = arg\min_{\beta \in \mathbb{F}^n} \|r(\beta)\|_2^2, \tag{30}$$

where $r \in \mathbb{F}^m$ and $m \geq n$, still remains to be carefully considered.

Matlab: Least Squares (Model Fitting). "Although the function in LS can be minimized using a general unconstrained minimization technique, as described in Basics of Unconstrained Optimization, certain characteristics of the problem can often be exploited to improve the iterative efficiency of the solution procedure. The gradient and Hessian matrix of LS have a special structure...Consider the efficiencies that are possible with the Gauss-Newton method. Gauss-Newton Method on Rosenbrock's Function shows the path to the minimum on Rosenbrock's

function when posed as a least-squares problem. The Gauss-Newton method converges after only 48 function evaluations using finite difference gradients, compared to 140 iterations using an unconstrained BFGS method."

1) Gauss-Newton Algorithm: Gauss-Newton algorithm (GNA) can be seen as a modification of the Newton's method. The difference is that GNA can only be used in least squares problems but without the labor of estimating the Hessian matrix.

Using the expansion of a vector function as shown in Eqn. (5), we have

$$r(\beta) \approx r(\beta^{(s)}) + J(r(\beta^{(s)}))(\beta - \beta^{(s)}), \tag{31}$$

where $J(r(\beta^{(s)}))$ is the Jacobian matrix of r over β at point $\beta^{(s)}$.

Substituting $r(\beta)$ in Eqn. (30) by (31), we have a linear least squares problem in the same form as Eqn. (28),

$$\beta^{\star} \approx arg \min_{\beta \in \mathbb{F}^n} \|r(\beta^{(s)}) + J(r(\beta^{(s)}))\Delta\beta\|_2^2, \tag{32}$$

where $\Delta\beta \triangleq \beta - \beta^{(s)}$. As Jacobian matrix is most likely fully ranked, the solution to this linear least squares problem is $\Delta\beta_{gna} = -J^{\dagger}r = -(J^H J)^{-1}J^H r$, which is exactly the search direction of GNA.

2) Levenberg-Marquardt Algorithm: The Levenberg-Marquardt algorithm seeks an interpolation between GNA and gradient descent method. Note that the gradient of $r(\beta)$ over β equals $-2J^H r$, this interpolation can be achieved by replacing $J^H J$ in $\Delta\beta_{gna}$ by $J^H J + \lambda I$:

$$\Delta\beta_{lma} = -(J^H J + \lambda I)^{-1}J^H r. \tag{33}$$

The smaller λ is, the closer it is to GNA; the bigger λ is, the closer it is to gradient descent.

V. Conjugate Gradient Descent

Conjugate gradient descent method is used to solve linear equations:

$$Ax = b, \tag{34}$$

where A is a $n \times n$ positive definite matrix and b is a $n \times 1$ vector.

A. Concept

Let x_* be the unique solution to it. As x_* is also the unique minimizer of the quadratic function,

$$f(x) = \frac{1}{2}x^T Ax - x^T b, \tag{35}$$

$f(x)$ is smaller if we are closer to x_*. Therefore we can redefine our problem from solving an equation set (34) to minimizing an object function (35).

Assume the initial guess is $x_0 = 0$ without loss of generality, otherwise we replace b with $b - Ax_0$. If we are using gradient descent method at step k, we will move in the direction of the negative gradient: $b - Ax_k \triangleq r_k$, and we can certainly do that here. However, the gradient descent method has low converging rate, for which we prefer to use the so-called conjugate gradient descent.

In conjugate gradient descent method, we form another set of orthogonal basis p_k using Gram-Schimidt orthonormalization,

$$p_k = r_k - \sum_{i<k} \frac{p_i^T A r_k}{p_i^T A p_i}. \tag{36}$$

And the update is given by,

$$x_{k+1} = x_k + \alpha_k p_k, \tag{37}$$

$$\alpha_k = \frac{p_k^T r_{k-1}}{p_k^T A p_k} \tag{38}$$

B. Convergence Properties

Theoretically, the conjugate descent method gives out the exact solution after a number of n iterations. But it is unstable with respect to even small round-off errors. Fortunately, the conjugate descent method as an iteration method can improve the solution x_k monotonically. This improvement is linear and its speed is determined by the condition number $\kappa(A)$: the larger $\kappa(A)$ is, the slower the improvement[5].

In order to accelarate the convergence, the preconditioner P^{-1} is often used when $\kappa(A)$ is large, i.e., solve the following problem,

$$P^{-1}(Ax - b) = 0, \tag{39}$$

where $\kappa(P^{-1}A)$ is smaller by a proper selected P.

REFERENCES

[1] T. P. Minka, "Old and new matrix algebra useful for statistics," *Citeseer*, 2001.

[2] K. B. Petersen and M. S. Pedersen, "The matrix cookbook. november 2008," November 2008.

[3] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge University Press, 2004.

[4] L. Hogben, *Handbook of linear algebra*. CRC Press, 2006.

[5] Y. Saad, *Iterative methods for sparse linear systems*. SIAM, 2003.